Learning Transfer Rules for Machine Translation from Parallel Corpora

Werner Winiwarter, University of Vienna
Department of Scientific Computing
Universitätsstraße 5,
A-1010 Vienna, Austria
werner.winiwarter@univie.ac.at

Journal of Digital
Information Management

ABSTRACT: *In this paper we present JETCAT, a Japanese-English transfer-based machine translation system. Our main research contribution is that the transfer rules are not handcrafted but are learnt automatically from a parallel corpus. The system has been implemented in Amzi! Prolog, which offers scalability for large rule bases, full Unicode support for Japanese characters, and several APIs for the seamless integration of the translation functionality into common office environments. As a first user interface we have developed a translation environment under Microsoft Word. The dynamic nature of our system allows for an easy customization of the rule base according to the user's personal preferences by simply post-editing the translation results, which leads to an automatic update. The user interface for Microsoft Word also provides the possibility for the user to display token lists, parse trees, and transfer rules, which makes JETCAT also a very useful tool for language students.*

Categories and Subject Descriptors
I.2.7 [**Natural Language Processing**]; Machine translation **I.2.8** [Problem Solving, Control Methods, and Search]:Graph and tree search strategies

General Terms
Machine translation, Unicode

Keywords: Japanese-English translation, Language transfer rules

Received 10 Sep. 2007; Reviewed and accepted 27 Jan. 2008

1. Introduction

Despite the large amount of effort invested in the development of machine translation systems, the achieved translation quality is still appalling. One major reason is the missing ability to learn from translation errors through incremental updates of the rule base.

In our research we use the bilingual data from the JENAAD corpus [27] comprising 150,000 Japanese-English sentence pairs from news articles to learn specific transfer rules from the examples through structural matching between the parse trees for the source and target language. The rules are generalized in a consolidation phase to avoid overtraining and to increase the coverage for new input. Based on the resulting rule base we have developed *JETCAT* (Japanese-English Translation using Corpus-based Acquisition of Transfer rules), a Japanese-English transfer-based machine translation system.

Our current research work originates from the PETRA project (Personal Embedded Translation and Reading Assistant) [29], in which we had developed a translation system from Japanese into German. One main problem for that language pair was the lack of training material, i.e. high quality Japanese-German parallel corpora, which was one of the reasons why we shifted our research focus to Japanese-English.

For the implementation of our machine translation system we have chosen Amzi! Prolog because it provides an expressive declarative programming language within the Eclipse Platform. It offers powerful unification operations required for the efficient application of the transfer rules and full Unicode support so that Japanese characters can be used as textual elements in the Prolog source code. Amzi! Prolog has also proven its scalability during past projects where we accessed large bilingual dictionaries stored as fact files with several 100,000 facts. Finally, it offers several APIs, which makes it possible to run the translation program in the background so that the users can invoke the translation functionality from their familiar text editor. For example, we have developed a prototype interface for Microsoft Word using Visual Basic macros.

The rest of the paper is organized as follows. We first provide a brief discussion of related work in Sect. 2 and an overview of the system architecture in Sect. 3. In Sect. 4 we describe the preprocessing stages to prepare the input for the acquisition and application of transfer rules. Section 5 gives a formal account of the three generic types of transfer rules that we use in our system along with several illustrative examples. The processing steps for the acquisition of new rules and the subsequent consolidation phase to avoid overtraining are presented in Sect. 6. Finally, Sect. 7 focusses on the application of the transfer rules during translation and the generation of the final natural language output. We close the paper with some concluding remarks and an outlook on future work in Sect. 8.

2. Related Work

Research on machine translation has a long tradition (for good overviews see [6, 7, 12, 21]). The state of the art in machine translation is that there are quite good solutions for narrow application domains with a limited vocabulary and concept space. However, it is the general opinion that fully automatic high quality translation without any limitations on the subject and without any human intervention is far beyond the scope of today's machine translation technology, and there is serious doubt that it will be ever possible in the future [8, 10].

It is very disappointing to have to notice that the translation quality has not much improved in the last 10 years [26]. One

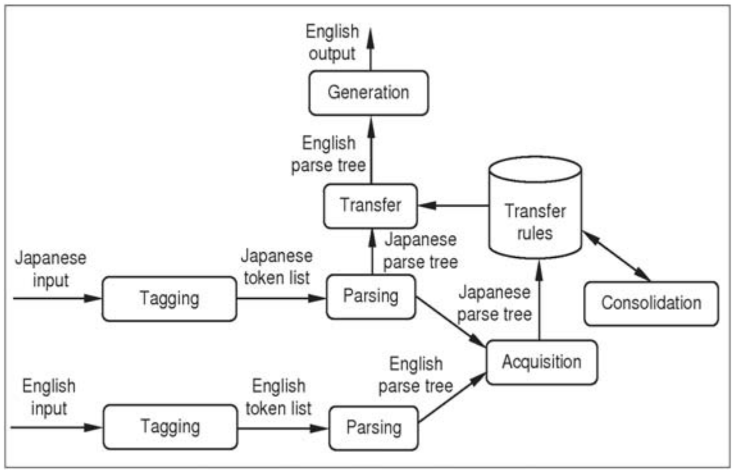


Figure 1. System architecture

main obstacle on the way to achieving better quality is seen in the fact that most of the current machine translation systems are not able to learn from their mistakes [9]. Most of the translation systems consist of large static rule bases with limited coverage, which have been compiled manually with huge intellectual effort. All the valuable effort spent by users on post-editing translation results is usually lost for future translations.

As a solution to this knowledge acquisition bottleneck, *corpus-based machine translation* tries to learn the transfer knowledge automatically on the basis of large bilingual corpora for the language pair [3, 15]. *Statistical machine translation* [2], in its pure form, uses no additional linguistic knowledge to train both a statistical translation and target language model. The two models are used to assign probabilities to translation candidates and then to choose the candidate with the maximum score. For the first few years the translation model was built only at the word level, however, as the limitations became apparent, several extensions towards phrase-based translation [16, 18, 22] and syntax-based translation [13, 28, 30] have been proposed. Although some improvements in the translation quality could be achieved, statistical machine translation has still one main disadvantage in common with rule-based translation, i.e. an incremental adaptation of the statistical model by the user is usually impossible.

The most prominent approach for the translation of Japanese has been *example-based machine translation* [11, 20, 25]. It uses a parallel corpus to create a database of translation examples for source language fragments. The different approaches vary in how they represent these fragments: as surface strings, structured representations, generalized templates with variables, etc. [1, 4, 5, 14]. However, most of the representations of translation examples used in example-based systems of reasonable size have to be manually crafted or at least reviewed for correctness [23] to achieve sufficient accuracy.

3. System Architecture

The three main tasks that we have to perform in our system are acquisition, consolidation, and translation. During *acquisition* (see Fig. 1) we derive new transfer rules by using a Japanese-English sentence pair as input. Both sentences are first analyzed by the *tagging* modules, which produce the correct segmentations into word tokens associated with their part-of-speech tags. The token lists are then transformed into parse trees by the *parsing* modules. The parse trees represent the input to the *acquisition* module, which uses a structural matching algorithm to discover new *transfer rules*. Whereas the transfer rules learnt during acquisition are very accurate and guarantee consistent translations, this specificity reduces the coverage for new unseen data. Therefore, the *consolidation* step generalizes transfer rules as long as such relaxations do not result in conflicts with other rules in the rule base.

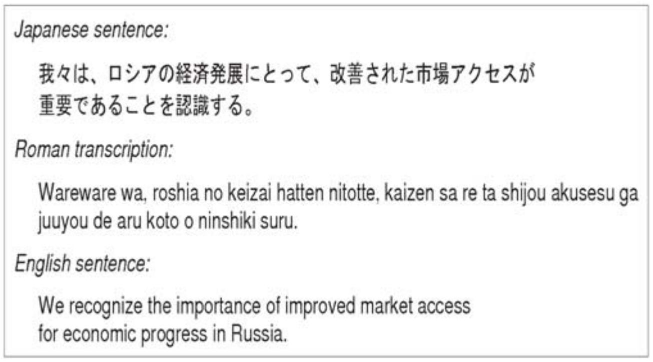
Finally, we perform the *translation* of a Japanese sentence by first tagging and parsing the sentence and then invoking the *transfer* module. It applies the transfer rules stored in the rule base to transform the Japanese parse tree into the corresponding English parse tree. The latter is the input to the *generation* module, which produces the surface form of the English sentence as character string. In the following sections we give a more detailed technical description of the individual modules. We illustrate their mode of operation by using the sentence pair in Fig. 2 as a running example throughout the rest of this paper.

4. Tagging and Parsing

We use Python scripts for the basic string operations to import the sentence pairs from the JENAAD corpus. For the part-of-speech tagging of Japanese sentences we use a Japanese lexicon, which was compiled automatically by applying the Japanese morphological analysis system ChaSen [19] to the JENAAD corpus. Figure 3 shows the result produced by the tagging module. It segments the Japanese input into morphemes and tags each morpheme with its pronunciation, base form, part-of-speech, conjugation type, and conjugation form. The last three columns are encoded as numerical categories. For the ease of the reader we have added Roman transcriptions, approximate English equivalents of the numerical categories, and contextual English translations.

The Japanese language uses postpositions instead of prepositions. The postposition O indicates the direct object, the theme particle WA the subject (in this sentence). The verb SURU is used to derive verbs from sahen nouns (e.g. recognize from recognition), and the phrase DE ARU KOTO changes an adjectival noun into a noun (e.g. important into importance). Finally, the verb suffix RERU indicates passive voice and the auxiliary verb TA past tense.

The acquisition of the Japanese lexicon is performed in two steps. We first create a lexicon entry for each Japanese word in the corpus based on the tags assigned by ChaSen. For words with several different tags, we store one default tag and, in a second step, learn word sense disambiguation rules based on the local context. We had to correct some wrong tag assignments by ChaSen, in particular regarding different uses of postpositions.



Japanese sentence:

我々は、ロシアの経済発展にとって、改善された市場アクセスが
重要であることを認識する。

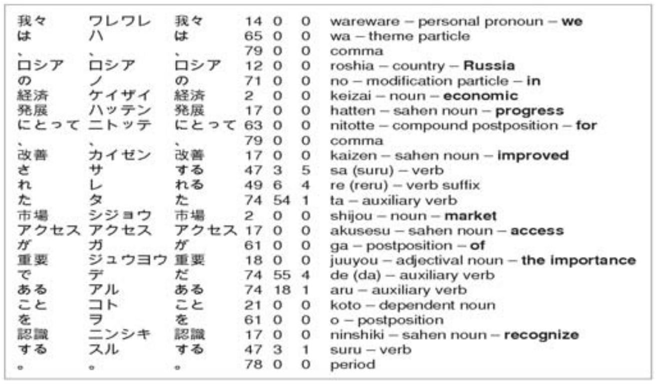
Roman transcription:

Wareware wa, roshia no keizai hatten nitotte, kaizen sa re ta shijou akusesu ga
juuyou de aru koto o ninshiki suru.

English sentence:

We recognize the importance of improved market access
for economic progress in Russia.

Figure 2. Example of sentence pair

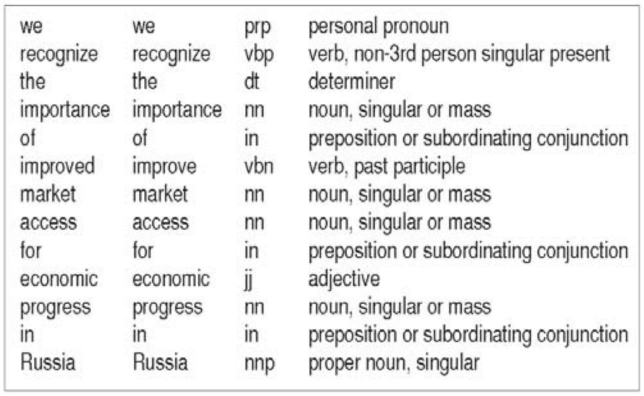


我々	ワレワレ	我々	14	0	0	wareware – personal pronoun – **we**
は	ハ	は	65	0	0	wa – theme particle
、	、	、	79	0	0	comma
ロシア	ロシア	ロシア	12	0	0	roshia – country – **Russia**
の	ノ	の	71	0	0	no – modification particle – **in**
経済	ケイザイ	経済	2	0	0	keizai – noun – **economic**
発展	ハッテン	発展	17	0	0	hatten – sahen noun – **progress**
にとって	ニトッテ	にとって	63	0	0	nitotte – compound postposition – **for**
、	、	、	79	0	0	comma
改善	カイゼン	改善	17	0	0	kaizen – sahen noun – **improved**
さ	サ	する	47	3	5	sa (suru) – verb
れ	レ	れる	49	6	4	re (reru) – verb suffix
た	タ	た	74	54	1	ta – auxiliary verb
市場	シジョウ	市場	2	0	0	shijou – noun – **market**
アクセス	アクセス	アクセス	17	0	0	akusesu – sahen noun – **access**
が	ガ	が	61	0	0	ga – postposition – **of**
重要	ジュウヨウ	重要	18	0	0	juuyou – adjectival noun – **the importance**
で	デ	だ	74	55	4	de (da) – auxiliary verb
ある	アル	ある	74	18	1	aru – auxiliary verb
こと	コト	こと	21	0	0	koto – dependent noun
を	ヲ	を	61	0	0	o – postposition
認識	ニンシキ	認識	17	0	0	ninshiki – sahen noun – **recognize**
する	スル	する	47	3	1	suru – verb
。	。	。	78	0	0	period

Figure 3. Example of Japanese token list

The English input is tagged by using an English lexicon, which is again automatically derived by applying the MontyTagger [17] to JENAAD. The tagging module segments the English input into morphemes, and tags each morpheme with its base form and part-of-speech tag from the Penn Treebank tagset [24]. Also the tag assignments by MontyTagger had to be corrected in several cases. Figure 4 shows the output for our example sentence, we added the verbose description of the part-of-speech tags for clarification.

The parsing modules compute the syntactic structure of sentences based on the information in the token lists. The Japanese and English grammars were learnt automatically from manually parsed sentences in the JENAAD corpus and are formulated as Definite Clause Grammar rules. A sentence is modeled as a list of constituents. A *constituent* is defined as a compound term of arity 1 with the *constituent category* as principal functor. We use three-letter acronyms to encode the constituent categories. Regarding the *argument* of a constituent we distinguish two types:



we	we	prp	personal pronoun
recognize	recognize	vbp	verb, non-3rd person singular present
the	the	dt	determiner
importance	importance	nn	noun, singular or mass
of	of	in	preposition or subordinating conjunction
improved	improve	vbn	verb, past participle
market	market	nn	noun, singular or mass
access	access	nn	noun, singular or mass
for	for	in	preposition or subordinating conjunction
economic	economic	jj	adjective
progress	progress	nn	noun, singular or mass
in	in	in	preposition or subordinating conjunction
Russia	Russia	nnp	proper noun, singular

Figure 4. Example of English token list

- *simple constituents* represent words or features (atom/atom or atom),
- *complex constituents* represent phrases modeled as lists of subconstituents.

Since the Japanese language uses postpositions and the general structure of a simple sentence is sentence-initial element, pre-verbal element, and verbal, it is much easier to parse a Japanese sentence from right to left. Therefore, we reverse the Japanese token list before we start with the parsing process. The output of the parser for our example sentence is (using Roman transcriptions of Japanese words):

```
[vbl([hea(SURU/47), hef(3/1), sjc(NINSHIKI/17)]),
dob([apo(O/61), hea(KOTO/21), mvp([vbl([hea(DA/74),
hef(55/4), aux([hea(ARU/74), hef(18/1)]),
cap([hea(JUUYOU/18)])]), sub([apo(GA/61), hea(AKUSESU/
17), mno([hea(SHIJOU/2)]), mvp([vbl([hea(SURU/47), hef(3/
5), aux([hea(RERU/49), hef(6/4)]), aux([hea(TA/74),
hef(54/1)]), sjc(KAIZEN/17)])])])])]), aob([apo(NITOTTE/
63), hea(HATTEN/17), mno([hea(KEIZAI/2)]), mnp([apo(NO/
71), hea(ROSHIA/12)])]), sub([apo(WA/65), hea(WAREWARE/
14)])]
```

Figure 5 provides a more human-readable presentation of the parse tree. We have added verbose descriptions of the constituent categories. The category "head form" encodes the conjugation type and conjugation form according to ChaSen. We use the categories "adposition" and "adpositional object" to also cover prepositions and prepositional objects in English parse trees.

The grammar acquisition is achieved by aligning the token lists with the parse trees and building a hierarchical graph as intermediate representation, which, at the end of the acquisition process, is translated into Definite Clause Grammar rules.

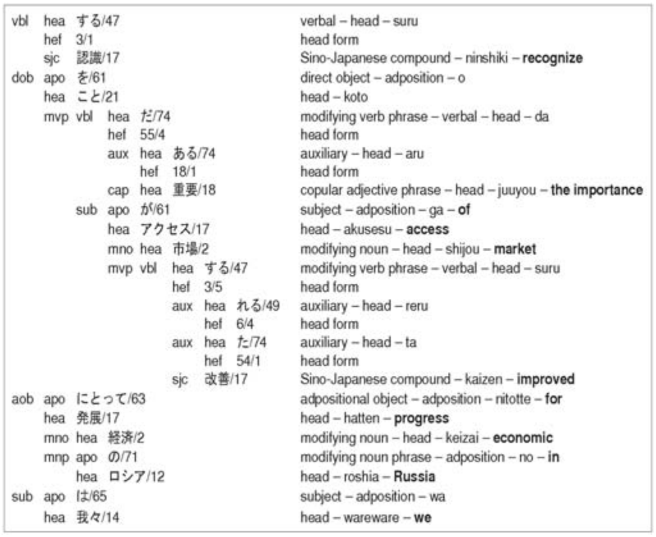
If a punctuation mark such as a comma is followed by a non-conjugated word, we use its tag as subtag for the following word to increase the discriminative power of the lexical tagging. We traverse the parse tree top-down and try to navigate the corresponding graph. Starting from an *initial node*, we follow *links* representing tags of words to *nodes* representing constituents. A link to a simple constituent removes the word from the token list, whereas a link to a complex constituent keeps the word in the list and navigates to the initial node of the corresponding subgraph. To leave a subgraph we follow an *exit link*, which again leaves the word intact and returns to the node from which the subgraph was called. All nodes and links are stored as dynamic facts to enable an easy incremental adaptation. If there is no existing link for a transition in the parse tree, we distinguish three cases:

- if the transition is an exit link or if the target node exists, then we add a new link,
- if the target node does not exist, then we also add a new node,
- if the target node represents a complex constituent and the corresponding subgraph does not exist, then we also add a new subgraph.

Figure 6 shows the hierarchical graph built from our example sentence. As can be seen we handle certain common complex constituents, e.g. *aux* or *mno*, which only contain *hea*, or *hea* and *hef* as subconstituents, like simple constituents to facilitate the acquisition process. We also learn rules to rearrange constituents for which the order in the token list does not correspond to the order in the parse tree. The resulting graph is highly non-deterministic, therefore, we replace ambiguous transitions with more specific conditions starting from words, word and tag sequences, two-word sequences, etc., until we reach a deterministic representation.

English sentences are parsed from left to right. To facilitate the structural matching between Japanese and English parse trees during acquisition we tried to align the use of constituent categories in the English grammar as best as possible with corresponding Japanese categories. In addition, we have chosen the same order of subconstituents as in the Japanese parse tree. Figure 7 shows the English parse tree for our example.

We have added explicit constituent categories for the determiner type (e.g. definite) and syntactic features such as number, comparison, tense, aspect, or voice. For each feature we have defined a default value (e.g. singular for number) so that we only indicate a feature in the parse tree if its value differs from the default value.



```
vbl  hea  する/47                              verbal – head – suru
     hef  3/1                                 head form
     sjc  認識/17                              Sino-Japanese compound – ninshiki – recognize
dob  apo  を/61                                direct object – adposition – o
     hea  こと/21                              head – koto
     mvp  vbl  hea  だ/74                      modifying verb phrase – verbal – head – da
               hef  55/4                      head form
               aux  hea  ある/74               auxiliary – head – aru
                    hef  18/1                 head form
               cap  hea  重要/18               copular adjective phrase – head – juuyou – the importance
          sub  apo  が/61                      subject – adposition – ga – of
               hea  アクセス/17                 head – akusesu – access
               mno  hea  市場/2                modifying noun – head – shijou – market
               mvp  vbl  hea  する/47          modifying verb phrase – verbal – head – suru
                         hef  3/5             head form
                         aux  hea  れる/49     auxiliary – head – reru
                              hef  6/4        head form
                         aux  hea  た/74       auxiliary – head – ta
                              hef  54/1       head form
                         sjc  改善/17          Sino-Japanese compound – kaizen – improved
aob  apo  にとって/63                           adpositional object – adposition – nitotte – for
     hea  発展/17                              head – hatten – progress
     mno  hea  経済/2                          modifying noun – head – keizai – economic
     mnp  apo  の/71                           modifying noun phrase – adposition – no – in
          hea  ロシア/12                        head – roshia – Russia
sub  apo  は/65                                subject – adposition – wa
     hea  我々/14                              head – wareware – we
```

Figure 5. Example of Japanese parse tree

5. Transfer Rules

One characteristic of our approach is that we model all translation problems with only three generic types of transfer rules. The transfer rules are stored as Prolog facts in the rule base. We have defined three Prolog predicates for the three different rule types. Therefore, for the rest of this paper, when we talk about "transfer rules" we always refer to transfer rules for machine translation in the more general sense, not to logical rules in the strict sense of Prolog terminology. We also use the terms "equal" and "equality" synonymously with "unifiable" and "unifiability".

In the next subsections we give an overview of the three different rule types along with illustrative examples. For the ease of the reader we use Roman transcription for the Japanese examples instead of the original Japanese characters.

5.1 Word Transfer Rules

For simple context-insensitive translations at the word level, the argument *A*1 of a simple constituent is changed to *A*2 by

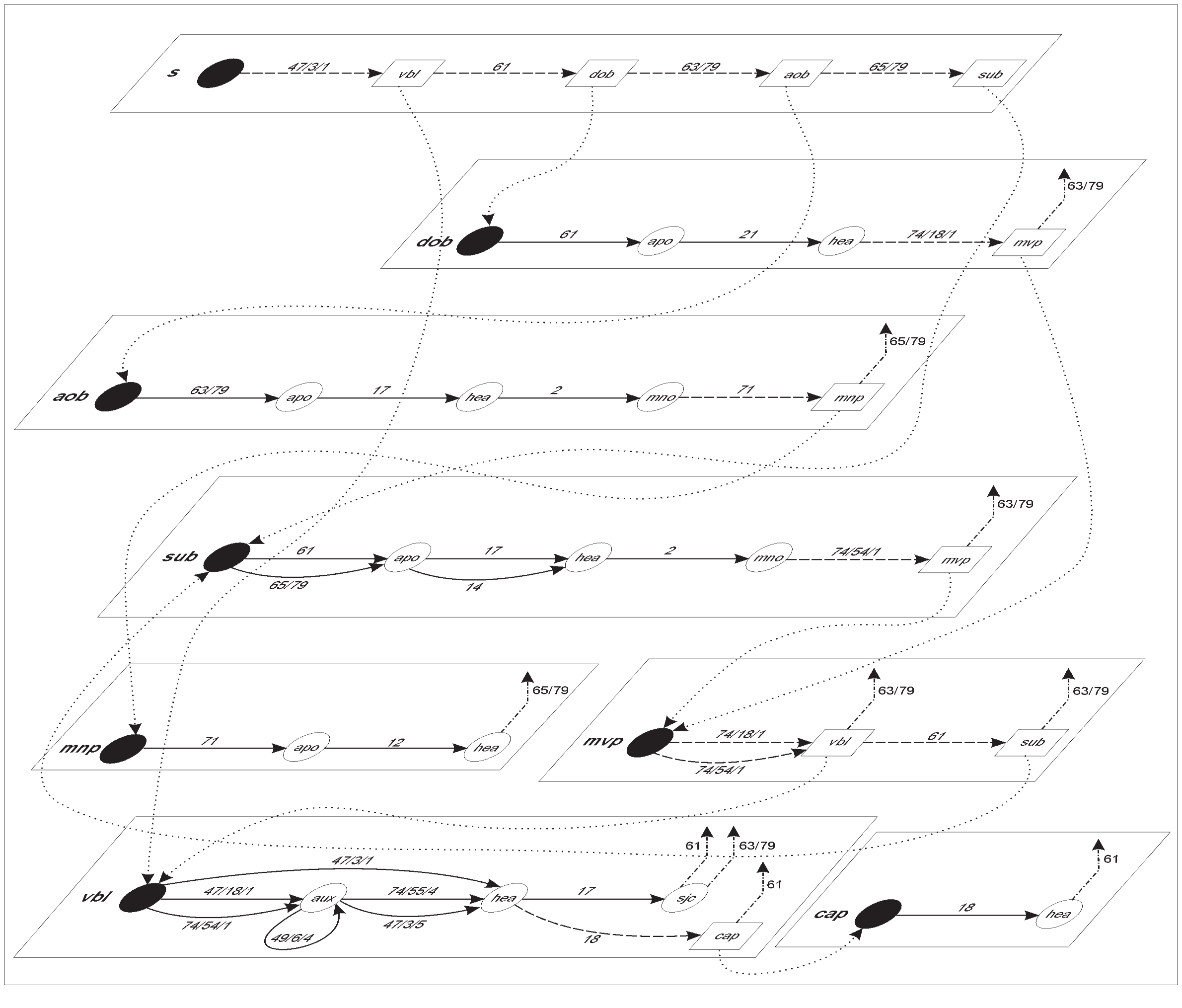


Figure 6. Example of grammar acquisition

applying the following predicate, i.e. if the argument of a simple constituent is equal to *argument condition A*1, it is replaced by *A*2:

wtr(*A*1, *A*2).

Example 1. The default transfer rule to translate the Japanese noun SEKAI into the English counterpart world is stated as the fact:

wtr(SEKAI/2, world/*nn*).

5.2 Constituent Transfer Rules

The second rule type concerns the translation of complex constituents to cover cases where both the category and the argument of a constituent have to be altered:

ctr(*C*1, *C*2, *Hea*, *A*1, *A*2).

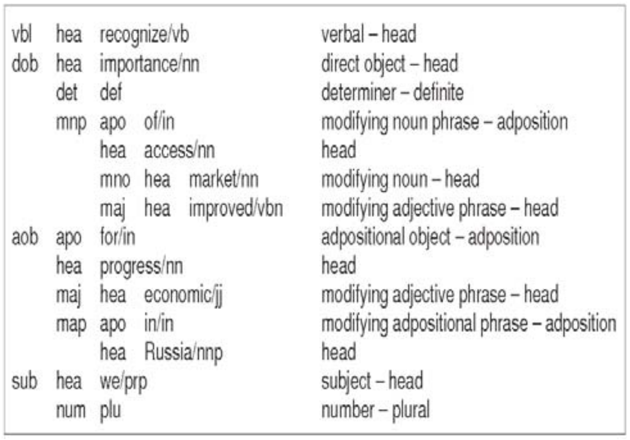
This changes a complex constituent *C*1(*A*1) to *C*2(*A*2) if the category is equal to *category condition C*1, the head is equal to *head condition Hea*, and the argument is equal to argument condition *A*1.

Example 2. The modifying noun (*mno*) with head KOKUSAI is translated as modifying adjective phrase (*maj*) with head international:

ctr(*mno*, *maj*, KOKUSAI/2, [*hea*(KOKUSAI/2)], [*hea*(international/*jj*)]).

The head condition serves as index for the fast retrieval of matching facts during the translation of a sentence and significantly reduces the number of facts for which the argument condition has to be tested.

Constituent transfer rules can contain shared variables for unification, which makes it possible to replace only certain parts of the argument and to leave the rest unchanged.



```
vbl  hea  recognize/vb                  verbal – head
dob  hea  importance/nn                 direct object – head
     det  def                           determiner – definite
     mnp  apo  of/in                    modifying noun phrase – adposition
          hea  access/nn                head
          mno  hea  market/nn           modifying noun – head
          maj  hea  improved/vbn        modifying adjective phrase – head
aob  apo  for/in                        adpositional object – adposition
     hea  progress/nn                   head
     maj  hea  economic/jj              modifying adjective phrase – head
     map  apo  in/in                    modifying adpositional phrase – adposition
          hea  Russia/nnp               head
sub  hea  we/prp                        subject – head
     num  plu                           number – plural
```

Figure 7. Example of English parse tree

Example 3. The Japanese modifying verb phrase (*mvp*) *X* NIMUKETA is translated as modifying adpositional phrase (*map*) toward *X*. In more detail, the Japanese modifying verb phrase contains a verbal with head MUKERU and auxiliary TA and an adpositional object with adposition NI:

ctr(*mvp*, *map*, MUKERU/47,
[*vbl*([*hea*(MUKERU/47),*hef*(6/4), *aux*([*hea*(TA), *hef*(54/1)])]), *aob*([*apo*(NI/61)| *X*])],
[*apo*(toward/*in*)| *X*]).

*C*1(*A*1) = *mvp*([*vbl*([*hea*(MUKERU/47), *hef*(6/4), *aux*([*hea*(TA), *hef*(54/1)])]),
aob([*apo*(NI/61), *hea*(MINSHU/2), *suf*(KA/31)])])

*C*2(*A*2) = *map*([*apo*(toward/*in*), *hea*(MINSHU/2), *suf*(KA/31)])

As can be seen, if we have to translate the phrase MINSHUKA NI MUKETA (toward democratization), the application of the above rule only translates NI MUKETA and leaves the translation of MINSHUKA to another transfer rule.

5.3 Phrase Transfer Rules

The most common and most versatile type of transfer rules are phrase transfer rules, which allow to define elaborate conditions and substitutions on phrases, i.e. arguments of complex constituents:

ptr(*C*, *Hea*, *Req*1, *Req*2).

Rules of this type change the argument of a complex constituent with category *C* from $A1 = Req1 \cup Add$ to $A2 = Req2 \cup Add$ if $hea(Hea) \in A1$. To enable the flexible application of phrase transfer rules, input *A*1 and argument condition *Req*1 are treated as sets and not as lists of subconstituents, i.e. the order of subconstituents does not affect the satisfiability of the argument condition. The application of a transfer rule requires that the set of subconstituents in *Req*1 is included in the argument *A*1 of the input constituent *C*(*A*1) to replace *Req*1 by *Req*2. Besides *Req*1 any additional constituents can be included in the input, which are transferred to the output unchanged. This allows for an efficient and robust realization of the transfer module because one rule application changes only certain aspects of a phrase whereas other aspects can be translated by other rules in subsequent steps. It is also possible to use the special constant notex as argument of a subconstituent in *Req*1, e.g. *sub*(notex). In that case the rule can only be applied if no subconstituent of this category is included in *A*1, e.g. if *A*1 includes no subject.

In addition to an exact match the generalized constituent categories *np* (noun phrase) and *vp* (verb phrase) can be used in the category condition, i.e. the condition is satisfied if the constituent category *C* is subsumed by the generalized category (e.g. $mvp \sqsubseteq vp$).

The head condition is again used to speed up the selection of possible candidates during the transfer step. If the applicability of a transfer rule does not depend on the head of the phrase, then the special constant nil is used as head condition. Another special case is the head condition notex. In analogy to the corresponding use in the argument condition this indicates that the rule can only be applied if *A*1 does not contain a head element.

Example 4. The Japanese verbal with head SURU and Sino-Japanese compound NINSHIKI is translated into an English verbal with head recognize:

ptr(*vbl*, SURU/47, [*hea*(SURU/47), *sjc*(NINSHIKI/17)], [*hea*(recognize/*vb*)]).

*A*1 = [*hea*(SURU/47), *hef*(3/1), *sjc*(NINSHIKI/17)]

*A*2 = [*hea*(recognize/*vb*), *hef*(3/1)]

As explained before, the order of the elements in *A*1 is of no importance, the rule is applied to *A*1 and the additional element *hef*(3/1) is added to the elements in *Req*2.

Just as in the case of constituent transfer rules, also the expressiveness of phrase transfer rules can be increased significantly by using shared variables for unification.

Example 5. The following rule states that a noun phrase with head KOTO and a modifying verb phrase with verbal JUUYOU DE ARU and a subject *X* is translated into a noun phrase with head importance, definite determiner, and a modifying noun phrase of *X*:

```
ptr(np, KOTO/21,
     [hea(KOTO/21), mvp([vbl([hea(DA/74, hef(55/4),
          aux([hea(ARU/74), hef(18/1)]), cap([hea(JUUYOU/18)])]),
          sub([apo(GA/61)| X ])])],
     [hea(importance/nn), det(def ), mnp([apo(of/in)| X ])]).
  A1 = [hea(KOTO/21), mvp([vbl([hea(DA/74, hef(55/4),
          aux([hea(ARU/74),hef(18/1)]), cap([hea(JUUYOU/18)])]),
          sub([hea(AKUSESU/17), apo(GA/61), mno(Y), mvp(Z)])])]
  A2 = [hea(importance/nn), det(def ), mnp([apo(of/in),
     hea(AKUSESU/17), mno(Y), mvp(Z)])]
```

The variables *Y*, *Z* are used for the convenience of the reader to shorten the example. An important point that becomes obvious from the example is that the set property for the argument condition does not only apply to the top level of *A*1 but extends recursively to any level of detail specified in *Req*1, e.g. to the subconstituents of *sub* in this example.

6. Acquisition and Consolidation

The acquisition module traverses the Japanese and English parse trees and derives new transfer rules, which are added to the rule base. We start the search for new rules at the sentence level by calling vp_match(vp, JapSent, EngSent). This predicate matches two verb phrases VPJ and VPE, the constituent category C is required for the category condition in the transfer rules:

```
vp_match(C, VPJ, VPE) :-
     reverse(VPJ, VPJR),
     reverse(VPE, VPER),
     vp_map(C, VPJR, VPER).
```

The predicate first reverses the two lists so that the leftmost constituents (in the sentences) are examined first, which facilitates the correct mapping of subconstituents with identical constituent category, e.g. several modifying nouns. It then calls the predicate vp_map, which is implemented as recursive predicate for the correct mapping of the individual subconstituents of VPJ:

```
vp_map(_, [], []).
...
vp_map(C, VPJ, VPE) :-
   map_dob(C, VPJ, VPE, VPJ2, VPE2),
  vp_map(C, VPJ2, VPE2).
...
vp_map(_, _, _).
```

Each rule for the predicate vp_map is responsible for the mapping of a specific Japanese subconstituent (possibly together with other subconstituents), e.g. map_dob looks for a subconstituent with category dob in VPJ and tries to derive a transfer rule to produce the corresponding translation in VPE. All subconstituents in VPJ and VPE that are covered by the new transfer rule are removed from the two lists to produce VPJ2 and VPE2. In that way all subconstituents are examined until the lists are empty or no more new rules can be found. Each derived rule is added to the rule base if it is not included yet.

Each predicate of type map_dob for the mapping of the individual subconstituents both covers special mappings as well as the default treatment:

```
...
     map_dob(_, VPJ, VPE, VPJ2, VPE2)
:-
          map_default(dob, VPJ, VPE,
VPJ2, VPE2).
     ...
map_default(C, J, E, J2, E2) :-
          remove_constituent(C, J,
ArgJ, J2),
          remove_constituent(C, E,
ArgE, E2),
          map_argument(C, ArgJ, ArgE).
     ...
     map_argument(dob, J, E) :-
          np_match(dob, J, E).
```

For the default mapping of direct objects both phrases must contain a subconstituent with category dob. The subconstituents are removed from the lists by calling the predicate remove_constituent. This predicate returns the argument of the removed constituent, it fails if the constituent does not exist. Finally, np_match is called, which is defined in analogy to vp_match, in order to derive transfer rules for the subconstituents of the two arguments.

The transfer rules that are derived by the acquisition module are very specific because they consider all context-dependent translation dependencies in full detail to avoid any conflict with existing rules in the rule base. This guarantees correct translations but leads to a huge number of complex rules, which has negative effects on computational efficiency. It also badly affects the coverage for unseen sentences. To avoid this overtraining we perform a consolidation step to prune the transfer rules as long as such new generalized rules are not in conflict with other rules. The relaxation of rules mainly concerns contextual translation dependencies of adpositions, head nouns, determiners, the number feature, and verbals. The most commonly performed transformations are:

- to simplify a phrase transfer rule or to replace it with a word transfer rule,
- to use the generalized categories np or vp in the category condition,
- to split a phrase transfer rule in two simpler rules.

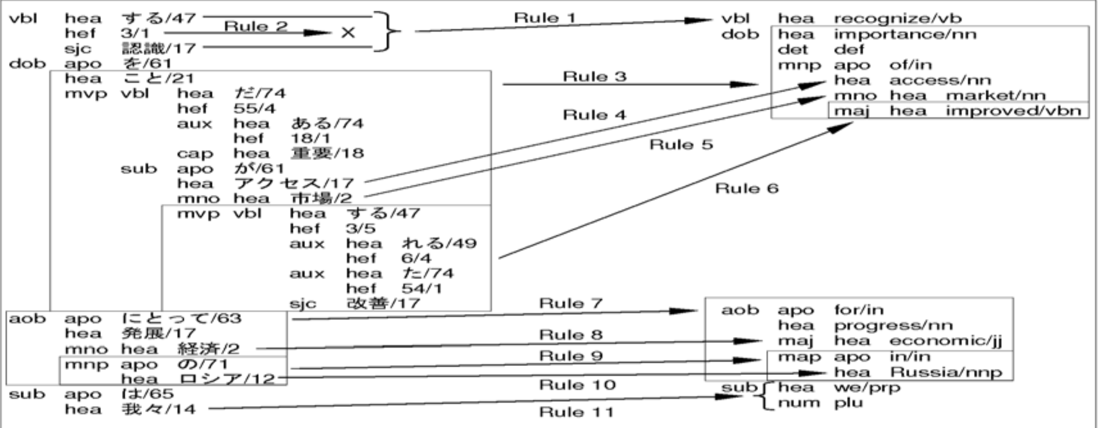
Figure 8 shows the transfer rules that were derived from our translation example. Rule 4 and Rule 10 are word transfer rules that were produced by the consolidation module (the original rules are struck out and written in angle brackets).

7. Transfer and Generation

The transfer module traverses the Japanese parse tree top-down and searches for transfer rules that can be applied. The chosen design of the transfer rules guarantees the robust processing of the parse tree. One rule only changes certain parts of a constituent into the English equivalent, other parts are left unchanged to be transformed by other rules during subsequent processing steps. Therefore, our transfer algorithm is able to work efficiently on a mixed Japanese-English parse tree, which gradually turns into a fully translated English parse tree.

At the top level we first apply phrase transfer rules (predicate apply_ptrules) to the sentence before we try to translate each constituent in the sentence individually (predicate transfer_const):

```
  transfer(JapSent, EngSent) :-
   apply_ptrules(vp, JapSent, IntermediateResult),
   transfer_const(IntermediateResult, EngSent).
 apply_ptrules(C, JapSent, EngSent) :-
   apply_ptr(C, JapSent, IntermediateResult),
   apply_ptrules(C, IntermediateResult, EngSent).
 apply_ptrules(_, Sent, Sent).
```



1. ptr(vbl, する/47, [hea(する/47), sjc(認識/17)], [hea(recognize/vb)]).
2. ptr(vbl, nil, [hef(3/1)], '[]').
3. ptr(np, こと/21, [hea(こと/21), mvp([vbl([hea(だ/74), hef(55/4), aux([hea(ある/74), hef(18/1)]), cap([hea(重要/18)])]), sub([apo(が/61) | X])])], [hea(importance/nn), det(def), mnp([apo(of/in) | X])]).
4. wtr(アクセス/17, access/nn). 〈~~ptr(np, アクセス/17, [hea(アクセス/17)], [hea(access/nn)]).~~〉
5. wtr(市場/2, market/nn).
6. ctr(mvp, maj, する/47, [vbl([hea(する/47), hef(3/5), aux([hea(れる/49), hef(6/4)]), aux([hea(た/74), hef(54/1)]), sjc(改善/17)])], [hea(improved/vbn)]).
7. ptr(vp, する/47, [aob([apo(にとって/63), hea(発展/17) | X])], [aob([apo(for/in), hea(progress/nn) | X])]).
8. ctr(mno, maj, 経済/2, [hea(経済/2)], [hea(economic/jj)]).
9. ptr(np, progress/nn, [mnp(X)], [map([apo(in/in) | X])]).
10. wtr(ロシア/12, 'Russia'/nnp) 〈~~ptr(np, ロシア/12, [hea(ロシア/12)], [hea('Russia'/nnp)]).~~〉
11. ptr(np, 我々/14, [hea('我々'/14)], [hea(we/prp), num(plu)]).

Figure 8. Example of transfer rules

The predicate apply_ptrules applies phrase transfer rules recursively until no further rule can be applied successfully. The application of a single phrase transfer rule (predicate apply_ptr) is divided in two steps. First, we select all rule candidates that satisfy the category, head, and argument conditions in the rule. Second, we rate each rule and choose the one with the highest score. The score is calculated based on the complexity of the argument condition. In addition, rules are ranked higher if:

- the head condition is not nil,
- the argument condition does not depend on the head,
- the argument condition contains notex.

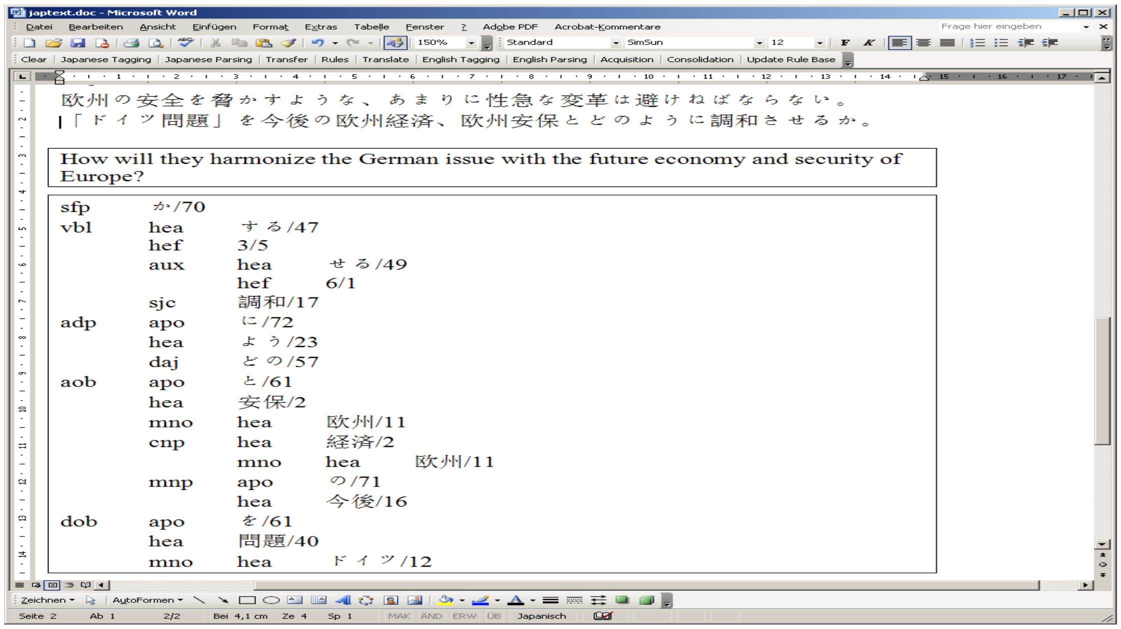
The most challenging task for selecting rule candidates is the verification of the argument condition because this involves testing for set inclusion (argument condition ⊆ input) at the top level as well as recursively testing for set equality of arguments of subconstituents. This is achieved by using the predicate split, which retrieves each element in the argument condition AC from the input I (at the same time binding free variables through unification) and returns the remaining constituents from the input as list of additional elements Add, which are then appended to the translation of the argument condition:

```
 split(I, AC, Add) :-
      once(split_rec(I, AC, AC, Add)).
      split_rec(Add, [], [], Add).
split_rec(I,[ConstAC|RestAC],[ConstAC2|RestAC2],
                                           Add):-
      once(retrieve_const(ConstAC, I, ConstAC2,
                                            I2)),
      split_rec(I2, RestAC, RestAC2, Add).
 retrieve_const(Const,[Const|RestI],Const,RestI). %(1)
retrieve_const(ConstAC, [ConstI|RestI],
                         ConstAC, RestI):-%(2)
      ConstAC =.. [Category, ArgAC],
      ConstI =..  [Category,  ArgI],
      equal_args(ArgI,  ArgAC).
 retrieve_const(ConstAC,
    [ConstI|RestI], ConstAC2,
                   [ConstI|RestI2]):-
  retrieve_const(ConstAC,  RestI,
                              ConstAC2, RestI2).
  equal_args(ArgI, ArgAC) :-
         once(unify_args(ArgI, ArgAC, ArgAC)).
  unify_args(ArgI, ArgAC, ArgAC2) :-   %(3)
         var(ArgAC), ArgAC2 = ArgI.
  unify_args([], [], []).   %(4)
  unify_args(ArgI, [ConstArgAC | RestArgAC],
              [ConstArgAC2 | RestArgAC2]) :-
    once(retrieve_const(ConstArgAC, ArgI,
                            ConstArgAC2, ArgI2)),
     unify_args(ArgI2, RestArgAC, RestArgAC2).
```

A constituent can be retrieved from the input, if the corresponding element from the argument condition can be (1) directly unified or (2) if the two categories are identical and the two arguments are equal sets. The equality of the arguments is tested by retrieving the argument condition subconstituents from the input argument until either (3) a free variable as tail (i.e. |X]) or (4) the end of the list is reached. For the convenience of the reader we have not shown the correct treatment of using notex in the argument condition, i.e. we also check that no such subconstituent is included in the input.

After applying phrase transfer rules at the sentence level, the predicate transfer_const examines each individual subconstituent. It first tries to apply constituent transfer rules before calling a predicate trans(C, JapArg, EngArg) for the category-specific transfer of the argument. For simple constituents this means the application of a word transfer rule, for complex constituents it involves again the application of phrase transfer rules (apply_ptrules), the recursive call of the predicate transfer_const, and some post-editing, e.g. removing the theme particle from a subject.

As last processing step of a translation, the generation module generates the surface form of the sentence as character string. For that purpose we traverse again the parse tree in a top-down fashion and transform the argument of each complex constituent into a list of surface strings, which


japtext.doc - Microsoft Word
Datei Bearbeiten Ansicht Einfügen Format Extras Tabelle Fenster ? Adobe PDF Acrobat-Kommentare
Frage hier eingeben
150% Standard SimSun 12
Clear | Japanese Tagging | Japanese Parsing | Transfer | Rules | Translate | English Tagging | English Parsing | Acquisition | Consolidation | Update Rule Base
Zeichnen AutoFormen
Seite 2 Ab 1 2/2 Bei 4,1 cm Ze 4 Sp 1 MAK ÄND ERW ÜB Japanisch

欧州の安全を脅かすような、あまりに性急な変革は避けねばならない。
「ドイツ問題」を今後の欧州経済、欧州安保とどのように調和させるか。

How will they harmonize the German issue with the future economy and security of Europe?

```
sfp     か/70
vbl     hea     する/47
        hef     3/5
        aux     hea     せる/49
                hef     6/1
        sjc     調和/17
adp     apo     に/72
        hea     よう/23
        daj     どの/57
aob     apo     と/61
        hea     安保/2
        mno     hea     欧州/11
        cnp     hea     経済/2
                mno     hea     欧州/11
        mnp     apo     の/71
                hea     今後/16
dob     apo     を/61
        hea     問題/40
        mno     hea     ドイツ/12
```

Figure 9. Example of user interface

is computed recursively from its subconstituents as nested list and flattened afterwards. The correct surface forms for words with irregular inflections is determined by accessing the English lexicon.

Since the order of the subconstituents in the argument of a complex constituent could have been arbitrarily rearranged through the application of phrase transfer rules, the generation module cannot derive the original sequence of several subconstituents with identical category (e.g. several modifying adjective phrases) from the information in the parse tree. However, to maintain the original sequence in the translation is an important default choice in such a case. Therefore, we have added an additional processing step after parsing a Japanese source sentence in which we add a sequence number as simple constituent seq(Seq) to each argument of a complex constituent. As a consequence we had to extend the transfer component so that it ignores but preserves this sequence information during the application of transfer rules.

8. Conclusion

In this paper we have presented JETCAT, a Japanese-English machine translation system based on the automatic acquisition of transfer rules from a parallel corpus. We have finished the implementation of the system and demonstrated the feasibility of the approach based on a small subset of the JENAAD corpus.

We also have just started implementing a Web interface using Java Server Faces with interactive features to manipulate lexical, syntactic, and translation knowledge. For example, it will be possible to correct parse trees by simple drag-and-drop and editing actions at the Web client. After submitting the changes, the grammar will be instantly updated. In the same way we plan to customize lexical data and transfer rules. The user only has to revise translation results to customize the rule base according to his personal preferences.We have also developed a convenient prototype interface to Microsoft Word (see Fig. 9 for an example screenshot). The user can simply click on a sentence and translate it. In addition, it is possible to view the token lists and parse trees for both source and target language, the application sequence of transfer rules during translation, and the resulting parse tree after the transfer step. Finally, the user can choose a Japanese-English sentence pair and watch the details of the acquisition and consolidation process. This enables the user to correct the result of a translation and verify the consequences for acquisition and consolidation before updating the rule base. The resulting system represents a valuable tool for language learning and customized machine translation.

Future work will focus on extending the coverage of the system so that we can process the full JENAAD corpus and perform a thorough quantitative evaluation of the translation quality using tenfold cross-validation. We also plan to make our system available to students of Japanese Studies at our university in order to receive valuable feedback from practical use.We also have just started implementing a Web interface using Java Server Faces with interactive features to manipulate lexical, syntactic, and translation knowledge. For example, it will be possible to correct parse trees by simple drag-and-drop and editing actions at the Web client. After submitting the changes, the grammar will be instantly updated. In the same way we plan to customize lexical data and transfer rules. The user only has to revise translation results to customize the rule base according to his personal preferences.

We have also agreed on a new research collaboration with Korean researchers from Korea Advanced Institute of Science and Technology (KAIST) in September 2007. Due to the similar nature of Japanese and Korean there is a high potential for a straightforward portation to Korean-English machine translation. One important intended extension of the machine translation framework is the incorporation of ontological knowledge to improve translation quality and to address advanced text understanding tasks such as question answering or summarization. Finally, the translation knowledge shall serve as a valuable resource for ontology mining and refinement.

References

[1] Brockett, C. et al. (2002). English-Japanese example-based machine translation using abstract linguistic representations. Proceedings of the COLING-2002 Workshop on Machine Translation in Asia.

[2] Brown, P. (1990). A statistical approach to machine translation, Computational Linguistics, 16 (2) 79–85.

[3] Carl, M. (1999). Toward a model of competence for corpus-based machine translation. *In*: Hybrid approaches to machine translation, ser. IAI Working Papers. Streiter, O., Carl, M., & Haller, J. (Eds.). IAI. vol. 36.

[4] Carl, C., Way, A. (Eds.). (2003). Recent advances in example-based machine translation. Kluwer.

[5] Furuse, O., Iida, H. (1992). Cooperation between transfer and analysis in example-based framework, Proceedings of the 14th International Conference on Computational Linguistics. 645–651.

[6] Hutchins, J. (1986). Machine translation: Past, present, future. Chichester: Ellis Horwood.

[7] Hutchins, J. (2001). Machine translation over 50 years. Histoire epistémologie langage*,* 23 (1) 7–31.

[8] Hutchins, J. (2003). Has machine translation improved? Some historical comparisons, *In*: Proceedings of the 9th MT Summit. 181–188.

[9] Hutchins, J. (2004). Machine translation and computer-based translation tools: What's available and how it's used. In: A new spectrum of translation studies. (p. 13–48). Bravo, J.M. (Ed.). Valladolid: University of Valladolid.

[10] Hutchins, J. (2005). Current commercial machine translation systems and computer-based translation tools. International Journal of Translation, 17 (1–2).

[11] Hutchins, J. (2005). Towards a definition of example-based machine translation. Proceedings of the 2nd Workshop on Example-Based Machine Translation at MT Summit X. Phuket, Thailand. 63–70.

[12] Hutchins, J., Somers, H. (1992). An introduction to machine translation. London: Academic Press.

[13] Imamura, K. (2001). Hierarchical phrase alignment harmonized with parsing, *In*: Proceedings of the 6th Natural Language Processing Pacific Rim Symposium. Tokyo, Japan.

[14] Kaji, H., Kida, Y., Morimoto, Y. (1992). Learning translation examples from bilingual text. Proceedings of the 14th International Conference on Computational Linguistics. 672–678.

[15] Knight, K. (1997). Automatic knowledge acquisition for machine translation. *AI Magazine* 18 (4) 81–96.

[16] Koehn, P., Och, F.J., Marcu, D. (2003). Statistical phrase-based translation. Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology. 48–54.

[17] Liu, H. (2004). MontyLingua: An end-to-end natural language processor with common sense. MIT Media Lab.

[18] Marcu, D., Wong, W. (2002). A phrase-based, joint probability model for statistical machine translation, *In*: Proceedings of the Conference on Empirical Methods in Natural Language Processing. 133–139.

[19] Matsumoto, Y. et al. (1999). Japanese morphological analysis system ChaSen version 2.0 manual. NAIST Technical Report, NAIST-IS-TR99009.

[20] Nagao, M. (1984). A framework of a mechanical translation between Japanese and English by analogy. In: Artificial and human intelligence (p. 173–180). Elithorn, A., & Banerji, R. (Eds.). Amsterdam: North Holland.

[21] Newton, J. (Ed.). (1992). Computers in translation: A practical appraisal. London: Routledge.

[22] Och, F.J., Tillmann, C., Ney, H. (1999). Improved alignment models for statistical machine translation. Proceedings of the Joint Conference of Empirical Methods in Natural Language Processing and Very Large Corpora. 20–28.

[23] Richardson, S. et al. (2001). Overcoming the customization bottleneck using example-based MT. Proceedings of the ACL Workshop on Data-driven Machine Translation. 9–16.

[24] Santorini, B. (1990). Part-of-speech tagging guidelines for the Penn Treebank project. Technical Report MS-CIS-90-47. Department of Computer and Information Science, University of Pennsylvania, 3rd revision, 2nd printing.

[25] Sato, S. (1991). Example-based machine translation. Ph.D Dissertation, Kyoto University.

[26] Somers, H. (Ed.). (2003). Computers and translation: A translator's guide. Amsterdam: John Benjamins.

[27] Utiyama, M., Isahara, H. (2003). Reliable measures for aligning Japanese-English news articles and sentences. Proceedings of the 41st Annual Meeting of the ACL. 72–79.

[28] Watanabe, T., Imamura, K., Sumita, E. (2002). Statistical machine translation based on hierarchical phrase alignment. Proceedings of the 9th International Conference on Theoretical and Methodological Issues in Machine Translation. 188–198.

[29] Winiwarter, W. (2005). Incremental learning of transfer rules for customized machine translation. In: Applications ofdeclarative programming and knowledge management (p. 47–64). Seipel, U. et al. (Eds.). LNAI, vol. 3392. Berlin: Springer-Verlag.

[30] Yamada, K. (2002). A syntax-based statistical translation model. Ph.D Dissertation. University of Southern California.

Biographical Note

Prof. Dr. Werner Winiwarter is the Vice Head of the Department of Scientific Computing, University of Vienna, Austria. He received his MS degree in 1990, his MA degree in 1992, and his PhD degree in 1995, all from the University of Vienna, Austria. The main research interest of Prof. Winiwarter is human language technology, in particular machine translation and computer-assisted language learning. In addition, he also works on data mining and machine learning, Semantic Web, information retrieval, electronic business, and education systems.