Motivations and Methods for Text Simplification

R. Chandrasekar*
Institute for Research in Cognitive Science & Center for the Advanced Study of India

Christine Doran
Department of Linguistics

B. Srinivas
Department of Computer & Information Science

University of Pennsylvania, Philadelphia, PA 19104
{mickeyc,cdoran,srini}@linc.cis.upenn.edu

Abstract

Long and complicated sentences prove to be a stumbling block for current systems relying on NL input. These systems stand to gain from methods that syntactically simplify such sentences. To simplify a sentence, we need an idea of the structure of the sentence, to identify the components to be separated out. Obviously a parser could be used to obtain the complete structure of the sentence. However, full parsing is slow and prone to failure, especially on complex sentences. In this paper, we consider two alternatives to full parsing which could be used for simplification. The first approach uses a Finite State Grammar (FSG) to produce noun and verb groups while the second uses a Supertagging model to produce dependency linkages. We discuss the impact of these two input representations on the simplification process.

1 Reasons for Text Simplification

Long and complicated sentences prove to be a stumbling block for current systems which rely on natural language input. These systems stand to gain from methods that preprocess such sentences so as to make them simpler. Consider, for example, the following sentence:

(1) *The embattled Major government survived a crucial vote on coal pits closure as its last-minute concessions curbed the extent of Tory revolt over an issue that generated unusual heat in the House of Commons and brought the miners to London streets.*

Such sentences are not uncommon in newswire texts. Compare this with the multi-sentence version which has been manually simplified:

(2) *The embattled Major government survived a crucial vote on coal pits closure. Its last-minute concessions curbed the extent of Tory revolt over the coal-mine issue. This issue generated unusual heat in the House of Commons. It also brought the miners to London streets.*

If complex text can be made simpler, sentences become easier to process, both for programs and humans. We discuss a simplification process which identifies components of a sentence that may be separated out, and transforms each of these into free-standing simpler sentences. Clearly, some nuances of meaning from the original text may be lost in the simplification process. Simplification is therefore inappropriate for texts (such as legal documents) where it is important not to lose any nuance. However, one can conceive of several areas of natural language processing where such simplification would be of great use. This is especially true in domains such as machine translation, which commonly have a manual post-processing stage, where semantic and pragmatic repairs may be carried out if necessary.

- Parsing: Syntactically complex sentences are likely to generate a large number of parses, and may cause parsers to fail altogether. Resolving ambiguities in attachment of constituents is non-trivial. This ambiguity is reduced for simpler sentences since they involve fewer constituents. Thus simpler sentences lead to faster parsing and less parse ambiguity. Once the parses for the simpler sentences are obtained, the subparses can be assembled to form a full parse, or left as is, depending on the application.
- Machine Translation (MT): As in the parsing case, simplification results in simpler sentential structures and reduced ambiguity. As argued in (Chandrasekar, 1994), this could lead to improvements in the quality of machine translation.
- Information Retrieval: IR systems usually retrieve large segments of texts of which only a part may be relevant. With simplified texts, it is possible to extract specific phrases or simple sentences of relevance in response to queries.

*On leave from the National Centre for Software Technology, Gulmohar Cross Road No. 9, Juhu, Bombay 400 049, India

- Summarization: With the overload of information that people face today, it would be very helpful to have text summarization tools that reduce large bodies of text to the salient minimum. Simplification can be used to weed out irrelevant text with greater precision, and thus aid in summarization.
- Clarity of Text: Assembly/use/maintenance manuals must be clear and simple to follow. Aircraft companies use a Simplified English for maintenance manuals precisely for this reason (Wojcik et al., 1993). However, it is not easy to create text in such an artificially constrained language. Automatic (or semi-automatic) simplification could be used to ensure that texts adhere to standards.

We view simplification as a two stage process. The first stage provides a structural representation for a sentence on which the second stage applies a sequence of rules to identify and extract the components that can be simplified. One could use a parser to obtain the complete structure of the sentence. If all the constituents of the sentence along with the dependency relations are given, simplification is straightforward. However, full parsing is slow and prone to failure, especially on complex sentences. To overcome the limitations of full parsers, researchers have adopted FSG based approaches to parsing (Abney, 1994; Hobbs et al., 1992; Grishman, 1995). These parsers are fast and reasonably robust; they produce sequences of noun and verb groups without any hierarchical structure. Section 3 discusses an FSG based approach to simplification. An alternative approach which is both fast and yields hierarchical structure is discussed in Section 4. In Section 5 we compare the two approaches, and address some general concerns for the simplification task in Section 6.

2 The Basics of Simplification

Text simplification uses the fact that complex texts typically contains complex syntax, some of which may be particular to specific domain of discourse, such as newswire texts. We assume that the simplification system will process one sentence at a time. Interactions across sentences will not be considered. We also assume that sentences have to be maximally simplified.

To simplify sentences, we need to know where we can split them. We define *articulation-points* to be those points at which sentences may be logically split. Possible articulation points include the beginnings and ends of phrases, punctuation marks, subordinating and coordinating conjunctions, and relative pronouns. These articulation points are general, and should apply to arbitrary English texts. These may, however, be augmented with domain-specific articulation points. We can use these articulation-points to define a set of rules which map from given sentence patterns to simpler sentences patterns. These rules are repeatedly applied on each sentence until they do not apply any more. For example, the sentence (3) with a relative clause can be simplified into two sentences (4).

(3) *Talwinder Singh, who masterminded the Kanishka crash in 1984, was killed in a fierce two-hour encounter ...*

(4) *Talwinder Singh was killed in a fierce two-hour encounter ... Talwinder Singh masterminded the Kanishka crash in 1984.*

3 FSG based Simplification

(Chandrasekar, 1994) discusses an approach that uses a FSG for text simplification as part of a machine aided translation prototype named Vaakya. In this approach, we consider sentences to be composed of sequence of word groups, or *chunks*. Chunk boundaries are regarded as potential articulation-points. Chunking allows us to define the syntax of a sentence and the structure of simplification rules at a coarser granularity, since we need no longer be concerned with the internal structure of the chunks.

In this approach, we first tag each word with its part-of-speech. Chunks are then identified using a FSG. Each chunk is a word group consisting of a verb phrase or a noun phrase, with some attached modifiers. The noun phrase recognizer also marks the number (singular/plural) of the phrase. The verb phrase recognizer provides some information on tense, voice and aspect. Chunks identified by this mechanism include phrases such as *the extent of Tory revolt* and *have recently been finalized*.

The chunked sentences are then simplified using a set of ordered simplification rules. The ordering of the rules is decided manually, to take care of more frequent transformations first, and to avoid unproductive rule interaction. An example rule that simplifies sentences with a relative pronoun is shown in (5).

(5) `X:NP, RelPron Y, Z → X:NP Z. X:NP Y.`

The rule is interpreted as follows. If a sentence starts with a noun phrase (`X:NP`), and is followed by a phrase with a relative pronoun, of the form (`, RelPron Y ,`) followed by some (`Z`), where `Y` and `Z` are arbitrary sequences of words, then the sentence may be simplified into two sentences, namely the sequence (`X`) followed by (`Z`), and (`X`) followed by (`Y`). The resulting sentences are then recursively simplified, to the extent possible.

The system has been tested on news text, and performs well on certain classes of sentences. See (Chandrasekar and Ramani, 1996) for details of quantitative evaluation of the system, including an evaluation of the acceptability of the resulting

sentences. A set of news stories, consisting of 224 sentences, was simplified by the prototype system, resulting in 369 simplified sentences.

However, there are certain weaknesses in this system, caused mostly by the relatively simple mechanisms used to detect phrases and attachments. Sentences which include long distance or crossed dependencies, and sentences which have multiply stacked appositives are not handled properly; nor are sentences with ambiguous or unclear attachments. Some of these problems can be handled by augmenting the rule set but what is really required is more syntactic firepower.

4 A Dependency-based model

A second approach to simplification is to use richer syntactic information, in terms of both constituency information and dependency information. We use partial parsing and simple dependency attachment techniques as an alternative to the FSG based simplification. This model (the DSM) is based on a simple dependency representation provided by Lexicalized Tree Adjoining Grammar (LTAG) and uses the "supertagging" techniques described in (Joshi and Srinivas, 1994).

4.1 Brief Overview of LTAGs

The primitive elements of LTAG formalism are **elementary trees**. Elementary trees are of two types: *initial trees* and *auxiliary trees*. Initial trees are minimal linguistic structures that contain no recursion, such as simple sentences, NPs, PPs etc. Auxiliary trees are recursive structures which represent constituents that are adjuncts to basic structure (e.g. relative clauses, sentential adjuncts, adverbials). For a more formal and detailed description of LTAGs see (Schabes et al., 1988).

4.2 Supertagging

The elementary trees of LTAG localize dependencies, including long distance dependencies, by requiring that all and only the dependent elements be present within the same tree. As a result of this localization, a lexical item may be (and almost always is) associated with more than one elementary tree. We call these elementary trees *supertags*, since they contain more information (such as subcategorization and agreement information) than standard part-of-speech tags. Hence, each word is associated with more than one supertag. At the end of a complete parse, each word is associated with just one supertag (assuming there is no global ambiguity), and the supertags of all the words in a sentence are combined by substitution and adjunction.

As in standard part-of-speech disambiguation, we can use local statistical information in the form of N-gram models based on the distribution of supertags in a LTAG parsed corpus for disambiguation. We use a trigram model to disambiguate the supertags so as to assign one supertag for each word, in a process termed supertagging. The trigram model of supertagging is very efficient (in linear time) and robust (Joshi and Srinivas, 1994).

To establish the dependency links among the words of the sentence, we exploit the dependency information present in the supertags. Each supertag associated with a word allocates slots for the arguments of the word. These slots have a polarity value reflecting their orientation with respect to the anchor of the supertag. Also associated with a supertag is a list of internal nodes (including the root node) that appear in the supertag. Using this information, a simple algorithm may be used to annotate the sentence with dependency links.

4.3 Simplification with Dependency links

The output provided by the dependency analyzer not only contains dependency links among words but also indicates the constituent structure as encoded by supertags. The constituent information is used to identify whether a supertag contains a clausal constituent and the dependency links are used to identify the span of the clause. Thus, embedded clauses can easily be identified and extracted, along with their arguments. Punctuation can be used to identify constituents such as appositives which can also be separated out. As with the finite-state approach, the resulting segments may be incomplete as independent clauses. If the segments are to be reassembled, no further processing need be done on them.

Figure 1 shows a rule for extracting relative clauses, in dependency notation. We first identify the relative clause tree (Z), and then extract the verb which anchors it along with all of its dependents. The right hand side shows the two resulting trees. The gap in the relative clause (Y) need only be filled if the clauses are not going to be recombined. Examples (6) and (7) show a sentence before and after this rule has applied.

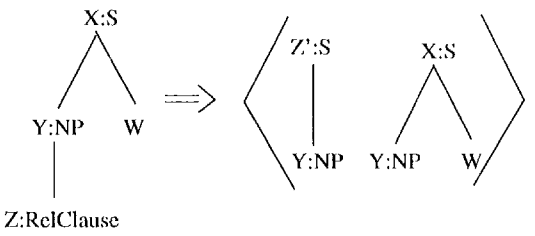

Figure 1: Rule for extracting relative clauses

(6) ... an issue [that **generated** unusual heat in the House of Commons] ...

(7) An issue [**generated** unusual heat in the House of Commons]. The issue ...

5 Evaluation

The objective of the evaluation is to examine the advantages of the DSM over the FSG-based model for simplification. In the FSG approach since the input to the simplifier is a set of noun and verb groups, the rules for the simplifier have to identify basic predicate argument relations to ensure that the right chunks remain together in the output. The simplifier in the DSM has access to information about argument structure, which makes it much easier to specify simplification patterns involving complete constituents. Consider example 8,

(8) *The creator of Air India, Mr. JRD Tata, believes that the airline, which celebrated 60 years today, could return to its old days of glory.*

The FSG-based model fails to recognize the relative clause on the embedded subject *the airline* in example (8), because Rule 5 looks for matrix subject NPs. On the other hand, the DSM correctly identifies the relative clause using the rule shown in Figure 1, which holds for relative clauses in all positions.

Other differences are in the areas of modifier attachment and rule generality. In contrast to the DSM approach, the FSG output does not have all modifiers attached, so the bulk of attachment decisions must be made by the simplification rules. The FSG approach is forced to enumerate all possible variants of the LHS of each simplification rule (eg. Subject versus Object relatives, singular versus plural NPs) whereas in the DSM approach, the rules, encoded in supertags and the associated constituent types, are more general.

Preliminary results using the DSM model are very promising. Using a corpus of newswire data, and only considering relative clause and appositive simplification, we correctly recovered 25 out of 28 relative clauses and 14 of 14 appositives. We generated 1 spurious relative clause and 2 spurious appositives. A version of the FSG model on the same data recovered 17 relative clauses and 3 appositives.

6 Discussion

Simplification can be used for two general classes of tasks. The first is as a preprocessor to a full parser so as to reduce the parse ambiguity for the parser. The second class of tasks demands that the output of the simplifier be free-standing sentences. Maintaining the coherence of the simplified text raises the following problems:

- Determining the relative order of the simplified sentences, which impacts the choice of referring expressions to be used and the overall coherence of the text.
- Choosing referring expressions: For instance, when separating relative clauses from the nouns they modify, copying the head noun into the relative clause is simple, but leads to quite awkward sounding texts. However, choosing an appropriate pronoun or choosing between definite and indefinite NPs involves knowledge of complex discourse information.
- Selecting the right tense when creating new sentences presents similar problems.
- No matter how sophisticated the simplification heuristics, the subtleties of meaning intended by the author may be diluted, if not lost altogether. For many computer applications, this disadvantage is outweighed by the advantages of simplification (i.e. gains of speed and/or accuracy), or may be corrected with the use of human post-processing.

Acknowledgements

This work is partially supported by NSF grant NSF-STC SBR 8920230, ARPA grant N00014-94 and ARO grant DAAH04-94-G0426.

References

Steven Abney. 1994. Dependency Grammars and Context-Free Grammars. Manuscript, University of Tubingen, March.

R. Chandrasekar and S. Ramani. 1996. Automatic Simplification of Natural Language Text. Manuscript, National Centre for Software Technology, Bombay.

R. Chandrasekar. 1994. *A Hybrid Approach to Machine Translation using Man Machine Communication.* Ph.D. thesis, Tata Institute of Fundamental Research/University of Bombay, Bombay.

Ralph Grishman. 1995. Where's the Syntax? The New York University MUC-6 System. In *Proceedings of the Sixth Message Understanding Conference*, Columbia, Maryland.

Jerry Hobbs, Doug Appelt, John Bear, David Israel, and W. Mary Tyson. 1992. FASTUS: a system for extracting information from natural language text. Technical Report 519, SRI.

Aravind K. Joshi and B. Srinivas. 1994. Disambiguation of Super Parts of Speech (or Supertags): Almost Parsing. In *Proceedings of the* 17^{th} *International Conference on Computational Linguistics (COLING '94)*, Kyoto, Japan, August.

Yves Schabes, Anne Abeillé, and Aravind K. Joshi. 1988. Parsing strategies with 'lexicalized' grammars: Application to Tree Adjoining Grammars. In *Proceedings of the* 12^{th} *International Conference on Computational Linguistics (COLING'88)*, Budapest, Hungary, August.

Richard H. Wojcik, Philip Harrison, and John Bremer. 1993. Using bracketed parses to evaluate a grammar checking application. In *Proceedings of the* 31^{st} *Conference of Association of Computational Linguistics, Ohio State University, Columbus, Ohio.*