Automatic Rewriting for Controlled Language Translation

Teruko Mitamura and Eric Nyberg
Language Technologies Institute,
Carnegie Mellon University
5000 Forbes Ave. Pittsburgh, PA 15213, USA
teruko@cs.cmu.edu

Abstract

This paper summarizes the goals of controlled language authoring for machine translation, and discusses why automatic rewriting is an important research area for controlled authoring[1]. We present the KANT controlled language rewriting architecture, which combines a rewriting engine with interactive dialogue. We also present some rewriting examples which are based on the results of an empirical analysis of real texts. We conclude with a discussion of various open issues for the implementation and deployment of automatic rewriting systems.

1 Introduction

Natural language texts are often written using complex and ambiguous sentences, and humans as well as computers may experience difficulty in understanding and translating them. The use of Controlled Language (CL) has been introduced to encourage authors to write texts simply and consistently. Improving a text through the use of Controlled Language will also improve the quality of any translations of that text, whether done by humans or machines.

The KANT system (Knowledge-based, Accurate Natural-language Translation) (Mitamura, et al., 1991; Nyberg and Mitamura, 1996) is focused on the translation of technical documents written in a controlled language (Mitamura, 1999). KANT has been adapted for multilingual translation in various domains, including heavy equipment documentation, computer manuals, automotive manuals and medical record texts.

Although the use of controlled language has demonstrable benefits, using a controlled language can be cumbersome, especially when the author has difficulty deciding how to re-write an existing sentence to conform to the controlled language rules. A CL system should provide feedback to the user, in the form of alternate phrasings of an incorrect input, to guide the process and make it more efficient. For this reason, automatic rewriting of sentences is an important research topic for practical CL systems.

In this paper, we first introduce the field of Controlled Language and the process of controlled language checking. We then discuss automatic rewriting for machine translation, and present the KANT controlled language rewriting architecture. We also raise some outstanding issues regarding automatic rewriting, and discuss ongoing and future work.

2 Controlled Language Rewriting

A controlled language (CL) is a form of language usage restricted by grammar and vocabulary rules. It is important to note that there is no single standardized, controlled version of a particular language (e.g. "English Controlled Language"). In reality, it is necessary to develop different types of controlled language depending on the domain and purpose of the texts that you are working on, since these will vary in vocabulary and writ-

[1]Proceedings of the NLPRS2001 Workshop on Automatic Paraphrasing: Theories and Applications

ing style. CL can be used solely as a guideline for authoring, with a checking tool to verify conformance, or in conjunction with machine translation. In both situations, the overall goals are to achieve consistent authoring, to reduce ambiguity and complexity, and encourage clear and direct writing which can improve the quality of the source text and the translation output (Mitamura 1999).

We can also characterize CL as human-oriented or machine-oriented. When CL is written to improve text comprehension by machines, restrictions must be precisely defined for computation. For example, the restriction "Do not make noun clusters of more than 4 nouns" can be evaluated computationally by a system with a lexicon and a noun phrase grammar. On the other hand, a CL that is intended to improve text comprehension by humans may contain restrictions that are more general and/or vague. For example, "Make your instructions as specific as possible", or "Present new and complex information slowly and carefully". These restrictions cannot be checked computationally, since characteristics such as "specific", "slowly" and "carefully" aren't precisely defined, but such restrictions can be useful informal guidelines for human authors. In this paper, we focus on machine-oriented CL rules, which are formally specified and directly computable.

2.1 CL Advantages and Challenges

In general, the use of CLs can improve the source text in terms of readability, comprehensibility, consistency and reusability. Lexical constraints, which restrict vocabulary size and meaning for a particular application domain, are a key element in controlling a source language. In previous work, we have shown that the most important way to reduce ambiguity of source text is to restrict the lexicon (Baker, et al., 1994). The reduction of homonymy and synonymy may enhance the readability, consistency and comprehensibility of the text. When the text is written using standard terminology and sentence structures, a uniformity of style is achieved and it may become easier to reuse the text elsewhere. When CL is used in conjunction with translation memory, this consistency makes text easy to reuse, and hence more cost-effective.

The primary challenge when adopting a CL approach is that the writing task may take longer. First of all, authors need to learn the CL rules in order to write texts. In addition, rewriting text from legacy documents which do not conform to the CL may be more time-consuming than writing new CL text. Rewriting involves more than just replacement of unapproved words with their approved counterparts. After initial rewriting is completed by the author, an additional editing and verification step may be undertaken by an experienced editor before final approval.

Another concern is the initial cost of developing a CL for a particular organization and document production process. New development of a custom CL and document production system may not be cost-effective for a small production volume. Before embarking on full-scale development, it is useful to complete an initial feasibility study to evaluate the benefits and costs in a particular customer scenario.

2.2 CL Checking

It is difficult for an individual author to memorize all of the lexical and grammatical rules in a CL, especially in a domain with a large vocabulary. Therefore it is useful to provide a software system that can help authors to verify that their sentences conform to the CL rules. A CL checker verifies that all words and phrases are approved, and verifies that all writing rules are obeyed. In addition, a CL checker may offer feedback to the author during checking, when words or sentences are identified which do not conform to the CL.

The KANT Controlled English Checker checks each sentence for conformance, and also supports interactive disambiguation. If more than one analysis is found for a sentence, the system determines whether the cause is a lexical ambiguity or a structural ambiguity, and attempts to resolve it.

Some ambiguities can be resolved automatically, using a semantic model of the domain or heuristic preferences (Mitamura et al., 1999). Unresolved ambiguities are highlighted for the user. For lexical ambiguities, the author is asked to choose the intended meaning for the ambiguous word. For structural ambiguities, the author is asked to choose the intended structural relationship; for example, prepositional phrases in English can potentially modify more than one constituent in a sentence. The system annotates the original input to indicate the author's choice, thus disambiguating the sentence for subsequent translation (Mitamura & Nyberg, 1995). The resulting sentence meets the constraints of controlled English, and encodes a single chosen meaning for each ambiguous lexical item or structural attachment.

3 CL Deployment Issues

When a controlled language is deployed for production use by a group of authors, we encounter a set of practical issues, including training, usability, productivity, and long-term maintenance.

3.1 Usability and Productivity

When CL is used in conjunction with a machine translation, the constraints on language tend to be more strict when compared to a CL intended just for authoring. For machine translation, it is important to have an unambiguous input for accurate translation. Therefore, a CL for machine translation tends to focus on disambiguation of input sentences. However, when deploying a controlled language, author usability and productivity are very important factors for CL acceptance. If CL is too strict and/or time-consuming, then authors may have difficulty using it effectively. Another usability concern is that the CL must retain expressiveness while restricting vocabulary and grammar. Limiting only vocabulary does not necessarily reduce the complexity of input sentences; in fact, authors may need to write long, convoluted sentences to express complicated meanings if sufficient terminology is not available[2]. The balance between vocabulary size and input complexity is very important for successful CL deployment.

In KANT Controlled English, the size of the vocabulary is not limited *a priori*, and only those lexical or grammatical constructions that may cause difficult ambiguity problems are ruled out. The result is a controlled language that is expressive enough to write technical documents, but limited in complexity, such that high-quality translations can be achieved.

3.2 Controlled Language Maintenance

If we don't need to update the terminology or grammar once a controlled language is defined, then controlled language maintenance is not a major issue. In practice, however, we often face an ongoing need to update the terminology and grammar to keep pace with the introduction of new products or new document types. When hundreds of authors are simultaneously creating documents using a controlled language, maintenance is not a trivial task. It is important to implement a formally-specified process for language maintenance, which includes change requests, problem reporting, terminology screening, process monitoring, and quality control through periodic reviews. Once a decision has been made to update the terminology and grammar, it is essential to have appropriate support tools which support rapid updates to the controlled language system and any MT systems (Kamprath et al., 1998).

3.3 Controlled Language Training

For successful deployment of controlled language, the authors must accept the notion of controlled language and be willing to receive appropriate training. When authors become accustomed to writing texts in their own style over many years, it may be difficult for them to change their writing style. Since author us-

[2] An overly limited vocabulary is one factor that led to the demise of Caterpillar's original Fundamental English (Kamprath, et al., 1998), the predecessor of the Caterpillar Technical English which is in use today.

ability and productivity are essential for success, providing comprehensive training with a supportive CL checker is crucial. Nevertheless, even with full commitment and training, CL checking can require additional time when compared with unrestricted authoring.

If the CL checker can use automatic rewriting to provide alternatives when the author's sentence doesn't pass the checker, author effort can be reduced. Since usability and productivity are the key issues for CL deployment, it is clear that automatic rewriting for controlled language is an important area for research and development, both for text authoring and machine translation (Mitamura, 1999).

4 Automatic Rewriting for Machine Translation

In our work on controlled language for machine translation, the term *automatic rewriting* describes any process which replaces a source language sentence with another source language sentence, before translation begins. Source rewriting can be undertaken for a variety of reasons, including:

- To replace an ungrammatical sentence with a grammatical one (error correction);
- To replace a grammatical (but stylistically undesirable) sentence with another (more stylistically "correct") sentence;
- To replace a sentence with a restructured sentence which is not necessarily "grammatical" for the human reader, but better suited to a particular machine translation algorithm. For example, syntactic rearrangement to support structural transfer for a particular target language (Shirai et al., 1998).

In this paper, we concentrate on the second type of rewriting, where one grammatical sentence is replaced with another grammatical sentence in order to improve its style, translatability, etc. In most existing controlled language systems, such rewriting is accomplished manually (by the author) with the aid of an automatic grammar checker (Mitamura & Nyberg, 1996; Knops & Depoortere, 1998; Wojcik, et al. 1998). There are two basic approaches to grammar checking, and the dichotomy also extends to rewriting:

- *Prescriptive Approach.* The controlled language is implemented by a grammar which describes all allowable sentences. Any sentence which cannot be parsed by the grammar is considered outside the controlled language, and must be rewritten. The developers must work very carefully to define all of the allowable sentence structures in the domain. This approach is taken by systems like Caterpillar Technical English (Kamprath et al. 1998) and the Controlled Automotive Service Language (Means & Godden, 1996).
- *Proscriptive Approach.* The controlled language is implemented by a set of patterns which will match any sentence that should be rewritten. Only sentences which match one of the patterns must be rewritten. The developers may limit their attention to only those sentence patterns which are considered unacceptable. We can consider this approach to be "partial checking", since there may be other problems with a sentence which aren't caught by the existing set of patterns. This approach is taken by systems like Diebold's controlled language checker (Moore, 2000).

An automatic rewriting system requires two fundamental steps:

1. Deciding whether a given input sentence requires rewriting (checking). In a prescriptive system, this involves parsing the sentence to show that it only contains valid lexical and syntactic structures. In a proscriptive system, this involves iterating through a set of pre-defined "error patterns" to see if the current sentence matches any of them.
2. Providing one or more possible rewritten alternatives (rewriting). The system

must provide rewrites that preserve the meaning of the original and pass the appropriate grammar or constraint checking.

In the case of a prescriptive approach, rewriting is triggered when a sentence cannot be analyzed by the grammar and lexicon. Deciding how to rewrite can be challenging unless the system provides some information regarding the reason(s) for the failure. In a proscriptive approach, on the other hand, the system knows exactly which constraint(s) failed; since each constraint is associated with a particular writing error, it is possible to proceed with an automatic correction without any further information. Each constraint can be paired with a *rewrite rule* that specifies how to rearrange the original sentence.

Rewrite rules can be based on the input string (e.g., string-based pattern matching and reordering), a syntactic analysis (e.g., structural transformation), or a semantic analysis (e.g., reasoning over a meaning representation). Depending on the language and domain, a rewriting system may require a variety of rules, some string-based, some syntactic, and some semantic.

If we consider the range of characteristics discussed so far, we can see that a variety of automatic rewriting systems are possible in principle. The simplest system would be a proscriptive, string-based rewriting system, which evalutes a small set of constraints by string matching and rewrites sentences at the level of the surface string. The most complex system would be a prescriptive, semantics-based rewriting system, which rewrites sentences by understanding their meanings and proposing acceptable paraphrases.

The approach we propose in this paper falls somewhere in the middle. We combine the KANTOO Analyzer[3], which is a prescriptive controlled language system, with a "relaxed" grammar and a set of proscriptive constraints.

[3]The KANTOO Analyzer, which is written in C++, is a redesign and reimplementation of the core algorithms of the earlier KANT Analyzer (Mitamura, et al. 1991).

5 KANT Controlled Language Rewriting Architecture

The architecture of the CL Rewriting system is illustrated in Figure 1. Sentences written by the author are analyzed by the Analyzer, which performs syntactic parsing using a controlled grammar and lexicon. When a sentence cannot be parsed, the Rewriting Engine attempts to rewrite the sentence to a form that can be analyzed. When a sentence parses successfully, the Rewriting Engine checks it for other optional improvements. We examine these two types of processing below (see Section 6 for examples from a particular corpus).

5.1 Rewriting Sentences that Don't Parse

When a sentence cannot be analyzed by the CL Analyzer, it is identified as a candidate for rewriting, and is passed to the Rewrite Engine (see Figure 1). The rewrite algorithm contains the following steps:

1. Iterate through the surface rewrite patterns. If one matches, perform the indicated surface transformation on the sentence. Re-check the sentence by passing the rewritten version to the KANTOO Analyzer.

2. If no surface pattern matched, or no surface pattern produced an acceptable rewrite, then try to do a recovery parse. Find the set of possible syntactic analyses using a relaxed (less constrained) lexicon, grammar, and semantic constraints. The relaxed grammar will loosen CL restrictions such as agreement (e.g., Subject-Verb agreement in English), grammatical constraints (e.g. allow conjoined VPs), and lexical constraints (e.g., unapproved synonyms).

   In KANTOO, the relaxed grammer (or *recovery grammar*) is a set of additional grammar rules which are added to the CL grammar during the recovery parse. Each recovery rule annotates the constituent it produces, to indicate precisely

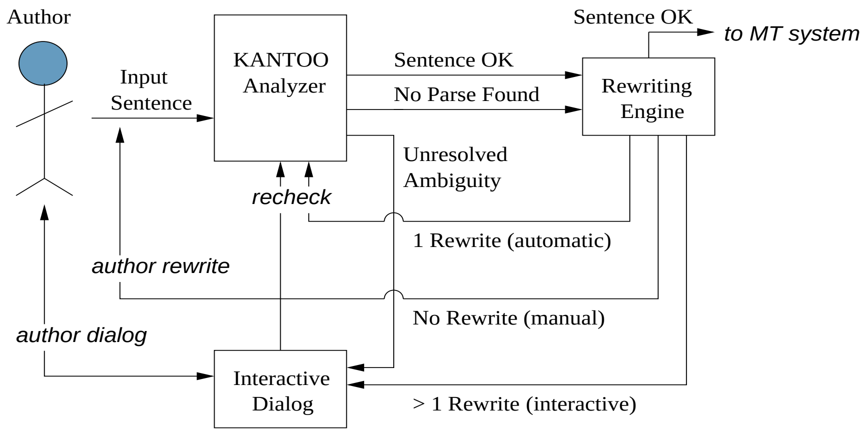

Figure 1: **KANT Controlled Language Rewriting Architecture.**

where a non-conforming constituent was found.

3. For each recovered parse, find all of the consituent nodes that are labelled with a recovery strategy. For each strategy, invoke a stored procedure which rewrites the original consituent corresponding to the recovered constituent, thus deriving a rewritten input.

4. Select one of the rewritten sentences, and pass it back to the KANTOO Analyzer; or, query the user and offer a set of rewritten sentences to choose from. If no possible rewrites are generated, then the system informs the user that there is an issue with the input, and the user is prompted to rewrite the original sentence manually.

5.2 Optional Rewriting of Grammatical Sentences

A sentence may pass the CL checker's grammar but still be considered for rewriting. As detailed in Section 6, there are grammatical sentences which are complex and/or ambiguous, and difficult to translate accurately; for example, long sentences with many conjoined phrases and clauses. It is also possible to rewrite a grammatical sentence to make it clearer and less ambiguous; for example, Figure 2 illustrates how the system might introduce an explicit preposition in a conjoined PP that elides the preposition in the second PP (see the example in Section 6.1).

```
   Input: marks on documents or scans
Readings: (or NP NP)
          (NP (or PP PP))

   Input: marks on documents or on scans
Readings: (NP (or PP PP))
```

Figure 2: **Optional Rewriting**

Since such sentences can be parsed by the first pass through the Analyzer, optional rewriting does not require a separate recovery parse. The optional rewriting strategies are implemented as syntactic transformations on f-structures (see the example in Appendix A).

5.3 Interactive Dialogue

Although our ultimate goal is to provide a rewritten input automatically when a sentence doesn't conform to the controlled language, in practice it may be difficult to select a rewrite automatically when there are several equivalent choices available. So we should consider making it possible for the user to view and select from the set of available rewrites (or best candidate rewrites, if some scoring mechanism has been used). Figure 3 gives an example of such an interactive dialogue. The original input, *Before working, the engine must warms up*, has been rewritten to fix two problems – the incorrect inflection of "warm" and the subjectless gerund "working". However, there are two possible rewrites. As shown in the figure, the gapped subject of "working" is ambiguous – the author could be referring to the reader, or the engine. Since the system cannot resolve this ambiguity, it decides to interact with the user by offering both rewrites for selection.

```
''Before working, the engine must warms up.''

This sentence must be rewritten. Please
select a rewrite, or Cancel to edit your
original sentence:

1. Before you work, the engine must warm up.
2. Before it works, the engine must warm up.
```

Figure 3: **Sample Dialogue**

6 Rewriting for KANT Controlled English

In order to examine how much rewriting is actually necessary for a legacy document to conform to the current KANT Controlled English, we selected about 220 sentences from a computer peripherals manual. The selected corpus contains more than 1000 unique lexical items. We used the KANTOO Analyzer to parse the original sentences[4]. The following summarizes the results:

1. 7% of the sentences parsed, but could be rewritten in order to improve style or reduce ambiguity.

2. 22% of the sentences did not parse and required a simple rewrite (generally to conform to a CL restriction intended to improve translation).

3. 13% of the sentences were rewritten because the original sentence was unclear, very long, or idiomatic.

4. 58% of the sentences did not require any rewriting.

From our experiment, we found that 35% of the sentences required rewriting in order to parse, and 7% parsed but could be optionally rewritten. Among these sentences, 113 instances of different phenomena were recognized as candidates for rewriting.

6.1 Optional Rewriting

An example of the first type of sentence, those which parse but could be improved to reduce ambiguity, can be seen in the following:

- **Original:** You may also need to clean these parts if there are smudges or other marks on documents or scans of documents.

- **Rewritten:** You may also need to clean these parts if there are smudges or other marks on documents or **on** scans of documents.

By inserting *on* before *scans of documents*, the structural analysis of which conjoins phrases *smudges*, *other marks on documents*, and *scans of documents* can be avoided and structural ambiguity can be reduced. Also, the sentence will become more clear in meaning for the human reader. This rewriting is not required in order to parse the sentence, but it is preferable for translation accuracy.

In order to insert *on* before *scans of documents* automatically, the analyzer needs to detect both *documents* and *scans of documents* modify *marks* or *smudges or other marks* and *scans of documents* do not conjoin with *smudges* or *other marks*. A semantic analysis is required to make such decision. When domain semantic knowledge is unavailable, an interactive dialogue with author would be initiated.

Another example of the first type of rewriting is seen in the following:

[4] Before parsing, we inserted SGML tags wherever necessary, since the KANTOO Analyzer assumes SGML tagging inside sentences for some constituents such as product numbers and names.

- **Original:** If a fax is being received while you are printing, the fax is stored in memory and then prints automatically after the print job finishes.

- **Rewritten:** If a fax is being received while you are printing, the fax is stored in memory. **Then, the fax** prints automatically after the print job finishes.

The original sentence is long but it can be broken down into two sentences to improve readability and reduce analysis complexity.

6.2 Simple Rewriting

Now we look at an example of second kind of rewriting, which requires a simple rewrite, as in the following:

- **Original:** This is also known as concatenated dialing.

- **Rewritten:** This is also **called** concatenated dialing.

This is a simple replacement of *known as* with *called*, in order to reduce syntactic ambiguity in the KANTOO Analyzer. This can be done with a simple pattern matching rule and does not require a syntactic analysis and transformation.

On the other hand, the following example requires syntactic parsing for rewriting:

- **Original:** Proofread the converted document carefully to ensure the characters have been correctly interpreted by the software.

- **Rewritten:** Proofread the converted document carefully **in order to** ensure **that** the characters have been correctly interpreted by the software.

Inserting *in order to* clearly indicates that this is a purpose clause. Since *proofread* does not subcategorize for an infinitival clause, it is assumed that *to ensure* is an adjunct clause. In the manual, we found that the infinitival adjunct clause is a purpose clause in most cases, so we automatically insert *in order* into the sentence. We found 11 cases in the corpus where this rewrite applied.

Another change that we make is to insert a complementizer, *that*. KANT controlled language requires a complementizer to avoid unnecessary ambiguity. Since *ensure* subcategorizes for a complement clause, *that* will be inserted when a sentence follows. We found 8 cases in the corpus where this rewrite applied.

Another example of the second type of rewriting is in the following:

- **Original:** After sending the fax, change this setting back to NO.

- **Rewritten:** After **you send** the fax, change this setting back to NO.

Omitting the subject of a subordinate clause is very common in technical writing, and it is assumed that the omitted subject is the same as the subject of the main clause. If this can be assumed to be always true, we insert the subject from the main clause. When the main clause is imperative, we insert *you* as the subject in the subordinate clause. However, this is not always appropriate:

- **Original:** Send an electronic fax while the printer makes copies, or scan a document while printing.

- **Rewritten:** Send an electronic fax while the printer makes copies, or scan a document while **a document is** printing.

In this sentence, it is better to write *while a document is printing*, instead of *while you are printing*. However, to handle this distinction, automatic rewriting requires more than just a simple syntactic analysis to determine the appropriate subject. If there is more than one choice, the system will query authors to choose the appropriate rewrite.

6.3 Author Rewriting

An example of the third type of rewriting is seen in the following idiomatic expression:

- **Original:** Area of faded print are often an indicator that the toner cartridge is near the end of its toner life.

- **Rewritten:** Area of faded print are often an indicator that the toner cartridge is **running out of toner.**

As an input to KANT machine translation, the rewritten sentence would be better, since it is a more direct expression that is less prone to mis-translation.

We sometimes encounter long, complicated sentences which are best split into two or more sentences. An example is seen in the following:

- **Original:** When you are scanning an item and choose Scan on the Scan Assistant or in the Scan Pro software, select the Scan to Text option, and click Start Scan, the Scan Pro software initiates the scan using the best settings for text.

- **Rewritten:** When you are scanning an item, choose Scan on the Scan Assistant or in the Scan Pro software, select the Scan to Text option, and click Start Scan. The Scan Pro software initiates the scan by using the best settings for the text.

The original sentence is not well-formed English, and can be broken down into two sentences to make the meaning more clear.

The above examples of the third type of rewriting are much more complicated to rewrite automatically, and the system will refer such sentences back to the author for rewriting.

In the case of approved domain idioms (which should be handled directly, not rewritten), we can enhance the grammar and lexicon in a straightforward manner to incorporate them into the CL.

7 Issues in Automatic Rewriting

In this section, we are going to discuss some issues related to rewriting design, controlled language rewriting and post-editing.

7.1 Rewriting Design Issues

Although the rewriting algorithm represents a fairly simple extension of the current KANTOO CL Checker control flow, there are several design issues which must be explored:

- *Determinism in Applying Surface Patterns.* If we have a system with n surface patterns, a question arises regarding how to apply them to the input and how to handle the result. Should we prefer sequential ordering, where we try each pattern one after the other to see which ones match? If so, then what sequential ordering of patterns is correct? We may also consider trying all the patterns in parallel, assuming that only one pattern should be tried at a time. Although this choice would remove the need to choose a sequential ordering of patterns *a priori*, we are left with the necessity to choose among several possible rewrites.

- *Determinism in Applying Syntactic Transformations.* If we have a recovery parse with more than one recovered constituent, is the rewriting process sensitive to the ordering of the transformations? In a bottom-up parser, we can assume that recovery of lower-order constituents is performed before parsing of higher-level constituents, so any resulting transformations must be carried out from bottom to top.

- *Automatic Selection of Rewrites.* Our rewriting algorithm has the ability to find more than one rewrite for a particular sentence through recovery parsing and structural transformation. In the case of automatic rewriting, where the user is not consulted, which rewrite should be chosen? A simple technique would be to prefer certain types of recovery over others, and to prefer recovered analyses which use the fewest number of rewrite steps. Using these two metrics, we could score the recovery analyses, partition them into equivalence classes based on score, and select from the analyses with the best score. In the absence of a scoring mechanism, we could use a statistical language model (SLM) to score the rewritten sentences (to pick the one that is most likely to be a corrent sentence in that language). However, it will probably be difficult to obtain a large enough corpus to build a useful SLM in technical domains with limited text resources.

- *Halting Problem.* Since the rewriting engine can create new inputs and send them back to the KANTOO Analyzer, an infinite loop cannot be ruled out *a priori.* It is possible to write two rewrite strategies, each of which rewrites its input to a new sentence matched by the other. With such rules the rewriting engine would enter an infinite loop. The rewriting engine could avoid this problem in a practical way by either limiting its application to user sentences only, and not to automatically rewritten sentences. Or the system could keep track of the rewriting inputs and notice when the same sentence had been re-submitted by the system as part of rewriting.

7.2 Controlled Language Rewriting Issues

The following issues should be considered from the perspective of the controlled language author:

- *Error Handling.* It is conceivable that the system might produce an automatic rewrite which passes subsequent checking, but contains some semantic error (i.e., it doesn't preserve the original meaning intended by the author). Such a scenario is likely to lead to translation inaccuracies in an MT system. A conservative approach might be to confirm all rewrites with the author, to avoid this problem; but the corresponding loss of productivity (through increased interaction) might not be justified if the frequency of actual problems is quite low. This trade-off must be examined empirically by comparing a) the loss of source and translation accuracy due to automatic rewriting errors, with b) the additional time that would be required from the author to confirm each automatic rewrite.

- *Degree of Automaticity.* If there is more than one rewrite, how do we decide whether to pick one automatically, or interact with the author? So far we have assumed that all rewriting rules are of equal importance and efficacy, but in practice some rules will be frequently applied and relatively "error free", while others may be infrequent, dealing with complex cases, and likely to produce errors if unconfirmed by the author. It is likely that some heuristic approach should be adopted, such that some rewrites always happen automatically, while others are always confirmed by the author. This could be implemented via per-rule preferences, or by setting an overall threshold on the number and type of rewrite rules that may be invoked before explicit confirmation is required.

- *Usability Study.* The impact of issues like error handling and automaticity must be empirically investigated, and hence require some measurement of their impact on author productivity and system acceptance. Therefore, it will be important to conduct a formal usability study in a realistic authoring context. The baseline for such a study would be the use of a controlled language checker without automatic rewriting. The experimental study would add rewriting to the basic checker, to accomplish A/B comparisons of automatic rewriting vs. interactive dialogue on a variety of documents. The intent is to measure the trade-offs between various factors, including:

  - Increase in author productivity (for automatic rewriting) vs. additional processing time (system time taken during rewriting) and frequency of rewriting errors;
  - Increase in author dialogue (for purely interactive rewriting) vs. an overall increase in productivity and reduction in rewriting errors.

The precise configuration of a rewriting system involves decisions about what should and shouldn't be automatic, which in turn will depend on empirical data from a particular domain.

7.3 Automatic Rewriting for Post-editing

Although our paper has been concentrated on automatic controlled language, which is a pre-editing part of KANT machine translation, we could also apply the same algorithm for automatic post-editing. Since KANT controlled English is used for multilingual machine translation, the controlled English is not designed to make it easier to translate a particular target language. When the MT generation module produces target sentences, we may want to rewrite them for grammatical as well as stylistical reasons. However, when we encounter some target grammatical issues, we update the KANT MT modules, so that we can generate correct sentences. The question on the division of effort between MT and automatic post-editing is beyond the scope of our research right now, although we recognize that this is a question for future work.

8 Future Work

In this paper, we have presented a linguistic framework and a software architecture for automatic rewriting which is based on the concepts of controlled language analysis and translation investigated in earlier systems. By making use of existing formalisms (unification grammar and syntactic transformations), we can implement a set of rewriting strategies which handle both recovery of ungrammatical sentences and optional rewriting of sentences which are grammatical, but stylistically less preferable.

We are currently implementing a set of recovery rules and transformations for the rewriting phenomena mentioned in Section 6. Once the rewriting strategies have been implemented, we intend to carry out a user study which examines the issues laid out in Section 7. The extensions to the KANTOO architecture are being implemented so that it will be possible to control the degree of automaticity and interaction in the system. This will make it possible to perform various kinds of user experiments, and also make it possible to tune the final system based on the results of the experiments.

Our experience with the industrial application of controlled language for MT has shown that an important criterion for system acceptance is author productivity, which is directly related to the amount of time the author must spend to rewrite sentences which don't conform to the controlled language. Continued research and development of automatic rewriting is essential, if controlled language systems are to achieve maximum utility in real-world applications.

References

Baker, K., Franz, A., Jordan, P., Mitamura, T., Nyberg, E. 1994. "Coping with Ambiguity in a Large-Scale Machine Translation System". *Proceedings of the 15th International Conference on Computational Linguistics (COLING-94*, pages 90-94, Kyoto, Japan.

Kamprath, C., E. Adolphson, T. Mitamura and E. Nyberg 1998. "Controlled Language for Multilingual Document Production: Experience with Caterpillar Technical English". *Proceedings of the Second International Workshop on Controlled Language Applications (CLAW-98)*, pages 51-61, Pittsburgh.

Knops, U. and B. Depoortere, "Controlled Language and Machine Translation". *Proceedings of the Second International Workshop on Controlled Language Applications (CLAW-98)*, pages 42-50, Pittsburgh.

Mitamura, T. 1999 "Controlled Language for Multilingual Machine Translation". *Proceedings of Machine Translation Summit VII*, pages 46-52, Singapore.

Mitamura, T., E. Nyberg, E. Torrejon and R. Igo 1999. "Multiple Strategies for Automatic Disambiguation in Technical Translation," *Proceedings of the 8th International Conference on Theoretical and Methodological Issues in Machine Translation (TMI-99)*, pages 218-227, Chester, England.

Mitamura, T., Nyberg, E. and Carbonell, J. 1991 "An Efficient Interlingua Translation System for Multi-lingual Document Production". *Proceedings of Machine Translation Summit III*, pages 55-61, Washington, DC.

Means, L. and K. Godden 1996. "The Controlled Automotive Service Language (CASL) Project", *Proceedings of the First International Workshop on Controlled Language Applications (CLAW-96)*, pages 106-114, Leuven, Belgium.

Moore, C. 2000 "Controlled Language at Diebold, Incorporated". *Proceedings of the Third International Workshop on Controlled Language Applications (CLAW-2000)*, pages 51-61, Seattle.

Nyberg, E. and T. Mitamura 1996. "Controlled Language and Knowledge-Based Machine Translation: Principles and Practice". *Proceedings of the First International Workshop on Controlled Language Applications (CLAW-96)*, pages 74-83, Leuven, Belgium.

Shirai, S., S. Ikehara, A. Yokoo, and Y. Ooyama 1998. "Automatic Rewiting Method for Internal Expressions in Japanese to English MT and Its Effects". pp 62-75, *Proceedings of the Second International Workshop on Controlled Language Applications (CLAW-98)*, pages 62-75, Pittsburgh.

Tomita, M. and E. Nyberg 1988. "Generation Kit and Transformation Kit Version 3.2: User's Manual", Technical Report CMU-CMT-88-MEMO, Center for Machine Translation, Carnegie Mellon University.

Wojcik, R., H. Holmback and J. Hoard 1998. "Boeing Technical English: An Extension of AECMA SE beyond the Aircraft Maintenance Domain". *Proceedings of the Second International Workshop on Controlled Language Applications (CLAW-98)*, pages 114-123, Pittsburgh.

Appendix A: Sample Transformation

Figure 4 illustrates a single structural transformation that is used to insert an explicit (distributed) preposition when a preposition appears with a conjoined NP as object (see Section 5.2). In KANTOO, tranformations are implemented using a rule formalism called PATRICK, which is a variation on the structural transformations (based on unification grammar) that were introduced in (Tomita & Nyberg, 1988).

The example contains a single transformation rule and a stored procedure, `make-conjoined-pp`. The rule will fire on any f-structure that matches the first three unification statements: any f-structure whose category is `prep`, is simple (not a conjunction of PPs), and contains a conjoined NP object.

The unification statements which follow create a new f-structure which is a conjoined prepositional phrase.

```
(*TRY*

; See if we have a single prep
; with conjoined object.
 ((%(fs cat) =c prep)
  (%(fs conj) = *UNDEFINED*)
  (%(fs obj conj) = *DEFINED*)

; Make a new FS which is a
; conjoined prep
  (%new-fs = %(fs obj))
  (%(new-fs root) <= "conj-pp")
  (%(new-fs tokens) =
    (*MULT* %(fs tokens)
            %(fs obj tokens)))

; Make the conjoined PP's
  (%(new-fs member) <=
    #make-conjoined-pp ((pp %fs)))

; Replace fs with new-fs
  (%fs <= %new-fs))
 )

; For each object in %fs,
; creates a pp w/ tail recursion

#make-conjoined-pp %fs
((*TRY*
  ((*NOT* #is-mult
          %(fs obj member))
   (%(fs obj) <= %(fs obj member))
   (*TEST* %fs))
  ((%pp <= %fs)
   (%(pp obj) = *REMOVE*)
   (%(pp obj) < %(fs obj member))
   (*TEST* (*MULT* %pp #make-conjoined-pp %fs)))
  ))
```

Figure 4: **Structural Transformation: Preposition Insertion**