1 Introduction

Automated paraphrasing is a sub-field of NLP that has been receiving increasing attention due to its potential for a wide range of NLP applications. Paraphrasing can be primarily viewed as a special case of translation, which has been intensively studied for decades in the NLP field; both transform the wording of a source text into another different wording, while preserving the meaning as much as possible. It must be noted, however, that paraphrasing has its own idiosyncrasies that should be explored as a different matter. Motivated by this context, we have developed a Japanese lexicostuctural paraphrasing engine KURA. It is designed as a computational environment in which a range of patterns of lexicostuctural paraphrasing can be easily implemented and tried on a large-scale corpus.

2 A system overview

Lexico-structural paraphrasing KURA is designed to easily implement a wide range of lexicostuctural paraphrasing patterns. Lexico-structural paraphrasing we consider here includes, for example, lexical/phrasal replacement, verb/case/voice alternation, topicalization, head switching and sentence aggregation/division.

Three-layered architecture The overall architecture is based on the division of labor between application-specific purposefulness assessment (e.g. readability assessment) and application-neutral linguistic transformation as illustrated in Figure 1.

Revision-based transformation A linguistic transformation is realized by a morpho-syntactic transfer followed by revisions (see Figure 1). The revision component rewrites or rejects ill-formed/inadequate paraphrase candidates produced by the non-deterministic transfer component. This division of labor reduces the complexity of paraphrasing knowledge.

Declarative and uniform knowledge representation A system user describes paraphrasing knowledge (a transfer and language generation models) uniformly as a set of declarative lexicostuctural transformation rules. Each transformation rule is automatically compiled into a sequence of dependency-tree processing operators, as illustrated in Figure 2, which are then executed by the linguistic transformation component.

Process efficiency Several basic techniques have been implemented to improve the computational efficiency of rule application. The system runs efficiently enough to be used for a large-scale experiment; for example, the current system, running on a 400MHz Sparc processor, takes no more than one hour to produce 8,769 paraphrases by applying pre-compiled 795 transfer rules to each of 8,329 prepared input sentences excerpted from newspaper articles.

GUI for process monitoring All the information of a linguistic transformation process is accumulated in a relational database. A GUI to the database is equipped for process monitoring, error analysis and rule debugging (Figure 3).

3 Implementation

The backbone of the current version of KURA is implemented on a Prolog-like typed feature unification system LiFeS 1. For morphosyntactic analysis and rule translation, we use Chasen 2 and Cabocha 3.

The whole system is available for free at www.pluto.ai.kyutech.ac.jp/plt/inui-lab/tools. It will be able to work wherever LiFeS, Chasen and Cabocha, which are all freeware, work properly.

Acknowledgments

The authors would like to thank the NTT Communication Science Laboratories for providing their case frame dictionary and thesaurus, Takaki Makino and his colleagues (Tokyo University) for allowing us to use LiFeS and their helpful advises to our programming, and Yuii Matsumoto and his colleagues (Nara Advanced Institute of Technology) for allowing us to use Chasen and Cabocha.

---

1 www.tsujii.is.s.u-tokyo.ac.jp/ mak/lifes/index.html
2 chasen.aist-nara.ac.jp/chasen/distribution.html
Figure 1: Three-layered architecture and revision-based linguistic transformation

Figure 2: Rule editing aid compilation

Figure 3: A snapshot of process monitoring: You can see (a) a source, intermediate and target sentences on the left-hand side of the background window, (b) the list of the rules applied in the current paraphrasing instance in the middle, (c) various tags annotated for evaluation and error analysis on the right-hand side, and (d) a list of paraphrasing instances where a certain rule is applied in the foreground window.