An Integrated Architecture for Example-Based Machine Translation

Alexander Franz, Keiko Horiguchi, Lei Duan,
Doris Ecker, Eugene Koontz, and Kazami Uchida
Spoken Language Technology, Sony US Research Labs
3300 Zanker Road
San Jose, CA 95134, USA
{amf,keiko,lei,doris,ekoontz,kuchida}@slt.sel.sony.com

Abstract

This paper describes a machine translation architecture that integrates the use of examples for flexible, idiomatic translations with the use of linguistic rules for broad coverage and grammatical accuracy. We have implemented a prototype for English-to-Japanese translation, and our evaluation shows that the system has good translation quality, and only requires reasonable computational resources.

1 Introduction

Machine translation by analogy to pairs of corresponding expressions in the source and target languages, or "example-based translation", was first proposed by (Nagao 1984). Recent work in the example-based framework includes memory-based translation (Sato & Nagao 1990), similarity-driven translation (Watanabe 1992), transfer-driven machine translation (Furuse & Iida 1996), and pattern-based machine translation (Watanabe & Takeda 1998).

The example-based approach promises easy translation knowledge acquisition, more flexible transfer than brittle rule-based approaches, and idiomatic translations. At the same time, the use of linguistic rules offers a number of important benefits. Detailed linguistic analysis can allow an example-based machine translation system to handle a wide variety of input, since rules can be used to factor out all linguistic variations that do not influence the example-based transfer. Rule-based language generation from detailed linguistic representations can lead to higher grammatical output quality. Finally, a modular system architecture that uses domain-independent linguistic regularities in separate linguistic modules allows extending the system to much broader domains. The HARMONY architecture for hybrid analogical and rule-based machine translation of naturally occurring colloquial language combines the advantages of both these approaches.

2 The Travel Domain

Our prototype implementation of the HARMONY architecture was designed to cover the "travel domain". This is composed of words, phrases, expressions, and sentences related to international travel, similar to what is covered by typical travel phrase books.

Two principles guided our detailed definition of the translation domain. First, the translation domain should not be limited to a narrow sub-domain, such as appointment scheduling or hotel reservations. Second, the expressions considered in the domain should reflect the fact that people quickly adapt to limitations in human-machine or machine-mediated communication by simplifying the input. For example, (Sugaya et al. 1999) found that the average length of actual human utterances in a hotel reservation task using speech translation was only 6.1 words, much shorter than some of the data that has been used in previous work on speech translation.

The current vocabulary of 7,500 words is divided into a group of general words, a number of extensible word groups (such as names of food items or diseases), and a number of area-specific word groups (such as names of cities or tourist destinations).The travel domain is divided into eight "situations": A general situation (including everyday conversation); transportation; accommodation; sightseeing;

shopping; wining, dining, and nightlife; banking and postal; and doctor and pharmacy.

We created a corpus for this domain, and divided it into a development set of 7,000 expressions, and a separate, unseen test set of 5,000 expressions. The development set is used for creation and refinement of the translation knowledge sources, and the test set is only used for evaluations. (Each evaluation uses a new, random 500-word sample from the 5,000 word test set.)

The corpus was balanced to illustrate the widest possible variety of types of words, phrases, syntactic structures, semantic patterns, and pragmatic functions. The average length of the expressions in the corpus is 6.5 words. Some examples from the development corpus are shown below. Even though this domain might seem rather limited, it still contains many challenges for machine translation.

- *Can I have your last name, please?*
- *Is this the bus for Shinagawa station?*
- *I would like to make a reservation for two people for eight nights.*
- *Can you tell us where we can see some Buddhist temples?*
- *Most supermarkets sell liquor.*
- *Can you recommend a good Chinese restaurant in this area?*
- *I'd like to change 500 Dollars in traveller's checks into Yen.*
- *Are there any English-speaking doctors at the hospital?*

3 NLP Infrastructure

The prototype implementation is constructed out of components that are based on a powerful infrastructure for natural language processing and language engineering. The three main aspects of this infrastructure are the Grammar Programming Language (GPL), the GPL compiler, and the GPL runtime environment.

3.1 The Grammar Programming Language

The Grammar Programming Language (GPL) is an imperative programming language for feature-structure-based rewrite grammars. GPL is a formalism that allows the direct expression of linguistic algorithms for parsing, transfer, and generation. Some ideas in GPL can be traced back to Tomita's pseudo-unification formalism (Tomita 1988), and to Lexical-Functional Grammar (Dalrymple et al. 1995). GPL includes variables, simple and complex tests, and various manipulation operators. GPL also includes control flow statements including if-then-else, switch, iteration over sub-feature-structures, and other features. An example of a simplified GPL rule for English generation is shown in Figure 1.

```
WH_SENT → NP YN_SENT {
          !exist[$m VP SUBJ WH];
          local-variable WH_VP = [$m VP];
          local-variable WH_PHRASE;
        $WH_PHRASE = find-subfstruct in $WH_VP
                            where (?exist[$x WH]);
        $d1 = [$WH_PHRASE SLOT-VALUE];
        [$WH_PHRASE SLOT-VALUE TRACE] = '+';
        $d2 = $m;
}
```

Figure 1 Example of a GPL Generation Rule

3.2 The GPL Compiler

GPL grammars are compiled into C code by the GPL compiler. The GPL compiler was created using the Unix tools lex and yacc (Levine et al. 1990). For each rewrite rule, the GPL compiler creates a main action function, which carries out most of the tests and manipulations specified by the GPL statements.

The GPL compiler handles disjunctive feature structures in an efficient manner by keeping track of sub-feature-structure references within each GPL rule, and by generating an expansion function that is called once before the action function. The compiler also tracks variable references, and generates and tracks separate test functions for nested test expressions.

3.3 The GPL Run-time Environment

The result of compiling a GPL grammar is an encapsulated object that can be accessed via a public interface function. This interface function serves as the link between the compiled GPL grammars, and the various language-independent and domain-independent software engines for parsing, transfer, generation, and others. This is illustrated in Figure 2.

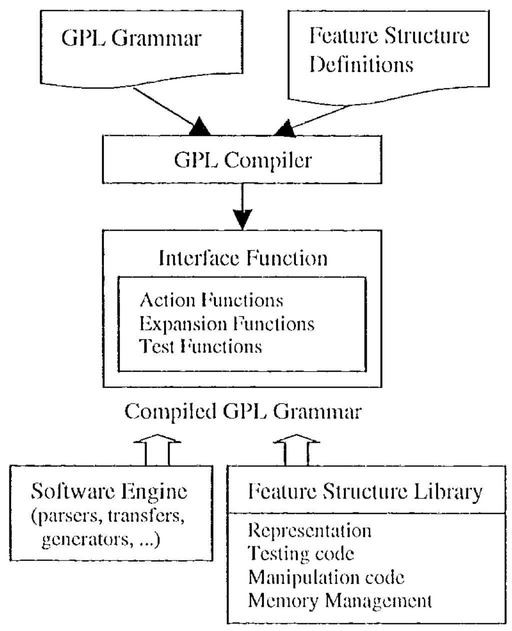
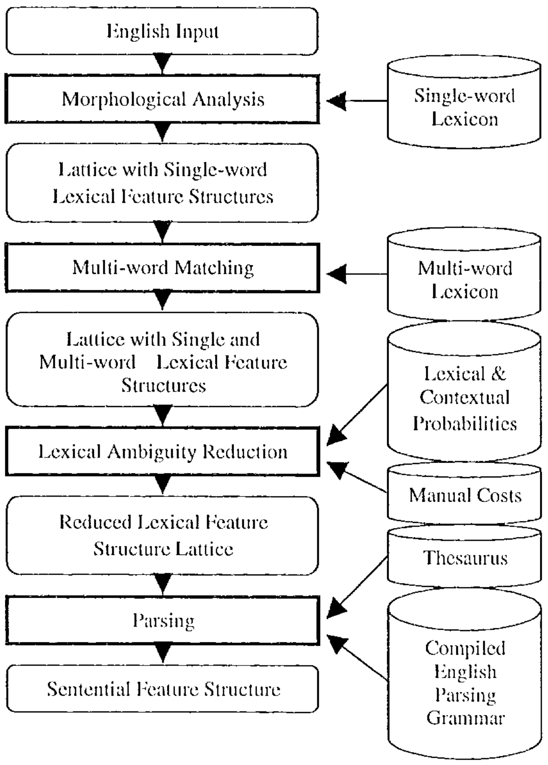


Figure 2 GPL Run-time Environment



English Input
Morphological Analysis
Single-word Lexicon
Lattice with Single-word Lexical Feature Structures
Multi-word Matching
Multi-word Lexicon
Lattice with Single and Multi-word Lexical Feature Structures
Lexical & Contextual Probabilities
Lexical Ambiguity Reduction
Manual Costs
Reduced Lexical Feature Structure Lattice
Thesaurus
Parsing
Compiled English Parsing Grammar
Sentential Feature Structure

Figure 3 Architecture of the Analysis Module

The compiled GPL grammars use the feature structure library, which provides services for efficiently representing, testing, manipulating, and managing memory for feature structures. A special-purpose memory manager maintains separate stacks of memory pages for each object size. This scheme allows garbage collection that is so fast that it can be performed after every attempted GPL rule execution. In our experiments with Japanese and English parsing, we found that per-rule garbage collection reduced the overall read/write memory requirements by as much as a factor of four to six.

4 Source Language Analysis

Translation is divided into the steps of analysis, transfer, and generation. Source-language analysis is illustrated in Figure 3.

English analysis begins with tokenization and morphological analysis, which creates a lattice that contains lexical feature structures. During multi-word matching, expressions from the multi-word lexicon (such as *White House* or *take on*) are detected in the word lattice, and new arcs with the appropriate lexical feature structures are added.

Lexical ambiguity reduction reduces the number of arcs in the word lattice. This module carries out part-of-speech tagging over the lattice, and reduces the lattice to those lexical feature structures that are part of the number of best paths that represents the best speed/accuracy trade-off (currently two). This calculation is based on the usual lexical and contextual bigram probabilities that were estimated from a training corpus, but it also takes into account manual costs that can be added to lexicon entries, or to individual part-of-speech bigrams.

The resulting reduced lattice with lexical single-word and multi-word feature structures is parsed using the GLR parsing algorithm extended to lattice input (Tomita 1986). The English parsing grammar consists of 540 GPL rules. The output is a sentential feature structure that represents the input to the transfer component.

5 Transfer

Transfer from the source-language sentential feature structure to the target-language sentential feature structure is accomplished with a hybrid rule-based and example-based method. This is illustrated in Figure 4.

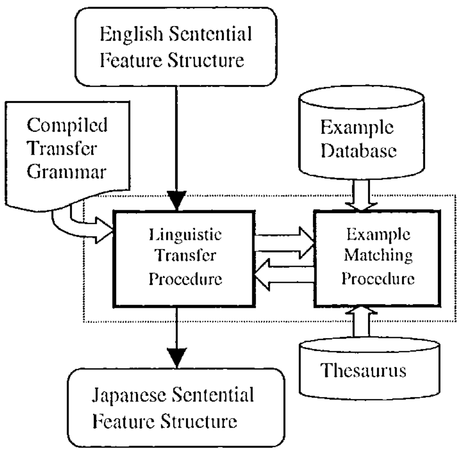

English Sentential Feature Structure
Compiled Transfer Grammar
Example Database
Linguistic Transfer Procedure
Example Matching Procedure
Thesaurus
Japanese Sentential Feature Structure

Figure 4 Architecture of the Transfer Module

The input feature structure is passed to the linguistic transfer procedure. This consists of a rule-rewriting software engine that executes the compiled English-to-Japanese transfer grammar. The transfer grammar consists of 140 GPL rules, and its job is to specify linguistic constraints on examples, combine multiple examples, transfer information that is beyond the scope of the example database, and perform various other transformations. The overall effect is to broaden the linguistic coverage, and to raise the grammatical accuracy far beyond the level of a traditional example-based transfer procedure.

The linguistic transfer procedure operates on the input feature structure in a recursive manner, and it invokes the example matching procedure to find the best translation example for various parts of the input. The example matching procedure retrieves the best translation examples from the example database, which contains 14,000 example pairs ranging from individual words to entire sentences. In an off-line step, the example pairs are parsed, disambiguated, and indexed for corresponding constituents using a Treebanking tool.

At each invocation of the example matching procedure, linguistic constraints from the transfer grammar are used to limit the search space to appropriate examples. In an off-line step, these constraints are pre-compiled into a complex index that allows a preliminary fast match. Examples that survive the fast match are matched and aligned with the input feature structure (or sub-feature-structure, during recursive invocations) using the thesaurus to calculate word similarity, and using various other constraints and costs for inserting, deleting, or altering slots and features. Rather than rely on the exact distance in the thesaurus to calculate lexical similarity, we use a scheme that is based on the information content of thesaurus nodes, similar to (Resnik 1995).

6 Target-language Generation

The Japanese target-language feature structure forms the input to the generation module, which is summarized in Figure 5 below. This module also consists of a rule-rewriting software engine, executing the compiled GPL Japanese generation grammar, which consists of 200 GPL rules. The generator uses the Japanese lexicon to create the Japanese target-language expression.

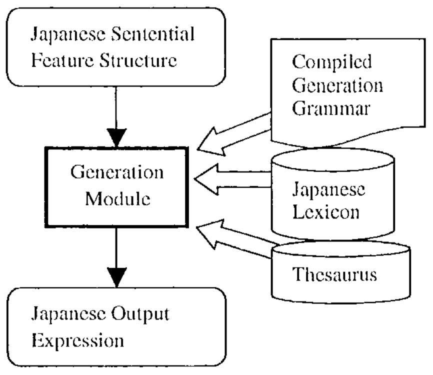

Japanese Sentential Feature Structure
Compiled Generation Grammar
Generation Module
Japanese Lexicon
Thesaurus
Japanese Output Expression

Figure 5 Architecture of the Generation Module

7 Evaluation and Conclusions

We evaluated the translation system using a random 500-expression sample from the unseen test set (see Section 2 above). The translations were manually assigned to one of the following categories of translation quality:

Failure. Complete translation failure, due to lack of coverage of a rule-based component.

Wrong. A translation that is completely wrong, or that has major errors in an important part, such as in the main clause.

Major Problem. A translation that has a missing, extra, or incorrect constituent, such as a subject, object, or adjectival/prepositional predicate.

Minor Problem. A translation that has a missing, extra, or incorrect minor part, such as an intensifier, tense, aspect, temporal or locative adjunct, adverb, adjective or other prenominal modifier, prepositional phrase, verb conjugation form, adjective form, or required word or constituent order.

Stylistic Problem. Stylistic problems include awkward but tolerable word order, incorrect Japanese particles, incorrect idioms, and similar.

Flawless. A translation that does not exhibit any of the above problems is considered flawless.

The results of the evaluation are shown in Table 1 below. Overall, 84% of the translations convey the meaning in an acceptable manner. We also evaluated the computational resource requirements of the system. On a Pentium III running at 500 MHz, the average translation speed was 0.44 seconds. The memory requirements are summarized in Table 2 below.

Flawless	60%
Stylistic Problem	9%
Minor Problem	14%
Acceptable with OOV	1%
Major Problem	9%
Wrong Translation	5%
Translation Failure	3%

Table 1 Translation Quality

Read-only Memory for Code and Data	6MB
Read-only Memory for Dictionary, Examples, Fast Match Index, etc.	23MB
Read/Write Memory for Feature Structures	14MB
Read/Write Memory for Software Engines	4MB

Table 2 Memory Requirements

Our plans for further work include extending the size of the input vocabulary, and developing mechanisms for closer integration with speech recognition and speech synthesis components for speech-to-speech translation. We are also working on the Japanese-to-English translation direction, and we plan to report results on this in the future.

Acknowledgements

Our thanks go to Robert Bowen, Benjamin Hartwell, Chigusa Inaba, Kaori Shibatani, Hirono Stonelake, and Kazue Watanabe for their language engineering efforts, and to Edward Ho for user interface and application development.

References

Dalrymple, M., R.M. Kaplan, J.T. Maxwell III, and A. Zaenen, eds. (1995) *Formal Issues in Lexical-Functional Grammar.* CSLI Lecture Notes 47, Stanford, CA.

Furuse, O. and H. Iida (1996) "Incremental translation utilizing constituent-boundary patterns", in *Proceedings of COLING-96*, pages 412-417.

Levine, J.R., T. Mason, and D. Brown (1990), *lex & yacc (Second Edition)*, O'Reilley and Associates, Sebastopol, CA.

Nagao, M. (1984) "A framework of a Machine Translation between Japanese and English by analogy principle", in *Artificial and Human Intelligence*, A. Elithorn and R. Banerji (eds.), North Holland, pages 173—180.

Resnik, P. (1995) "Using information content to evaluate semantic similarity in a taxonomy", in *Proceedings of IJCAI-95*.

Sato, S. and M. Nagao (1990) "Toward memory-based translation", in *Proceedings of COLING-90*, vol. 3, Helsinki, Finland, pages 247—252.

Sugaya, F., T. Takezawam A. Yokoo, and S. Yamamoto (1999) "End-to-end evaluation in ATR-Matrix", in *Proceedings of Eurospeech-99*, Budapest, Hungary, pages 2431—2434.

Tomita, M. (1988) *The Generalized LR Parser/Compiler (Version 8.1): User's Guide.* Technical Memorandum CMU-CMT-88-Memo, Center for Machine Translation, Carnegie Mellon University.

Tomita, M., "An efficient word lattice parsing algorithm for continuous speech recognition", in *Proceedings of ICASSP-86*, Tokyo, Japan, pages 1569-1572.

Watanabe, H. (1992) "A similarity-driven transfer system", in *Proceedings of COLING-92*, Nantes, France, pages 770-776.

Watanabe, H. and K. Takeda (1998) "A pattern-based Machine Translation system extended by example-based processing", in *Proceedings of ACL-COLING-98*, pages 1369-1373.