To Parse or Not to Parse: Relation-Driven Text Skimming

Paul S. Jacobs
Artificial Intelligence Program
GE Research and Development Center
Schenectady, NY 12301 USA
psjacobs@crd.ge.com

Abstract

We have designed and implemented a text processing system that can extract important information from hundreds of paragraphs per hour and can be transported within weeks to a new domain. The system performs efficiently because it determines the level of processing required to understand a text. This "skimming" method identifies surface relations in the input text that are likely to contribute to its interpretation in a domain. This approach differs from previous skimming techniques in that it uses conceptual information as part of bottom-up linguistic processing, thus using linguistic knowledge more fully while limiting grammatical complexity.

1 Introduction

Natural language systems that extract information from volumes of text have matured during the last several years. While these systems still operate in fairly limited domains, they can produce useful structured information from text with reasonable accuracy. Volumes of text information, low cost computing power, and scarce labor resources increase the motivation for using computers to manage information. The question that stands in the way of the widespread installation of text processing systems is "Are they good enough yet?"

Performance is a major issue in the evolution of text processing systems from "toy" research problems to real applications. In spite of the rapid advances in computing technology, most text processing systems are simply too slow, because applications and thorough testing both demand higher throughput. Excess grammatical complexity clearly accounts for much of the performance shortfall. Most computational methods of analysis overlook the human-like capability to "skim" texts, limiting computation to sections of importance.

Unfortunately, the "skimming" approach has come to be identified with systems that abandon grammatical *knowledge* as well as syntactic processing, thus sacrificing accuracy and other performance features. The ideal skimming system would need no less linguistic knowledge than a full syntactic parser; in fact, knowing what *not* to parse may require as much linguistic information as parsing itself. Our approach to skimming is *relation-driven*: the program makes a low-cost first pass through the texts, determining what conceptual relations could be relevant. Then it segments the text to limit grammatical analysis to sections that affect those conceptual relations. In sections of text with high information content, the program still performs a complete analysis. In other cases, the skimming technique leads to three types of performance improvements. The following examples illustrate these features in the context of SCISOR, a system that reads news stories about corporate takeovers [Rau and Jacobs, 1988]:

1. *Skipping irrelevant text.*
   Example: The company said it expects 1989 daily output to average 10,500 barrels of oil and liquids.
   Effect: Do not parse the sentence.
2. *Limited processing of intervening phrases.*
   Example: Fidelity Federal Savings & Loan Association said its board held a meeting on Dec. 2 and declined a proposal to amend the acquisition agreement by BEI Holdings Ltd.
   Effect: Parse to determine the roles of "declined" and "acquisition" only.
3. *Limited attachment.*
   Example: Revere said it had received an offer from an investor group to be acquired for $16 a share, or about $127 million.
   Effect: Break the sentence into three sections to limit complexity.

In each of the above examples, relation-driven skimming limits processing by concentrating on information that is necessary to satisfy the information requirements of the program, such as identifying the target and suitor of a takeover. In the first case, the program can skip an entire sentence because it does not contribute at all. In the second example, the sentence contains relevant information, but one clause (about the meeting of the board) does not really contribute. In the third example, the sentence is packed with relevant material, but the program can still limit grammatical complexity by discarding alternatives that do not contribute to semantic processing.

Relation-driven skimming is part of our text processing system, which has been applied to several prototype domains. The first accomplishment of the algorithm was to deliver a factor of six improvement in performance to SCISOR, which reads Dow Jones

financial news stories at a rate of over 500 per hour. The same skimming program applied successfully to an evaluation set of naval operations reports used in the Message Understanding Conference (MUCK-II) held in San Diego in June, 1989 [Sundheim, 1990]. This paper will describe the relation-driven skimming algorithm and discuss the sort of performance improvements that can be expected from this approach.

2 The Text Extraction Task and the Skimming Problem

Skimming is of little use if the processing task demands a complete analysis. The skimming method presented here is aimed at the task of information extraction from text, where the program looks for relatively superficial facts that appear in texts with constrained content. Since the program is looking only for certain key information, it should spend most of its time analyzing sections that contain that information. The more extraneous information there is in a text, the more skimming improves performance over full parsing.

The skimming problem assumes that a text processing program works from set of predefined conceptual roles that represent some of the information from a text. These conceptual roles form a "template" that the program fills in while scanning a text. The words "relevant", "important", and "extraneous" used throughout this paper describe the relationship of portions of text to this template-filling task.

For example, the following is a typical Dow Jones new story, along with the template structure representing the information extracted by SCISOR:

Input Text:

> Brunswick Corp Up; Active amid Continued Takeover Talk
>
> New York -DJ- Traders and market sources say shares of Brunswick Corp., the world's largest manufacturer of recreational boats, are trading actively for the second consecutive week amid growing speculation that someone is accumulating a position in the company. Brunswick is up 5-8 at 20 on NYSE-composite voluime (sic) of 1,160,400 shares, compared with an average daily volume of 435,200 shares. The stock rose 1-2 yesterday on more than 1.1 million shares. In the past two weeks, Brunswick shares have traded above average daily volume on all but one day, inching up from a low of 16 3-4 on March 14. The most often rumored suitor for Brunswick is Minneapolis investor and boat company owner Irwin Jacobs, whose name surfaced about a year ago when Brunswick shares made a similar move on unfounded speculation. One trader tells Dow Jones Professional Investor Report a New Jersey-based "tape reading" service today named the stock as the target of a $30-a-share bid from Jacobs. The service can't be reached for confirmation. A secretary to Jacobs said late yesterday he'll be out of his office until Friday. Jacobs has become a popular rumored shark since walking away with cash from his failed bid earlier this month for Shaklee Corp.. He's also one of several rumored suitors for NWA Inc.

Template Structure Produced:

Corporate-Takeover-Core
Event: Rumor
Suitor: Jacobs
Target: Brunswick Corp.
Per-share-price: $30 *[price of the offer]*
Effect-of-rumor: Up 5/8

The system could obtain the same minimal information if the text read as follows:

> Brunswick is up 5-8 at 20....rumored....the target of a $30-a-share bid from Jacobs.

The objective of skimming is to read the text as if it were closer to this condensed form. The problem for text skimming is thus (1) to identify sections of text that will contribute to information extraction, and (2) to limit processing in those sections. The introduction outlined three different types of performance improvements that come from skimming. The next section describes the relation-driven method and how it achieves these improvements.

3 Relation-Driven Skimming

Relation-driven skimming takes advantage of the theory that most conceptual information derives from linguistic relations that do not depend on a complete surface structure. Such relations, like *subject-predicate* and *verb-complement*, carry constraints such as agreement or selectional restrictions. Since the bulk of the complexity of most language analyzers comes from the combinatorics of parsing, finding relations without a complete syntactic analysis helps performance.

The relation-driven skimming algorithm has three components:

- *The concept activation component* makes a first pass through the text and selects candidate concepts that may contribute to its semantic interpretation.
- *The segmentation component* tells the program what to parse, what to skip, and where to use semantic information for attachment.
- *The attachment component* identifies linguistic relations in the input text that contribute to its semantic interpretation, even where segments have been skipped.

The trick to relation-driven skimming is to perform attachment as accurately as possible with as little grammatical analysis as possible. This is no simple task, because phrases with no relevant semantic content can always affect the attachment of relevant phrases. In the sections that follow, we will give for each of the components above an observation of why it works, its main activity, and an example or two of its operation.

3.1 Concept Activation

Concept activation uses lexical analysis of words, combinations, and other features to determine

whether a portion of text is likely to be relevant. The concept activation component makes a single pass through the input text, producing a sequence of conceptual categories that may contribute to the conceptual interpretation.

Observation: The density of relevant content words in a section of text generally determines the degree of processing required for semantic analysis.

Activity: Divide content words into two categories: "triggers", or relation heads, and role fillers. Scan the text using domain knowledge for words or combinations that might be triggers, and for words or combinations that might be fillers.

Example: The following is the input text and output of concept activation for the Revere example:

> **Input:** Revere said it had received an offer from an investor group to be acquired for $16 a share, or about $127 million.
>
> **Output:** (Company) (receive-offer) (investor-group) (acquire) (dollar) (number) (share) (dollar) (number).

This process is more than a lexical lookup. Some words, like *rumor* or *target* in the corporate takeover stories, are indeed "triggers" directly associated with important concepts. However, considering all words that *might* contribute to an important concept is inefficient; words such as *make, take, issue* or *increase* require more analysis of the surrounding context. In these cases, the skimmer looks for combinations of words or concepts (such as *received* and *offer* above). This prevents the parser from doing a lot of processing around low-content words.

Whether a word is contentful or not depends on context. Some words, like *plan*, do not themselves carry much information but must be understood because they distinguish the agent of another action. "Acme rejects an offer" and "Acme plans an offer" place Acme in different roles (i.e. the *target* and *suitor*, respectively). A concept like *plan*, therefore, appears in the list of activated concepts only when there are takeover events in the local context.

Concept activation eliminates processing of unimportant sentences and clauses and helps efficiency in contentful sections, mainly by determining relations (such as role-filler) that help to guide syntactic analysis. The next phase, text segmentation, uses the results of concept activation to control parsing.

3.2 Text Segmentation

The text segmentation phase groups the text around words that are concept activators, identifying noun groups and complement structures after verbs, and finding punctuation or words that separate segments of text. This phase determines (1) where to skip and (2) where to limit parsing.

Observation: It is generally possible to reconstuct the important relations of a text in spite of skipping over intervening words and phrases.

Activity: Skip over empty sentences and phrases, and break the combinatorics of parsing where a single parse will do.

Example: In the Revere example, the segmented (and marked) text is as follows:

> Revere *skip* it had received an offer from an investor group *break* to be acquired for $16 a share *break* *skip* $127 million.

The **skip** token indicates to the parser that there is intervening information, while the **break** token indicates that it should "reduce" or complete all active linguistic structures. Both help to limit complexity—skipping tends to avoid wasted parsing as well as the combinatorics of attachment, while the breaks help to avoid considering multiple attachments where syntax contributes little or no information.

The segmentation algorithm includes most important noun phrases, even when separation information prevents them from being attached. This is because, as in the above example, these noun phrases implicitly play a role in anaphoric references or infinitive phrases.

A side effect of text segmentation is to mark the original text, highlighting sections that are considered relevant. This has a dual effect: (1) It helps to debug the skimming algorithm by showing visually what sections of the text the program has read, and (2) It allows the users of the program quickly to spot key information.

For example, a typical merger & acquisition story from the Dow Jones examples will appear with relevant sections in boldface, as shown below:

> **Mayfair Gets Buyout Proposal**
>
> **Mayfair Super Markets Inc.** said that **Stanley P. Kaufelt**, its chairman, president and chief executive, has proposed a business combination with **Mayfair** in which **the holders of Mayfair's outstanding common stock would receive $23.50 a share** in cash.

Text segmentation confines processing to sections of text that contain important information. The correct semantic interpretation of these text sections often depends on correct syntactic attachment, as described below.

3.3 Attachment

The attachment phase produces linguistic relations from the segmented text. This phase is part of the bottom-up parsing process; the main difference between attachment and full parsing is that the parser must attempt to form linguistic relations where it has skipped sections of text. Attachment in the absence of a complete parse relies on rules that combine linguistic and conceptual information; for example, "attach a verb phrase or infinitive to the most recent clause-level semanticly valid noun phrase".

Observation: Attachments by default are much less costly computationally than attachments by exhaustive consideration of possibilities.

Activity: Prefer attachments within boundaries separated by breaks, and use semantics and recency to guide attachment otherwise.

Example: In the Revere example, the following relations guide the interpretation process:

> Revere received an offer... (NP-VP)
> an offer from an investment group (NP-PP)

Revere to be acquired... (NP-INFPHR)
acquired for $16 a share (VP-PP)
$160 million (NP)

The combination of these simple relations permits the correct semantic interpretation without a complete parse (see [Rau and Jacobs, 1988] for a discussion of the use of these relations for analysis). In this example, the skimming algorithm reduces the number of parses considered by a factor of six. This is in spite of the fact that the Revere sentence contains a fair amount of useful information; in less dense text the skimming program can skip sentences entirely or extract only two or three relations from a complex sentence (as in the Fidelity example given in the introduction).

The Revere example is more complex than most of the cases that occur in these texts because it illustrates a number of interacting rules and preferences. It is, however, unusual in that full syntactic parsing of this example could be misleading because it would tend to attach "to be acquired" to "investor group" rather than "Revere". The point of this example is *not*, however, that full syntactic processing is *bad*, but rather that skimming can make the necessary attachments without full parsing.

Complications of Limited Attachment

The attachment mechanism makes use of several heuristics for constructing relations from the text, such as the infinitive phrase rule given earlier, resolving references before attaching pronouns, reconstructing sentences from verb phrases in incomplete sentences, and determining voice before attaching conjunctive verb phrases. These rules have derived from the analysis of fairly large bodies of text. The following are some observations about the sorts of examples where "limited attachment" is necessary:

- *Dangling Phrases.* In many longer texts, prepositional phrases, infinitives, and other adjunct information "hang off" the ends of sentences. Typically, such phrases can be attached syntactically to multiple heads. The effect of the skimming algorithm is to give more weight to the semantic attachment of these phrases. Since many such examples contain temporal, spatial, or other information associated with events, this semantic attachment seems to provide an advantage over syntactic preferences.
- *Conjunctive Clauses.* Conjunctions introduce linguistic complexity. If only part of a conjunctive clause contains useful information, the program can identify a relation involving one portion of the coordinated clause without parsing the whole sentence. For example, one news story reads "Investor William Farley...said he plans to seek a special meeting to discuss his proposal and to wage a proxy fight for control of the board". Only the second clause contains useful information, although the first clause can help to attach the second.
- *Negative Information.* The skimming algorithm, in its application of linguistic relations, must use both positive and negative information in determining where to attach phrases. Lack of agreement, for example, can override the attachment of a verb phrase to a semanticly valid subject. Case constraints often guide the analysis of pronouns. Semantic information tends to provide positive information in these cases, while syntactic information provides negative information. Oddly, this is the reverse of the more typical parsing strategy of using semantics to filter out invalid interpretations.

It might seem that these complications present enough problems that it would be easier to perform full parsing than to try to derive new heuristics for attachment in all these examples. This is true in some cases, but the vast majority of examples we have encountered require only a few simple attachment preferences. In these "easier" examples, the performance payoff has been enough to keep us from degrading to full parsing whenever possible.

4 Comparison with Other Approaches

Most work in skimming or partial parsing [DeJong, 1979; Lebowitz, 1983; Lytinen and Gershman, 1986; Young and Hayes, 1985] uses template-based or memory-based strategies, effectively using conceptual information in place of linguistic constraints. This approach seems to work in highly constrained texts where conceptual knowledge is sufficient for determining role relationships. In the domains that we have tested, the pure template-based approach fails because some role relationships are determined almost entirely from linguistic information such as complement structure or agreement. For example, the target and suitor of corporate mergers are both companies; thus there is little conceptual information (other than the size of the companies) that helps to determine role-filling. In many classes of tactical operations reports, the agent and object are both military forces, thus correct linguistic attachment is essential in this domain as well.

Although the overall parsing style of our system integrates template-based and language-based strategies [Rau and Jacobs, 1988], the skimming algorithm is actually more bottom-up or language-based. Like some of the other major text processing systems such as PROTEUS, PUNDIT, and TACITUS [Hobbs, 1986; Grishman and Hirschman, 1986], the skimming program applies linguistic constraints and maps linguistic structures into conceptual roles. In these other systems, however, the bottom-up approach may cause the program to waste time on irrelevant sections of text. The difference is that these programs do not really use conceptual information until after the parser has generated its candidate structures. The relation-driven skimming process shortcuts bottom-up analysis by first using conceptual knowledge to block some fruitless paths. As we have had the benefit of comparing our system in some detail with these other programs after operation on a common task, we believe that many such systems could achieve an order of magnitude improvement in processing speed by incorporating a similar method.

5 System Status and Current Directions

The SCISOR system [Rau and Jacobs, 1988] was the initial testbed for this algorithm, is a completed prototype that reads news stories at the rate of about 500 per hour. It extracts certain key information from stories about corporate takeovers (typically about 10% of the texts), identifying target, suitor, purchase price, and other information with about 90% accuracy.

The generic text processing components of SCISOR, known as the GE NLToolset [Jacobs and Rau, 1990], are used in applications in the operations of GE. Our group applied this core of text processing tools, including the skimming procedures described here, to the MUCK-II task, which consisted of generating database templates from naval operations messages, during a period of several weeks before the conference. The skimming algorithm of the NLToolset was the key to producing good results so rapidly. The same text processing system has since applied to a number of message sets in other domains.

The improvements in speed from skimming have so far come without a degradation in accuracy. This does not, however, mean that the attachment heuristics are infallible. Clearly, examples can occur where text that has been skipped influences the attachement of key phrases, especially when texts contain ellipsis, anaphoric references, and complex coordinated structures. Future enhancements to our algorithm must refine the attachment rules for these cases, and degrade to full parsing where necessary.

6 Conclusion

Natural language text processing has reached a point where efficiency is a real issue. While it might be possible to design fundamentally faster text processing algorithms, a more fruitful approach in the near term is to try to eliminate much of the wasted processing that is done in parsing text. This does not mean using less syntactic *knowledge*, but does mean less syntactic processing. Our approach uses the identification of linguistic relations as a driver for producing conceptual information while eliminating some of the detail of parsing. This approach has been successful in producing a program that operates almost an order of magnitude faster in multiple domains without major effect on the design or accuracy of the system.

References

[DeJong, 1979] Gerald DeJong. Skimming stories in real time: An experiment in integrated understanding. Research Report 158, Department of Computer Science, Yale University, 1979.

[Grishman and Hirschman, 1986] Ralph Grishman and Lynette Hirschman. PROTEUS and PUNDIT: Research in text understanding. PROTEUS Project Memorandum 1, NYU, 1986.

[Hobbs, 1986] Jerry R. Hobbs. Site report: Overview of the TACITUS project. *Computational Linguistics*, 12(3):220–222, 1986.

[Jacobs and Rau, 1990] Paul S. Jacobs and Lisa F. Rau. The GE NLToolset: A software foundation for intelligence text processing. In *Proceedings of the Thirteenth International Conference on Computational Linguistics*, Helsinki, Finland, 1990.

[Lebowitz, 1983] M. Lebowitz. Memory-based parsing. *Artificial Intelligence*, 21(4), 1983.

[Lytinen and Gershman, 1986] Steven Lytinen and Anatole Gershman. ATRANS: Automatic processing of money transfer messages. In *Proceedings of the Fifth National Conference on Artificial Intelligence*, Philadelphia, 1986.

[Rau and Jacobs, 1988] Lisa F. Rau and Paul S. Jacobs. Integrating top-down and bottom-up strategies in a text processing system. In *Proceedings of Second Conference on Applied Natural Language Processing*, pages 129–135, Morristown, NJ, Feb 1988. ACL.

[Sundheim, 1990] Beth Sundheim. Second message understanding conference (MUCK-II) test report. Technical Report 1328, Naval Ocean Systems Center, San Diego, CA, 1990.

[Young and Hayes, 1985] S. Young and P. Hayes. Automatic classification and summarization of banking telexes. In *The Second Conference on Artificial Intelligence Applications*, pages 402–208. IEEE Press, 1985.