Incremental Learning of Transfer Rules for Customized Machine Translation

Werner Winiwarter

Faculty of Computer Science,
University of Vienna, Liebiggasse 4, A-1010 Vienna, Austria,
werner.winiwarter@univie.ac.at,
WWW home page: http://www.ifs.univie.ac.at/~ww/

Abstract. In this paper we present a machine translation system, which translates Japanese into German. We have developed a transfer-based architecture in which the transfer rules are learnt incrementally from translation examples provided by a user. This means that there are no handcrafted rules, but, on the contrary, the user can customize the system according to his own preferences. The translation system has been implemented by using Amzi! Prolog. This programming environment had the big advantage of offering sufficient scalability even for large lexicons and rule bases, powerful unification operations for the application of transfer rules, and full Unicode support for Japanese characters. Finally, the application programming interface to Visual Basic made it possible to design an embedded translation environment so that the user can use Microsoft Word to work with the Japanese text and invoke the translation features directly from within the text editor. We have integrated the machine translation system into a language learning environment for German-speaking language students to create a Personal Embedded Translation and Reading Assistant (PETRA).

1 Introduction

For language students and other people interested in Japanese documents, the Web makes available a wealth of information. In general, after reaching a certain level of linguistic competence in a foreign language, the reading of written material represents an excellent way to improve the fluency by learning new terminology or grammatical structures in their natural context with comparatively little effort.

However, this approach to language acquisition, which works so well with many languages, is seriously hampered by the complexity of the Japanese writing system. Japanese texts are a mixture of the two syllabaries *hiragana* and *katakana* as well as the Japanese versions of Chinese characters called *kanji*. The two syllabaries are relatively easy to learn with only 46 different characters each, but there are several thousand, mostly quite complex kanji of which the pronunciations or *readings* often depend on the textual context. Another severe problem in Japanese is that the individual words are not separated by spaces so that the reader has to guess the word boundaries.

All these difficulties make reading and translating Japanese sentences a cumbersome and tedious process. If the reader reaches an inscrutable text passage, he must first guess where an unknown word starts and then consult a dictionary. To look up the word in a bilingual dictionary is quite straightforward as long as the reader is sure about the correct pronunciation, otherwise he has to consult a kanji dictionary, which lists kanji and their readings categorized by 214 basic elements or *radicals*. The retrieval of this kanji information is again a time-consuming task, especially because the radicals appear in different shapes depending on the position within the kanji.

Online documents have the great advantage that they enable the use of convenient tools, which assist the reader in comprehending the meaning of the Japanese text. Today, there exist several Web sites that offer information about kanji as well as English or German translations of Japanese words as pop-up hints just by pointing with the mouse at a certain text position, e.g. POPjisho[1] or Rikai[2]. Even if these tools are very useful, there are still often problems with the correct segmentation and the retrieval of conjugated words.

In a previous project we developed a reading tool for the use within Microsoft Word. We implemented this environment by using Amzi! Prolog, which provides full Unicode support so that Japanese characters can be used freely in the Prolog source code. Its application programming interface to Visual Basic enabled us to embed the Prolog program into the text editor. The implemented functionality of our reading tool included correct segmentation, the lookup of conjugated words, and the addition of new word definitions. This application represented also an evaluation of the scalability of Amzi! Prolog. We could achieve excellent performance although we searched 6,355 entries extracted from the kanji dictionary KANJIDIC[3], 100,014 entries from the Japanese-English dictionary EDICT[4], and 190,251 entries from the Japanese-German dictionary WaDokuJT[5].

Another, less satisfying observation with using our reading environment was that even with all this information available, it was still often not possible to correctly reproduce the intended meaning of a Japanese text. The main reason for this lies in the complexity of the translation task for the language pair Japanese–German caused by the very different grammars of the two languages. Whereas German grammar has a very specific system of declensions and conjugations to express number, gender, case, tense, mood, voice, etc., Japanese is highly ambiguous regarding most of these features, e.g. there exist no articles to indicate gender or definiteness, no declension to indicate number or case, and only two tenses. The ambiguity is further increased dramatically by the extensive use of ellipsis in Japanese. Therefore, a machine translation system requires sophisticated disambiguation techniques [1, 17, 19, 20] and anaphoric resolution strategies [12, 16, 18, 27].

---

[1] `http://www.popjisyo.com.`
[2] `http://www.rikai.com/perl/Home.pl.`
[3] `http://www.csse.monash.edu.au/~jwb/kanjidic.html.`
[4] `http://www.csse.monash.edu.au/~jwb/edict.html.`
[5] `http://www.wadoku.de.`

Instead of a lengthy discussion of the state of the art of systems available for Japanese translation, we show the results of an entertaining experiment in Fig. 1. The figure lists the attempts of several machine translation programs to translate a sentence about producing a parchment codex. We could only find one program that also translates into German, all others translate only into English. All the examples are taken from free online translation Web sites, except the last entry, which was produced by a commercial product.

As can be seen, the results are far from satisfactory. All the systems are certainly not suitable for fully automatic high quality machine translation. It is sometimes even hard or impossible to grasp the exact meaning of a Japanese sentence from the mutilated translations.

This unsatisfactory situation was the motivation for us to meet the challenge of developing a high quality machine translation system from Japanese into German. In our approach the system learns the transfer rules incrementally from translation examples by using structural matching between the syntax trees. This way the user can customize the system according to his personal preferences. If the user is not satisfied with a translation result, he can simply correct the translation and activate the adaptive learning module, which results in an update of the translation rule base.

We have integrated our machine translation system with the previously developed reading tool to create the *Personal Embedded Translation and Reading Assistant (PETRA)*. PETRA's main aim is to assist German-speaking language students in reading and translating Japanese documents. PETRA offers the students valuable information, which the students apply to solve the translation task at hand. This encourages a bidirectional knowledge transfer so that the students play an active role during their whole interaction with PETRA. Therefore, studying Japanese becomes more interesting and entertaining.

The rest of the paper is organized as follows. In Sect. 2 we first provide a brief discussion of related work. Then we give an overview of the system architecture in Sect. 3 before we describe the technical details of the individual components of our translation environment in Sect. 4, i.e. tokenization, parsing, learning, transfer, and generation. Finally, we close the paper with concluding remarks and an outlook on future work.

2 Related Work

Research on machine translation has a long tradition (for good overviews see [7–9, 11, 22]). 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. For more general use only systems for very similar language pairs promise to produce output that is acceptable or at least comprehensible. 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 [10].

Japanese sentence:

これは、片面だけに字を書いて、同じ大きさに切りそろえたものを、
何枚も革のひもでとじた。

Roman transcription:

Kore wa, katamen dake ni ji o kaite, onaji ookisa ni kirisoraeta mono o,
nan mai mo kawa no himo de tojita.

Human translation into German:

Man hat es nur einseitig beschrieben und mehrere auf gleiche Größe
zurechtgeschnittene Blätter mit Lederriemen zusammengebunden.

Human translation into English:

It was written on only one side, and several sheets, trimmed to the same size,
were bound together with leather laces.

Machine translation by WorldLingo
(www.worldlingo.com/products_services/worldlingo_translator.html):

Dieses, den Brief auf gerade die eine Seite, die schreibend, die sie in die
gleiche Größe trimmt, geschlossen vielen mit der Zeichenkette des Leders.

This, writing the letter on just the one side, those which it trims in the same size,
closed many with the string of the leather.

Machine translation by Excite
(www.excite.co.jp/world/url/):

This is leather many sheets about what wrote the character only to one side
and was cut to an even length in the same size. With a string It closed.

Machine translation by @nifty
(www.nifty.com/globalgate/):

This wrote the character only to one side and also closed many things cut to
an even length in the same size with the string of leather.

Machine translation by TransLand
(www.brother.co.jp/jp/honyaku/demo/index.html):

A letter was written only to the settlement side, and this じ how many sheets
of things which cut it into the same size and which was completed with the
leather string, too.

Machine translation by iTranslator
(itranslator.mendez.com/BGSX/BGSXeng_us-EntryPage.htm):

For this, as writing a character, I cut and leveled it for a similar size for one
side only, and any sheets are ひもでとじた of a leather.

Machine translation by 訳せ!!ゴマ
(ai2you.com/goma/):

This bound the one, that writes only to one side and evenly cut a/the
character to the same size with the strings of many sheets of leather.

Fig. 1. Example output of machine translation systems

This is true for *transfer-based* machine translation systems, which try to find mappings between specific language pairs, and even more so for *interlingua-based* machine translation systems aiming to find a language-independent representation that mediates among arbitrary languages. The latter are also often referred to as *knowledge-based* machine translation systems [15, 23, 24] because in most cases a semantic representation of the sentence meaning is used as interlingua. The most ambitious initiative in this direction is probably UNL[6]; one recent system limited to the translation of Japanese, Spanish, and Arabic texts into English is GAZELLE [6].

It is very disappointing to have to notice that the translation quality has not much improved in the last 10 years [28]. One 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. 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 (for a good survey and discussion see [14]). *Statistical* machine translation [3, 4] basically translates word-for-word and rearranges the words afterwards in the right order. Such systems have only been of some success for very similar language pairs. For applying statistical machine translation to Japanese several hybrid approaches have been proposed that also make use of syntactic knowledge [29, 30].

The most prominent approach for the translation of Japanese has been *example-based* machine translation [21, 26]. The basic idea is to collect translation examples for phrases and to use a best match algorithm to find the closest example for a given source phrase. The translation of a complete sentence is then built by combining the retrieved target phrases. The different approaches vary in the representation of the translation examples. Whereas some approaches store structured representations for all concrete examples [2], others explicitly use variables to produce generalized templates [5, 13]. However, the main drawback remains that 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 [25].

To summarize, we are faced with the dilemma that by relying on the available approaches one can either spend several years of effort in creating hand-coded transfer rules or a knowledge-based interlingua – ending up with a large knowledge base that is difficult to maintain – or put one's trust in statistical machine translation based on huge bilingual corpora resulting in mediocre translations caused by the use of inaccurate approximations. Example-based machine translation somehow offers a compromise: one can choose how much effort one wants to invest in adding or correcting translation examples in order to improve the translation quality.

[6] `www.undl.org`.

3 System Architecture

In our approach we use translation examples provided by the user to learn the transfer rules incrementally by using structural matching between the corresponding syntax trees. There were several considerations that guided us towards this design choice:

– as our aim was to develop a domain-independent machine translation system, an interlingua-based approach was out of the question,
– we did not have the resources to manually build a large transfer rule base, also a handcrafted rule base is in conflict with our need for flexible adaptation,
– we had no huge bilingual corpus available for Japanese–German, also the insufficient data quality of today's large corpora would interfere with our demand for high quality translations,
– even if we had an adequate corpus, the poor results achieved by statistical techniques and the manual effort to compile translation templates of sufficient quality for the use in example-based machine translation prohibit the use of existing approaches,
– in our opinion there exists no "perfect" translation but only a preferred one for a certain user, therefore we aim at full customization of our machine translation system,
– the interactive improvement of translation results has also an important pedagogical benefit for the language students because it turns a boring translation task into an entertaining hands-on experience,
– the structured representation in the syntax trees proved to be an efficient input to the learning algorithm, and we can display the trees to language students as additional valuable information.

The operation of our machine translation system can be divided into a learning mode and a translation mode. In the *learning mode* (see Fig. 2) we derive new transfer rules by using a Japanese–German sentence pair as input. Both sentences are first analyzed by the *tokenization* modules, which produce the correct segmentations into word tokens associated with their part-of-speech (POS) tags. Both token lists are then transformed into syntax trees by the *parsing* modules. The syntax trees represent the input to the *learning* module, which uses a structural matching algorithm to discover new transfer rules.

In the *translation mode* (see Fig. 3) we translate a Japanese sentence into the corresponding German sentence by invoking the *transfer* module. It applies the transfer rules stored in the rule base to transform the Japanese syntax tree into the corresponding German syntax tree. Finally, the task of the *generation* module is to produce the surface form of the German sentence as a character string. Of course, the user can correct the translation result and activate the learning mode to incrementally improve the quality of the transfer rule base.

In Sect. 4 we give a more detailed technical description of the individual modules. We illustrate their mode of operation by using the sentence in Fig. 1 as a running example throughout the rest of this paper.

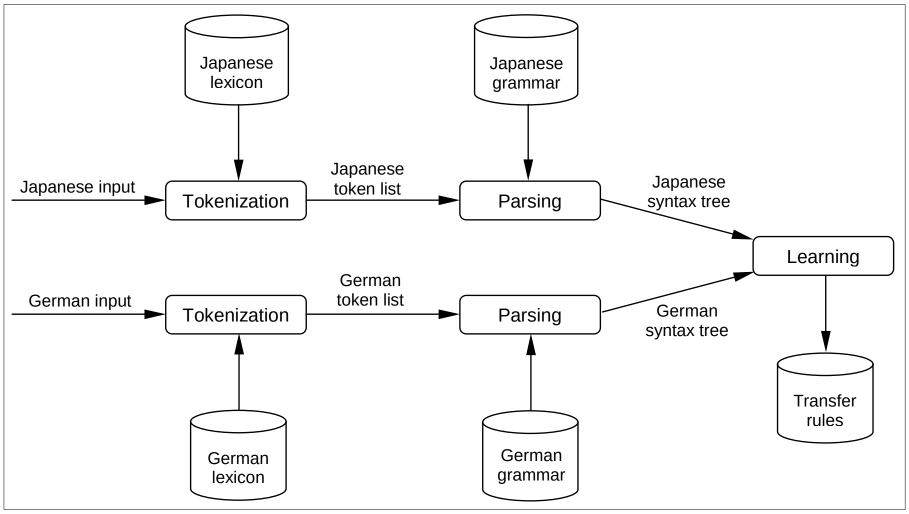


Fig. 2. Learning mode

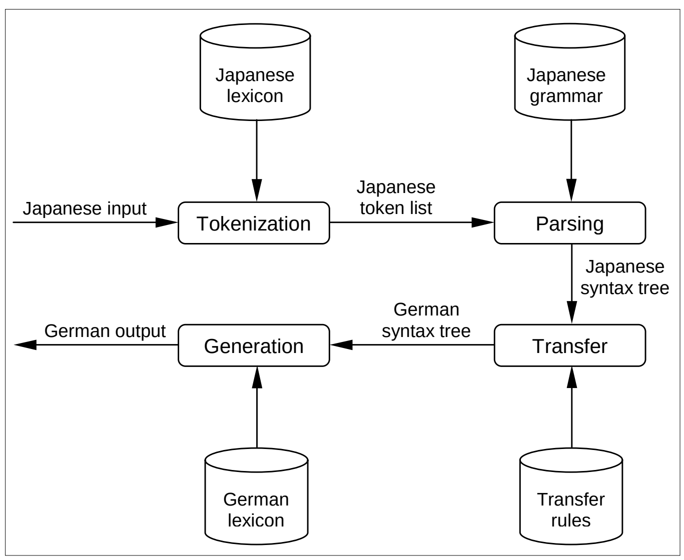


Fig. 3. Translation mode

4 System Description

4.1 Tokenization

The task of the tokenization module is to analyze the surface string of a sentence, to divide the string into words, to lemmatize the words (i.e. to reduce inflectional and variant forms of a word to their base form), and to annotate the base forms with POS tags. Figure 4 shows the token list for our example sentence. The demonstrative pronoun "kore" is an anaphoric reference to "the parchment", which was introduced before in the Japanese text. The *ta*-form of a verb indicates English past tense (expressed as perfect tense in German), whereas the *te*-form is the connective form. The expression "nan mai mo" (literally "what thin objects also") means "several sheets" in this context.

Japanese sentence:

これは、片面だけに字を書いて、同じ大きさに切りそろえた
ものを、何枚も革のひもでとじた。

Segmentation:

これ|は|、|片面|だけ|に|字|を|書いて|、|同じ|大きさ|に|切りそろえた|
もの|を|、|何|枚|も|革|の|ひも|で|とじた|。

Roman transcription:

Kore wa, katamen dake ni ji o kaite, onaji ookisa ni kirisoraeta
mono o, nan mai mo kawa no himo de tojita.

これ/dpr	demonstrative pronoun – kore – it
は/par	particle – wa – (topic indicator)
、/cma	comma
片面/nou	noun – katamen – one side
だけ/suf	suffix – dake – only
に/par	particle – ni – on
字/nou	noun – ji – character
を/par	particle – o – (direct object indicator)
書く/vte	verb te-form – kaku – to write
、/cma	comma
同じ/ano	adjectival noun – onaji – same
大きさ/nou	noun – ookisa – size
に/par	particle – ni – to
切りそろえる/vta	verb ta-form – kirisoraeru – to trim
もの/nou	noun – mono – thing
を/par	particle – o – (direct object indicator)
、/cma	comma
何/ipr	interrogative pronoun – nan – what
枚/cou	counter – mai – thin object
も/par	particle – mo – also
革/nou	noun – kawa – leather
の/par	particle – no – (attribution indicator)
ひも/nou	noun – mono – lace
で/par	particle – de – with
とじる/vta	verb ta-form – tojiru – to bind together
。/per	period

Fig. 4. Example of Japanese token list

Since Japanese writing does not use word delimiters (such as space characters), we have to represent a Japanese sentence as one single atom during segmentation. We have to find and remove the correct word token that is the left subatom of the sentence:

```
segment(Sentence, [BaseForm/POS|TokenList]) :-
    find_token(Sentence, BaseForm, WordLength, POS),
    remove_token(Sentence, WordLength, TokenList).
```

To remove the word token from the sentence we use the information about the word length to calculate the subatom that has to be extracted. Then we continue recursively with the retrieval of the next word token. The recursion ends when the word length equals the length of the remaining partial sentence:

```
remove_token(Sentence, WordLength, []) :-
    atom_length(Sentence, WordLength).
remove_token(Sentence, WordLength, TokenList) :-
    atom_length(Sentence, SentenceLength),
    StartPos is 1 + WordLength,
    RestLength is SentenceLength - WordLength,
    sub_atom(String, StartPos, RestLength, RestSentence),
    segment(RestSentence, TokenList).
```

For the identification of the correct word token we retrieve all words from the Japanese lexicon that are left subatoms of the sentence. From the list of word candidates we choose the correct word by applying some disambiguation rules. The default choice is the longest matching sequence:

```
find_token(Sentence, BaseForm, WordLength, POS) :-
    findall(W:B:P, find_word(Sentence, W, B, P), Candidates),
    select_word(Candidates, BaseForm, WordLength, POS).
```

The retrieval of a word from the Japanese lexicon is performed by matching it with the beginning of the sentence:

```
find_word(Sentence, Word, Word, POS) :-
    jap_lex_entry(Word, POS),
    atom_length(Word, WordLength),
    sub_atom(Sentence, 1, WordLength, Word).
```

Since Japanese has quite a complex system of conjugations for verbs and adjectives, we also have to search for all concatenations of word stems and endings for these two word classes. The base form of conjugated words is computed by concatenating the stem and the correct base form ending depending on the conjugation class:

```
find_word(Sentence, Word, BaseForm, POS) :-
    jap_lex_verbadj(Stem, ConjClass),
    atom_length(Stem, StemLength),
    sub_atom(Sentence, 1, StemLength, Stem),
    jap_ending(ConjClass, Ending, POS),
    atom_length(Ending, EndLength),
    StartPos is StemLength + 1,
    sub_atom(Sentence, StartPos, EndLength, Ending),
    atom_concat(Stem, Ending, Word),
    jap_baseform_ending(ConjClass, BaseFormEnding),
    atom_concat(Stem, BaseFormEnding, BaseForm).
```

The tokenization of Japanese sentences requires a lot of processing power, but is solved by Amzi! Prolog even for large lexicons without any problems.

Compared to this, tokenization of German sentences is quite a trivial task. It can be solved by simply using the predicate `string_tokens` to transform the sentence into a list of tokens, which can then be lemmatized separately. Figure 5 shows the German token list for our translation example. Some ambiguities regarding syntactic features are resolved later during parsing. For example, for the noun “Lederriemen” plural and singular forms are identical so that the decision about the correct number is left to the parsing module. Within the PETRA environment, the language students can consult the token lists to offer them valuable information at the word level.

German sentence:

Man hat es nur einseitig beschrieben und mehrere auf gleiche Größe zurechtgeschnittene Blätter mit Lederriemen zusammengebunden.

man/npr	indefinite pronoun – one
haben/apr	auxiliary verb present tense – to have
es/pep	personal pronoun – it
nur/adv	adverb – only
einseitig/apo	adjective positive comparison – on one side
beschreiben/vpp	verb past participle – to write
und/con	conjunction – and
mehrere/npr	indefinite pronoun – several
auf/prp	preposition – to
gleich/apo	adjective positive comparison – same
Größe/nsg	noun singular – size
zurechtschneiden/vap	verb attributive past participle – to trim
Blatt/npl	noun plural – sheet
mit/prp	preposition – with
Lederriemen/nsp	noun singular or plural – leather lace
zusammenbinden/vpp	verb past participle – to bind together
. /per	period

Fig. 5. Example of German token list

4.2 Parsing

The parsing modules compute the syntactic structure of sentences from their token lists. One interesting property of Japanese grammar is that it uses postpositions instead of prepositions and that the predicate is at the end of the sentence. Therefore, it is easier to parse a Japanese sentence from right to left. Figure 6 shows the syntax tree for our example sentence. As can be seen, the POS tag for conjugated word forms is indicated as feature `hwf` (head word form).

hew	ver	とじる					head word – verb – tojiru – to bind together
hwf	vta						head word form – verb ta-form
pob	hew	nou	ひも				postpositional object – head word – noun – himo – lace
	php	par	で				phrase particle – particle – de – with
	anp	hew	nou	革			attributive noun phrase – head word – noun – kawa – leather
dob	hew	nou	もの				direct object – head word – noun – mono – thing
	amo	hew	cou	枚			amount – head word – counter – mai – thin object
		php	par	も			phrase particle – particle – mo – also
		qua	ipr	何			quantity – interrogative pronoun – nan – what
	avp	hew	ver	切りそろえる			attributive verb phrase – head word – verb – kirisoraeru – to trim
		hwf	vta				head word form – verb ta-form
		pob	hew	nou	大きさ		postpositional object – head word – noun – ookisa – size
			php	par	に		phrase particle – particle – ni – to
			aap	hew	ano	同じ	attributive adjective phrase – head word – adjectival noun – onaji – same
pcl	hew	ver	書く				preceding clause – head word – verb – kaku – to write
	hwf	vte					head word form – verb te-form
	dob	hew	nou	字			direct object – head word – noun – ji – character
	adp	hew	nou	片面			adverbial phrase – head word – noun – katamen – one side
		php	par	に			phrase particle – particle – ni – on
		asf	suf	だけ			attributive suffix – suffix – dake – only
	sub	dpr	これ				subject – demonstrative pronoun – kore – it

Fig. 6. Example of Japanese syntax tree

We use the Definite Clause Grammar (DCG) preprocessor of Amzi! Prolog to write the grammar rules. Instead of using a fixed structure to represent the syntax tree, we opted for a more flexible and robust representation by using *sets* modeled as Prolog lists. A sentence is a set of constituents, and each constituent is a compound term of arity 1 with the constituent name as principal functor and the argument being either

- a *simple constituent* (`feature_value` or `word/word_class`) or
- a *complex constituent* (set of subconstituents).

This flexible representation has the advantage that it is compact, because empty optional constituents are not stored explicitly, and is not affected by the ordering of the different subconstituents. The latter is important for a robust and effective realization of the transfer module so that the transfer rules can change the syntax tree without having to consider any sequencing information.

During parsing we collect arguments for all possible subconstituents and then eliminate empty subconstituents by using the predicate `compress` to remove all list entries with argument `nil`.

In the following we show some (strongly simplified) grammar rules for a noun phrase with an optional attributive suffix and an optional attributive noun phrase (we use the Roman transcription of the particle “no” just in this example):

```
noun_phrase(NP) --> attr_suffix(Asf), [N/nou], attr_np(Anp),
    {compress([hew(N), asf(Asf), anp(Anp)], Np)}.
attr_suffix(Asf) --> [Asf/suf].
attr_suffix(nil) --> [].
attr_np(Anp) --> [no/par], noun_phrase(Anp).
attr_np(nil) --> [].
```

To facilitate the matching between Japanese and German syntax trees (see Sect. 4.3) we tried to align the German grammar as best as possible with the Japanese one. Therefore, we also parse German sentences from right to left. For that purpose we have to perform a preprocessing step on the token list in which we shift all prepositions to the end of prepositional phrases so that they are parsed first. Figure 7 shows the German syntax tree for our translation example. As mentioned in Sect. 4.1 we resolve ambiguous feature values during parsing, e.g. now we can assign the correct number plural to “Lederriemen”.

```
hew  ver  zusammenbinden                head word – verb – to bind together
ten  per                                tense – present perfect
pob  hew  nou  Lederriemen              prepositional object – head word – noun – leather lace
     php  prp  mit                      phrase particle – preposition – with
     det  nod                           determiner type – no determiner
     num  plu                           number – plural
dob  hew  nou  Blatt                    direct object – head word – noun – sheet
     det  nod                           determiner type – no determiner
     num  plu                           number – plural
     aip  npr  mehrere                  attributive indefinite pronoun – indefinite pronoun – several
     avp  hew  ver  zurechtschneiden    attributive verb phrase – head word  – verb – to trim
          ten  per                      tense – present perfect
          pob  hew  nou  Größe          prepositional object – head word – noun – size
               php  prp  auf            phrase particle – preposition – to
               det  nod                 determiner type – no determiner
               num  sng                 number – singular
               aap  hew  adj  gleich    attributive adjective phrase – head word – adjective – same
                    com  pos            comparison – positive
sub  npr  man                           subject – indefinite pronoun – one
pcl  hew  ver  beschreiben              preceding clause – head word – verb – to write
     ten  per                           tense – present perfect
     php  con  und                      phrase particle – conjunction – and
     pap  hew  adj  einseitig           predicative adjective phrase – head word – adjective – on one side
          com  pos                      comparison – positive
          aav  adv  nur                 attributive adverb – adverb – only
     dob  pep  es                       direct object – personal pronoun – it
```

Fig. 7. Example of German syntax tree

For displaying the parsing trees to the user we have implemented one generic display module for both Japanese and German syntax trees, which is also able to deal with mixed representations caused by missing coverage of the transfer rule base. This way we can show the limitations of the translation system to the language student who can easily fix them with an update of the rule base.

4.3 Learning

The learning module traverses the Japanese and German syntax trees and derives new transfer rules, which are added to the rule base. For that purpose we have implemented generic predicates for the simultaneous navigation in two complex constituents. We start to search for new rules at the sentence level before we look for corresponding constituents to continue the search for finer-grained transfer rules recursively. We always perform a complete traversal, i.e. new rules are learnt even if they are not required for the translation of the Japanese sentence in order to extract as much information as possible from the example.

We distinguish between four different types of transfer rules for simple constituents (SC) and complex constituents (CC). The transfer rules are stored as facts in Prolog:

- `tr_sc(C1,C2,A1,A2)`: changes the SC `C1(A1)` to `C2(A2)`,
- `tr_asc(A1,A2)`: changes the argument of an SC from `A1` to `A2`,
- `tr_cc(C1,C2,Hew,Req1,Req2)`: changes the CC `C1(A1)`, `A1`=`Req1`∪`Opt`, to `C2(A2)`, `A2`=`Req2`∪`Opt`, if `hew(Hew)`∈`A1`,
- `tr_acc(Hew,Req1,Req2)`: changes the argument of a CC from `A1`=`Req1`∪`Add`, to `A2`=`Req2`∪`Add` if `hew(Hew)`∈`A1`.

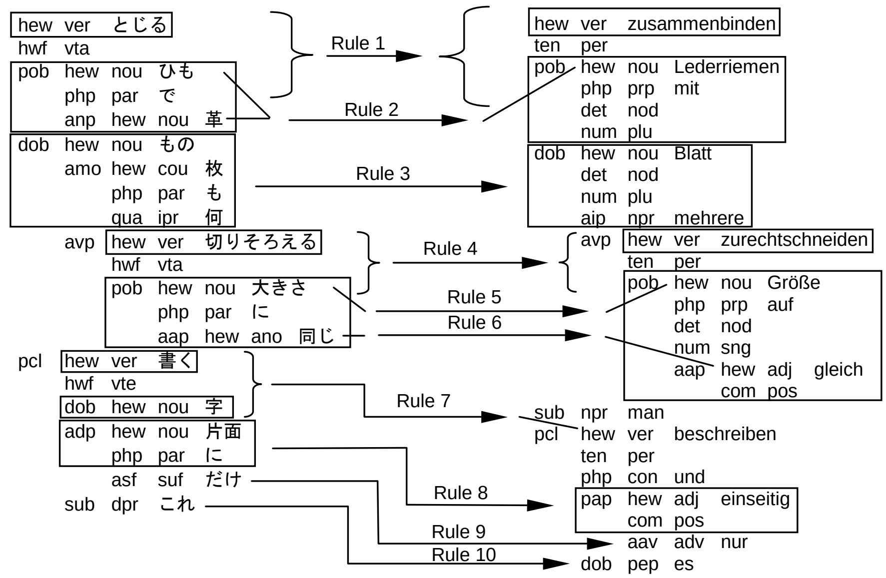
`Hew` serves as index for the fast retrieval of matching rules and the reduction of the number of rules that have to be analyzed. For transfer rules of type `tr_acc` any additional subconstituents are allowed in `Add`, whereas `Opt` in rules of type `tr_cc` can only contain certain optional subconstituents. Transfer rules for complex constituents can use shared variables for unification in `Req1` and `Req2`. In addition to those four generic rule types, we also use several more specific types, e.g. for the correct translation of conjunctions and syntactic features.

Figure 8 shows the transfer rules that we can learn from our translation example. We omit the default rules for deriving the perfect tense from the head word form `vta`, for deriving the conjunction "und" from the head word form `vte`, and for inserting the indefinite pronoun "man" for the missing subject. The suffix "dake" is an optional subconstituent, which can extend the set of required subconstituents in Rule 8. Rule 1 and Rule 4 are two examples of transfer rules that use shared variables for unification.

The principal steps for performing the structural matching between two complex constituents are:

- we either derive transfer rules of type `tr_asc` for the head word or transfer rules of type `tr_acc` for head/modifier combinations,
- we derive transfer rules of type `tr_sc` or `tr_cc` to translate a Japanese subconstituent into a different German subconstituent,
- we search for corresponding subconstituents and apply the matching recursively to those subconstituents,
- we derive transfer rules for conjunctions and syntactic features.

Each rule is validated against the existing rules to resolve all conflicts arising from adding the new rule to the rule base. The resolution is achieved by making the conflicting rules more specific.



1. tr_acc(とじる/ver, [hew(とじる/ver), pob([php(で/par)|X])],
   [hew(zusammenbinden/ver), pob([php(mit/prp), det(nod), num(plu)|X])]).
2. tr_acc(ひも/nou, [hew(ひも/nou), anp([hew(革/nou)])], [hew('Lederriemen'/nou)]).
3. tr_acc(もの/nou, [hew(もの/nou), amo([hew(枚/cou), qua(何/ipr), php(も/par)])],
   [hew('Blatt'/nou), num(plu), det(nod), aip(mehrere/npr)]).
4. tr_acc(切りそろえる/ver, [hew(切りそろえる/ver), pob([php(に/par)|X])],
   [hew(zurechtschneiden/ver), pob([php(auf/prp), det(nod), num(sng)|X])]).
5. tr_asc(大きさ/nou, 'Größe'/nou).
6. tr_asc(同じ/ano, gleich/adj).
7. tr_acc(書く/ver, [hew(書く/ver), dob([hew(字/nou)])], [hew(beschreiben/ver)]).
8. tr_cc(adp, pap, 片面/nou, [php(に/par), hew(片面/nou)], [hew(einseitig/adj), com(pos)]).
9. tr_sc(asf, aav, だけ/suf, nur/adv).
10. tr_sc(sub, dob, これ/dpr, es/pep).

Fig. 8. Example of learning transfer rules

4.4 Transfer

The transfer module traverses the Japanese syntax tree and searches for transfer rules that can be applied. The flexible definition of the rules enables a robust processing of the syntax tree. One rule only changes certain parts of a constituent into the German equivalent, other parts are left unchanged to be transformed later on. Thus, our transfer algorithm deals efficiently with a mixture of Japanese–German, which gradually turns into a correct German syntax tree.

To translate the argument `A1` of a constituent `C(A1)` into `A2` we have defined the predicate `tf_arg(C, A1, A2)`. For simple constituents we just apply transfer rules of type `tr_asc`, for complex constituents we first apply transfer rules of type `tr_acc` (predicate `tf_acc(A1, A2)`) as well as rules for conjunctions and syntactic features before we recursively call the predicate `tf_sub(Csub, A1, A2)` for the translation of each subconstituent `Csub(Asub)`:

```
tf_sub(Csub, A1, A2) :-
    find_subconstituent(Csub, A1, Asub),
    tf_sub_arg(Csub, Asub, A1, A2).
tf_sub(_, A, A).
```

The predicate `find_subconstituent` retrieves the argument `Asub` for the subconstituent `Csub(Asub)`. It fails if no subconstituent with constituent name `Csub` is included in `A1`. The predicate `tf_sub_arg` first tries to apply rules of type `tr_sc` and `tr_cc` to replace the Japanese subconstituent with a different German subconstituent before it recursively calls the predicate `tf_arg` to translate the argument `Asub`:

```
tf_sub_arg(Csub, Asub, A1, A2) :-
    tr_sc(Csub, Csub2, Asub, Asub2),
    replace_subconstituent(Csub, Csub2, A1, Asub2, A2).
tf_sub_arg(Csub, Asub, A1, A2) :-
    tf_cc(Csub, Csub2, Asub, Asub2),
    replace_subconstituent(Csub, Csub2, A1, Asub2, A2).
tf_sub_arg(Csub, Asub, A1, A2) :-
    tf_arg(Csub, Asub, Asub2),
    Asub \== Asub2,
    replace_arg_subconstituent(Csub, A1, Asub2, A2).
tf_sub_arg(_, _, A, A).
```

To apply transfer rules of type `tr_acc` we retrieve the head word from `A1` as index for the access to matching transfer rules and then call `split` to unify the subconstituents in `Req1` with the corresponding subconstituents in `A1`:

```
tf_acc(A1, A2) :-
    find_subconstituent(hew, A1, Hew),
    tr_acc(Hew, Req1, Req2),
    split(A1, Req1, Add),
    append(Req2, Add, A2).
tf_acc(A, A).
```

The predicate `split` takes every subconstituent in `Req1`, retrieves the corresponding subconstituent in `A1` and unifies the two structures. This way we can guarantee that the unification is not affected by the order of the subconstituents in `Req1` and `A1`. As a byproduct of this sorting procedure, `split` returns the set of additional subconstituents `Add`=`A1`\`Req1`, i.e. all subconstituents in `A1` that were not retrieved. Figure 9 shows an example of the application of a transfer rule of type `tr_acc` with a shared variable for unification (Rule 4 in Fig. 8). The predicate `tf_cc` for the application of transfer rules of type `tr_cc` is defined in a similar way.

```
tr_acc(切りそろえる/ver,
          [hew(切りそろえる/ver), pob([php(に/par)|X])],
          [hew(zurechtschneiden/ver), pob([php(auf/prp), det(nod), num(sng)|X])]).
Req1 =  [hew(切りそろえる/ver), pob([php(に/par)|X])]
Req2 =  [hew(zurechtschneiden/ver), pob([php(auf/prp), det(nod), num(sng)|X])]
A1 =    [pob([php(に/par), hew(大きさ/nou), aap([hew(同じ/ano)])]),
          hwf(vta), hew(切りそろえる/ver)]
Req1' = [hew(切りそろえる/ver), pob([php(に/par),
               hew(大きさ/nou), aap([hew(同じ/ano)])])]
Add =   [hwf(vta)]
Req2' = [hew(zurechtschneiden/ver), pob([php(auf/prp), det(nod), num(sng),
               hew(大きさ/nou), aap([hew(同じ/ano)])])]
A2 =    [hew(zurechtschneiden/ver), pob([php(auf/prp), det(nod), num(sng),
               hew(大きさ/nou), aap([hew(同じ/ano)])]), hwf(vta)]
```

Fig. 9. Example of the application of a transfer rule

4.5 Generation

To generate the surface form of a German sentence, we traverse the syntax tree in a top-down fashion. For each complex constituent we transform its argument into a list of surface strings, which is computed recursively from its subconstituents as nested list and flattened afterwards. The syntactic features to compute the correct determiners and the declensions and conjugations of German words are partly included in the German syntax tree, e.g. number or tense, and partly retrieved from the German lexicon, e.g. gender. In the following we show the (strongly simplified) predicate to generate the list of surface strings for a noun phrase (the predicate `find_optional_subconstituent` returns `nil` if it cannot find the subconstituent):

```
generate_np(nil, _, []).
generate_np(NP, Case, StringList) :-
    find_subconstituent(hew, NP, Hew/nou),
    find_subconstituent(det, NP, Det),
    find_subconstituent(num, NP, Num),
    ger_lex_noun(Hew, Gender, DeclClass),
    generate_det(Det, Num, Case, Gender, Determiner),
    find_optional_subconstituent(aap, NP, Aap),
    generate_aap(Aap, Det, Num, Case, Gender, Adjective),
    generate_hew(Hew, Num, Case, DeclClass, Noun),
    flatten([Determiner, Adjective, Noun], StringList).
generate_np(_, _, []).
```

After the complete traversal of the syntax tree, the resulting flat list of surface strings is transformed into a single character string by inserting spaces where appropriate.

Finally, we provide some means for the resolution of simple intersentential anaphora by storing candidates for antecedents in previous sentences, e.g. to compute the correct surface form of a personal pronoun.

5 Conclusion

In this paper we have presented a customizable machine translation system, which incrementally learns transfer rules from translation examples provided by a user. We have completed the implementation of the translation system and the integration into the language learning environment PETRA. We are now in the process of filling the transfer rule base with the help of several language students from the University of Vienna. So far, the feedback from the students has been very positive. For some, PETRA has already become an invaluable companion throughout their language studies.

Whereas at the moment language students are our main target audience, we hope to reach a level of linguistic competence in the near future that will make it also possible for non-specialist users to benefit from our translation environment. In addition to constantly extending the coverage of our machine translation system, future work will also concentrate on a thorough evaluation of the system according to the FEMTI[7] framework.

References

1. Bond, F., Ogura, K., Kawaoka, T.: Noun phrase reference in Japanese-to-English machine translation. Proceedings of the 7th International Conference on Theoretical and Methodological Issues in Machine Translation, Leuven, Belgium (1995)
2. Brockett, C. et al.: English-Japanese example-based machine translation using abstract linguistic representations. Proceedings of the COLING-2002 Workshop on Machine Translation in Asia, Taipei, Taiwan (2002)
3. Brown, P.: A statistical approach to machine translation. Computational Linguistics **16(2)** (1990) 79–85
4. Brown, P. et al.: The mathematics of statistical machine translation: Parameter estimation. Computational Linguistics **19(2)** (1993) 263–311
5. Furuse, O., Iida, H.: Cooperation between transfer and analysis in example-based framework. Proceedings of the 14th International Conference on Computational Linguistics, Nantes, France (1992) 645–651
6. Germann, U.: Making semantic interpretation parser-independent. Proceedings of the 3rd AMTA Conference, Longhorne, USA (1998) 286–299
7. Hutchins, J.: Machine Translation: Past, Present, Future. Ellis Horwood (1986)
8. Hutchins, J.: Machine translation over 50 years. Histoire epistémologie langage **23(1)** (2001) 7–31
9. Hutchins, J.: Has machine translation improved? Some historical comparisons. Proceedings of the 9th MT Summit, New Orleans, USA (2003) 181–188
10. Hutchins, J.: Machine translation and computer-based translation tools: What's available and how it's used. In: Bravo, J. M., ed.: A New Spectrum of Translation Studies. University of Valladolid (2003)
11. Hutchins, J., Somers, H.: An Introduction to Machine Translation. Academic Press (1992)
12. Isozaki, H., Hirao, T.: Japanese zero pronoun resolution based on ranking rules and machine learning. Proceedings of the Conference on Empirical Methods in Natural Language Processing, Sapporo, Japan (2003) 184–191

[7] `www.isi.edu/natural-language/mteval/`.

13. Kaji, H., Kida, Y., Morimoto, Y.: Learning translation examples from bilingual text. Proceedings of the 14th International Conference on Computational Linguistics, Nantes, France (1992) 672–678
14. Knight, K.: Automatic knowledge acquisition for machine translation. AI Magazine **18(4)** (1997) 81–96
15. Leavitt, J. R. R., Lonsdale, D. W., Franz, A. M.: A reasoned interlingua for knowledge-based machine translation. Proceedings of the 10th Canadian Conference on Artificial Intelligence, Banff, Canada (1994)
16. Murata, M., Isahara, H., Nagao, M.: Pronoun resolution in Japanese sentences using surface expressions and examples. Proceedings of the ACL-99 Workshop on Coreference and its Applications, Maryland, USA (1999)
17. Murata, M., Nagao, M.: Determination of refential property and number of nouns in Japanese sentences for machine translation into English. Proceedings of the 5th International Conference on Theoretical and Methodological Issues in Machine Translation, Kyoto, Japan (1993) 218–225
18. Murata, M., Nagao, M.: Resolution of verb ellipsis in Japanese sentence using surface expressions and examples. Proceedings of the Natural Language Processing Pacific Rim Symposium, Phuket, Thailand (1997) 75–80
19. Murata, M. et al.: An example-based approach to Japanese-to-English translation of tense, aspect, and modality. Proceedings of the 8th International Conference on Theoretical and Methodological Issues in Machine Translation, Chester, United Kingdom (1999) 66–76
20. Murata, M. et al.: A machine-learning approach to estimating the referential properties of Japanese noun phrases. Proceedings of the CICLing-2001 Conference on Intelligent Text Processing and Computational Linguistics, Mexico City, Mexico (2001)
21. Nagao, M.: A framework of a mechanical translation between Japanese and English by analogy principle. In: Elithorn, A., Banerji, R., eds.: Artificial and Human Intelligence. NATO Publications (1984)
22. Newton, J., ed.: Computers in translation: A practical appraisal. Routledge (1992)
23. Nirenberg, S. et al.: Machine Translation: A Knowledge-Based Approach. Morgan Kaufmann Publishers (1992)
24. Onyshkevych, B., Nirenburg, S.: A lexicon for knowledge-based MT. Machine Translation **10(1-2)** (1995) 5–57
25. Richardson, S. et al.: Overcoming the customization bottleneck using example-based MT. Proceedings of the ACL Workshop on Data-driven Machine Translation, Toulouse, France (2001) 9–16
26. Sato, S.: Example-Based Machine Translation. PhD thesis, Kyoto University (1991)
27. Seki, K., Atsushi, F., Ishikawa, T.: A probabilistic method for analyzing Japanese anaphora integrating zero pronoun detection and resolution. Proceedings of the 19th International Conference on Computational Linguistics, Taipei, Taiwan (2002) 911–917
28. Somers, H., ed.: Computers and Translation: A Translator's Guide. John Benjamins (2003)
29. Watanabe, T., Imamura, K., Sumita, E.: Statistical machine translation based on hierarchical phrase alignment. Proceedings of the 9th International Conference on Theoretical and Methodological Issues in Machine Translation, Keihanna, Japan (2002) 188–198
30. Yamada, K.: A Syntax-Based Statistical Translation Model. PhD thesis, University of Southern California (2002)