# Automatic Induction of Rules for Text Simplification

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

**B. Srinivas**
Department of Computer & Information Science

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

**Abstract**

Long and complicated sentences pose various problems to many state-of-the-art natural language technologies. We have been exploring methods to automatically transform such sentences as to make them simpler. These methods involve the use of a rule-based system, driven by the syntax of the text in the domain of interest. Hand-crafting rules for every domain is time-consuming and impractical. This paper describes an algorithm and an implementation by which generalized rules for simplification are automatically induced from annotated training material with a novel partial parsing technique which combines constituent structure and dependency information. This algorithm described in the paper employs example-based generalizations on linguistically-motivated structures.

## 1 The Need for Text Simplification

Long and complicated sentences pose various problems to many state-of-the-art natural language technologies. For example, in parsing, as sentences become syntactically more complex, the number of parses increases, and there is a greater likelihood for an incorrect parse. In machine translation, complex sentences lead to increased ambiguity and potentially unsatisfactory translations. Complicated sentences can also lead to confusion in assembly/use/maintenance manuals for complex equipment.

We have been exploring methods to automatically simplify such sentences [Chandrasekar, 1994], [Chandrasekar et al, 1996]. Consider, for example, the following sentence:

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

(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 an equivalent (manually) simplified multi-sentence version:

(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.*

Most of the problems listed above are either eliminated or substantially reduced for the simplified version shown in (2). For instance, simpler sentences have fewer constituents, hence fewer ambiguities in identifying attachments and thus are parsed faster. Simplification would also be of great use in several areas of natural language processing such as machine translation, information retrieval and in applications where clarity of text is imperative. Of course, one may lose some nuances of meaning from the original text in the simplification process.

There has been interest in simplified English from companies such as Boeing and Xerox. Researchers at Boeing [Hoard et al, 1992], [Wojcik et al, 1993] have developed a Simplified English Checker. However, their focus is on carefully constraining the use of words in a specific domain, and in providing a tool to authors of machine maintenance/operation manuals to help them adhere to guidelines aimed at clear written communication. In contrast, our aim is to develop a system to (semi-)automatically simplify text from any domain.

The following is the outline of this paper. In Section 2, we present an architecture for simplification. The method used for analysis of input is discussed in Section 3 and the notion of supertags is outlined. In Section 4, we describe a method by which generalized rules are automatically induced from annotated training material of newspaper text in English. We discuss some of the issues pertaining to simplification in Section 5.

## 2 The Architecture of Simplification

Our simplification system processes one sentence at a time. Discourse related issues are not considered. We view simplification as a two stage process: analysis followed by transformation. The analysis stage provides a structural description of the input, and the transformation stage uses this representation for simplification.

The most obvious choice for the analysis stage is to use a full parser to obtain the complete structure of a sentence. If all the constituents of the sentence along with the dependency relations are given, simplification is very straightforward. However, it is well-known that, as sentences become syntactically more complex, the number of parses increases, and there is a greater likelihood for an incorrect parse.

We have discussed two alternative approaches to analyzing text, using a finite state grammar approach [Chandrasekar, 1994] and a dependency based approach [Chandrasekar et al, 1996]. We summarize the dependency based approach in the next section. Note that this approach is different from a full parsing approach in that a complete constituent structure is not created.

We define *articulation-points* to be those points where sentences may be split for simplification. Segments of a sentence between two articulation points may be extracted as simplified sentences. The nature of the segments delineated by the articulation points depends on the type of the structural analysis performed. If the sentences are viewed as linear strings of words, we could define articulation points to be, say, punctuation marks. If the words in the input are also tagged with part of speech information, we can split sentences based on the category information, for instance at relative pronouns. With part of speech information, subordinating and coordinating conjunctions may also be detected and used as articulation points. However, with just this information, the span of the subordinating/coordinating clause would be difficult to determine. On the other hand, if the sentence is annotated with phrasal bracketings, the beginnings and ends of phrases could also be articulation points.

For example, the sentence (3) with a relative clause, annotated with phrasal bracketing, can be simplified into two sentences as shown in (4), using a rule such as the one shown in (5) that relies on skeletal phrasal structure and punctuation information.

(3) *[Talwinder Singh]:NP, who:RelPron masterminded:V [the 1984 Kanishka crash]:NP, [was killed]:V [in [a fierce two-hour encounter]:NP]:PP.*

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

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

The rule is interpreted as follows. If a sentence starts with some segment W and a noun phrase (`X:NP`), and is then followed by a phrase 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 (`W X`) followed by (`Z`), and the sequence (`X`) followed by (`Y`).

However, the rule shown above does not handle reduced relatives, such as the one in sentence (6).

(6) *[The creator of Air India, Mr. JRD Tata]:NP, [believes]:V [that]:COMP [the airline]:NP, [known]:V [for [its on-board service]:NP]:PP, [could return]:V [to [its old days of glory]:NP]:PP.*

To solve such problems, we use a representation which combines dependency information with constituent structure, providing attachment and scope information. This representation is described in the next section.

We need a variety of rules to simplify text from any particular domain. However, hand-crafting simplification rules is time-consuming and not very practical. While some of the rules are likely to be common across domains, several are likely to be domain-specific. We ideally need a method to develop rules which can be easily induced for a new domain. In this paper, we present an algorithm and an implementation to automatically induce rules for simplification given an annotated aligned corpus of complex and simple text.

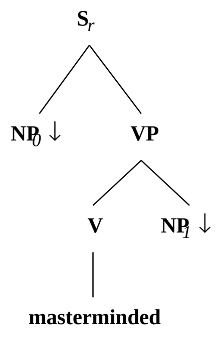
In addition to developing rules, we need gap-filling routines. For example, if we separate a relative clause from a sentence (for example (4)), we must insert a copy of the head noun at the gap in the relative clause. The exact choice of the gap fillers is a complicated task based on a variety of pragmatic factors, and will not be discussed in this paper.

## 3 Analysis of Input

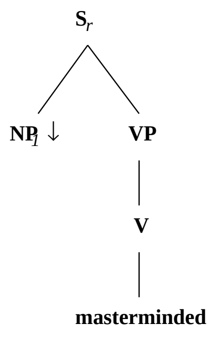
Our approach to the analysis stage of simplification uses rich syntactic information, that combines constituency and dependency information. We use partial parsing and simple dependency attachment techniques for fast and robust parsing. This model is based on a simple dependency representation provided by Lexicalized Tree Adjoining Grammar (LTAG) [Joshi, 1985], [Schabes et al, 1988] and uses the "supertagging" techniques described in [Joshi and Srinivas, 1994].

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 is associated with more than one elementary tree. The example in Figure 1 shows a selection of the elementary trees associated with the word "masterminded".

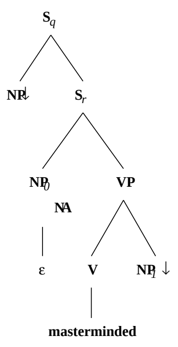
These elementary trees are called *supertags*, since they contain more information (such as subcategorization, agreement information) than standard



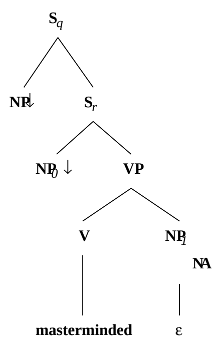
(a) Transitive Supertag



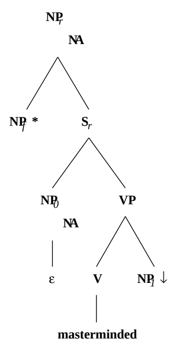
(b) Passive Supertag



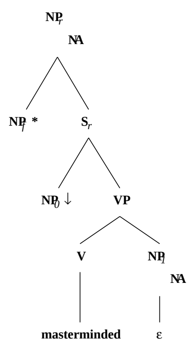
(c) Subject Extraction Supertag



(d) Object Extraction Supertag



(e) Subject Relative Supertag



(f) Object Relative Supertag

Figure 1: A selection of the supertags associated with the word *masterminded*

part-of-speech tags. Each word of an input sentence is initially associated with many such supertags. In a complete parse, each word would be associated with just one supertag (assuming there is no global ambiguity). The supertags for all the words in the sentence are combined by the operations used in LTAG, namely, substitution and adjunction.

Instead of relying on parsing to disambiguate supertags, we can use local statistical information in the form of N-gram models [Church, 1988] based on the distribution of supertags in a LTAG parsed corpus. Further, using the information coded in supertags, such as subcategorization and dependency information, we have implemented a system, a Lightweight Dependency Analyzer (LDA) [Srinivas et al, 1996], to heuristically determine the constituent structure and dependencies between constituents. For the purpose of simplification, the constituent information is used to determine 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 located and extracted, along with their arguments. Punctuation can be used to identify constituents such as appositives which can also be separated out.

## 4 Induction of Rules for Simplification

Our approach to automatically inducing rules from training data is described in this section. The training data is an aligned text corpus that links complex sentences to corresponding simplified sentences. This data are analyzed using LDA, and simplification rules are induced which are subsequently generalized using techniques similar to those used in Explanation Based Learning.

The training procedure for rule induction is detailed below, and illustrated with a running example.

1. The training data consists of a set of input sentences (such as (7)) along with a set of equivalent manually simplified sentences (such as (8)) corresponding to each of the input sentences.

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

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

2. The sentences in the training data are first processed to identify phrases that denote names of people, names of places or designations. These phrases are converted effectively to single lexical items.

3. Each training sentence $S_i$, along with its associated $j$ (simplified) sentences $S_{i1}$ to $S_{ij}$, is then processed using the Lightweight Dependency Analyzer (LDA).

4. The resulting dependency representations of $S_i$ and $S_{i1}$ through $S_{ij}$ are 'chunked'. Chunking collapses certain substructures of the dependency representation (noun phrases and verb groups) and allows us to define the syntax of a sentence at a coarser granularity. Chunking also makes the phrasal structure explicit, while maintaining dependency information. Thus, this approach has the benefit of both phrasal and dependency representations.

   The chunked LDA representation for the example sentence and its simplified version is illustrated in Figure 2. The nodes of this representation consist of word groups which are linked by dependency information. Each node is also associated with a supertag, such as the Subject Relative Supertag (Rel $\beta$) and the Transitive Supertag (Trans $\alpha$) in Figure 2.

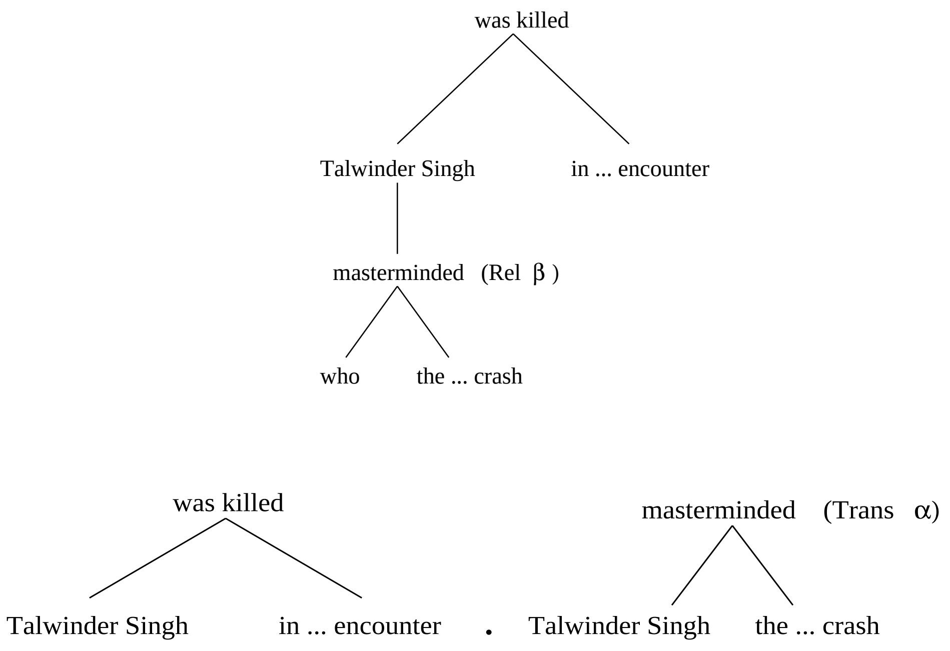


Figure 2: Chunked LDA representation of a complex sentence and its simplified versions

5. The chunked dependency representation of the complex sentence is compared with that of the simpler sentences using a tree-comparison algorithm.

This algorithm uses the immediate dominance (parent–child) relation and computes the tree-to-trees transformations required to convert $S_i$ to $S_{i1}$ through $S_{ij}$. The transformations include variables which are instantiated using a constraint satisfaction mechanism. The resulting rule is generalized, from the level of the specific words and word features in the sentences, to the supertags associated with each word. Recursive substructures are identified using supertag information, and marked as potentially repeating structures.

In our example, there are three changes between the complex and the simple versions:

(a) The Subject Relative Supertag (Rel $\beta$) changes to the Transitive Supertag (Trans $\alpha$).

(b) The head of the relative clause (represented by the parent of the Rel $\beta$ node in the LDA representation) is copied in place of the relative pronoun.[1]

(c) The Subject Relative Supertag (Rel $\beta$), and its dependents are separated out.

Note that the same rule will apply to all sentences which have relative clauses, regardless of the argument being relativized (subject/object/indirect object). Note also the level of generalization achieved already: it is not important if the verb in the relative clause is *masterminded* or not; the rule will apply to any verb which is associated with the subject-relative transitive supertag. In fact, it will be true of any morphological variant of the verb; so verbs such as *masterminds, mastermind* etc. will also show the same behaviour in a similar context.

6. All input sentences $S_i$ are processed using steps 2 through 4, and duplicate rules removed. This results in a set of generalized simplification rules.

7. Each rule is indexed on its articulation points, and stored appropriately. The articulation point defines the link (or edge) to be cut for simplification. For example, this rule is indexed on the Subject Relative Supertag (Rel $\beta$).

In the rule application phase, every new sentence is first analyzed using the LDA, and then chunked. Every node in the chunked LDA representation is a potential articulation point. The system retrieves all rules associated with the categories of these articulation points, and attempts to apply each of them. All rules that match the given structure are applied.

Consider the sentence shown in (9):

[1]Reduced relative clauses will have empty relative pronouns.

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

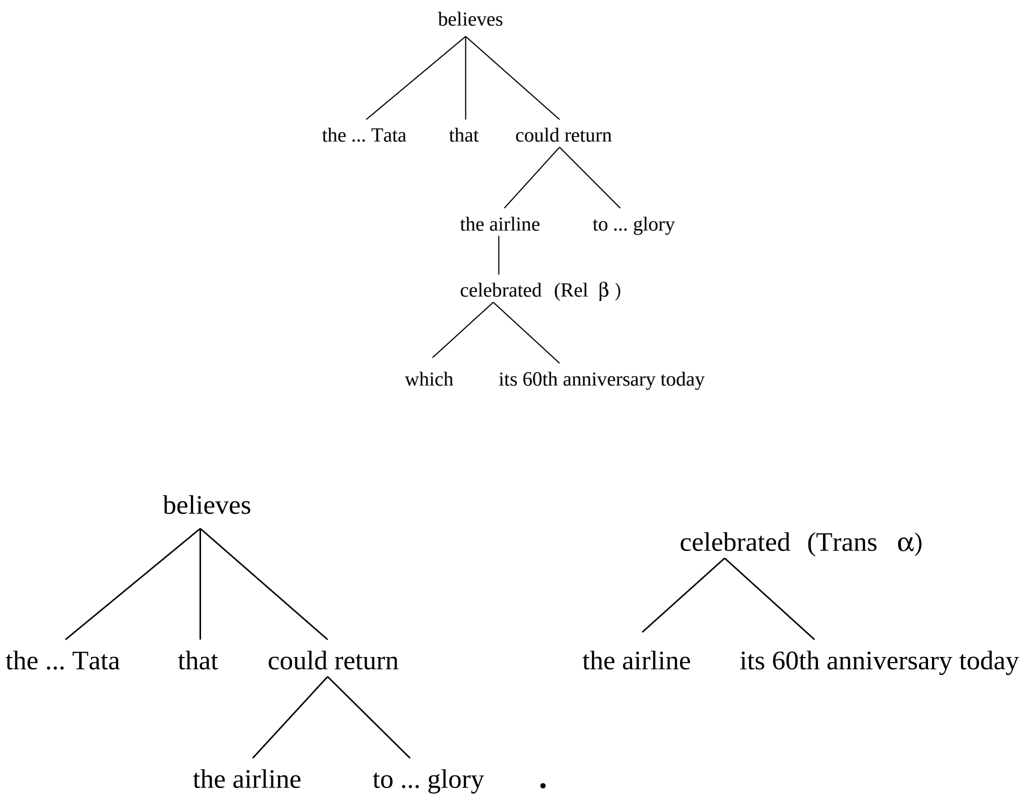


Figure 3: Chunked LDA representation of a complex sentence and its simplified versions

Figure 3 shows the chunked LDA representations of the original text and the result of applying the rule induced in the training phase. *Note that while the structure at the sentence level is significantly different from the training example, there is a similarity in the sub-structure, and the rule is applicable on this component.*

The training data for this system was culled from a set of sixty-five stories from a leading Indian newspaper, published in English. A simplified version of these stories was manually created. For the present, we have concentrated on simplifying sentences with relative clauses. We are extending this to handle other syntactic phenomena. The system has been coded as a series of interconnected PERL programs.

```
(a)

(who ,) (the_1984_Kanishka_crash masterminded) (masterminded who)
  ====>
(masterminded .) (masterminded the_1984_Kanishka_crash)

(b)

(B_COMPs B_PUs) (A_NXN B_N0nx0Vnx1) (B_N0nx0Vnx1 B_COMPs)
  ====>
(A_nx0Vnx1 B_sPU) (A_nx0Vnx1 A_NXN)
```

Figure 4: Example of an induced rule (a) before generalization and (b) after generalization.

An example rule induced by the program given the sentences in examples 7 and 8 is shown in Figure 4 before and after generalization. The tuples indicate parent–child relations. The terms on the LHS of the rule represent a conjunction of constraints which must be satisfied for the rule to fire. The generalized tags (B_COMPs, A_NXN etc.) are the appropriate supertags assigned to the words given the context of the sentences.

## 5 Discussion

In this paper, we have presented a novel approach to induce rules for simplification of text using the representation provided by supertags, which combines phrasal and dependency information in a uniform manner. Supertags localize all the dependencies of a word to one structure. As a result, the dependents of a word in the LDA representation appear as children of that word. The simplification rules that are induced operate on these localized representations, and have a local domain of influence. Therefore, these rules do not interact with each other with regard to their applicability. Also, the result of simplification is independent of the order of rule application.

As in many rule-based systems, hand-crafting rules is a time-consuming, tedious and error-prone process. An automated method of rule induction facilitates improved coverage of the system in terms of the phenomena handled, and the induction of rule sets for new domains with manageable effort. It provides us the opportunity to experiment with texts of different genres, and with a variety of preprocessing and post-processing software. In this work we have also integrated the transparency and interpretability afforded by rule-based representation with the robustness provided by the training process on corpora. We believe that this is an important advance in simplification.

There are several problems of interest in the area of simplification. For example, the ordering of simplified sentences, the choice of referring or gap-filling expressions, and the maintenance of discourse coherence as a whole deserve attention. Another aspect that deserves attention is the evaluation of simplification. We believe that the performance of simplification can be best evaluated in the context of an application where simplification is used as a component.

## Acknowledgments

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

## References

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

[Chandrasekar et al, 1996]
Chandrasekar R, Doran C and Srinivas B. *Motivations and Methods for Text Simplification*, Poster paper. In *Proceedings of the* 16$^{th}$ *International Conference on Computational Linguistics (COLING'96)*, Copenhagen, Sweden, August 1996.

[Church, 1988]
Church, KW. A Stochastic Parts Program and Noun Phrase Parser for Unrestricted Text. In *Proc. 2nd Applied Natural Language Processing Conference, Austin, Texas, 1988,* pp. 136–143.

[Hoard et al, 1992]
Hoard JE, Wojcik RH and Holzhauser KC. An automated grammar and style checker for writers of Simplified English, In PO Holt and N Williams (eds.), *Computers and Writing: State of the Art*, Kluwer, 1992.

[Joshi, 1985]
Joshi, AK. Tree Adjoining Grammars: How much context sensitivity is required to provide a reasonable structural description, In D Dowty, I Karttunen and A Zwicky (eds.), *Natural Language Parsing*, Cambridge University Press, Cambridge, UK, 1985.

[Joshi and Srinivas, 1994]
Joshi AK and Srinivas B. Disambiguation of Super Parts of Speech (or Supertags): Almost Parsing, In *Proceedings of the* 15th *International Conference on Computational Linguistics (COLING'94)*, Kyoto University, Japan, August 1994.

[Schabes et al, 1988]
Schabes Y, Abeillé A and Joshi AK. Parsing strategies with 'lexicalized' grammars: Application to tree adjoining grammars. In *Proceedings of the* 12th *International Conference on Computational Linguistics (COLING'88)*, Budapest, Hungary, August 1988.

[Srinivas et al, 1996]
Srinivas B, Doran C, Hockey BA and Joshi AK. An approach to Robust Partial Parsing and Evaluation Metrics, In *Proceedings of the Workshop on Robust Parsing at European Summer School in Logic, Language and Information*, Prague, August 1996.

[Wojcik et al, 1993]
Wojcik RH, Harrison P and Bremer J. Using bracketed parses to evaluate a grammar checking application. In *Proceedings of the* 31st *Annual Meeting of the Association for Computational Linguistics (ACL93)*, Ohio State University, Columbus, Ohio, 1993.