Modeling informational novelty in a conversational system with a hybrid statistical and grammar-based approach to surface natural language generation

Adwait Ratnaparkhi
IBM TJ Watson Research Center
PO Box 218
Yorktown Heights, NY 10598
aratnapa@us.ibm.com

Abstract

We present a hybrid statistical and grammar-based system for surface natural language generation (NLG) that uses grammar rules, conditions on using those grammar rules, and corpus statistics to determine the word order. We also describe how this surface NLG module is implemented in a prototype conversational system, and how it attempts to model informational novelty by varying the word order. Using a combination of rules and statistical information, the conversational system expresses the novel information differently than the given information, based on the run-time dialog state. We also discuss our plans for evaluating the generation strategy.

1 Introduction

We present a module for surface natural language generation (NLG) that is capable of dynamically reordering words based on information in the run-time dialog state of a conversational system in the air travel domain. For our purposes, we make the distinction that deep NLG is the process of deciding what information to convey, whereas surface NLG is the process of rendering that information in natural language. The surface NLG module is used by the conversational system to express the new information differently than old information, similar to how people might express informational novelty in a human-human conversation. The eventual goal of this experiment is to test if strategically re-ordering the words in the response of a conversational system can address a widely held criticism, namely, that such systems "don't sound human."

2 Previous Approaches

The most popular technique for surface NLG is templates. A template for describing a flight noun phrase in the air travel domain might be "flight departing from $city-fr at $time-dep and arriving in $city-to at $time-arr" where the words starting with "$" are actually variables —representing the departure city, and departure time, the arrival city, and the arrival time, respectively— whose values will be extracted from the environment in which the template is used. This approach requires the programmer to write a different template for every possible word ordering, and may be impractical for domains in which many word orderings are necessary.

There are more sophisticated surface generation packages, such as FUF/SURGE (Elhadad and Robin, 1996), KPML (Bateman, 1996), MUMBLE (Meteer et al., 1987), and RealPro (Lavoie and Rambow, 1997), which produce natural language text from abstract semantic representations. These packages use many rules written by linguistic experts to map the input representations to textual output.

In order to partially automate the process of mapping input representations to textual output, several researchers have recently investigated the use of statistics in generation. Our approach is in the same spirit as other recent work, such as such as (Langkilde and Knight, 1998), (Ratnaparkhi, 2000), and (Bangalore and Rambow, 2000), in which statistics from a corpus have been used to disambiguate or rank candidates for surface generation. We compare our approach with these approaches in section 7.

3 Our approach: A Hybrid Grammar and Statistical surface NLG system

The motivation for our approach is the desire for a surface generation framework that allows dynamic word re-ordering in the context of a conversational system. People naturally re-order words in a conversation in a way that maximizes their communicative power, and we hope to duplicate this behavior in an automated conversational system. We describe our surface generation framework, the linguistic behavior it tries to duplicate, and its implementation in an air travel conversational system.

The central idea of our surface NLG module is that given a dependency-like grammar, it generates many word sequences that are consistent with the grammar rules and rule conditions, and uses corpus statistics to find the word sequence that most resembles the real utterances of people. In our framework, the input to the surface NLG module consists of

- a *mandatory* set of attribute-value pairs, A_1
- an *optional* set of attribute-value pairs, A_2

In the air travel domain, an example attribute might be "$city-fr", denoting the departure city, while an example value might be "New York". (Attributes are denoted here by the "$" prefix.) The surface NLG is required to express every attribute in the set A_1, but does not need to express anything from the set A_2. In practice, A_1 carries the intended meaning of the utterance to be generated, while A_2 carries miscellaneous information that is accessible to the NLG module. The sets A_1 and A_2 are extracted from the dialog state of the conversational system, after the dialog manager has determined that it needs to speak an utterance to the user, and after the deep generation module has determined the content of A_1 and A_2. We assume that any relevant fact about the discourse history exists in the dialog state, and can be encoded as an attribute-value pair in either A_1 or A_2. We make the assumption that attribute-value pairs are sufficient to describe the meaning of the utterance we intend to generate. This assumption is reasonable for small domains like air travel.

3.1 Grammar

In our approach, grammar rules define the possible dependency trees the NLG module may generate in the context of the current dialog state. Any dependency tree generated by our grammar can be converted to a flat word sequence by a deterministic procedure.

A grammar rule in this system specifies a relationship between a parent, and one more children, using the following structure:

Parent: This is the parent, and is usually the linguistic "head" of the phrase.

Direction: is either - (left) or + (right), and indicates the intended word order of the children relative to the parent.

Children: One or more words that are children to the parent

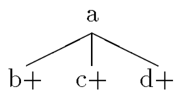
Condition: A code fragment that evaluates to either true or false in the current state of the dialog system (A_1, A_2)

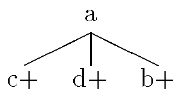
The order of the children specified in a single rule is fixed; it is merely the order written by the programmer. However, there is no ordering constraint between the children of *different* rules with the same head. For example, take the following grammar (in which the **Condition** section has been omitted for clarity),

Parent	Direction	Children
a	+	b
a	+	c d
c	-	f

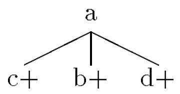
This grammar will allow the following dependency trees,

each of whom represent the application of one grammar rule for the head "a". The + sign denotes a right child, whereas the - sign denotes a left child (regardless of whether the child is visually typeset to the left or right of its parent). The following trees reflect the use of the first two rules, in both possible orderings

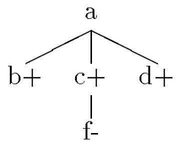




The siblings "c" and "d" cannot be re-ordered or broken up with respect to each other, since they were specified in a single rule. So the tree



is disallowed. The children can themselves be recursively expanded, so the tree



is allowed. The dependency trees can be converted to word sequences (i.e., linearized), by recursively traversing the left children, the parent, and the right children. The word sequence corresponding to the tree above is "a b f c d". It is possible for different dependency trees to yield the same word sequence.

The **Condition** section of the rule specifies an arbitrary code fragment that is evaluated in the context of the attribute-value sets A_1, A_2, which are derived from the current dialog state. The rule is used only if the code fragment evaluates to true. A rule condition associates an element of meaning with its realization as a phrase in natural language. For example, in the air travel domain, Table 1 lists a grammar rule, with a condition in pseudo-code, that might be used to describe a departure city. This rule would allow "flights from New York" if the departure city, as specified in the dialog state, is "New York".

The rule condition can be more complex if necessary; in our implementation, the rule condition is an arbitrary fragment of code in the language Tcl. Also, the rules in our implementation are slightly more abstract in that they may contain attributes in addition to words. In this case the attribute will be instantiated with a value of interest at some later

Parent	Direction	Children	Condition
flights	+	from New York	value of departure-city in dialog state is "New York"

Table 1: Sample rule to describe departure city

Parent	Direction	Children	Condition
flights	+	from \$city-fr	the \$city-fr attribute exists in the dialog state

Table 2: Sample rule with attribute to describe departure city

point. Table 2 shows a rule for describing departure cities that uses attributes. With attributes allowable as children, the output of the NLG module is essentially just a template. The difference between our system and the template method for NLG is that the programmer need only specify template *fragments* in the form of parent/children relationships, instead of the entire template.

Lists with conjunctions are linguistic phenonema that need to be generated frequently in the air travel domain. For example, a list with one item a_1 is realized simply as "a_1", but two items are realized as "a_1 and a_2", while $n > 2$ items are realized as "a_1, ..., a_{n-1}, and a_n". We found it easier to properly generate the conjunction and commas with a built-in construct, as opposed a programmer-supplied grammar rule. We define the constructs "&" and "|" to denote that the children of a rule must be generated with conjunctions ("and" and "or", respectively) and commas. For example, the grammar (with the conditions omitted):

Parent	Direction	Children
a	+&	b
a	+&	c
a	+&	d

will generate (among others) the word sequences

- a b
- a b and c
- a b , c , and d

Of course, the words "and" and "or", are dependent on the language and perhaps even the genre of the language. The comma notation is dependent on the application of interest. It is critical to have the commas placed properly if the generated text will eventually be synthesized into speech; the speech synthesizer relies on commas to generate the appropriate pauses in the speech output.

The system currently has no facility that is specially designed to handle the generation of morphological variants. For example, in the air travel domain, the word "flight" should be realized as "flights", if the number of flights is greater than 1. Similarly, the verb "arrive" must agree with "flight" in the phrase "flight that arrives" versus "flights that arrive". We instead use a generic token re-write facility, in which the programmer, for example, can tell the system to re-write the word "flight" as "flights" based on information in A_1, A_2. This facility can be flexible since different uses of the same word in the grammar can be represented by distinct tokens, (e.g. flight-subj, flight-obj) which are later realized into their morphologically correct spellings. Our system is not meant to be a general purpose generator, and in the future, we plan to extend our system to better handle the generation of morphological variants.

3.1.1 Assigning scores to trees

We assign scores to a dependency tree t by first converting it to a word sequence $w_1 \ldots w_n$, and by using an interpolated n-gram language model on the word sequence:

$$P(w_1 \ldots w_n) = \prod_{i=1}^{n} P(w_i | w_{i-1} \ldots w_1)$$

$$P(w_i | w_{i-1} \ldots w_1) = \sum_{j=1}^{4} \lambda_j P_j(w_i | w_{i-1} \ldots w_1)$$

The probability models P_j are computed from statistics derived from roughly 8000 utterances in the air travel domain. The probability model P_1, P_2, and P_3 are derived from trigram, bigram, and unigram statistics, while P_4 is the uniform model. The λ_j are set heuristically such that $\lambda_j \geq 0$ and $\sum_{j=1}^{4} \lambda_j = 1$.

3.2 Searching for the best dependency tree

The goal of the system is to find the highest-scoring dependency tree that is consistent with the grammar. The strategy is, given an existing tree t, to enumerate all the ways of creating new dependency trees $t_1 \ldots t_n$ that are consistent with the grammar, and to only keep the top N scoring trees for consideration in the next search iteration. The search terminates when N A-completed[1] trees are found, where an A-completed tree mentions all of the attributes in the mandatory attribute-value set A_1 *exactly once*. We justify the restriction to A-completed trees because trees that have omitted one or more attributes are clearly not expressing the meaning of A_1, while we view trees that mention the same attribute more than once as containing redundant information. The

[1] A mnemonic for "attribute-completed"

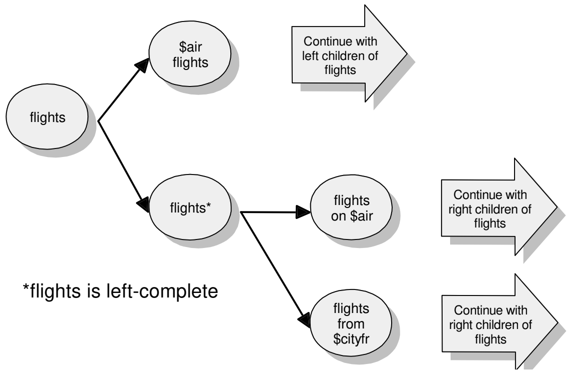


Figure 1: Pictorial depiction of search algorithm

highest scoring A-completed tree is the answer returned by the NLG module. This search strategy is heuristic in nature; it is not guaranteed to find the highest-scoring tree.

On each search iteration, the system finds the top scoring N trees $t_1 \ldots t_n$ that are currently under consideration, and attempts to create a new set of trees by applying the following algorithm to each tree t in the set.

- Check to see if t is A-complete. If so, remove it from consideration. If N trees are A-completed, terminate the search.
- If t is not A-complete, the system determines the *active parent*, by starting at the root of t, and recursively checking the left children, the right children, and then the parent itself, for the first tree node that is not *completed*. A tree node is completed if
  - it is *left-complete*, meaning that all of its left children have been generated, and
  - it is *right-complete*, meaning that all of its right children have been generated
- If no active parent is found, t is discarded, since we cannot apply more rules to make t A-complete.
- If p is the active parent, the system decides to work in the left direction if p is not left-complete, otherwise it works in the right direction.
- Either
  - Apply a rule: once the direction is settled, the system applies a rule r in the grammar if
    - $*$ the parent specified in r is equal to the active parent p
    - $*$ the condition of r evaluates to true
    - $*$ r has not been previously used to generate children for the parent p
    - $*$ the attributes mentioned in the children have not been mentioned elsewhere in the tree
  - If the rule can be applied, add the children in the rule to the active parent. Add from right-to-left if we are adding left children, add from left-to-right if we are adding right children.
  - Use the new tree t' for consideration in the next search iteration
- or mark the tree
  - left-complete, if we were adding in the left direction
  - right-complete, if we were adding in the right direction
  - use the new tree t'' for consideration in the next search iteration

The point of the search algorithm is that it explores many possible word sequences, while requiring the programmer to only specify template fragments in the form of dependency tree parent/children relationships. The programmer can specify several ways to express any given attribute; the search guarantees that any attribute given in A_1 will be mentioned only once in the generated utterance. Intuitively, the system takes the fragments of natural language given by the programmer, explores many ways of "pasting" them together such that they respect the grammar, and returns the "best" way with respect to the scoring function. Figure 1 gives a pictorial depiction of the search procedure looking for ways to express the attributes $city-fr and $air, which represent the departure city and air carrier, respectively. The attributes are instantiated with their corresponding values after the search has found the best candidate for surface generation.

4 Using word order to express informational novelty

It has been long argued that utterances have an *information structure*, such that one part refers to preexisting information in the discourse, while the other part refers to information that is newly introduced into the discourse. There are several existing dichotomies, which capture the same general idea but differ in their details, such as *theme vs rheme*, *topic* *vs. comment*, *presupposed vs. focus*. See (Prevost, 1996) for a summary of different information structure schemes.

We want to model informational novelty, which correlates roughly with the theme vs. rheme distinction, so that old information (theme) is expressed differently than the new information (rheme). Furthermore, at this time, we wish to do it without modifying the pitch, only with word ordering. (Steedman, 1996) gives a more fine-grained information structure, and points out that sub-elements of the theme can also contain new information, and are often emphasized with pitch. However, for our purposes, we choose to model the more simplistic structure of new versus old information, since this is the only distinction we can reliably make in our current dialog system.

In a spoken conversational system, it is usually necessary to confirm to the user what was spoken and understood by the computer in the last turn. This way, the user can ascertain if system's speech recognition module and natural language understanding module are working correctly, and can repeat any information that the computer misunderstood. Another reason for confirmation messages is to remind the user of information that was understood several turns ago. Ideally, we should analyze a sample of text in the domain of interest (air travel) and annotate how the confirmation information (new vs. old) is expressed. However, most confirmations in human-human conversations do not contain both new and old information, and happen in a manner that is not easily reproducible with a speech-to-speech conversational system, as shown in Table 3. In this type of dialog, the user interrupts the travel agent in order to confirm what has been recently spoken, the confirmation is done with an "mm hmm" sound. We have noticed that some confirmations, do contain both old and new information, as shown in Table 4. In this case, the old information ("Buffalo to Chicago") is spoken for confirmational purposes.

Many of the human-human dialogs in the air travel domain that do contain old and new information are expressed in a way such that the old information precedes the new information. At this time, we are still accumulating quantitative evidence in a corpus of transcribed human-human dialogs in the air travel domain to make this claim more precise. Furthermore, in English, it has been long noted that there is a tendency to specify old information before new information (e.g., see (Sornicola, 1999) for a review of many studies).

Note that word order is not the only indication of novelty! It is clear that other factors, such as pitch and loudness, also convey novelty, even when the word order is fixed. We assume that pitch and loudness are roughly constant, and that we can model novelty by only varying the word order. Also, using only word order to model novelty has the advantage that it leaves open the possibility of using our NLG technique for non-spoken text, e.g., an interactive web page.

Agent:	we have you returning on the seventeenth of September on US Air flight five zero seven
User:	mm hmm...
Agent:	out of Syracuse at seven fifty a.m. into Pittsburgh at nine oh five a.m.
User:	mm hmm...

Table 3: Example of confirmation only in human-human dialog

User:	What was the Buffalo to Chicago flight ?
Agent:	ah Buffalo to Chicago is three ninety three

Table 4: Example of agent confirming old information and introducing new information in human-human dialog

5 Application: Modeling informational novelty in a conversational system

The hybrid surface NLG module has been integrated into a telephony conversational system for air travel reservations, developed for the DARPA COMMUNICATOR effort, and described in (Axelrod, 2000). Most system utterances are generated using an existing template-based approach, while a certain class of utterances are generated with the NLG system described in this paper.

The conversational system first collects information from the user, and then consults a flight database to find flights that match the user's constraints. If one flight was found, it asks the user to confirm it. If no flights were found, it asks the user to relax some of the constraints, whereas if many flights were found, it prompts the user to further constrain the flight list. In the case where either many or no flights were found, the first utterance given by the system is called a *summary sentence*, whose purpose is to give a one-sentence summary of the results from the flight database. In the current system, the summary sentence is generated using templates, where some template fragments are "optional", so that they are printed only in the presence of certain attributes. In the existing approach, the generation of certain words is optional, but the order in which they are presented is fixed.

In our new approach, the word order in the summary sentence depends on three sources of information

Grammar: This is the dependency grammar specified by the programmer. Approximately 50 rules were needed to generate the possible summary sentences.

Statistics: These are derived from a corpus of roughly 8000 utterances in the air travel domain, and are used by the NLG module's scoring function.

Attribute Novelty: Each attribute in set of mandatory attributes A_1 is marked as either *old* or *new*. For our purposes, *new* attributes are those which were given to the system in the last user turn. Anything not marked new is assumed to be *old*.

The attributes are marked as old or new by the deep generation component, by using information in the dialog history and some heuristics.

We use the surface NLG module to express the new attributes differently than the old attributes in the summary sentence. The surface NLG system allows us to detect the novelty of an attribute in any of the rule conditions of the grammar, e.g., with a function call that takes the name of an attribute and returns either true or false. Therefore, we can have two kinds of rules for every attribute: one rule to express it when it is new), and another rule to express it when it is old. The grammar for the summary sentence is written in a way to produce sentences having general structure shown in Figure 2. The [old information] is the area in the sentence in which the old information will be expressed, while the [new information] is the area in the sentence in which the new information will be expressed. In the case where there is more than one old and one new attribute, novelty alone does not determine the word order; it merely tells the NLG module the area in the sentence that will contain the phrase expressing the attribute. The complete word order is attained with the scoring function (applied in the search procedure), which ranks the possible different orderings that are consistent with the grammar structure.

Table 5 shows how several grammar rules can be used to control the placement of old and new information. Table 6 contains a dialog with the existing conversational air travel system. Table 7 contains the dialog, with the same user input, but with the system using the NLG module discussed in this paper. Approximately 50 grammar rules were used to generate the summary sentence in our prototype conversational system.

In our approach, the definition of novelty is rather peculiar. We mark an attribute as *new* if it has been introduced by the user in the previous turn, and mark it as *old* otherwise. Using this definition, all of the attributes in the summary sentence are "old" in the sense that they have already been introduced into the dialog by the user; the labels *new* and *old* only distinguish the time at which the user introduced the attributes. Truly novel information, i.e., information from the flight database that has not previously been heard by the user or said by the system, is not expressed in the summary sentence. Unfortunately, in our current approach, we confine the use of the hybrid NLG system to the summary sentence. In future work, we plan to extend it to other kinds of sentences in the system's response in the hopes of modeling the "truly novel" information.

User	i want to fly from new york to pittsburgh tomorrow at ten
Sys	do you want to leave about ten A M or at ten P M?
User	ten A M
Sys	**i see a couple of flights from new-york to pittsburgh which depart about ten A M on september nineteenth.** the first flight is on T W A.. It departs at nine forty A M and arrives at eleven sixteen A M... the second flight is on U S air. It departs at ten twenty five A M and arrives at eleven forty one A M..
User	i want delta
Sys	**i can't find any flights from new-york to pittsburgh on delta which depart about ten A M on september nineteenth.** you may want to try changing the airline to T W A or U S air.

Table 6: Dialog with existing conversational system. Summary sentence is in bold face.

6 Evaluation

Evaluating our surface NLG system in isolation is difficult, because its output is dependent on information in the dialog state of a conversation. We propose to evaluate it in the context of an entire conversational system, by recording the time it takes for a user to complete certain tasks using the existing template generation, and comparing it with how long it takes users to complete the same tasks using the surface NLG module described in this paper. Such evaluations are already underway in the DARPA COMMUNICATOR effort (Walker et al., 2000). We believe that strategically setting the word order will reduce the amount of attention and mental effort necessary from the user in order to successfully use a conversational system. We hope that this reduction in mental effort will allow users to have higher success rates with the system, faster completion times, and eventually, the ability to multitask, i.e., to ability to use the system while engaged in some other secondary activity.

There are N flights [old information] that [new information]

Figure 2: General structure for flight database summary sentence in air travel conversational system

Parent	Direction	Children	Condition
flights	+	from $city-fr	the $city-fr attribute exists in A_1 , $city-fr is old
leave	+	from $city-fr	the $city-fr attribute exists in A_1, $city-fr is new
flights	+	that	if at least one attribute is marked new
that	+	leave	if $city-fr, $date-dep, or $time-dep are new

Table 5: Sample rules to describe departure city as either old or new information

User	i want to fly from new york to pittsburgh tomorrow at ten
Sys	do you want to leave about ten A M or at ten P M?
User	ten A M
Sys	**there are several flights from new-york to pittsburgh on september nineteenth that *leave around ten A M.*** the first flight is on T W A.. It departs at nine forty A M and arrives at eleven sixteen A M... the second flight is on U S air. It departs at ten twenty five A M and arrives at eleven forty one A M..
User	i want delta
Sys	**there are no flights from new-york to pittsburgh on september nineteenth around ten A M that *are served by delta*.** you may want to try changing the airline to T W A or U S air.

Table 7: Dialog with conversational system using surface NLG module for summary sentence. Summary sentence is in bold face. New attributes are italicized.

7 Comparison with other work

Our work is similar to (Langkilde and Knight, 1998), (Bangalore and Rambow, 2000), and (Ratnaparkhi, 2000) in that we use statistical information to select between multiple candidates for surface generation. (Langkilde and Knight, 1998) use statistics to select the best generation candidate from a word lattice generated from a grammar, while (Bangalore and Rambow, 2000) use statistics to select the word order of an underspecified dependency tree generated from a grammar. (Ratnaparkhi, 2000) uses statistics to rank candidates given by a grammar induced from a dependency tree annotated corpus.

Our approach differs from previous approaches in that it is specifically directed towards modeling informational novelty in a conversational system. In our system, the programmer can impose *partial* constraints on the ordering, using rule conditions to finely control the ordering in some cases, e.g., the novel attributes, while leaving other cases for the statistics to disambiguate. To our knowledge, previous approaches have not addressed informational novelty in a conversational system, although we suspect that they could be adapted to do so as well.

The hybrid surface NLG module described in this paper is *not* meant to be a general purpose generation package. Instead, it is designed to generate utterances in a small domain, such as air travel, and provides a framework to experiment with the ability to express additional shades of meaning by varying the word order at run-time. While our hybrid surface NLG system is not linguistically sophisticated as other full-fledged generation packages, the grammar rules are easy to write, and do not require much linguistic expertise. For this reason, we believe it is more practical than the other full-fledged generation packages. Our hope is that programmers will be able to implement NLG in a conversational system without needing to know how to specify detailed linguistic descriptions, as are usually required by the more sophisticated NLG packages. We hope to extend the framework with some useful facilities as they are needed by our conversational system. For example, we hope to add a facility to pass semantic “features” from a parent to a child, and an interface with a morphological database, which will more properly deal with phenomena such as agreement and inflection. Furthermore, we hope to add these extensions without compromising the simplicity of the grammar rule structure.

8 Conclusion

We have presented a system for surface natural language generation that uses grammar rules, rule conditions, and statistical information to decide the word order at run time. It takes template fragments given by the programmer, and attempts to paste them together in a way that is both consistent with the grammar and optimal with respect to the scoring function. We have integrated it into a conversational system for air travel and have attempted to model the linguistic notion of focus with attribute novelty. To our knowledge, we are the first to model informational novelty in a surface generation system with a combination of grammar rules and statistics, and we are also the first to integrate this into a practical conversational system. Our NLG module is not intended as a general purpose generator, and appears adequate for domains of low complexity. We hope to more extensively use our surface NLG module in our conversational system, and we hope that future evaluations will reveal that strategically varying the word order makes the system talk more like a real person.

References

Scott Axelrod. 2000. Natural Language Generation in the IBM Flight Information System. In *ANLP/NAACL Workshop: Conversational Systems*, Seattle, Washington.

Srinivas Bangalore and Owen Rambow. 2000. Exploiting a Probabilistic Hierarchical Model for Generation. In *International Conference on Computational Linguistics (COLING)*, Saarbrucken, Germany, August.

John Bateman. 1996. KPML Development Environment – Multilingual linguistic resource development and sentence generation. Technical report, German Centre for Information Technology (GMD), Institute for Integrated Information and Publication Systems (IPSI), Darmstadt, Germany.

Michael Elhadad and Jacques Robin. 1996. An Overview of SURGE: a Reusable Comprehensive Syntactic Realization Component. Technical Report 96-03, Ben Gurion University, Beer Sheva, Israel.

Irene Langkilde and Kevin Knight. 1998. Generation that Exploits Corpus-Based Statistical Knowledge. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics*, University of Montreal, Montreal, Quebec, Canada.

Benoit Lavoie and Owen Rambow. 1997. A Fast and Portable Realizer for Text Generation Systems. In *Proceedings of the Fifth Conference on Applied Natural Language Processing*, pages 265–268, Washington D.C., March 31–April 3.

M. W. Meteer, D. D. McDonald, S.D. Anderson, D. Forster, L.S. Gay, A.K. Huettner, and P. Sibun. 1987. Mumble-86: Design and implementation. Technical Report Technical Report COINS 87-87, University of Massachusetts at Amherst.

Scott Allan Prevost. 1996. *A Semantics of Contrast and Information Structure for Specifying Intonation in Spoken Language Generation*. Ph.D. thesis, University of Pennsylvania.

Adwait Ratnaparkhi. 2000. Trainable Methods for Surface Natural Language Generation. In *First Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL)*, Seattle, Washington, April 29 — May 4.

R. Sornicola. 1999. Topic, Focus, and Word Order. In Keith Brown and Jim Miller, editors, *Concise Encyclopedia of Grammatical Categories*, pages 375–383. Elsevier.

Mark Steedman. 1996. Representing discourse information for spoken dialogue generation. In *International Symposium on Spoken Dialogue, held in conjunction with ICSLP-96*, Philadelphia, Oct.

Marilyn A. Walker, Lynette Hirschman, and John Aberdeen. 2000. Evaluation For Darpa Communicator Spoken Dialogue Systems. In *Language Resource and Evaluation Conference (LREC)*.