Generating Concise Natural Language Summaries

Kathleen McKeown **kathy@cs.columbia.edu**
Jacques Robin **robin@cs.columbia.edu**
Department of Computer Science
450 Computer Science Building
Columbia University
New York, N.Y. 10027

Karen Kukich **kukich@bellcore.com**
Bell Communication Research
Morristown, N.J.

Abstract: Summaries typically convey maximal information in minimal space. In this paper, we describe an approach to summary generation that opportunistically folds information from multiple facts into a single sentence using concise linguistic constructions. Unlike previous work in generation, how information gets added into a summary depends in part on constraints from how the text is worded so far. This approach allows the construction of concise summaries, containing complex sentences that pack in information. The resulting summary sentences are, in fact, longer than sentences generated by previous systems. We describe two applications we have developed using this approach, one of which produces summaries of basketball games (STREAK) while the other (PLANDOC) produces summaries of telephone network planning activity; both systems summarize input data as opposed to full text. The applications implement opportunistic summary generation using complementary approaches. STREAK uses revision, creating a draft of essential facts and then using revision rules constrained by the draft wording to add in additional facts as the text allows. PLANDOC uses discourse planning, looking ahead in its text plan to group together facts which can expressed concisely using conjunction and deleting repetitions. In this paper, we describe the problems for summary generation, the two domains, the linguistic constructions that the systems use to convey information concisely and the textual constraints that determine what information gets included.

1 Introduction

Summarization is a task that seems to require solutions in both natural language interpretation and generation. Summarizing an article, for example, involves interpretation of the article to identify the most important facts conveyed and generation to produce a paragraph that conveys those facts concisely. This need for simultaneous solutions in two subareas has made automated text summarization an elusive goal. In fact, full text summarization using natural language interpretation and generation in unrestricted domains is very much an open research problem, although there are some systems that use statistical or language analysis techniques to select a representative sentence from the input text to serve as a summary [Rau 87], [Cox 93], [Paice 90].

Contrary to this view, we have identified several summarization tasks involving generation only. We are working in two domains, summarization of sports events [Robin and McKeown 93], [Robin 93] and automated documentation of telephone planning engineer activities [Kukich et. al. 93]. In the first of these domains, input is a set of box scores for a basketball game and the task for the system is to summarize the highlights of the game, underscoring their significance in the light of previous games. Output is a short summary such as the lead sentences found in newspaper sports reports. In the second domain, the system must produce a report documenting how an engineer investigated what new technology is needed in a telephone route to meet demand through use of a sophisticated software planning system, LEIS-PLAN[1]. Input to the generation system is a trace of user interaction with PLAN and output is a 1-2 page report, including a paragraph summary of PLAN's solution, a summary of refinements that an engineer made to the system solution, and a closing paragraph summarizing the engineer's final proposed plan. Since input to summary generation in both domains is data as opposed to full text, this has allowed us to focus on the problem of generating summaries as opposed to extracting information from text.

Problems for summary generation are quite distinct from the generic task of language generation. Summaries must convey maximal information in a minimal amount of space. This requires selecting linguistic structures that can convey information concisely. This means the use of complex sentence structure, including multiple modifiers of a noun or verb, conjunction (e.g., "*and*"), and ellipsis (i.e., deletion of repetitions across conjoined phrases). It also means the selection of words that can convey multiple aspects of the information to be communicated. Our analysis of example summaries in both domains supports these summarization needs [Robin and McKeown 93], [Kukich et. al. 93]. In fact, we found that sentences in sports summaries range from 21-46 words in length, much longer and more complex than sentences generated by any previous generation system. Furthermore, our analysis shows that some information is opportunistically added into the summary, as the words and syntactic structure already used allow. These observations suggest the need for modifications to a typical language generation approach (e.g., [McKeown 85a], [Hovy 88], [McDonald 83], [Kukich 83]) in order to produce summaries.

---

[1]LEIS is a registered trademark of Bell Communications Research, Piscataway, NJ.

We characterize problems for summary generation as falling into two separate classes, *conceptual summarization*, or determining what information should be included in a summary, and *linguistic summarization*, the task of determining how to convey as much information as possible in a short amount of text. In this paper, our focus is on linguistic summarization, although, as we shall show, linguistic constraints often determine when information is included or omitted from a summary. Our systems take as input information which must be included, as well as information which potentially can be included in the summary. We characterize linguistic summarization as centering around the following issues:

- How to use syntactic and lexical devices to convey information concisely
- Given the choice of a particular word or syntactic structure, how does this constrain (or allow) the attachment of additional information
- How to fold multiple pieces of information into a single linguistic construction

Our approach to summary generation is revision based, opportunistically adding in new information that can be folded into the summary text generated so far. A draft is first generated from information that must be included. Additional, potentially interesting, information is added when specific lexical, syntactic and semantic constraints match the draft text. Thus, the summary content is determined by linguistic constraints, as well as the usual conceptual constraints, an approach unique to our work. This revision based approach allows the generation of the complex, information rich sentences found in summaries. In this paper, we detail different linguistic constraints and constructions that can be used to concisely add information into a text, showing the computational rules our systems used to implement these constructions. In particular, we show how the following four constructions allow information from multiple facts in the input to be folded into the same sentence, avoiding repetition of similar information across sentences and resulting in a concise text:

1. Single word conveys multiple information
2. Modifiers of nouns or verbs
3. Conjunction and ellipsis
4. Abridged references

We describe the implementation of opportunistic summary generation in both STREAK and PLANDOC, showing how they use these constraints and constructions. While STREAK and PLANDOC both opportunistically revise the summary, they use different, but complementary, techniques. STREAK uses a set of revisions rules, constructing a draft of the essential input facts; that is, those that absolutely must appear in the summary. Constraints from draft wording determine how additional information is added to the text often changing the original wording. PLANDOC uses a discourse planning approach, looking beyond input for the current sentence in its content plan, using syntactic constraints from the current sentence to select information appearing ahead in the plan that can be folded in. It thus reorganizes the content plan, changing the order in which information will appear in the text. By selecting information that was originally planned to appear further ahead in the summary, it can often be collapsed into the current sentence, avoiding repetition and producing a much less verbose text.

In the following sections, we first describe traditional approaches to language generation which

stand in contrast to the approach we claim is needed for summary generation. We then describe each application domain, showing both how summaries can be generated from structured data and the complex sentences that are required. From here, we turn to a discussion of linguistic summarization, describing our techniques for conveying maximal information in minimal space. We detail specific syntactic and lexical constructions from our domains that are used to convey information concisely, showing how wording constraints determine how information is opportunistically added to a summary. While conceptual summarization is not the primary focus of our work, we briefly describe how it can be carried out in the two application domains, pointing to places where decisions about whether information should be included are left open for linguistic summarization to resolve. We close by describing related work in the language generation field and our current directions in summarization.

2 Traditional Language Generation Approaches

Traditionally, language generation systems are divided into two modules, a content planner and a surface sentence generator. It is the task of the content planner to determine what information should be communicated in the text, while the surface sentence generator takes the conceptual representation of text content produced by the planner and realizes it in natural language. This latter task involves determining the overall sentence structure for each input proposition, selecting the words, and ordering the words in a sentence by building syntactic structure and enforcing syntactic constraints between constituents (e.g., ensuring that the subject and main verb agree in person and number). While some researchers have argued about the kind of interaction that should occur between these two modules (e.g., [Danlos 86], [Appelt 85], [Hovy 85], [Suthers 91], [Rubinoff 88]), there seems to be general agreement that some division between the two modules is needed as evidenced by the large number of systems that have taken this approach [McKeown 85a], [McCoy 86], [Paris 87], [Meteer, McDonald and al. 87], [Maybury 90], [Moore and Paris 89], [Kukich 83], [Elhadad 93a], [Reiter 91], [Dale 92], [Wahlster et al. 89], [McKeown and Swartout 87], [Mann 83], [Jacobs 85] [Feiner and McKeown 91].

A content planner uses conceptual information and reasoning strategies to select information from an underlying knowledge source and to determine its overall organization in the text. Many systems use some form of rhetorical knowledge in this task, whether embodied in schemas [McKeown 85a], [Paris 87], [McKeown et al. 90], [Rambow & Korelsky 92], rhetorical structure theory [Hovy 91], [Moore and Paris 89], [Wahlster et al. 89], or plans [Dale 92], [Suthers 91]. Other forms of knowledge used in this task include information about the user's goals and background [Paris 87], [Wahlster et al. 89], [Wolz et. al. 89], [Elhadad 93a], knowledge about information communicated in previous discourse [Zukerman and Pearl 86] [McKeown et al. 90] and domain knowledge [Rambow & Korelsky 92]. While some researchers have shown that determining wording of a text can also influence what information gets included [Appelt 85], [McKeown et.al. 93], [Danlos 86], [Kukich 83], usually this has been shown to happen at specific points in generation and is implemented by allowing backtracking across the two modules. Thus, the issue of how linguistic form influences what gets said has generally not been given much attention.

Typically, the content planner selects information in proposition size chunks which each get realized in English as a sentence. A list of these propositions, or facts, are passed to the surface sentence generator which generates a sentence for each fact. To do so, the sentence generator must decide on the overall sentence structure, choosing for example whether to generate a question or declarative sentence, selecting a main verb and mapping elements in the fact to verb arguments. It also selects words for each verb argument and finally, builds a syntactic tree for the sentence, and enforcing syntactic constraints, produces the words in linear order. There are a few exceptions to this approach. Some systems choose words after the syntactic tree has been constructed [Matthiessen 91], while others choose words and build the tree simultaneously [Kukich 83], [Hovy 85] [Jacobs 85]. While in most, a fact selected by the content planner will appear in the text as a separate sentence, in limited circumstances systems are able to combine a few, consecutive facts within a single sentence in specific situations. This is usually done only locally and in a domain specific way.

3 The Domains

In this section we describe the main characteristics of the two domains. In each, we describe input to the system showing how it is readily available from a non-text source, provide a system overview, and describe the state of system implementation. In both domains, we used an analysis of naturally occurring reports on which to base the generation system. These analyses revealed the textual techniques which can be used to convey information succinctly and they suggested a strategy for producing the complex sentences required.

Quick development of both systems was possible because of the existence of robust language generation tools for the surface sentence generator. Both systems use the FUF/SURGE package [Elhadad 93b], [Elhadad 93a] developed at Columbia for the lexicalizer and syntactic generation modules. FUF is a programming language based on functional unification [Kay 79] and SURGE is a large systemic grammar of English that was developed for FUF. The FUF/SURGE package handles the generation of individual sentences once the overall sentence structure has been determined and open class words chosen. It was used for this task in STREAK and PLANDOC as well as many other generation systems developed at Columbia University and elsewhere as well. FUF is used for the task of lexicalization, providing a language and interpreter in which to write a lexical chooser. However, the actual lexical chooser, encoding constraints on word choice and selecting overall sentence form, had to be developed separately for both domains.

3.1 Generation of Summaries of Quantitative Data

Summarization of quantitative data raises several challenges for language generation systems. First, sentences in such reports are very complex (e.g., in newswire basketball game summaries the lead sentence ranges from 21 to 46 words in length). Second, while the essential facts consistently appear in *fixed* locations across reports (e.g., game results are always conveyed in the lead sentence), other potentially interesting facts *float*, appearing anywhere in a report and at different depths within a given sentence. Floating facts appear to be opportunistically placed where the form of the surrounding text allows. They vary significantly both in the form that is used to convey the same type of fact and in where they are placed in a report; the

same fact type can appear anywhere from beginning to end of report. Examples 1-3 below show how the form can vary; sentences 2 and 3 result from adding the same background information to sentence 1: that is, information that the player is only a reserve. Each example uses different syntactic categories at distinct structural levels.

1. Draft sentence: *''Hartford, CT -- Jay Humphries scored 24 points Friday night as the Utah Jazz defeated the Boston Celtics 98 - 94.''*
2. Add a noun modifier: *''Hartford, CT --* ***Reserve*** *Jay Humphries scored 24 points Friday night as the Utah Jazz defeated the Boston Celtics 98 - 94.''*
3. Add a prepositional phrase to modify clause: *''Hartford, CT -- Jay Humphries scored 24 points* ***off the bench*** *Friday night as the Utah Jazz defeated the Boston Celtics 98 - 94.''*
4. Use new main clause, embedding old clause: *''Hartford, CT -- Jay Humphries* ***came off the bench to*** *score 24 points Friday night as the Utah Jazz defeated the Boston Celtics 98 - 94.''*

Although optional in any given sentence, floating facts cannot be ignored. In our domain, they account for over 40% of lead sentence content, with information conveying the historical significance of facts *only* conveyed as floating structures. Earlier report generators [Kukich 83], [Bourbeau et al. 90], however, could not include historical information precisely because of the fact that it floats.

To determine how floating facts can be incorporated in a draft, we analyzed a corpus of basketball reports[2], pairing sentences that differ semantically by a single floating fact and identifying the minimal syntactic transformation between them. Figure 1 shows an example of how this analysis was done. We started from the simplest sentences in the corpus, those containing only fixed information (i.e., location, teams, game result, score, date, and most significant player statistic). Sentence S0 is an example of such sentence. Then we looked in the corpus for sentences that contain one more fact than S0 but otherwise follow the same syntactic structure. This is the case for S1, which is just like S0, except that it also explains what makes the reported player statistic significant. From this pair of sentences, we learn that such background facts can be slipped into a report by way of a noun phrase modifier (boldfaced in S1). We then proceeded to search for sentences with one more fact than S1. S2 is an example; it differs from S1 only in that it relates the game result to the winner's previous results. From this pair of sentences, we learn that this type of information can be added opportunistically by attaching a "*for*" prepositional phrase (boldfaced) to the second clause of the conjunction. We repeated this pairwise analysis starting with each distinct basic sentence structure and ending with the most complex sentence found.

The result is a set of *revision tools*, specifying precise semantic and syntactic constraints on (1) where a particular type of floating fact can be added in a draft and (2) what linguistic constructs can be used for the addition. Note that the actual words and syntactic structure of the

[2]Taken from the UPI newswire.

Sentence S0 of the simplest type:
Chicago -- Michael Jordan scored 30 points Saturday night and the Chicago Bulls defeated the Cleveland Cavaliers 108 92.

More complex sentence S1 with one more fact than S0 in the first clause:
*Minneapolis -- Pooh Richardson scored **a career high** 35 points Saturday night and the Minnesota Timberwolves beat the Golden State Warriors 121 113.*

Even more complex sentence S2 with one more fact than S1 in the second clause:
*Houston (UPI) -- Buck Johnson scored a season high 26 points Thursday night and the Houston Rockets routed the Orlando Magic 119 95 **for their sixth straight win**.*

Figure 1: Paired Corpora Sentences Showing Revisions

draft, in addition to the semantic information it conveys, can determine when new information gets added in. These revision tools are directly used as rules in the underlying implementation. In Section 4, we show how these rules are implemented to generate specific types of lexical and syntactic constructions to convey information concisely.

This approach to generation allows STREAK to incrementally produce the complex sentences required for summarization by opportunistically adding optional historical information into simple sentences generated as a first draft. Example output sentences from STREAK illustrating how this works is shown below in Figure 2. Here, five different revision tools are applied in sequence, each adding in new information.

1. **Initial draft (basic sentence pattern)**: "Hartford, CT -- Karl Malone scored 39 points Friday night as the Utah Jazz defeated the Boston Celtics 118 94."
2. **adjunctization**: "Hartford, CT -- Karl Malone **tied a season high with** *39 points* Friday night as the Utah Jazz defeated the Boston Celtics 118 94."
3. **conjoin**: "Hartford, CT -- Karl Malone tied a season high with 39 points **and Jay Humphries added 24** Friday night as the Utah Jazz defeated the Boston Celtics 118 94."
4. **absorb**: "Hartford, CT -- Karl Malone tied a season high with 39 points and Jay Humphries **came off the bench** *to add 24* Friday night as the Utah Jazz defeated the Boston Celtics 118 94."
5. **nominalization**: "Hartford, CT -- Karl Malone tied a season high with 39 points and Jay Humphries came off the bench to add 24 Friday night as the Utah Jazz **handed** the Boston Celtics **their sixth straight home** *defeat* 118 94."
6. **adjoin**: "Hartford, CT -- Karl Malone tied a season high with 39 points and Jay Humphries came off the bench to add 24 Friday night as the Utah Jazz handed the Boston Celtics their **franchise record** sixth straight home defeat 118 94."

Figure 2: Incremental generation of a complex sentence using various revision tools

A shown in Figure 3, there are five main modules in STREAK: the fact generator, the sentence planner, the lexicalizer, the sentence reviser and SURGE. The fact generator, which is still at the design stage[3], will take as input the game boxscore augmented by a database of historical statistics and will produce as output a list of essential facts that must be included and a list of floating facts to convey opportunistically. Each fact is a conceptual network encoded as a feature structure (an example of such a network is given in Figure 7). The fact generator thus has the task of conceptual summarization.

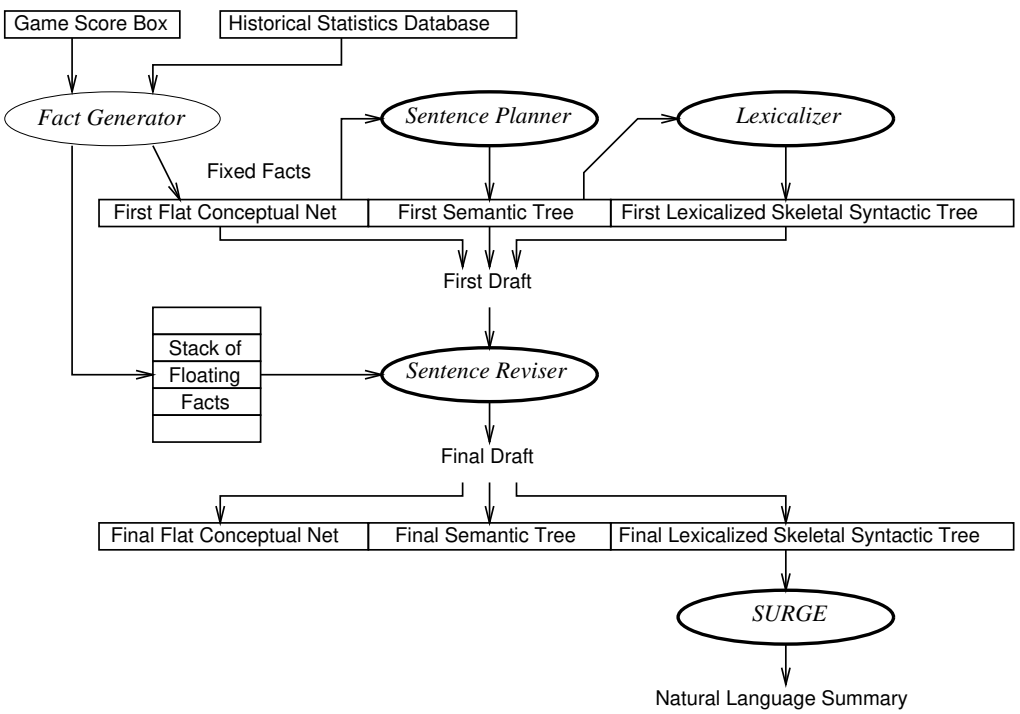

Figure 3: An Overview of STREAK's Architecture

Linguistic summarization is carried out in STREAK by the next three modules: sentence planner, lexicalizer and sentence reviser. The sentence planner takes as input the flat conceptual network containing all essential facts and maps it onto a semantic tree which represents overall sentence structure. The lexicalizer then maps the semantic tree onto a lexicalized skeletal syntactic tree; in the process open-class words[4] and overall sentence structure are determined. Both the sentence planner and the lexicalizer are implemented through functional unification, using FUF.

The sentence reviser takes as input both the draft (note that the draft encodes both meaning,

[3]The current implementation takes hand coded symbolic input like the input shown in Figure 15.

[4]Open-class words are words from a syntactic category that contain an unlimited number of different words (e.g., nouns). In contrast, closed-class words are from a category that contains a small set of different possible words (e.g., prepositions).

in the semantic tree, as well as any words or syntactic structures that will be used, in the thematic feature structure) and the list of potential facts. It consists of a revision rule interpreter with a set of revision rules drawn from the corpus analysis. The interpreter is also written in FUF. The output of the sentence reviser is a final draft incorporating both the essential facts of the first draft and as many potential facts that could be added following the domain, linguistic and space constraints encoded in the revision rules. To keep the rules general, they indicate only *where* and *how* to attach the new fact onto the draft. The precise wording and syntactic form of the attached phrase are chosen by re-calling the sentence planner and the lexicalizer in context from the reviser. After revision, the actual natural language text is produced by extracting the lexicalized structure from the final draft and unifying it with SURGE. All four linguistic summarization modules are fully implemented. For more details on the implementation of STREAK see [Robin 94].

3.2 Automated Documentation of Planning Engineer Activity

Jointly with Bellcore, we are developing a system, PLANDOC, that will document the activity of planning engineers as they study telephone routes[5]. Planning engineers currently use a software tool, the Bellcore PLAN system, that helps them derive a 20-year plan based on growth forecasts and economic constraints. Documentation of the activity is helpful for informing managers who are responsible for authorizing expenditures as well as for auditors and external regulators. PLANDOC takes as input a trace of the engineer's interaction with PLAN and produces a 1-2 page summary.

The telephone network planning engineer's job is to derive a capacity expansion (relief) plan specifying when, where, and how much new copper, fiber, multiplexing and other equipment to install in the local network to avoid facilities exhaustion. PLAN first automatically derives a plan for a route using economic constraints only. The engineer then uses PLAN to refine this plan using information about any company strategic policies (e.g., conversion to fiber), intangibles in the route (e.g., crowded manholes preventing further installations), and any long or short term economic constraints. During this interaction the engineer can experiment with hypothetical scenarios, exploring the results of different possible changes. Management will check that all possibilities have been explored before approving a particular plan for change. This is but one way in which the PLANDOC system is important; in providing a summary of the engineer's activity, it can show managers that other possible changes were explored and rejected as not beneficial.

We based development of PLANDOC on a user-needs analysis (see [Kukich et. al. 93]), including a set of model narratives written by an experienced planning engineer. This corpus provided a general model for the documentation, beginning with a summary of the plan produced by PLAN alone, followed by a summary of the engineer's refinements to this plan, and concluding with a summary of the plan recommended by the engineer (which usually includes elements of the program generated plan and human refinements). Currently, PLANDOC can produce the refine-

[5] PLANDOC is being developed collaboratively with Karen Kukich and Neal Morgan of Bellcore and James Shaw, Jacques Robin, and Jong Lim of Columbia University.

ment summary (one paragraph from a refinement summary is shown below in Figure 4) and we are beginning work on opening and closing summaries of the report.

PLAN Tracking Information:
Sample fact generator Input

1. RUNID fiberall FIBER 6/19/93 act yes
2. FA 1301 2 1995
3. FA 1201 2 1995
4. FA 1401 2 1995
5. FA 1501 2 1995
6. ANF co 1103 2 1995 48
7. ANF 1201 1301 2 1995 24
8. ANF 1401 1501 2 1995 24
END. 856.0 670.2

PLANDOC Output:

Run-ID: FIBERALL

This saved fiber refinement included all DLC changes in RUNID ALLDLC. RUNID FIBERALL demanded that PLAN activate fiber for CSAs 1201, 1301, 1401 and 1501 in 1995 Q2. It requested the placement of a 48-fiber cable from the CO to section 1103 and the placement of 24-fiber cables from section 1201 to section 1301 and from section 1401 to section 1501 in the second quarter of 1995. For this refinement, the resulting 20 year route PWE was $856.00K, a $64.11K savings over the BASE plan and the resulting 5 year IFC was $670.20K, a $60.55K savings over the BASE plan.

Explanation of Input:

Facts 2-5 are all fiber activations (FA). The second column for these facts indicates the site (e.g., CSA 1301 for line 2), the third indicates the quarter (quarter 2 for all of them) and the fourth column, the year. These three lines are all generated as the second sentence of the summary. Lines 6-8 are all cable placements (indicated by the code ANF). Column 2 indicates the starting section where the cable is placed (realized by "from section" in the summary), column 3 the ending section, column 4 the quarter, column 5 the year, and column 6 indicates the number of fibers in the cable. Facts 6-8 are realized as the third sentence of the summary.

Figure 4: PLANDOC input and output

PLANDOC's architecture, which is shown in Figure 5, draws on our previous text generation and report generation work [McKeown 85a], [Kukich 83] and includes a fact generator, a discourse

Figure 5: An Overview of PLANDOC's Architecture

planner, a lexicalizer, and a sentence generator. Input to PLANDOC is a trace of the engineer's activity with PLAN. It indicates, in table format, the different types of changes to the route the engineer experimented with and how this affects overall cost. Input for a single refinement scenario is shown in Figure 4 along with the summary paragraph generated. Since the PLAN system is implemented in C and most of PLANDOC is implemented in Lisp, a fact generator module[6] serves as an interface between PLAN and PLANDOC. The fact generator produces a fact in the form of a semantic feature structure for each refinement action in a PLAN tracking report. In Figure 6, a fact corresponding to a single line of PLANDOC input is shown along with the part of the sentence generated for this fact.

INPUT:

RUNID fiberall FIBER 6/19/93 act yes
FA 1301 2 1995

Fact Generated by PLANDOC:

```
((cat fact)
 (admin ((PLANDoc-fact-name RFA)        ; administrative information
         (track-tag FA)                 ; providing unique identifiers
         (seq-num 19)
         (runid r-fiberall)
         (prev-runid r-alldlc)
         (status act)
         (saved yes)))
 (class refinement)                     ; indicates that this is an engineer's refinement
 (ref-type FIBER)                       ; equipment is of type fiber
 (action activation)                    ; fiber will be activated
 (csa-site 1301)                        ; site of activation
 (date ((year 1995) (quarter 2))))      ; date of activation
```

Sentence Generated by PLANDOC:

RUNID FIBERALL demanded that PLAN activate fiber for CSA 1301
in the second quarter of 1995.

Figure 6: PLANDOC input and corresponding fact

[6]written in C by N. Morgan

This set of facts is first passed to an *ontologizer*[7], that enriches each fact with semantic knowledge from PLAN's domain. The enriched facts are then passed to a discourse planner[8], whose job is to determine which information in the facts should appear in the various paragraphs and to organize the overall narrative. For our initial prototype, we have focused only on the generation of paragraphs that summarize refinement actions and not on the generation of the paragraphs that summarize the base plan and the final plan. Thus, the discourse planner's task is to combine individual facts to produce the input for complex sentences. The output of the discourse planner is a 'condensed' set of complex facts, each still a semantic feature structure.

As in STREAK, the lexicalizer module[9] maps the semantic feature structure representing the fact onto a lexicalized thematic structure. It thus chooses the overall sentence form, open-class words and syntactic features associated with the words and does so while maintaining constraints imposed by the discourse planner. Finally, this lexicalized structure is unified with SURGE to produce the full syntactic tree, which is linearized to yield an English sentence. PLANDOC is fully implemented for refinement summaries and will be tested with actual users over the next several months.

Just as for STREAK, one major issue for PLANDOC in generating the refinement summaries is to combine individual facts into a single sentence. If separate sentences were generated for each input fact, the resulting summary would be quite verbose and repetitive. PLANDOC uses a top-down, recursive method, which is complementary to STREAK's use of revision, for this problem. It uses discourse knowledge to look ahead for facts that have equivalent features, combining them into a single complex fact. The resulting complex fact is then passed to the lexical chooser. While making word choices, the lexical chooser also selects a complex sentence structure using conjunction. It determines which pieces of the sentence can be deleted since they occur in another part of the conjunction, thus producing a concise sentence.

4 Linguistic Summarization Devices

Summaries are necessarily much shorter in length than the full text they summarize. This can be achieved in part through conceptual summarization by leaving out inessential information. But more information can be conveyed if the summary language is concise, packing in as many facts as possible into a short amount of space. In this section, we show the language constructions our systems use to convey information concisely for this task of linguistic summarization.

A summary generator is given as input a set of facts to convey. In our systems, these facts are divided into two classes: essential (must be included) and potential (include if possible). For example, in STREAK game results are essential while the historical significance of the results can potentially be included. In PLANDOC, the refinements eventually used in the proposed plan

[7] written in FUF by J. Robin and J. Shaw

[8] written in Lisp by J. Robin and J. Shaw

[9] written in FUF by J. Shaw with input from J. Robin, D. Radev, J. Lim, M. Elhadad and D. Horowitz

are essential, while those that were explored and rejected are potential. It is the job of the language generator to determine both where in the summary to place each fact and how to realize it in English. A simplistic approach, for example, might realize each fact as a sentence. Such a scheme would tend to produce a verbose text, as some information may be repeated over several facts.

In this section, we show how a single word can convey multiple facts, how multiple modifiers can be used to pack information into a single sentence, how conjunction with ellipsis can be used to convey multiple facts concisely, and finally, how referring expressions can be abridged to save space for other facts.

4.1 Single Word, Multiple Facts

In some cases, two facts can be merged into one sentence by selecting a word that can simultaneously convey two or more pieces of information (*content units*) contained in the two facts. This is usually only the case if some information is repeated in both facts. Usually, the word conveying more than one content unit is a verb and the information that is repeated from one fact to the next appears as arguments to the verb. We term this situation *content conflation*.

For example, consider the corpus sentence:
"Portland outlasted Utah 101 97."

Both the fact that Portland won and the fact it was a tight game in which Portland took the decisive advantage only at the very end are conflated in the single verb "*to outlast*". Without using such a semantically rich verb, a much longer paraphrase, would have been needed:
"Portland defeated Utah 101 97, in a tight game where the lead kept changing hands until late in the fourth quarter."

This type of conflation is the simplest case since it involves a single word choice which does not affect the rest of the sentence. For more detail on simple content conflation, see [Elhadad and Robin 92]. In more complex cases, content conflation is possible only if it is accompanied by some re-organization of the rest the sentence's content.

As an example of conflation requiring reorganization, consider the situation, again in the case of sport summaries, where the system is given two related facts, shown as two networks in Figure 7. If these facts were each realized as separate sentences, the top one would produce the sentence: "*Karl Malone scored 39 points.*"

while the bottom one would produce
"*Karl Malone's 39 point performance is equal to his season high.*".

Note that several pieces of information are repeated from one fact to the next; the player is the same and the same score is mentioned. The new information included in the second fact is the "*season high*" and the fact that his score is "*equal*" to the season high. These two facts can be conveyed simultaneously by choosing the verb "*tie*" to produce the single sentence:
"*Karl Malone tied his season high with 39 points.*"

In this sentence, the verb "*tie*" simultaneously conveys that Malone scored these 39 points and that they equal the season high.

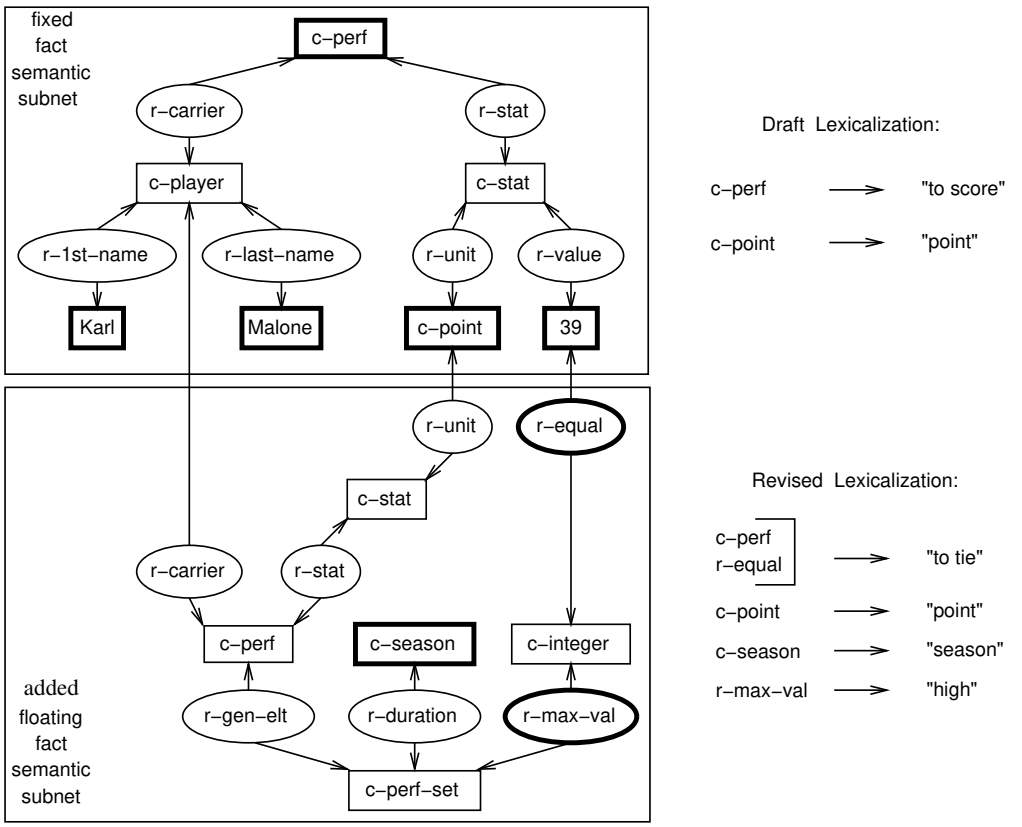

Figure 7: Input and lexicalization of sentence revised by adjunctization

STREAK combines these two facts using its draft and revision approach. On first pass, it generates the draft "*Karl Malone scored 39 points.*" saving a representation of both the conceptual fact and its various intermediate linguistic representations. Note that this fact is fixed and must appear in the summary as it conveys the most significant player statistic.

On second pass, it uses a revision rule that adds the two new facts at the same time to the draft by selecting a new verb and placing the object of the tie (i.e., the season high) as the object of the new verb, yielding:
"*Karl Malone tied his season high.*".

This displaces the object of the old verb (the score); in order to include it, it is made into an adjunct, in this case a prepositional phrase that conveys the quantity of the score yielding the full sentence: *"Karl Malone tied his season high* ***with 39 points"***. The mapping from the concepts and relations of the networks[10] for both the draft and revised sentences is indicated at

[10]respectively prefixed by "c-" and "r-".

the right of Figure 7. *''tie''* conveys both the "c-perf" concept which was conveyed by *''to score''* in the draft and the added "r-equal" relation.

The revision rule is called *adjunctization of object into instrument*, precisely because it selects a new verb with new object and makes the old object into an instrument adjunct. More generally, we call *adjunctization* any revision where a verb argument (**Bc2**) becomes an adjunct to accommodate new content (**Vf** and **Ac**) as shown in Figure 8.

Figure 8: General Adjunctization Rule

4.2 Modification

Another way to combine two facts into a single sentence is by realizing one fact as a modification of some element of the first. In this case, the first fact forms a simple sentence, with its action as the verb of the sentence. If the two facts both refer to the same element, which is an argument of the verb, then the second fact can be expressed more compactly by making it a modifier of the verb argument. Consider, for example, the following situation from the sports domain: a player scores many points and it is all the more remarkable because he played the game only as a reserve. The scoring action and his reserve status correspond to two distinct facts in the generator's input. A system with no other linguistic summarization abilities than straightforward pronominalization would have to convey these two facts by two sentences, e.g.: *"Jay Humphries scored 24 points. He came in as a reserve."*

Instead, the content of the second sentence can be compactly conveyed by adding a modifier up front in the first: "***Reserve*** *Jay Humphries scored 24 point."*

This revision is achieved by using the rule called *nominal-rank adjoin of pre-modifier* since it adds a modifier at the beginning of a noun phrase (i.e., nominal rank). This rule has syntactic constraints in that the information shared across facts must appear as a noun phrase in the first sentence and that the second fact must be expressed as a pre-modifier. It has no semantic constraints since it can apply to any facts that meet these conditions. In general, any rule involving the simple addition of an optional dependent constituent (**Ac**) is called an *adjoin* as shown in Figure 9.

Like conflation, there are both simple cases of modification (the *adjoin* revision just presented), and more complex cases where the modification can occur only after some sentence content

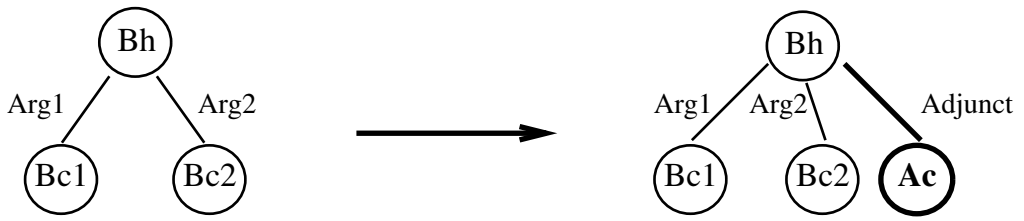

Figure 9: General Adjoin Rule

reorganization.

Nominalization is an example of a revision rule involving a complex modification. It replaces a meaning carrying verb (**Vf**) in the draft sentence, with a collocation consisting of a support verb (**Vs**) plus a noun (**Nf**), where the noun is a nominalization of the original verb. This type of revision allows additional modifiers to be added onto the nominalization in subsequent revisions. For example, in revising sentence 4 into sentence 5 in Figure 2 above, the verb structure "X *defeat* Y" is replaced by the collocation "X *hand* Y *a defeat*." Once nominalized, "*defeat*" can then be pre-modified by the phrase "*their sixth straight home*" providing historical background. This rule embodies both a syntactic and lexical constraint on adding in new information modifying an action; only if there is a nominal synonym to the verb describing the action can this revision rule be selected. The general structural schema of this revision, is shown in Figure 10.

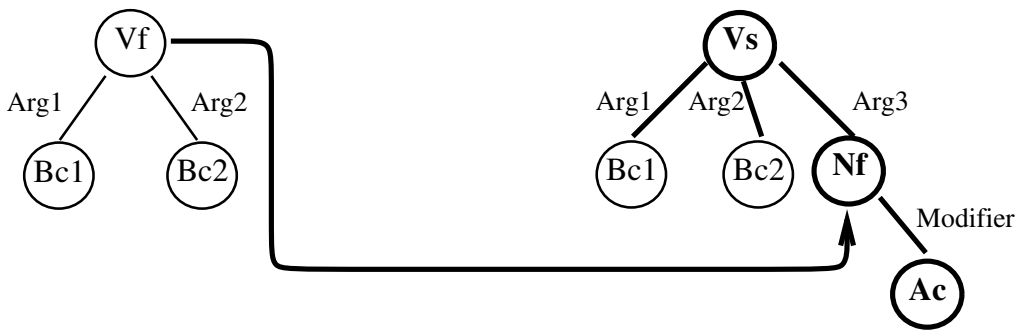

Figure 10: General Nominalization Rule

The FUF conjunctions encoding this revision are shown in Figures 11 and 12. The first precondition in the nominalization rule shown in these figures constrains its application to clauses of the form "WINNER verb LOSER" where:

1. The verb is a full-verb realizing the result itself (this is indicated by the chain of REALZ features in Figure 12, starting from the process feature of the syntactic tree - the one where the draft verb is specified - down to the BEAT role relation of the

```
;;; Revision rule for nominalization
(def-conj nominalization
  (preconditions
   (;; Applies to draft clauses headed by a full verb conveying the game result
    (old-draft ((:& basic-game-result-clause)))

    ;; Can be used to add a losing streak from the floating fact stack
    (floats-top ((:& losing-streak-extension)))))

  (actions
   ((delete {new-draft})  ;; delete the matching draft clause

    ;; Replace it by the frame of a transfer of possession clause headed by
    ;; a support verb e.g., ''WINNER hand LOSER     a defeat''
    ;;                                 possessor possessed (nominalized)
    ;;                        agent    affected
    (insert ((surfsemcat rhetor)
             (struct hypotax)
             (root ((surfsemcat rhetor)
                    (struct event)
                    (root transf-poss))))
            {new-draft sem-tree})

    ;; Locally call the lexicalizer on this new clause frame to choose a
    ;; support verb to head it.
    (lexicalize {new-draft sem-tree} {new-draft synt-tree})

    ;; Copy the fixed predicate modifiers and participants of the clause,
    ;; from the original clause.
    (copy {old-draft sem-tree rels score} {new-draft sem-tree rels score})
    ...
    < more copy actions not shown here for agent and affected roles >
    ...
    ;; Locally call the phrase planner and lexicalizer to realize the possessed
    ;; role of the new clause containing the nominalized game result
    (plan {floats-top} {new-draft sem-tree args possessed})
    (lexicalize {new-draft sem-tree args possessed}
                {new-draft synt-tree participants possessed}))))
```

Figure 11: FUG implementation of nominalization revision rule

corresponding conceptual network). This is a lexical constraint matched by the class of verbs including "to beat", "to defeat", "to triumph against" etc.

2. The winner relation is realized by an AGENT participant in the syntactic tree (this is a semantic constraint)
3. The loser relation is realized by an AFFECTED participant in the syntactic tree (this is a semantic constraint)

This combination of semantic, syntactic and lexical constraints are shown in detail in the feature conjunction BASIC-GAME-RESULT-CLAUSE. The second precondition is that a losing streak is present in the stack of floating fact. In practice, nominalization can apply to other pairs <draft form, new-content-type> though only one is shown here.

The first nominalization action is to delete the clause to revise from the new-draft (which is initially a copy of the old draft). This clause is replaced by a transfer of possession clause frame (e.g., as "to hand," "to give," "to bring"). The lexicalizer is then called from the reviser to realize

```
;; Precondition pattern for basic a clauses realizing the game result and
headed by a
;; full-verb e.g., ''WINNER defeat LOSER SCORE''
(def-conj basic-game-result-clause
  (synt-tree
   ((cat clause)
    ; 1. VERB IS A FULL VERB REALIZED BY 1A. BELOW
    (process ((realz {sem-tree root concept})))
    (participants ((agent ((realz {sem-tree root args agent})))
                   (affected ((realz {sem-tree root args affected})))))
    (predicate-modifiers ((score ((realz {sem-tree rels score})))))))
  (sem-tree
   ((surfsemcat rhetor)
    (struct hypotax)
    (root
     ((surfsemcat encyclo)
      ; 1A. THIS IS JUST A POINTER TO 4. BELOW
      (concept {concept-net results rels result role})
      ; 2. AGENT IS WINNER
      (args ((agent
              ((surfsemcat encyclo)
               (realz {concept-net results rels result args winner})))
      ; 3. AFFECTED IS LOSER
             (affected
              ((surfsemcat encyclo)
               (realz {concept-net results rels result args loser})))))))
    (rels ((score ((surfsemcat encyclo)
                   (realz {concept-net results entities score})))))))
  (concept-net
   ((deepsemcat entity)
    (concept game)
    (results
     ((entities ((host ((deepsemcat entity) (concept team)))
                 (visitor ((deepsemcat entity) (concept team)))
                 (score ((deepsemcat entity) (concept score)))))
     ; 4. VERB MUST BE CHOSEN FROM SYNONYMS OF ''BEAT''
     (rels ((result ((deepsemcat relation)
                             (role beat)
                             (args ((winner {concept-net results ents host})
                             (loser {concept-net results ents host})))))))))))
```

Figure 12: FUG implementation of nominalization revision rule

the head verb of this frame. The participants and predicate modifiers that are unchanged by the revision are then copied in turn from the old draft to the new draft (in this case, the AGENT and AFFECTED). Then the phrase planner is called on the floating fact to incorporate it into the draft. The resulting plan fills the POSSESSED role in the new clause frame. The lexicalizer is then called on this phrase plan and produces a new POSSESSED role whose head is a nominal synonym of the old draft head verb (e.g., "a defeat," "a loss"). At the surface form level the successive effects of this series of action on a particular draft could be for example:

- precondition matched: "... while Chicago defeated New York 99-84."
- delete action: "... while"
- insert and perform the first lexicalize action: "... while ... handed"
- copy actions: "... while Chicago handed New York ... 99-84"
- plan and perform the second lexicalize action: "... while Chicago handed New York

their third straight defeat 99-84"

4.3 Conjunction

A common way to combine separate facts into a single sentence is to use conjunction (i.e., a list of constituents separated by the word "*and*" or "*or*"). In order to use conjunction, there must be a common piece of information occurring across all facts. Both PLANDOC and STREAK use conjunction. PLANDOC handles conjunction through discourse planning, while STREAK uses revision.

For PLANDOC, conjunction with ellipsis is the primary means of creating a concise summary. Its discourse planner includes a fact combining stage, where it collects facts with common information (in PLANDOC, information in facts is represented as feature structures, so common information would be features shared across facts). It primarily looks for facts with the same action and for these facts, notes which features in the separate facts are equivalent. It creates a new fact, with the features separated into those that are *common* to all facts and those that are *distinct* in each. Creating a sentence that contains conjunction is complex primarily because the number of different feature combinations in facts is large and each different combination of common and distinct groupings results in a different kind of conjunction. Depending on which information is common across facts, different amounts of the sentence can be *elided* (i.e., removed) in adjoining clauses.

Figure 13 shows four different examples of conjunction. Each example includes a pair, where the first part of the pair shows the facts combined using conjunction, but with no ellipsis. The second part of the pair gives the same example, but with the common information across facts left implicit on subsequent mention. These examples show that the amount of information that is elided varies tremendously. In the first example, all information is common to the facts except for the CSA site. A conjunction of nominals, each denoting one site can therefore be used following *"for"*, resulting in the eight facts being very concisely expressed by only a single clause instead of eight.

Examples 2-4 show how the different combinations of common and distinct information can get quite complex and yet still result in a conjoined sentence. In Example 2, all CSA's were activated for the same date except one, but they could be successfully combined by using two layers of conjunction. The third example shows a situation where two features are in distinct (in Example 1 only the CSA site was distinct). Finally, in the fourth case, only the refinement and the action is the same, but all other features are different. This results in almost full sentences appearing in each clause of the conjunction.

We have implemented a general algorithm which groups together input features having the same value, so that as many separate messages as possible can be collapsed into a single message. In the first step, messages with the same action are identified and grouped together for possible coordination, called *common-action group*. Since the action is realized in a verb form or nominalization in the sentence, this means that the action will be common across all

Example 1:

- RUNID R-DLC1 activated ALL-DLC for CSA 8105 in the first quarter of 1994 and
  RUNID R-DLC1 activated ALL-DLC for CSA 8107 in the first quarter of 1994 and
  RUNID R-DLC1 activated ALL-DLC for CSA 8113 in the first quarter of 1994 and
  RUNID R-DLC1 activated ALL-DLC for CSA 8118 in the first quarter of 1994 and
  RUNID R-DLC1 activated ALL-DLC for CSA 8121 in the first quarter of 1994 and
  RUNID R-DLC1 activated ALL-DLC for CSA 8126 in the first quarter of 1994 and
  RUNID R-DLC1 activated ALL-DLC for CSA 8128 in the first quarter of 1994 and
  RUNID R-DLC1 activated ALL-DLC for CSA 8132 in the first quarter of 1994.
- RUNID R-DLC1 activated ALL-DLC for CSAs 8105, 8107, 8113, 8118, 8121, 8126, 8128 and 8132 in the first quarter of 1994.

Example 2:

- This refinement activated fiber for CSA 4109 in the first quarter of 1995 and
  this refinement activated fiber for CSA 4114 in the first quarter of 1995 and
  this refinement activated fiber for CSA 4211 in the first quarter of 1995 and
  this refinement activated fiber for CSA 4403 in the first quarter of 1995 and
  this refinement activated fiber for CSA 4502 in the first quarter of 1995 and
  this refinement activated fiber for CSA 4104 in the second quarter of 1995.
- This refinement activated fiber for CSAs 4109, 4114, 4211, 4403 and 4502 in the first quarter of 1995 and for CSA 4104 in the second quarter of 1995.

Example 3:

- This refinement used a cutover strategy of GROWTH for CSA 1401 and
  this refinement used a cutover strategy of ALL for CSA 1318 and
  this refinement used a cutover strategy of MIN for CSA 1001.
- This refinement used a cutover strategy of GROWTH for CSA 1401, of ALL for CSA 1318 and of MIN for CSA 1001.

Example 4:

- It requested the placement of a 48-fiber cable from the CO to section 1103 in the second quarter of 1995 and it requested the placement of 24-fiber cables from section 1201 to section 1301 in the second quarter of 1995 and it requested the placement of 24-fiber cables from section 1401 to section 1501 in the second quarter of 1995.
- It requested the placement of a 48-fiber cable from the CO to section 1103 and the placement of 24-fiber cables from section 1201 to section 1301 and from section 1401 to section 1501 in the second quarter of 1995.

Figure 13: Conjunction and ellipsis in PLANDOC

clauses. Depending on which of the remaining features (in the group of messages with common action) are common or distinct, one or more of the roles (e.g., subject, object, complement) will be conjoined. Next PLANDOC determines whether conjunction can be contained within one constituents (preferable because there is less chance for ambiguity [Quirk et al. 85] or whether conjunction must be treated as between full clauses requiring deletion in the final sentence of common constituents. To do this, PLANDOC identifies any subsets of the messages in the current *common-action group* which differ by only one feature. These subsets are immediately collapsed into a single message where the distinct features are made into a list, and the resulting sentence will contain conjunction within the constituent realizing this feature. Example 1, Figure 13, illustrates this case. Here, conjunction is contained within a noun phrase that realizes the list of CSA sites.

When more than one message remains in the *common-action group* after this collapsing, then more than one feature is distinct and conjunction must be done across the clause as a whole. To determine how to form groups of the remaining messages, PLANDOC determines which feature is shared by the greatest number of remaining messages. The remaining messages are then grouped into subsets where each subset has the same value for the feature just identified. This allows the most concise sentence to be generated because the greatest number of common constituents can be deleted in the sentence. Finally, the messages are sorted by the date feature so that the conjoined clauses appear in temporal order [Quirk et al. 85]. This algorithm can be summarized in the following steps:

1. Group messages by common action
2. For each common-action group:
    a. Collapse any messages which differ by one feature into a single message
    b. If more than one message remains:
        i. Determine which feature is shared across most messages
        ii. Sort messages by most common shared feature to form subgroups
        iii. Within each subgroup, sort again by most common shared feature of subgroup, repeating until all nested subgroups have been sorted.
        iv. Sort by date feature so that clauses are conjoined in temporal order

To see how this works, consider Example 4. Here, there are three distinct features, equipment type (with values of 48-fiber cable, 24-fiber cable, and 24-fiber cable), from-section (with values of CO, section 1201, and section 1401), and two-section (with values of 1103, 1301, and 1501). Of these, equipment type has the largest number of shared values across messages (2 shared values, 24-fiber cable) while the other features have no shared values. Thus, two conjoined clauses are formed, where the second has an embedded conjunction. Next each common feature is deleted. Action is common to all messages so only appears in the first clause ("it requested"), while "the placement of 24-fiber cables" is deleted in the last clause. The clauses are ordered so that the simplest (i.e., with fewest messages and/or number of distinct features) is placed first. This avoids sentences that are difficult to understand because very complex constituents must be processed first in the conjunction. In such difficult sentences, the subject

of the first clause is also the subject of the second clause, but is separated from the second clause by many words due to the complex constituent. This creates difficultly in understanding because the reader must remember the subject while processing all the intervening material before finally getting to the verb. For technical writing and proscriptive, this is described as a situation to avoid [Gopen and Swan 90], [Quirk et al. 85].[11]

Ellipsis in conjunction is difficult to handle because without careful control, it can give rise to ambiguity. Ambiguity can be triggered by *surface form factors* like ordering of movable constituents and comparative length of adjacent constituents [Quirk et al. 85], [Gopen and Swan 90]. Overuse of ellipsis can also result in sentences difficult to comprehend. Such overuse is illustrated by the following example:

"*This refinement extended fiber from fiber hub 8107 to CSAs 8128, 8126, 8121 and 8113, from fiber hub 8107 and the CO to CSAs 8105 and 8120 and from the C0 to CSAs 8118 and 8107}* ***in 1994 Q1*** *and from the CO to CSAs 8120 in 1994 Q3,* ***with both the active fibers and the protection fibers placed on the primary path.***"

This sentence conveys a complex feature sharing pattern that occurred in the PLANDOC domain. It is not only barely intelligible, but also ambiguous: it is unclear what preceding portion of the sentence each boldfaced prepositional phrase modifies. For example, both the active and protection fibers could have been placed on the primary path only for the last extension (i.e., to CSA 8120) or they could have been placed on the primary path for all extensions. PLANDOC prevents sentences such as this from being generated. Currently this is done by blocking conjunction for message types which include too many conjoined features such as the one in this example. We are working on devising a general linguistic constraint which would selectively disallow conjunction between very complex constituents. This might be done by counting the number of messages being conjoined and/or the number of distinct features within each message or the number of embedded conjunctions within clauses, disallowing those where the count is too high.

As an alternative strategy to PLANDOC's, STREAK relies on explicit surface form criteria to avoid generating sentences too hard to read. After each revision adding a new piece of information in the draft, the complexity of the draft is checked against the maximum complexity observed in the model corpus of the human- written summaries. Two criteria are used to define this complexity: the number of words, whose observed maximum is 45 and the depth of syntactic embedding, whose observed maximum is 10. This complexity check occurs right before the final sentence is generated. If the last revision has pushed the draft over one of these two thresholds, this last revision is then undone before the output is displayed. This constraint applies to any complex linguistic construction, whether or not it uses conjunction.

[11]It should be noted that although there is a good deal of literature on conjunction, much of it addresses issues involving competence rather than performance (the latter being closer to the constraints on processing complexity that we seek).

4.4 Abridged References

When facts get packed into the same sentence, it is sometimes the case that the same piece of information must be referred to more than one time. The summary can be made more concise by making a second reference to the same object shorter.

In PLANDOC this is done in the discourse planner by examining the content of the summary generated so far (in natural language, this is often termed "the previous discourse"). If the object was first referred to in the previous sentence, PLANDOC generates a shorter reference using just the head of the original noun phrase, changing the determiner as well. For example, on first reference to a refinement, PLANDOC would use the full refinement identifier as in "*Refinement 2A193*". On second reference in the subsequent sentence, PLANDOC generates "*This refinement.*" On third and subsequent references, PLANDOC uses a pronoun, "*it.*" PLANDOC uses essentially the *most recent mention* algorithm for interpreting anaphora [Hobbs 85], but reversed for generation.

STREAK can shorten a reference as part of revision. For example, suppose that it generates the draft sentence:
"*San Antonio, TX -- David Robinson scored 32 points Friday night lifting the San Antonio Spurs to a 127 111 victory over* the Denver Nuggets.",

To this, STREAK chooses to add a fact about past results of the Denver Nuggets using conjunction. Then it will generate:
"*San Antonio, TX -- David Robinson scored 32 points Friday night lifting the San Antonio Spurs to a 127 111 victory over Denver* ***and handing*** *the Nuggets* ***their seventh straight loss***".

In this revised sentence the original reference to the losing team is shortened from the default form "*the Denver Nuggets*" to just "*Denver*". The second reference added by the conjunction can then become just "*the Nuggets*". This avoids a redundant and verbose form like:
? "*San Antonio, TX -- David Robinson scored 32 points Friday night lifting the San Antonio Spurs to a 127 111 victory over the Denver Nuggets* ***and handing*** *the Denver Nuggets* ***their seventh straight loss***".

This also exemplifies how maintaining proper style while adding new content necessitates changing the existing draft. It also allows a more balanced division of the referent properties than is the case if a more traditional pronoun is used as second reference:
"*San Antonio, TX -- David Robinson scored 32 points Friday night lifting the San Antonio Spurs to a 127 111 victory over the Denver Nuggets* ***and handing*** *them* ***their seventh straight loss***".

5 Conceptual Summarization

Conceptual summarization requires determining the content of the summary, and to a certain extent, its organization. When summarizing data, the task involves determining which data are important enough to include in the summary. When summarizing text, the task is to determine which facts or characteristics of the text are important. Whether information is considered important will vary depending on the goal of summarization. For example, if the goal is to give the

reader enough information to determine whether or not to read the full text, the summary might be closer to an *indicative* abstract [Borko 75], including more information about text characteristics (e.g., its length, style, etc). In contrast, if the goal is to produce a shorter version of the text which the reader can read in place of the text itself, then the key content of the text is more important resulting in a summary much like an *informative* abstract [Borko 75]. Here again, depending on the reader's goals, key content can vary. For example, one sports fan might be more interested in tracking the achievements of a particular player, while another may want to track game results of all teams within an entire league.

For our applications, we are interested in the production of informative summaries. If there were a full text (e.g., a full newspaper article on the basketball game), the summary could be read in place in the full text. Thus, conceptual summarization must determine what key content should be conveyed. One of our claims is that the line between information that should or should not appear in a summary is fuzzy. If a summary could be longer, perhaps more information could be included. Our approach, therefore, involves a gradation of essential information to interesting information that could be included if space exists, but need not. We divide information into two classes, that which *must* appear, and potential information, which *may* be included if there is room. Potential information is ordered from more important to less important. Exactly how this information could be automatically generated is dependent on the domain and is discussed separately for STREAK and PLANDOC below.

Determining what information goes into these two classes (essential and potential) is part of conceptual summarization. But, as we just described in the previous section, determining which potential information actually gets included in a summary is done during linguistic summarization. Using both conceptual and linguistic constraints to fully determine the content of the summary is a unique contribution of our work. In this section we focus on the construction of these two classes of information. While we have not implemented conceptual summarization for STREAK, we describe our design and show how implementation is quite feasible.

Once we have the facts, then the techniques we have developed allow the generation of concise summaries. If the facts are obtained from a text, these same techniques are needed and can be used for certain kinds of text summarization tasks, namely informative summaries. To do this, the task of conceptual summarization could be handled in either one of two ways. Either a natural language interpretation system could process the full text, producing a set of facts representing text meaning. From this set of facts, a conceptual summarizer could select the subset that convey the key content, depending on the reader's interest. Alternatively, the bulk of the conceptual summarization task could be embedded in the interpretation system which, instead of "understanding" the full text, could instead extract only the facts of interest. Given the state of the art of natural language interpretation, the second alternative appears more feasible at this point in time and, in fact, is possible for certain domains and text types (e.g., newspaper articles on terrorism) as demonstrated by the ARPA message understanding systems (e.g., [Krupka, Jacobs and Rau 92]). Integrating a summary generator such as described here with the output of ARPA message understanding systems would allow the production of informative summaries of full text newspaper articles on terrorist events where the summaries contain only specific information such as type of terrorist event, victims, perpetrators, etc.,

regardless of the range of information included in the article.

In the remainder of this section, we describe the task of conceptual summarization for STREAK and PLANDOC.

5.1 Conceptual Summarization in STREAK

We determined which information was essential and which could potentially be included in the summary through the corpus analysis of newspaper basketball reports. Our first observation in the analysis was that all reports followed the *inverted pyramid structure with summary lead* where the most essential facts are included in the first sentence [Fensch 88]. We used the lead sentences (800 in total) in the corpus to guide determination of summary content. We noted that all 800 lead sentences contained the game result (*e.g.*, *"Utah beat Miami 105-95"*), its location, date and at least one final game statistic: the most remarkable statistic of a winning team player. These constitute the essential facts as they consistently appear in a fixed location in the report.

These facts can be obtained from the tabular formatted box score statistics. A box-score is a table containing a standard set of statistics for one game. In daily newspapers, there is one box-score accompanying each game report. An example box-score with the corresponding report is given in Figure 14. In this report, fine-grained statistics *not* available in the box-score are emphasized by an italic font (historical information is, as usual, emphasized by a boldface font). Such finer grained statistics come from complete game charts [Anderson 85]. Box-scores are available on-line through newswire and sport statistic computer services, whereas game charts are not[12]. Therefore, only box-scores constitute a realistic statistical input for a report generation system at this point in time.

For input to the summary generator, we need to convert purely quantitative data to a symbolic form that is suitable for content planning and that represents the meaning of numeric values (e.g., whether a numeric score is a win or a loss). Thus, the box score input is represented as a flat conceptual network that expresses relations between what were the columns and rows of the table. The symbolic representation of two facts (Kenny Smith's significant statistic and the game result) is given in Figure 15. This appears as fixed information in the first sentence of the report and appears in the box score as the last column of the 5th row entry in the Houston Rocket's table (Smith's statistic) and the heading of the two tables (game result and score). The symbolic form explicitly identifies winner and loser along with the scores. Currently, STREAK accepts this symbolic form as input. While the conversion from numeric, tabular form to symbolic, conceptual net has not yet been implemented, it would be very straightforward.

In addition to fixed, essential facts, input to STREAK also contains a symbolic representation of potential facts. We used the most commonly occurring floating facts that we found in the corpus:

- Other notable final game statistics of individual players (*e.g.*, *"Stockton finished*

[12]To the best of our knowledge.

ORLANDO MAGIC					24	24 (48)		21 (69)		25 (**94**)	
		fg	**3pt-fg**	**ft**	**rb**						
Players	**mn**	**m-a**	**m-a**	**m-a**	**o-t**	**bl**	**st**	**as**	**to**	**pf**	**tp**
Catledge	44	8-15	0-0	0-0	4-11	0	3	3	2	3	16
Reynolds	27	3-9	1-1	2-2	0-2	1	1	1	1	3	9
Roberts	16	4-7	0-0	1-2	2-6	2	0	1	1	5	9
Vincent	22	4-12	0-0	0-0	3-6	0	1	3	4	0	8
Anderson	30	6-10	0-0	2-4	1-2	0	1	1	0	2	14
Kite	22	2-4	0-0	0-0	3-4	0	0	0	0	2	4
O.Smith	26	4-11	0-0	1-2	0-1	0	2	1	0	2	9
Skiles	26	8-15	0-3	5-5	1-2	0	4	7	1	3	21
Scott	14	1-7	0-1	2-2	0-0	0	0	0	1	1	4
Je. Turner	4	0-1	0-0	0-0	0-1	0	0	1	0	3	0
Williams	9	0-1	0-0	0-0	0-1	0	0	0	0	0	0
Team totals	240	40-92	1-5	13-17	14-36	2	12	18	12	24	**94**
Team %		.435	.200	.765							

HOUSTON ROCKETS					20	25 (45)		31 (76)		23 (**99**)	
		fg	**3pt-fg**	**ft**	**rb**						
Players	**mn**	**m-a**	**m-a**	**m-a**	**o-t**	**bl**	**st**	**as**	**to**	**pf**	**tp**
Johnson	38	4-10	0-1	2-4	2-8	1	2	4	1	3	10
Thorpe	40	5-11	0-0	4-4	5-10	0	0	2	1	3	14
Olajuwon	33	6-16	0-0	5-6	3-14	8	2	4	3	6	17
Maxwell	42	4-14	3-9	3-4	0-3	0	1	5	3	3	14
K.Smith	38	12-15	1-1	3-3	1-4	0	2	9	6	0	**28**
Rollins	14	0-0	0-0	0-0	1-2	1	0	0	0	2	0
Floyd	19	2-5	1-2	7-8	0-3	0	0	4	4	2	12
Herrera	8	1-3	0-0	2-4	0-1	0	0	0	1	2	4
Jo. Turner	5	0-0	0-0	0-0	0-1	0	0	0	1	2	0
Bullard	3	0-0	0-0	0-0	0-0	0	0	0	0	0	0
Team totals	240	34-74	5-13	26-33	12-46	10	7	28	20	21	**99**
Team %		.549	.385	.788							

ORLANDO, Fla. (UPI) – Kenny Smith scored 28 points Sunday night to pace the Houston Rockets to a 99-94 victory over Orlando, **giving the Magic their league-high 10th straight loss**.

Hakeem Olajuwon contributed 17 points, 14 rebounds and eight blocked shots before fouling out with 3:15 remaining in the game.

The Magic led 48-45 at halftime, *but Houston outscored Orlando 23-12 in the first eight minutes of the third quarter to take the lead for good.*

Smith converted 12 of 15 shots from the field and dished out nine assists **to give the Rockets their sixth win in eight meetings with Orlando**.

Scott Skiles provided spark of the bench with 21 points, seven assists and four steals for the Magic, **which lost for the 14th time in 15 games**.

Orlando was outrebounded 46-36 and shot just 43.5-percent from the field. **The Magic have dropped their last six home games**.

Figure 14: Standard Box Score and Corresponding Natural Language Report

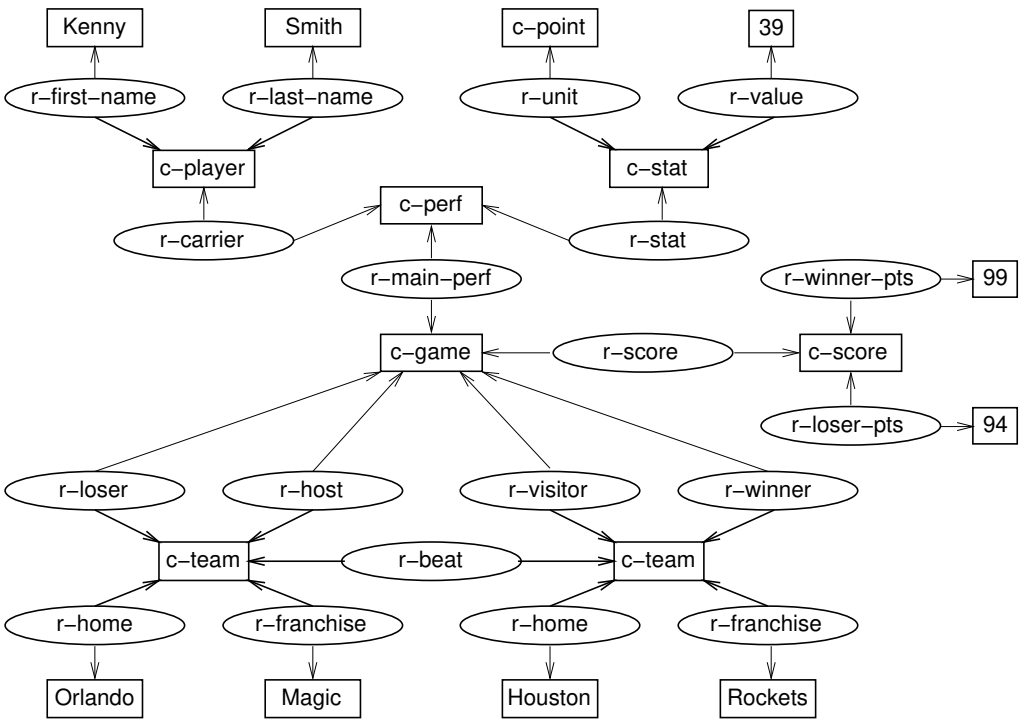

Figure 15: Symbolic Input for Facts in Box Score

with 27 points'').

- Game result streaks (*e.g.*, *''Utah recorded its fourth straight win''*).
- Extremum performances such as maximums or minimums (*e.g.*, *''Stockton scored* **a season-high** *27 points''*).

The first of these is termed "new information" and can be obtained from the box-score statistics. Box-score statistics are the most common type of new information. It was the sole type of new information in 50.2% of the corpus sentences.[13]. New statistics which are not fixed (i.e., other than the most significant player statistic) can be obtained directly from the box-score and placed on the list of potential information.

The second two types of potential information are historical. They place current game results and player statistics in historical context, underscoring their significance. The fact generator will ultimately query a database of historical statistics to retrieve for each new statistic a set of related historical statistics. In the sports domain, access to such historical databases through a modem is offered by several companies that specialize in such services. This historical data is used to assess the relevance of the new data and vice-versa. Thus, a new statistic will be ranked higher on the potential fact list if it is historically significant and the historical facts that best relate to significant new statistics will be ordered higher.

[13]Note also in Figure 14 that most of the information conveyed in the report comes directly from the box-score.

Note that the inclusion of historical information in report generation triggers a combinatorial explosion of the number of relevant propositions. This explosion is due to the fact that the historical context multiplies the dimensions along which to evaluate the significance of each input statistic. Consider for example, a player's scoring statistic. Ignoring the historical context there are basically only two ways in which this statistic can be significant: when compared to the scoring statistics of all the other players in the game (game-high) and when compared to the scoring statistics of his teammates only (team-high). However such a statistic can be *historically* significant in a combinatorially explosive number of ways: when compared to that player's scoring average (or high or low) over the season (or over his entire career or since he joined his current team, etc), when compared to the highest scoring performance of any player on his team (or on any team in a given division, conference, etc) this season (or over the last 10 games, or ever) while playing at home (or on the road) against a particular opponent (or against any opponent) etc.

Such facts can be determined by comparing a statistic or game result against varying periods times and against different groups of players. While we have not implemented this, it could be done by encoding a systematic search for maximums, minimums, and consecutive sequences of similar results. Because of the sheer number of historical facts, any such fact is likely to be of similar relevance as many others. Therefore, content production results in a much larger set of candidate propositions with many more shades of relative importance. This makes content selection much harder. General rules based on purely domain grounds (*e.g.,* if a player scores more than 20 points or if he scores more than anybody in his team, then include his scoring in the report) can be used to provide some ordering of the facts.

However, there are still likely to be too many facts to include in the final report. Other factors need to be considered. One such factor is whether the possible forms of the candidate propositions can be combined with the draft summary in cohesive and stylistically felicitous surface forms. The ability to convey historical information thus entails the ability *to perform final content selection under surface form constraints*. Thus, while conceptual summarization can generate and order potential new and historical facts, linguistic summarization must be called into play to determine which of these will actually be included.

5.2 Conceptual Summarization in PLANDOC

Conceptual summarization in PLANDOC is currently much simpler than in STREAK, although this will change once we begin generating the base plan and proposed plan paragraphs of the full report along with enriched refinement paragraphs.

The current refinements summary is intended to provide information about the scenarios explored by the engineer that led to the proposed final plan. Thus, the essential facts are those refinements which the engineer made and included in the final plan. Recall that the engineer may also experiment with refinements to the base plan and finally, reject these refinements if she decides they are not optimal for any reason. For example, the engineer may wish to test some hypothetical scenarios to show management that other possibilities were explored but rejected because they were not optimal. Alternatively, the engineer may just want to get a feel

for the route before seriously beginning making changes to the system generated plan (the base plan). Refinements which were tried but not included in the final plan are potential candidates for inclusion in the summary. Whether the refinement is included depends on whether it is a plausible alternative to the final plan which management will want to know about. This is very domain specific information which is not currently encoded anywhere in the system (and since it relies heavily on human expertise, would be quite difficult to encode). Only the planning engineer knows why she tried and rejected any refinements. Thus, in our current version of the system, we ask the planning engineer to indicate which refinement scenarios should be included in the summary when a report is requested.

In the final version of the system, output will include an introductory paragraph summarizing the base plan (i.e., the plan which the system produced), enriched refinement summary paragraphs (i.e., that include additional facts related to refinement actions), and a closing paragraph that summarizes the proposed plan (i.e., that indicates which base plan elements were included and which refinements were included). We expect that determining the content of these additional paragraphs and sentences will require some domain heuristics. Knowledge engineering techniques such as interviewing planning experts and analyzing human-generated model summaries will be required to devise such heuristics. These heuristics could then be used to identify salient facts from among all those available in the data representing PLAN's base plan results, and to categorize those facts as essential or potential facts. Once facts have been identified and categorized, the linguistic summarization devices already in use for both STREAK and PLANDOC can be applied. We have just begun to analyze a small sample of model summaries.

Figure 16 provides an example of the kind of information one expert felt should be included in a base plan summary paragraph. This example is informative with respect to both potential heuristics for conceptual content inclusion and useful linguistic summarization devices. Conceptually, this summary suggests heuristics such as "limit essential facts to those occurring within the first 10 years of the study", "mark as essential all facts related to activations, DLC transfers and placements", and "create negation facts for those actions that are notable by their absence" (e.g., no coarse gauge cable placements or duct placements). Linguistically, note the use of the colon-enumeration format for concise expression of similar facts. Note also the use of the *General Adjunctization Rule* in items (2) and (3) by which a placement fact is succinctly folded into an activation sentence via a "with" clause. Item (3) achieves further conciseness by eliding the subordinate verb "placed". Finally, the common feature *negation* allows three facts to be combined and ellipsis to be used in the negation sentence.

In our corpus of human-generated refinement summaries, we found that refinement paragraphs were sometimes enriched with "non-refinement" sentences, i.e., statements that referred to facts that were *not* among those in the input trace of the refinement scenario in process. Two such non-refinement sentences are shown in bold-faced type in Figure 17.

The first non-refinement sentence refers to facts that would be found in the route model

The BASE plan for this route calls for the following actions in the first 10 years of the study: (1) CSA 6501 is activated for DLC in 1992Q1; (2) CSA 6122 and CSA 6131 are activated for fiber in 1993Q1 with 150mb FOT's placed; (3) CSA 6501 is activated for fiber in 1995 with a 150mb FOT; and (4) 35 lines are transferred from copper to DLC in CSA 6801. There are no copper terminations, coarse gauge cable or duct placements in the first 10 years of the study. The BASE plan 20 year route PWE is $855K and the 5 year IFC is $463K.

Figure 16: One Model Base Plan Summary

After starting over with the BASE plan, this refinement demanded a fiber activation for CSA 2551 in the first quarter of 1994. **CSA 2551 is an existing DLC site with 180 conditioned copper pairs and 0 spare.** ...

RUN-ID DLC94 activated CSA 4111 for DLC in 1994 Q4. **CSA 4111 was activated in 1997 in the BASE plan**. ...

Figure 17: Enriched Refinement Summary Paragraphs

database; the second refers to a fact found in the base plan data; other non-refinement sentences we've encountered refer to facts found in previous refinement scenarios. Since non-refinement facts do not appear in the trace of the refinement scenario currently in process, PLANDOC's content-planner must proactively seek out these facts from the total pool of route model facts, base plan facts and previous (and perhaps subsequent) refinement facts, and it must do so intelligently. That is, it must exploit both domain heuristics and discourse structure knowledge to determine when and how an non-refinement fact might be appropriately included. One such domain heuristic suggested by the second example in Figure 17 is: "search the previous refinement fact pool for facts sharing at least three common features and one contrastive feature with the current fact in focus". In this case three common features are CSA-site (4111), action (activation), and equipment (DLC) and one contrastive feature is date. This heuristic could also be invoked by a *contrastive* rhetorical relation.. [McKeown 85b], [Mann 84] The first non-refinement example above employs the *General Adjunctization Rule* to concisely express two facts.

The closing paragraph, i.e., the proposed plan summary, will essentially serve as a summary of the entire previous text. Due to length restrictions it most likely will only be able to contain the most important changes from both the base plan and the engineer refinements. For these summaries, domain heuristics can be used to indicate which types of refinements are more important than others. In addition, linguistic summarization techniques can be used to condense the final version. One clear issue in the organization of proposed plan summaries is the need for decisions about what information should be included in the summary to be influenced by how information can be grouped together in a single sentence. In order to convey information more concisely, similar refinements are often grouped together and short phrases that can refer to the group selected, usually using conjunction and ellipsis. Thus, for example, in one of our model summaries shown below in Figure 18, the summary groups together a set of refinements which activated a specific type of equipment and uses a nominalization to refer to

them ("*DLC activations*"). Thus, as is the case in the refinements summary, wording and syntactic constraints will determine what information gets included in the summary and where. However, unlike the current implementation, the system needs to be able to determine which refinements should not be mentioned and when additional information should be included. For example, the summary should mention aspects of the BASE plan which are recommended and it may sometimes need to provide supporting information from other databases that justifies the recommended plan.

The refinement "fib_2551" is the best solution for this route. This plan includes the BASE plan DLC activations of CSA 2907 in 1997 and CSA 2119 in 1998. It also includes the BASE plan fiber activation of CSA 2317 in 1994Q4. ...

Figure 18: Portion of a Model Proposed Plan Summary

6 Related Work

The research presented in this paper concerns two subareas of generation: report generation, relevant to both PLANDOC and STREAK, and revision- based generation, relevant to STREAK. We compare previous work in each of these two subareas with our approach in what follows.

6.1 Report Generation

At least four previous systems use natural language to summarize quantitative data: SEMTEX [Roesner, D. 87], LFS [Iordanskaja et al. 92], FOG [Bourbeau et al. 90] and ANA [Kukich 83].

SEMTEX and LFS each generate summaries of labor market statistics. SEMTEX generates German text; LFS generates bilingual French/English reports. SEMTEX does both conceptual and linguistic summarization, each in a separate module. The conceptual module takes raw labor market statistics as input and outputs a list of frame structures representing the text's content. The frame structures are converted to functional structures representing thematic and lexical information by SEMTEX's linguistic module. SEMTEX's functional structures are similar to but less detailed than those of STREAK and PLANDOC. A surface generator converts SEMTEX's functional structures to linearized German text. SEMTEX's linguistic module emphasizes the need for conciseness in summaries by using ellipsis to avoid repetition of redundant information whenever possible, e.g., information about the time period concerned. However, it does not make systematic use of conjunction as does PLANDOC. LFS is similar to SEMTEX in that both are implemented for similar domains and produce summaries of similar content and style [Iordanskaja et al. 92]. However, the focus in LFS is on the use of Meaning-Text Theory and lexical functions [Mel'cuk 81] that allow for the computation of collocations and semantically related lexemes, resulting in a system that employs systematic paraphrasing. Important types of conjunction and ellipsis are also covered in its extensive grammar, but the full range of linguistic summarization techniques we describe here is not a focus of their work.

FOG generates marine weather forecasts from meteorological data and remains to date the only

generator in everyday industrial use. ANA generates stock market reports from updates arriving on the half hour of financial index values. FOG performs only *conceptual* summarization, by picking only a few facts among the large amount of input data available. It does not perform *linguistic* summarization, discussed in this paper: it generates many short sentences in a telegraphic style in which fluency is essentially a non-issue.

This is in contrast with ANA whose main theme was the generation of multiple clause sentences fluently combining several related facts. ANA performs both conceptual and linguistic summarization. It achieves a high degree of fluency for fairly complex sentences by relying on a phrasal lexicon. The entries in this lexicon are phrases, comprising up to 8 words, simultaneously realizing several facts in a idiomatic and concise way. The advantage of this approach it is that it circumvents the identification of the complex constraints that influence the generation of these phrases. It has two drawbacks. First, it cannot exploit summarization potential lying below the clause rank such as the nominalization, head noun modification or nominal conjunctions examples presented in this paper. Second, it makes scaling up the paraphrasing power prohibitively costly, since it requires the hand-coding of a combinatorially explosive number of phrases.

In the two systems presented here, STREAK and PLANDOC, the lexicon entries are individual words or collocations, each realizing one to three fine-grained content units. Phrases that were stored as a whole in ANA's lexicon are instead dynamically built at generation-time. Though other generation systems have used such a compositional approach [McKeown et al. 90], [Dale 92], [Elhadad 93a], PLANDOC and STREAK are the first to exploit it to perform pervasive linguistic summarization. By using revision in addition to a word-based lexicon, STREAK takes this compositional approach one step further, resulting in a recognized benefit of compositionality: enhanced expressive flexibility and reduced scale-up cost. However, such extreme compositionality also involves an increase in the overall complexity of the generation process.

6.2 Revision

Previous work viewed revision uniquely as a way of improving stylistic quality while preserving content. Analyzing a corpus of work sessions from professional editors, [Meteer 91] presented an interesting set of revision operations to leave draft content intact while improving its style. She did not attempt to implement these operations in a generator. [Inui, Tokunaga and Tanaka 92] presented a generation system in which similar types of content-preserving revisions are implemented.

In STREAK, revision is viewed as a way to gradually and opportunistically improve informative content of the report while keeping it short and concise. This view brings together revision with another line of research: incremental generation [Joshi 87], [De Smedt 90]. This line of research, however, limited itself to study only elaborations that *preserve* the linguistic form of the draft. The most sophisticated revision tools of STREAK, such as nominalization with adjoin discussed above, show that it is often necessary to *change* the linguistic form of the draft in order to accommodate additional background facts in the most compact way. STREAK is unique in that it performs revisions that alter both the content *and* the linguistic form of the draft and

simultaneously improves both the coverage and compactness of the summary.

7 Current Directions

For STREAK, we have looked at the portability of the revision based approach by doing an analysis of summaries in another quantitative domain, the stock market. This analysis showed that many of STREAK's revision rules could be directly used in this domain. For PLANDOC, we are currently turning our attention towards the generation of summaries of PLAN's base plan and of the final recommended plan, as described earlier. We are also beginning a new summarization application, the generation of full text summaries, using input from natural language information extraction systems. In this section, we discuss these directions briefly.

7.1 Portability of STREAK's Revision Based Approach

While the use of revision to generate concise, compact summaries appears promising, we were interested in determining whether any of the specific revision rules we had identified could be used when moving to a new domain. In particular, we were curious whether the rules could be used directly for similar domains or whether they were specific to sports reports.

To assess the degree of domain dependence of the revision rules we abstracted from our analysis of sports summaries, we looked for examples of their use in stock market summaries. Use of a revision rule in a corpus can be identified by the simultaneous presence of:

1. A simple sentence whose content types and form match the triggering conditions of the rule.
2. A more complex sentence whose content types and form can be derived from the simple one by applying the rule.

For example, the rule *Nominalization with Ordinal Adjoin* was originally abstracted from the following pair of sports sentences:

- *''helping Detroit defeat Indiana 114 112''*
- *''helping the Milwaukee Bucks **hand** the Denver Nuggets **their fourth straight defeat**, 115-98.''*

Here, the verb "*to defeat*" is nominalized in order to modify it by an ordinal pre-modifier providing the historical context of the defeat in just two words, something that could not have been done as concisely by directly modifying the verb or clause. The rule for this revision was presented in Section 4.2

This tool is used in the stock market as well, as demonstrated by the following pair of corpus sentences:

- *''the Tokyo stock market plunged''*
- *''the Tokyo stock market **posted its third consecutive** decline''*

where the verb "*to plunge*" is nominalized into the noun "*decline*" also because the latter can be concisely modified with streak information ("*third consecutive*").

This portability evaluation was partially automated by using CREP [Duford 93], a software tool that retrieves corpus sentences matching a regular expression of words and/or parts-of-speech tags. We used such expressions to approximate the basic and revised sentence patterns characterizing each revision rule observed in the sports domain. CREP was run on a corpus of over 18,000 sentences in the financial domain compiled from several newswires.

The revision rules identified in the sports domain form (and are implemented as) a hierarchy characterizing the observed transformations at various degrees of specificity. On top of this hierarchy sits nine general syntactic transformations such as *nominalization, adjunctization, adjoin, conjoin* etc. At this coarse grain, we found that portability was 100%: all nine of these general transformation were used at least once in our test financial corpus. Each of transformation is refined down the hierarchy in terms of the syntactic category and semantic roles of each constituent involved, yielded more specific revision rules such as *adjunctization of object NP into instrument PP, adjoin of non-finite clause to NP* etc. At this finer grain, portability fell to 71.4%. For example, no usage of *adjoin of non-finite clause to NP* was found in our financial corpus even though cases of *adjoin of relative clause to NP* were found[14]. The most specialized rules incorporate stylistic transformations such as abridged references in addition to syntactic ones. Such transformations tend not to port very well. Consider, for example, the strategy for abridging references presented in section 4.4. It relies on the fact that in the sports domain all entities have compound names (*e.g.,* "Denver Nuggets", "Shaquille O'Neal") different parts of which can be used unambiguously in different references for the targeted audience, i.e., sports fans. This is not the case for all financial entities. Nevertheless, even at this finest grain, a majority of revision rules (51.9%) were portable.

This is a very encouraging result for the portability potential of STREAK to other summarization domains and more generally for the potential of scaling up the revision-based generation approach. For an in-depth presentation of this portability assessment see [Robin 94]

7.2 Generating Summaries from News Articles

Given the recent results of the natural language information extraction systems developed under ARPA's message understanding conferences [MUC-4 92] along with recent advances in natural language generation, the stage is now set for full text summarization. Information extraction systems use natural language analysis to extract key facts from the newspaper articles in a given domain. For example, when applied to articles about terrorism, these systems extract facts such as the victim, the perpetrator, the level of confidence, the source, the time, the location, etc. If these facts were passed as input to a natural language summarizer such as the ones we have discussed here, it would be possible to produce summaries of news articles in specific domains. This is a task which we have begun to work on[15].

[14]Note that a larger financial corpus may have contained such cases. The results reported here are thus lower bounds on portability.

[15]This work is being carried out by Dragomir Radev.

Input to our system will be a set of articles on a single event. The generation system will use the set of templates produced by the information extraction system for each article. We are currently using the published "correct" answers during development as planned input for our system[16]. In any case, this means the summarization system will be used to produce a textual summary, where the input facts have been extracted from a text. From our initial analysis of a corpus of articles used for the message understanding system evaluation, there seems to be several ways the input articles can vary. For example, the sources (e.g., AP newswire versus Reuters) can vary while the account of the event is essentially the same. Alternatively, the sources may be different and the accounts contradictory. Perhaps the most interesting case is when the source is the same, but the time of the account varies. In this scenario, the articles relate a changing view of the same event over time.

In addition to the dimensions along which the articles can vary, we have also noted that summaries take advantage of general knowledge that tends to be fixed over time. This may or may not be explicitly mentioned in the input news article. Even if it is, the summarizer must be able to distinguish between new facts about an event (what is being reported) and facts that remain constant over time and set the stage for the article. For example, assassination of an official may be reported along with the fact that he is President of his country. We are exploring whether a distinction between these kinds of facts is similar to the distinction between essential facts (game results) and potential (historical significance) in the basketball domain. Here, we suspect that well-known, facts that have been true for some period of time and serve to set the stage for the "news" may function similarly to historical information; they get included if they fit and if they can be folded into the draft. Note that in some cases, such facts may be essential, but they still may be more easily conveyed in modifier form.

While this summarization domain involves summarizing of qualitative data and the work we have reported here involves summarizing quantitative data, note that once input has been produced in symbolic form (that is, once the conceptual summarizer has completed its task), there is nothing in our approach that is dependent on quantitative input. Linguistic summarization only presupposes that the input will include a list of essential facts and a list of potential facts; whether or not potential facts are included in the summary depends on how they can be realized and the form of the draft, criteria that can be applied to any form of input fact. However, simply interfacing an message understanding system with summarizers such as we describe will neither fully resolve the text summarization problem nor result in a full system. First, an integration of these two types of systems will produce a restricted class of summaries. The summaries will be informative and can only include information extracted by the message understanding system. Note that if information on a terrorist event can be extracted from the article, the summary will convey only that information regardless of whether the article was about a terrorist event or not. Thus, if the article included a digression on a terrorist event in the course of relating some other story, the summary will include no information on that other story. Another problem that must be addressed in developing a fully integrated text

[16]Evaluations of these systems are carried out by comparing system output with "correct" answers that have been prepared ahead of time by hand by the evaluation committee. It is these answers that we are using.

summarization system from these two existing systems is knowledge abstraction. Generalizations from individual facts extracted from separate articles may result in a single fact which adequately and more concisely conveys the several individual facts. This is part of what we are working on.

Our work is still at early stages, but we have noted the different forms of summaries that can be used depending on the kinds of generalizations that can be made about the input facts. We will explore integrating the approaches taken in STREAK and in PLANDOC so that the summarization system can use both discourse planning and revision in combination to produce the summaries.

8 Conclusion

Our work identifies problems unique to concise summarization of different types of information and demonstrates an approach that has been successfully implemented in two separate systems. Underlying our approach is the basic tenet that summary generation requires a different approach than used in traditional language generation. Summaries attempt to fit as much information as possible into as short a space as possible. Developing a system that can do this requires changing the generation process. We have shown that the following features are essential to a summary generator:

- A summary generator must be able to opportunistically add information to the summary, incrementally building complex sentences.
- The addition of new information can be triggered by the surface form of an existing text, along with semantic constraints. Thus, the syntax or words used in an existing text can allow or disallow the addition of specific new information.
- Producing a concise summary is dependent on finding concise constructions that take advantage of similarities between information already in the text and new facts, thus compactly expressing new facts through small changes to existing sentences.
- Opportunistic addition of information implies a division of information into essential information that *must* appear in the summary and additional facts that are added *if possible*.

In both domains, the examples illustrate that selection of information to include in a summary can depend on lexical and syntactic constraints. In STREAK, the wording of the draft summary can determine whether information gets included at all in the final version, for example addition of an optional fact can be triggered by a verb which can be nominalized. In PLANDOC, choosing what optional facts to include and how to order them depends on whether they can be concisely conjoined using ellipsis. Influence of wording and surface form on the inclusion and ordering of information is quite unique to our work; previous generation systems used content planning [McKeown 85a], [Hovy 91], [Moore and Paris 89], [Paris 87] that relied on rhetorical strategies, relations, or planning about the user's knowledge to determine the content and organization of a text. These are all reasoning strategies that rely on conceptual information only and ignore the text surface form.

This approach of incrementally adding in information as it fits makes it possible for a generator

to produce the complex sentences required in summaries. Sentences of almost 50 words in length can be generated, much longer than sentences produced by any previous language generation system. Moreover, these sentences are generated from individual words[17] and use concise linguistic forms. This happens since the summary generator chooses various modifying constructions that gradually add information and build up the summary sentences to full length.

In this paper, we described four main constructions that could be used to concisely add information, exploiting repetitions across facts; these included the selection of a single word to simultaneously convey multiple pieces of information, the selection of modifiers to convey full facts, the use of conjunction along with ellipsis to group together similar facts while deleting repetitions, and the use of abridged references when multiple references to the same object are required.

STREAK and PLANDOC use different, but complementary, approaches to the task of opportunistically adding information based on linguistic constraints. STREAK uses revision to incrementally add information, constructing a full draft of essential facts, and using revision rules that encode lexical, syntactic and semantic constraints to determine how to fold in new information to the draft. PLANDOC embodies a more traditional language generation approach, using discourse planning to look ahead and gather together facts that can be combined in systematic ways using conjunctions of various different syntactic constituents, reasoning about how to delete repetitions. Ultimately, these two approaches can be combined to yield a more complete approach to summarization. STREAK's approach allows local changes within a sentence, while PLANDOC's can pull information together from distant parts of its text plan using long distance knowledge to make modifications. Both local and discourse knowledge must eventually be coordinated in the summarization task.

Finally, both applications illustrate the feasibility of summarizing input data as opposed to full text. In both domains, the input is currently readily available as data. For STREAK, box scores for a game are accessible from online services; for PLANDOC, input data is drawn from an existing application that planning engineers currently use in production at various regional companies. Summarization is thus practical today, without also having to carry out the difficult task of interpreting texts in the domain. Moreover, the summarization techniques we have developed are general. Were information to be extracted from a text using natural language interpretation, the same summarization techniques we have described here could be applied to produce a natural language summary of a text. In fact, as we have described in Section 7, we believe that today's technology is currently poised for a breakthrough in this area; by using the output of information extraction systems developed in ARPA's message understanding program, which can extract key facts from news articles in restricted domains, and feeding this information as input to a summary generator we could see the realization of full text summarization for a limited class of summary types, namely informative summaries conveying specific types of key content.

---

[17] As opposed to hard-wired phrases.

Acknowledgements

This work was supported in part by Bellcore through sponsored research agreement *Bellcore CU01403301, by a joint grant from DARPA and ONR under contract N00014-89-J-1782, by NSF GER-90-24069, and by New York State Center for Advanced Technology Contract NYSSTF-CAT(91)-053. We would like to acknowledge the contributions of James Shaw in the development of conjunction in PLANDOC and for his continuing work on this system.

References

[Anderson 85] Anderson, D.
Contemporary sports reporting.
Nelson-Hall, Chicago, IL, 1985.

[Appelt 85] Appelt, D.E.
Planning English Sentences.
Cambridge University Press, Cambridge, England, 1985.

[Borko 75] Borko, H.
Abstracting Concepts and Methods.
Academic Press, New York, NY, 1975.

[Bourbeau et al. 90]
Bourbeau, L., Carcagno, D., Goldberg, E., Kittredge, R. & Polguere, A.
Bilingual generation of weather forecasts in an operations environment.
In *Proceedings of the 13th International Conference on Computational Linguistics*. COLING, 1990.

[Cox 93] Cox, J.
Text-analysis server to simplify queries.
Communication Week :, April, 1993.

[Dale 92] Dale, R.
Generating Referring Expressions.
ACL-MIT Press Series in Natural Language Processing, Cambridge, Ma., 1992.

[Danlos 86] Danlos, L.
Studies in Natural Language Processing: *The linguistic basis of text generation.*
Cambridge University Press, 1986.

[De Smedt 90] De Smedt, K.J.M.J.
IPF: an incremental parallel formulator.
In R. Dale, C.S. Mellish, & M. Zock (editor), *Current Research in Natural Language Generation*. Academic Press, 1990.

[Duford 93] Duford, D.
CREP: a regular expression-matching textual corpus tool.
Technical Report CUCS-005-93, Columbia University, 1993.

[Elhadad 93a] Elhadad, M.
Using argumentation to control lexical choice: a unification-based implementation.
PhD thesis, Computer Science Department, Columbia University, 1993.

[Elhadad 93b] Elhadad, M.
FUF: The universal unifier - user manual, version 5.2.
Technical Report CUCS-038-91, Columbia University, 1993.

[Elhadad and Robin 92]
Elhadad, M., & Robin, J.
Controlling Content Realization with Functional Unification Grammars.
In R. Dale, H. Hovy, D. Roesner, & O. Stock (editor), *Aspects of Automated Natural Language Generation*, pages 89-104. Springler Verlag, 1992.

[Feiner and McKeown 91]
Feiner, S., & McKeown, K.R.
Automating the Generation of Coordinated Multimedia Explanations.
IEEE Computer 24(10):33-41, October, 1991.

[Fensch 88] Fensch, T.
The sports writing handbook.
Lawrence Erlbaum Associates, Hillsdale, NJ, 1988.

[Gopen and Swan 90]
Gopen, G.D., & Swan, J.A.
The Science of Scientific Writing.
American Scientist (78):550-558, 1990.

[Hobbs 85] Hobbs, J.R.
Ontological promiscuity.
In *Proceedings of the 23rd Annual meeting of the ACL*, pages 61-69. ACL, 1985.

[Hovy 85] Hovy, E.
Integrating Text Planning & Production in Generation.
In *Proceedings of the 9th International Joint Conference on Artificial Intelligence*. IJCAI, 1985.

[Hovy 88] Hovy, E.
Generating natural language under pragmatic constraints.
L. Erlbaum Associates, Hillsdale, N.J., 1988.

[Hovy 91] Hovy, E.
Approaches to the planning of coherent text.
In C. Paris, W. Swartout, & W.C. Mann (editor), *Natural Language Generation in Artificial Intelligence and Computational Linguistics*. Kluwer Academic Publishers, 1991.

[Inui, Tokunaga and Tanaka 92]
Inui, K., Tokunaga, T., & Tanaka, H.
Text revision: a model and its implementation.
In R. Dale, E. Hovy, D. Roesner, & O. Stock (editor), *Aspects of Automated Natural Language Generation*, pages 215-230. Springler-Verlag, 1992.

[Iordanskaja et al. 92]
Iordanskaja, L., Kim, M., Kittredge, R., Lavoie, B. & Polguere, A.
Generation of Extended Bilingual Statistical Reports.
In *Proceedings of COLING-92*, pages 1019-1023. COLING, 1992.

[Jacobs 85] Jacobs, P.
PHRED: a generator for natural language interfaces.
Computational Linguistics 11(4):219-242, 1985.

[Joshi 87] Joshi, A.K.
The relevance of Tree-Adjoining Grammar to generation.
In G. Kempen (editor), *Natural Language Generation: New Results in Artificial Intellligence, Psychology and Linguistics*. Martinus Ninjhoff Publishers, 1987.

[Kay 79] Kay, M.
Functional Grammar.
In *Proceedings of the 5th Annual Meeting of the Berkeley Linguistic Society*. 1979.

[Krupka, Jacobs and Rau 92]
George R. Krupka and Paul S. Jacobs and Lisa F. Rau.
GE NLToolset: Description of the System as Used for MUC-4.
In *Proceedings of the Fourth Message Understanding Conference (MUC-4)*, pages . , Morgan Kaufmann Publishers, San Mateo, CA, June, 1992.

[Kukich 83] Kukich, K.
The design of a knowledge-based report generation.
In *Proceedings of the 21st Conference of the ACL*. ACL, 1983.

[Kukich et. al. 93] Kukich, K., McKeown, K., Morgan, N., Phillips, J., Robin, J., Shaw, J., & Lim, J.
User-Needs Analysis and Design Methodology for an Automated Documentation Generator.
In *Proceedings of the Bellcore/BCC Symposium on User-Centered Design: ''People and Technology''*. Piscataway, N.J., October, 1993.

[Mann 83] Mann, W.C.
An overview of the PENMAN text generation system.
Technical Report ISI/RR-83-114, ISI, Marina del Rey, CA, 1983.

[Mann 84] Mann, W.C.
Discourse Structure for Text Generation.
Technical Report ISI/RR-84-127, Information Sciences Institute, February, 1984.
Marina del Rey, CA.

[Matthiessen 91] Matthiessen, C.M.
Lexicogrammatical choice in text generation.
In C. Paris, W. Swartout, & W.C. Mann (editor), *Natutal Language Generation in Artificial Intelligence and Computational Linguistics*. Kluwer Academic Publishers, 1991.

[Maybury 90] Maybury, M.T.
Using discourse focus, temporal focus and spatial focus to generate multisentential text.
In *Proceedings of the 5th International Workshop on Natural Language Generation*. Pittsburgh, PA, 1990.

[McCoy 86] McCoy, K. F.
The ROMPER System: Responding to Object-Related Misconceptions Using Perspective.
In *Proceedings of the 24th Annual Meeting of the ACL*. Association of Computational Linguistics, New York City, New York, June, 1986.

[McDonald 83] McDonald, D.
Description Directed Control: its Implications for Natural Language Generation.
In B. Grosz, K. Sparck-Jones, & B. Webber (editor), *Readings in Natural Language Processing*. Morgan Kaufmann Publishers, 1983.
also in Computers & Mathematics #9(1).

[McKeown 85a] McKeown, K. R.
Studies in Natural Language Processing: *Using Discourse Strategies and Focus Constraints to Generate Natural Language Text.*
Cambridge University Press, 1985.

[McKeown 85b] McKeown, K. R.
The need for text generation.
Technical Report CUCS-173-85, Columbia University, 1985.

[McKeown and Swartout 87]
McKeown, K.R. and W. R. Swartout.
Language Generation and Explanation.
In J. Traub, B. Grosz, B. Lampson, & N. Nilsson (editors), *Annual Review of Computer Science*. Annual Reviews Inc., Palo Alto, Ca., 1987.

[McKeown et al. 90]
McKeown, K.R., Elhadad, M., Fukumoto, Y., Lim, J.G., Lombardi, C., Robin, J., & Smadja, F.A.
Text generation in COMET.
In R. Dale, C.S. Mellish, & M. Zock (editor), *Current Research in Natural Language Generation*. Academic Press, 1990.

[McKeown et.al. 93]
McKeown, K.R., Robin, J., and Tanenblatt, M. .
Tailoring lexical choice to the user's vocabulary in multimedia explanation generation.
In *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics*. Columbus, Oh., June, 1993.

[Mel'cuk 81] Mel'cuk, I.A.
Meaning-Text Models: a Recent Trend in Soviet Linguitics.
The Annual Review of Anthropology (10), 1981.

[Meteer 91] Meteer, M.
The implications of revisions for natural language generation.
In C. Paris, W. Swartout, & W.C. Mann (editor), *Natural Language Generation in Artificial Intelligence and Computational Linguistics*. Kluwer Academic Publishers, 1991.

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

[Moore and Paris 89]
Moore, J.D., & Paris, C.L.
Planning Text for Advisory Dialogues.
In *Proceedings of the 27th Annual Meeting of the Association for Computational Linguistics*, pages 203-11. Asscoication for Computational Linguistics, Vancouver, B.C., June, 1989.

[MUC-4 92] DARPA Software and Intelligent Systems Technology Office.
Proceedings of the Fourth Message Understanding Conference, 1992.

[Paice 90] Paice, C.
Constructing literature abstracts by computer: techniques and prospects.
Information Processing and Management 1(26):171-186, 1990.

[Paris 87] Paris, C.L.
The Use of Explicit User models in Text Generation: Tailoring to a User's Level of Expertise.
PhD thesis, Columbia University, 1987.

[Quirk et al. 85] Quirk, R. and Greenbaum, S. and Leech, G. and Svartvik, J.
A comprehensive grammar of the English language.
Longman, 1985.

[Rambow & Korelsky 92]
Rambow, O., & Korelsky, T.
Applied Text Generation.
In *Proceedings of the 3rd Conference on Applied Natural Language Processing*, pages 40-47. Association for Computational Linguistics, Trento, Italy, April, 1992.

[Rau 87] Rau, L. F.
Information Retrieval in Never-ending Stories.
In *Proceedings of the Sixth National Conference on Artificial Intelligence*, pages 317--321. Morgan Kaufmann Inc., Seattle, Washington, July, 1987.

[Reiter 91] Reiter, E.B.
A New Model for Lexical Choice for Open-Class Words.
Computational Intelligence (7), December, 1991.

[Robin 93] Robin, J.
A Revision-Based Generation Architecture for Reporting Facts in their Historical Context.
In H. Horacek, & M. Zock (editor), *New Concepts in Natural Language Generation: Planning, Realization and Systems*. Frances Pinter, London and New York, 1993.

[Robin 94] Robin, J.
Revision-Based Generation of Natural Language Summaries: Corpus-Based Analysis, Design, Implementation and Evaluation.
PhD thesis, Computer Science Department, Columbia University, 1994.
Forthcoming.

[Robin and McKeown 93]
Robin, J. and McKeown, K.R.
Corpus Analysis for Revision-Based Generation of Complex Sentences.
In *Proceedings of the National Conference on Artificial Intelligence*. Washington, D.C., July, 1993.

[Roesner, D. 87] Roesner, D.
SEMTEX: A Text Generator for German.
In G. Kempen (editor), *Natural Language Generation: New Results in Artificial Intelligence, Psychology and Linguistics*, pages 133-148. Martinus Ninjhoff Publishers, 1987.

[Rubinoff 88] Rubinoff, R.
A Cooperative Model of Strategy and Tactics in Generation.
Presented at the Workshop on Natural Language Generation.
Forthcoming, 1988.

[Suthers 91] Suthers, D.
A Task-Appropriate Architecture for Explanation.
Computational Intelligence 7(4):315-333, 1991.

[Wahlster et al. 89]Wahlster, W., Andre, E., Hecking, M., & Rist, T.
WIP: Knowledge-based Presentation of Information.
Technical Report WIP-1, German Research Center for Artificial Intelligence, Saarbruecken, FRG, May, 1989.

[Wolz et. al. 89] Wolz, U., McKeown, K.R., & Kaiser, G.
Automated Tutoring in Interactive Environments: A Task Centered Approach.
Journal of Machine Mediated Learning , 1989.

[Zukerman and Pearl 86]
Zukerman, I., & Pearl, J.
Comprehension-driven generation of meta-technical utterances in math tutoring.
In *Proceedings of AAAI-86*. Philadelphia, PA, 1986.