Machine Translation Using Corpus-based Acquisition of Transfer Rules

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

Abstract

In this paper we present 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. We learn specific transfer rules from sentence pairs, which are then generalized in a consolidation phase to avoid overtraining. 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, e.g. Microsoft Word. The dynamic nature of our system allows for an easy customization of the rule base according to the user's personal preferences.

1. Introduction

Research on machine translation has a long tradition [2]. 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. 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 [3].

It is very disappointing to have to notice that the translation quality has not much improved in the last 10 years [10]. 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 [4]. *Statistical* machine translation [1] basically translates word-for-word and rearranges the words afterwards in the right order. Such systems have only been of some success for similar language pairs. In the last few years, there have been several attempts to extend this word-based translation approach towards phrase-based translation [5]. Recently, also hybrid approaches that make use of syntactic knowledge have been proposed for translating Japanese [12, 14].

The most prominent approach for Japanese has been *example-based* machine translation [9]. 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 examples: some store structured representations for all concrete examples, others explicitly use variables to produce generalized templates. 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 [8].

In our approach we use a transfer-based machine translation architecture, however, we learn all the transfer rules automatically from translation examples by using structural matching between the parse trees. Our current research work originates from the PETRA project (Personal Embedded Translation and Reading Assistant, [13]) 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. Fortunately, the situation looks much brighter for Japanese-English as there are several large high quality parallel corpora available. In particular, we use the JENAAD corpus [11], which is freely available for research or educational purposes and contains 150,000 sentence pairs from news articles.

For the implementation of our machine translation system we have chosen Amzi! Prolog because it provides an

1-4244-1476-8/07/$25.00 ©2007 IEEE.

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 an overview of the system architecture in Sect. 2. Section 3 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 consolidation phase to avoid overtraining are presented in Sect. 4. Finally, Sect. 5 focusses on the application of the transfer rules during translation and the generation of the final natural language output.

2. 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. For Japanese tagging we use ChaSen [7], for English MontyTagger [6]. ChaSen produces numerical tags whereas MontyTagger uses the Penn Treebank tagset.

The token lists are then transformed into parse trees by the *parsing* modules by using the Definite Clause Grammar (DCG) preprocessor of Amzi! Prolog. 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 *simple constituents* representing words (`atom/atom`) or features (`atom`), and *complex constituents* representing phrases modeled as lists of subconstituents. Japanese sentences are parsed from right to left, we have chosen the same order for the subconstituents in English parse trees. The parse trees form 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. Irregular inflections of English words are generated by applying *morphology rules* which are learnt while parsing English sentences in the acquisition phase.

3. 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. 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.

3.1. Word transfer rules

For simple context-insensitive translations at the word level, the argument $A1$ of a simple constituent is changed to $A2$ by applying the predicate $\textit{wtr}(A1, A2)$, i.e. if the argument of a simple constituent is equal to the *argument condition* $A1$, it is replaced by $A2$.

Example 1. The default transfer rule to translate the Japanese noun SEKAI into the English counterpart `world` is stated as the fact: $\textit{wtr}(\text{SEKAI}/2, \texttt{world}/nn)$.

3.2. 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: $\textit{ctr}(C1, C2, \textit{Hea}, A1, A2)$.

This changes a complex constituent $C1(A1)$ to $C2(A2)$ if the category is equal to *category condition* $C1$, the head is equal to *head condition Hea*, and the argument is equal to argument condition $A1$.

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

$$\textit{ctr}(\textit{mno}, \textit{maj}, \text{KOKUSAI}/2, [\textit{hea}(\text{KOKUSAI}/2)], [\textit{hea}(\texttt{international}/\textit{jj})]).$$

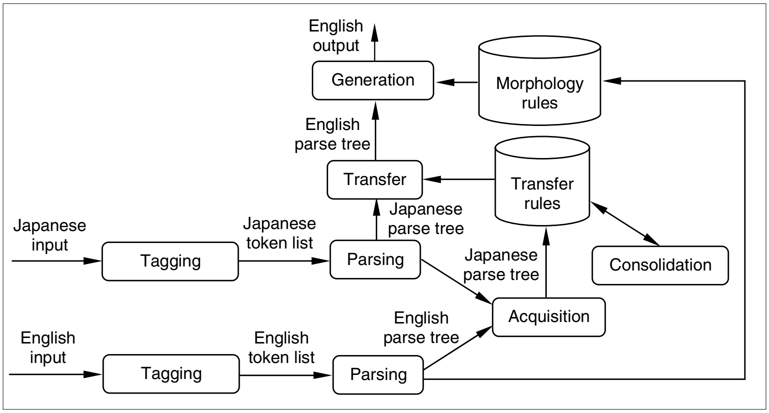


Figure 1. System architecture.

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.

Example 3. The modifying verb phrase (*mvp*) X NI MUKETA is translated as modifying adpositional phrase (*map*) `toward` X. It contains a verbal with head MUKERU and auxiliary TA and an adpositional object with adposition NI (*hef* encodes the conjugation type and form):

$$\begin{aligned}&\mathit{ctr}(\mathit{mvp}, \mathit{map}, \text{MUKERU}/47, [\mathit{vbl}([\mathit{hea}(\text{MUKERU}/47),\\ &\quad \mathit{hef}(6/4), \mathit{aux}([\mathit{hea}(\text{TA}), \mathit{hef}(54/1)])]),\\ &\quad \mathit{aob}([\mathit{apo}(\text{NI}/61)|X])], [\mathit{apo}(\texttt{toward}/\mathit{in})|X]).\\ &C1(A1) = \mathit{mvp}([\mathit{vbl}([\mathit{hea}(\text{MUKERU}/47), \mathit{hef}(6/4),\\ &\quad \mathit{aux}([\mathit{hea}(\text{TA}), \mathit{hef}(54/1)])]),\\ &\quad \mathit{aob}([\mathit{apo}(\text{NI}/61), \mathit{hea}(\text{MINSHU}/2), \mathit{suf}(\text{KA}/31)])])\\ &C2(A2) = \mathit{map}([\mathit{apo}(\texttt{toward}/\mathit{in}), \mathit{hea}(\text{MINSHU}/2),\\ &\quad \mathit{suf}(\text{KA}/31)])\end{aligned}$$

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.

3.3. 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: $\mathit{ptr}(C, \mathit{Hea}, Req1, Req2)$.

Rules of this type change the argument of a complex constituent with category C from $A1 = Req1 \cup \mathit{Add}$ to $A2 = Req2 \cup \mathit{Add}$ if $\mathit{hea}(\mathit{Hea}) \in A1$. To enable the flexible application of phrase transfer rules, input $A1$ and argument condition $Req1$ 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 $Req1$ is included in the argument $A1$ of the input constituent $C(A1)$ to replace $Req1$ by $Req2$. Besides $Req1$ 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.

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. $\mathit{mvp} \sqsubseteq \mathit{vp}$).

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.

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

$$\begin{aligned}&\mathit{ptr}(\mathit{vbl}, \text{SURU}/47, [\mathit{hea}(\text{SURU}/47), \mathit{sjc}(\text{NINSHIKI}/17)],\\ &\quad [\mathit{hea}(\texttt{recognize}/\mathit{vb})]).\\ &A1 = [\mathit{hea}(\text{SURU}/47), \mathit{hef}(3/1), \mathit{sjc}(\text{NINSHIKI}/17)]\\ &A2 = [\mathit{hea}(\texttt{recognize}/\mathit{vb}), \mathit{hef}(3/1)]\end{aligned}$$

As explained before, the order of the elements in $A1$ is of no importance, the rule is applied to $A1$ and the additional element $\mathit{hef}(3/1)$ is added to the elements in $Req2$.

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:

$$\begin{aligned}&\textit{ptr}(\textit{np}, \text{KOTO}/21, [\textit{hea}(\text{KOTO}/21), \textit{mvp}([\textit{vbl}([\textit{hea}(\text{DA}/74),\\ &\quad \textit{hef}(55/4), \textit{aux}([\textit{hea}(\text{ARU}/74), \textit{hef}(18/1)]),\\ &\quad \textit{cap}([\textit{hea}(\text{JUUYOU}/18)])]), \textit{sub}([\textit{apo}(\text{GA}/61)|X])])],\\ &\quad [\textit{hea}(\texttt{importance}/nn), \textit{det}(\textit{def}),\\ &\quad \textit{mnp}([\textit{apo}(\texttt{of}/\textit{in})|X])]).\\ &A1 = [\textit{hea}(\text{KOTO}/21), \textit{mvp}([\textit{vbl}([\textit{hea}(\text{DA}/74), \textit{hef}(55/4),\\ &\quad \textit{aux}([\textit{hea}(\text{ARU}/74), \textit{hef}(18/1)]),\\ &\quad \textit{cap}([\textit{hea}(\text{JUUYOU}/18)])]),\\ &\quad \textit{sub}([\textit{hea}(\text{AKUSESU}/17), \textit{apo}(\text{GA}/61),\\ &\quad \textit{mno}(Y), \textit{mvp}(Z)])])]\\ &A2 = [\textit{hea}(\texttt{importance}/nn), \textit{det}(\textit{def}), \textit{mnp}([\\ &\quad \textit{apo}(\texttt{of}/\textit{in}), \textit{hea}(\text{AKUSESU}/17), \textit{mno}(Y), \textit{mvp}(Z)])]\end{aligned}$$

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 $A1$ but extends recursively to any level of detail specified in $Req1$, e.g. to the subconstituents of *sub* in this example.

4. 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 left-most constituents in the sentences are examined first, which facilitates the correct mapping of subconstituents with identical constituent categories, e.g. several modifying nouns. It then calls `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.

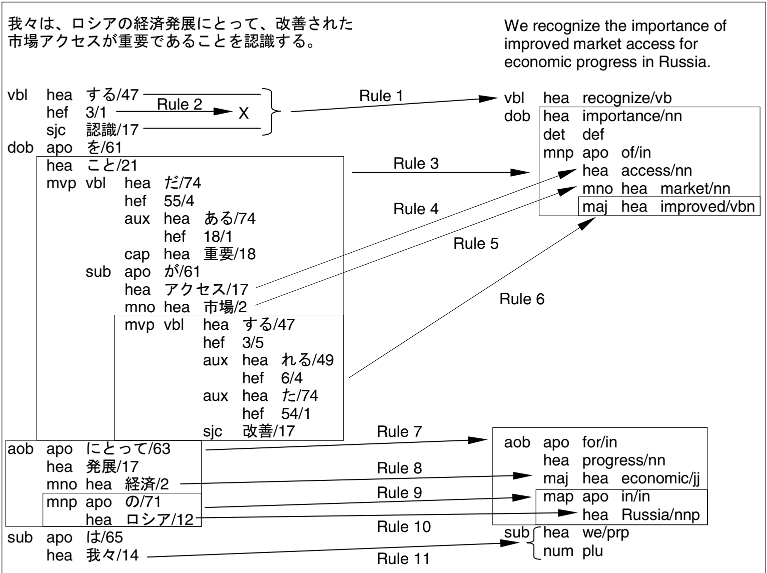
Each predicate of type `map_dob` for the mapping of the individual subconstituents both covers special mappings as well as the default treatment:

```
... 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, or to split a phrase transfer rule in two simpler rules.

Figure 2 shows an example of rule acquisition, 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).



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(acess/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 2. Example of rule acquisition.

5. 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. 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 (`apply_ptrules`) to the sentence before we try to translate each constituent in the sentence individually (`transfer_const`):

```
transfer(J,E):-apply_ptrules(vp,J,I),
  transfer_const(I,E).
apply_ptrules(C,J,E):-apply_ptr(C,J,I),
  apply_ptrules(C,I,E).
apply_ptrules(_,Sent,Sent).
```

The predicate `apply_ptrules` applies phrase transfer rules recursively until no further rule can be applied. The application of a single phrase transfer rule (`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` or the argument condition does not depend on the head.

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,[CoAC|ReAC],
    [CoAC2|ReAC2],Add):-
  once(retrieve_co(CoAC,I,CoAC2,I2)),
  split_rec(I2,ReAC,ReAC2,Add).
retrieve_co(Co,[Co|ReI],Co,ReI).%(1)
retrieve_co(CoAC,[CoI|ReI],CoAC,ReI):-%(2)
  CoAC =.. [Cat,ArgAC],CoI =.. [Cat,ArgI],
  equal_args(ArgI,ArgAC).
retrieve_co(CoAC,[CoI|ReI],CoAC2,
    [CoI|ReI2]):-
  retrieve_co(CoAC,ReI,CoAC2,ReI2).
equal_args(ArgI,ArgAC):-
  once(unify_args(ArgI,ArgAC,ArgAC)).
unify_args(ArgI,ArgAC,ArgAC2):-
  var(ArgAC),ArgAC2 = ArgI.%(3)
unify_args([],[],[]).%(4)
unify_args(ArgI,[CoArgAC|ReArgAC],
    [CoArgAC2|ReArgAC2]):-
  once(retrieve_co(CoArgAC,ArgI,
    CoArgAC2,ArgI2)),
  unify_args(ArgI2,ReArgAC,ReArgAC2).
```

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.

After applying phrase transfer rules at the sentence level, `transfer_const` examines each individual subconstituent. It first tries to apply constituent transfer rules before calling the 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 `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 is computed recursively from its subconstituents as nested list and flattened afterwards. As mentioned before, we use morphology rules derived while parsing English training sentences to produce the correct surface forms for words with irregular inflections.

6. Conclusion

In this paper we have presented 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 including a prototype interface to Microsoft Word and have demonstrated the feasibility of the approach based on a small subset of the JENAAD corpus.

Future work will focus on extending the coverage of the system so that we can process the full JENAAD corpus and perform a thorough 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.

References

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

[2] J. Hutchins. Has machine translation improved? Some historical comparisons. In *Proc. 9th MT Summit*, pages 181–188, New Orleans, USA, 2003.

[3] J. Hutchins. Machine translation and computer-based translation tools: What's available and how it's used. In J. M. Bravo, editor, *A New Spectrum of Translation Studies*, pages 13–48. Univ. Valladolid, Valladolid, 2004.

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

[5] P. Koehn, F. J. Och, and D. Marcu. Statistical phrase-based translation. In *Proc. Human Language Technology Conf. of the North American Chapter of the ACL*, pages 48–54, Edmonton, Canada, 2003.

[6] H. Liu. MontyLingua: An end-to-end natural language processor with common sense. Technical report, MIT Media Lab, 2004.

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

[8] S. Richardson et al. Overcoming the customization bottleneck using example-based MT. In *Proc. ACL Workshop on Data-Driven Machine Translation*, pages 9–16, Toulouse, France, 2001.

[9] S. Sato. *Example-Based Machine Translation*. PhD thesis, Kyoto University, 1991.

[10] H. Somers, editor. *Computers and Translation: A Translator's Guide*. John Benjamins, Amsterdam, 2003.

[11] M. Utiyama and H. Isahara. Reliable measures for aligning Japanese-English news articles and sentences. In *Proc. 41st Annual Meeting of the ACL*, pages 72–79, Sapporo, Japan, 2003.

[12] T. Watanabe, K. Imamura, and E. Sumita. Statistical machine translation based on hierarchical phrase alignment. In *Proc. 9th Intl. Conf. on Theoretical and Methodological Issues in Machine Translation*, pages 188–198, Keihanna, Japan, 2002.

[13] W. Winiwarter. Incremental learning of transfer rules for customized machine translation. In U. Seipel et al., editors, *Applications of Declarative Programming and Knowledge Management*, volume 3392 of *LNAI*, pages 47–64. Springer-Verlag, Berlin, 2005.

[14] K. Yamada. *A Syntax-Based Statistical Translation Model*. PhD thesis, Kyoto University, 1999.