December 2015

A Machine-Aided Approach to Generating Grammar Rules from Japanese Source Text for Use in Hybrid and Rule-based Machine Translation Systems

Sean Michael Jones
University of Wisconsin-Milwaukee

A MACHINE-AIDED APPROACH TO GENERATING GRAMMAR RULES FROM JAPANESE SOURCE TEXT FOR USE IN HYBRID AND RULE-BASED MACHINE TRANSLATION SYSTEMS

by

Sean Jones

A Thesis Submitted in

Partial Fulfillment of the

Requirements for the Degree of

Master of Science

in Computer Science

at

The University of Wisconsin-Milwaukee

December 2015

ABSTRACT

A MACHINE-AIDED APPROACH TO GENERATING GRAMMAR RULES FROM JAPANESE SOURCE TEXT FOR USE IN HYBRID STATISTICAL RULE-BASED MACHINE TRANSLATION SYSTEMS

by

Sean Jones

The University of Wisconsin-Milwaukee, 2015
Under the Supervision of Professor Susan McRoy

Many automatic machine translation systems available today use a hybrid of pure statistical translation and rule-based grammatical translations. This is largely due to the shortcomings of each individual approach, requiring a large amount of time for linguistics experts to hand-code grammar rules for a rule-based system and requiring large amounts of source text to generate accurate statistical models. By automating a portion of the rule generation process, the creation of grammar rules could be made to be faster, more efficient and less costly. By doing statistical analysis on a bilingual corpus, common grammar rules can be inferred and exported to a hybrid system. The resulting rules then provide a base grammar for the system. This helps to reduce the time needed for experts to hand-code grammar rules and make a hybrid system more effective.

DEDICATION

This thesis became a reality with the kind support and help of many individuals. I would like to extend my sincere thanks to all of them:

To my family, especially my wife Sumi, for putting up with my continued absence while pursuing degree after degree, missing important dates, neglecting phone calls and being away from home far too often while working on this research.

To my academic advisor, Dr. Susan McRoy, for the very tough job of guiding me through this work, offering her knowledge and expertise, dealing with my often insane schedule and teaching me the tools I needed to complete this work.

To the members of my thesis defense committee, Dr. Ichiro Suzuki and Dr. Nicholas Fleisher, for their time, their knowledge and kindness.

And to all my colleagues at UW-Milwaukee and MATC who helped me in ways too numerous to list here.

Chapter 1.Introduction 1
1.1.Background 1
1.2.Current Practices 2
1.3.MT System Advantages and Disadvantages 4
1.4.Hybrid Systems 6
1.5.Interactive Machine Translation 8
1.6.Research Aim 9
1.7.Structure 10
Chapter 2.Linguistic Concepts 12
2.1.Part-of-Speech Tagging 12
2.2.Phrase Structure Grammar and Parse Trees 14
2.3.Ambiguity 16
2.3.1.Sentence Boundary Ambiguity 16
2.3.2.Word Sense Ambiguity 17
Chapter 3.System Introduction 19
3.1.The Language Pair: English-Japanese 19
3.2.Some Difficulties in Translating Between English and Japanese 20
3.2.1.Sentence Structure 20
3.2.2.Weak Inflection vs Overt Inflection 22
3.3.The Base System 23
3.3.1.Apertium 23
3.3.2.Bilingual Corpus and Dictionary 24
Chapter 4.Apertium 26
4.1.Apertium Overview 26
4.2.Monolingual Dictionaries 26
4.3.Bilingual Dictionary 29
4.4.Transfer Rules 30
4.5.Apertium Dictionary Builder 35
Chapter 5.Generating Rules 37
5.1.Preprocessing 37
5.2.Parsing rules into Apertium 39

5.2.1.Interpreting the Rules....39
5.2.2.Formatting the Rules for Apertium....41
5.3.Evaluation....42
6.Results 44
6.1.Overview....44
6.2.Test Results....45
6.Conclusion 47
6.1.Summary....47
6.2.Limitations....48
6.3.Future Work....49
Bibliography 51
Appendix A.Apertium Files 56
apertium-en.en.dix....56
apertium-jp.jp.dix....62
apertium-jp-en.jp-en.dix....66
apertium-jp-en.jp-en.t1x....68
Appendix B.Test Results 81
II.1.Rule List....81
II.2.Raw Test Output....82
Appendix C.Apertium Dictionary Builder 86

LIST OF FIGURES

Fig 1 POST Example using a simplified tag set.....13

Fig 2 Parse Tree from the Penn tree bank.....16
for the sentence “Pierre Vinken, 61
years old, will join the board as a non
executive director Nov. 29.

Fig 3 Word Sense Ambiguity Examples.....19

Fig 4 An example with a source sentence F.....23
reordered into target order F’, and
its corresponding target sentence E. D is
one of the permutations that can produce
this ordering

Fig 5 An example of an English sentence that.....25
could have many viable translations in
Japanese, but cannot infer which
translation based solely on the source
sentence.

Fig 6 Excerpts from WWWJDIC (left) and the.....27
Tanaka Corpus (right)

Fig 7 Constituent parts of an Apertium .dix.....29
file. <sdefs> define the POS tags used
in the system, <pardefs> define the

paradigms, or grammar rules, of the dictionary entries and the main section is filled with <e> dictionary entries.

Fig 8 The Apertium pipeline, taken from the Apertium for Dummies section of Apertium's wiki 30

Fig 9 Examples of bids entries 33

Fig 10 Category and Attribute definitions in a transfer rule file 35

Fig 11 A simple transfer rule from an interchunk file 36

Fig 12 A System flow diagram of the grammar analysis, rule generation and Apertium system building. The green modules are automatically processed, while the blue modules require manual intervention. The output of this system is a shallow-transfer language pair that may be used by Apertium. 41

Fig 13 The pattern 1$2$4$5 (1 = noun, 2 = particle, 3 = verb, 4 = verb suffix) mapped to the ID numbers of all the sentences containing that phrase type in the corpus. 42

Fig 14 Japanese particles altering the meaning of 43

a sentence

Chapter 1.Introduction

1.1.Background

In recent years, research into *Machine Translation*(MT) has seen quite a surge in popularity. This is the result of a number of factors, the most influential being the increased interaction between disparate language-speaking people online. MT has been an area of research since the 1940's, fueled initially in large part by the U.S. Defense Department's interest in Russian-English translation during the Cold War. Overly optimistic expectations and computational limitations of the time caused opinion to turn against MT research, and, in 1966, the Automatic Language Processing Advisory Committee(ALPAC) recommended to stop funding MT research, greatly reducing the amount of research for decades(Hutchins, 2007).

As computing power, insights into related fields such as Linguistics and Software Engineering advanced and with the explosion of data on the web, MT systems began to improve dramatically. Today, there are many commercial and freely available online translation systems that support nearly one hundred languages(Google, 2015).

However, translating one human language to another is not a simple matter. Any one natural language, when looked at from a purely grammatical and syntactical point of view, is a complicated system, with many interconnecting rules and exceptions to those rules. When semantics are added, the task of understanding even a single language is great. In any MT system, this needs to be done, simultaneously, for *two* separate languages. As a result, these systems and the approaches they use have their flaws.

1.2.Current Practices

The state-of-the-art in MT approaches has changed significantly over the last 20 years. The 1990's marked a trend towards turning away from morphological and syntactical analyses that used linguistic rules to determine the appropriate translation and towards a more purely statistical approach. Initially *word-based*(Brown, 1993), and later *phrase-based* models(Koehn, Och, & Marcu, 2003; Zens, Och, & Ney, 2002) relied primarily on statistical analysis of a sententially aligned bilingual corpus, called a *parallel corpus* or *bitext,* to generate a language model based on relative frequencies of words and phrases. Later systems used *Maximum Likelihood Estimation(*MLE) to determine the probability of a translation based on multiple features to train

their language models(Berger, 1996; Och & Ney, 2002). These corpus-based systems came in two distinct types(Bijimol & Abraham, 2014):

1. *Statistical Machine Translation*(SMT), where the above statistical models are used with a bilingual corpus to generate translations based on the probabilities of word usage, order and syntax.
2. *Example-based Machine Translation*(EBMT), which takes example sentences from the corpus that are similarly constructed and does a word-by-word translation, using the same orderings from the target and source language examples.

Rule-based systems, however, were and still are being researched and used. By the late 1990's, SMT was considered the mainstream, but *Rule-Based Machine Translation*(RBMT) was still being pursued(Somers, 1999). RBMT systems tend to fall into three separate categories:

1. *Dictionary systems*, where sentences are translated from the source language to the target language without an intermediate representation

2. *Transfer systems*, where the source language is converted into a more general language representation using grammar rules and bilingual dictionaries
3. *Interlingual systems*, where the source language is converted into an intermediate representation that is not language-dependent

More recent RBMT systems, such as Apertium(Forcada, 2011) and ANN(Akeel & Mishra, 2014) have been developed and are still being actively researched, Interlingual-type systems such as UNL(Uchida, Zhu, & Della Senta, 2005) are being expanded to cover a growing number of languages and other language tools, such as WordNet(Princeton, 2014), a knowledge base that maps groups of words to a semantic meaning, is being implemented in more and more languages.

1.3.MT System Advantages and Disadvantages

These systems both have strengths and shortcomings. According to Bijimol and Abraham, the benefits of RBMT systems are "easy customization and predictability", meaning there is simple, direct access to the dictionary or intermediate representation of the language that can be quickly modified, and that the output is highly predictable, as the system can be logically

followed from start to finish. Additionally, rules may often be reusable among languages and among different language pairs, particularly in interlingual systems, where the mappings will remain the same regardless of the second language.

On the contrary, corpus-based systems often have unexpected results. Being purely lexical in nature, these systems will return a translation based solely on probability calculated from the existing corpus. Any novel words or sentence structures would be processed solely on its similarity to another sentence or phrase structure, often yielding unexpected results. Additionally, the models that are generated from these systems are quite opaque, consisting of a variety of probabilities, scores and other measurements for word mappings. Any desired change to the system's behavior would require a regeneration of the language model with additional data or altered corpus, which can become quite time-consuming. Further, the altering of the model may end up changing previously accurate output.

SMT systems, on the other hand, have the benefits of not requiring linguistic expertise to build, meaning less human and temporal cost is required to set the system up. Also, CPU resource requirements are, in general, less for SMT systems,

making it faster to return translations. And finally, SMT is greatly dependent on the amount of source material available, and there is not much linguistic diversity available in a digital format. In 2013, Google had indexed around 30 trillion unique web pages(Koetsier, 2013), yet currently Google Translate, the company's proprietary SMT system, only supports 90 languages(Google, 2015), or about 1.2% of the more that 7,100 languages spoken in the world(Ethnologue, 2015).

RBMT, on the other hand, currently requires a good deal of manual entry for linguistic rule creation and dictionary building, both of which require trained expert assistance to generate. Also, as the RBMT system grows, it becomes more difficult to maintain and manage a large amount of rule interactions.

1.4.Hybrid Systems

Of course, there are many systems that have attempted to take advantage of techniques from both SMT and RBMT, so-called *Hybrid Machine Translation Systems* (HMT). These systems use varying amounts of SMT and RBMT in an attempt to mitigate some of the difficulties mentioned above and, in fact, many modern systems use some combinations of approaches. An early Hierarchical

Phrase-Based model, for example, used a standard Phrased-Based SMT model, but extends the model to a hierarchy of phrases, rather than simple *n*-gram phrases to assist in the recognition and reordering of phrases(Chiang, 2005). Other hybrid systems began with a RBMT system, then used SMT as a way to post-process the translation to match a trained model(Sánchez Martínez, Forcada Zubizarreta, & Way, 2009).

As these models began to evolve, hybrid systems started combining the two system types, using both at different points during the translation process in an attempt to gain the best features of each system type while mitigating their shortcomings(España Bonet, Màrquez Villodre, Labaka, Díaz de Ilarraza Sánchez, & Sarasola Gabiola, 2011), including combining HMT systems into a different hybrid system(Baker et al., 2014); (J. Li, Marton, Resnik, & Daumé III, 2014). These later HMT systems often attempt to take a SMT system and inform it with some linguistic, syntactic or semantic knowledge from a RBMT or HMT to improve the fluency of the translation or to shape it for a specific knowledge domain, such as medical translation(Costa-jussá, Farrús, & Pons, 2012);(Lu et al., 2014). Another common use is that of phrase and word reordering between languages with

different syntactic structures(Farzi, 2013);(Zhang, Liu, Li, Zhou, & Zong, 2015).

The state-of-the-art sees HMT systems being used for the previously mentioned knowledge domain improvement and phrase and word reordering, as well as standalone systems that attempt to improve translations through filtering SMT results through a RBMT in some fashion, or by improving the tokenization and/or chunking of the input data of an SMT through a RBMT(Park, Kwon, Kim, & Kim);(H. Li, Zhu, & Jin, 2015)

1.5.Interactive Machine Translation

Interactive Machine Translation (IMT) is a related, but distinctly different approach to computer-aided translation. Using concepts from SMT and RBMT, IMT shifts the decision-making to the user. These implementation are a more of a translation assistance machine than an automated translation system, and rely on the expertise of the user.

Early IMT systems were largely focused on source-text disambiguation. They would ask for user input to add context and clarifications, then generate a translation(Bisbey & Kay, 1972; Tomita, 1985). This approach often relies on a certain level of

knowledge from the user, though systems were designed for non-expert users, as well(Maruyama, Watanabe, & Ogino, 1990). Later systems began using a "Target-Text" model, focusing on output corrections by the user (Foster, Isabelle, & Plamondon, 1997). Current systems are also implementing machine learning into the system and updating the language model after every user input(Ortiz-Martınez & Casacuberta, 2014).

IMT systems are not purely computer-based, and as such have some inherent weaknesses. The output quality is, in the end, dependent on the linguistic skill level of the user, and there are often cases, especially in dissimilar language pairs, where a deep understanding of the source language is required for correct output. The initial, often SMT, output given to the user can be confusing, ungrammatical and incorrect. IMT also, due to the interaction required, is significantly slower than other MT methods. Currently, these types of systems are often used by professional translators to increase their translation speed.

1.6.Research Aim

These approaches are typical of what is currently being published. While they have improved on the shortcomings of SMT and RBMT, there are still many issues that have yet to be

adequately addressed, such as domain-specific translations, slang, low-resourced languages, contrasting sentential structure and others. Furthermore, all of these approaches, with the exception of pure SMT, still rely on some form of linguistic rules to, at a minimum, improve the fluency of the translation, which still relies heavily on linguistics experts hand-coding and revising complex rules. The author chose to tackle this last issue, rule creation, specifically looking at a way to reduce the time and human cost of generating grammar rules for any MT systems that make use of them. There has been some similar work done previously by Victor Sánchez-Cartagena using Apertium that took an existing grammar rule set for a language pair and inferred additional rules from them to improve translation(Sánchez-Cartagena, 2014). What this work will do is take a language that does not have a rule set and, based on a bilingual corpus, a dictionary file and a rule set for one other language, generate some basic rules for the new language.

1.7.Structure

The goal of this research is to explore a technique to assist linguistic rule creation for RBMT and HMT systems by automatically generating the structure of some of these needed rules. Using a monolingual dictionary, a bilingual corpus and a

set of grammar rules for one language, an automated method will be used to infer common grammar rules for the second language from the corpus and built into the Apertium RBMT format. Common words will be extracted from the corpus and dictionary files for the new language pair will be built using a custom tool. Chapter 2 will be an overview of the linguistic concepts and approaches that will be used and referenced in this system; Part-of-Speech Tagging and Parse Trees being the foundations for representing language. One of the open problems of MT, that of Ambiguity, will also be discussed. Chapter 3, will detail the system built to generate grammar rules and a base dictionary, explaining the chosen language pair, the existing tools and the custom tools used to generate the rules and dictionary files. Chapter 4 will discuss the RBMT System, Apertium, in greater detail. Chapter 5 show the results of the system with examples. Chapter 6 will be an overview of the specific issues tackled in this paper, conclusions regarding this type of system and an outlook on possible future work.

Chapter 2.Linguistic Concepts

2.1.Part-of-Speech Tagging

One of the key tools used in any natural language system is *Part-of-Speech Tagging*(POST). POST is the process of marking up, or *tagging*, words in a text with a corresponding part-of-speech label as it appears in a given sentence. These tags are used to disambiguate words that, depending on their position in a sentence, have a different definition, context, or meaning based on their relationship with the adjacent and otherwise related words within a given phrase, sentence or even paragraph. Tags will denote a given word, phrase, suffix, prefix or other portion of a word as particular grammatical concepts, such as noun, verb, adjective, and so on. Figure 1 shows a simplified tag set and how it is used to mark up a sentence.

Part-of-Speech	POS Tag
Noun	N
Verb	V
Adjective	ADJ
Determiner	DET

Input: The dog is brown.

Tagged: The[DET] dog[N] is[V] brown[ADJ].

Fig. 1 POST example using a simplified tag set

Initial POST systems used hand-crafted grammar rules to determine what part-of-speech(POS) to apply to each term based on the adjacent terms(Francis, 1964), while modern systems use a

variety of statistical methods, primarily the Hidden Markov Model(HMM), that may take into account larger groups of terms and calculate the probability that a given series of POS tags will occur, given earlier sentences in the corpus(Church, 1988). These groups of terms are called *n-grams*, where *n* is the number of terms being analyzed (2 terms, being a 2-gram, or bigram, 3 terms would be a 3-gram or trigram, etc). While simply looking at patterns of previously tagged sentences will give no knowledge of the grammar rules of the source language, for many languages, this approach works remarkably well. In fact, it has been shown that assigning the most common tag to each known word and the tag "proper noun" to all unknown words in a corpus will approach 90% accuracy, as most words are unambiguous(Charniak, 1997).

The size and complexity of a POST tag set varies greatly. The Penn TreeBank(Marcus, Marcinkiewicz, & Santorini, 1993), the most popular English-language tag set, has 36 general POS tags that divide types of nouns, verbs, and adjectives into subtypes, such as singular nouns(NN), plural nouns(NNS), and so on. Other tag sets, designed for translation, are smaller, only focusing on main POS tags, as different languages often do not share the same grammar usages(Petrov, Das, & McDonald, 2011).

2.2.Phrase Structure Grammar and Parse Trees

Of course, natural language is made up of more than just the parts of speech. Various POS work together to create larger semantic units, *phrases*. In order to organize all the various parts of speech into a cohesive and standard structure from which inferences can be made, linguists have developed grammar formalisms. *Phrase Structure Grammar*(PSG) is based on the notion of sentences as a collection of phrases of varying POS types: noun phrase(NP), verb phrase(VP), adjectival phrase(ADJP), prepositional phrase(PP), and so on(Koehn, 2009).

These grammars use a tree data structure called a *syntax tree* or *parse tree.* It is defined as an ordered, rooted tree that represents the syntactic structure of a string according to some *context-free grammar*(CFG)(Carnie, 2013). The *nonterminals* of the CFG are made up of POS tags and phrase types, the *terminals* being the words. Figure 2,
taken from the Penn Tree Bank, is a parse tree for a complex English sentence, showing the relationships between the phrases and words. The root of a parse tree is the sentence node *S*, and its children are general phrases, usually the main subject noun phrase and verb phrase. From there, based on the CFG, the top-

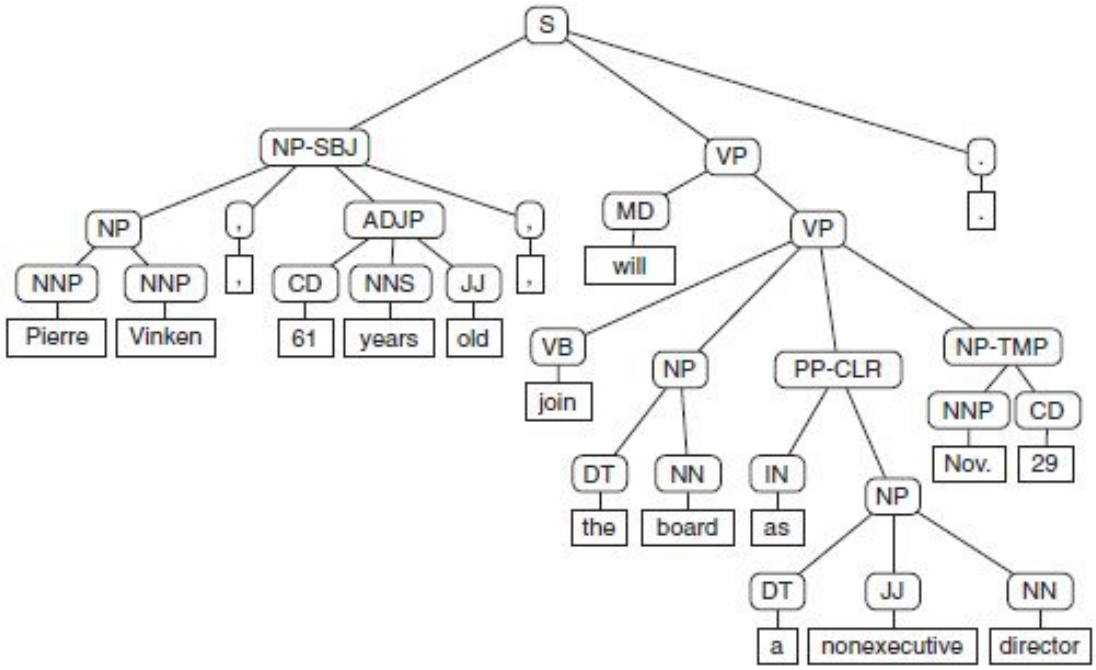


Fig. 2 Parse Tree from the Penn tree bank for the sentence "Pierre Vinken, 61 years old, will join the board as a non executive director Nov. 29.

level phrases break further into phrases and POS until reaching terminal nodes, where the words themselves reside. This allows us different levels of abstraction with which to analyze a sentence, grouping words and phrases into larger phrases. Phrase trees are integral in allowing MT systems to split sentences into larger chunks than individual words. This chunking is especially useful when sentences do not map directly word-for-word between languages, which is most often the case, and for reordering of sentences between languages.

2.3.Ambiguity

The previous two sections explained how sentences are stored and organized using POS tagging and parse trees and it should seem obvious that automatic translation is mostly a matter of lining up corresponding words between the two languages, based on a dictionary translation and a POS tag, in a language pair and using some grammar rules to reorder the target language to a natural language state. When explained in such a way, it seems like a trivial task that could be solved, albeit a bit slowly, deterministically. However all natural language has some inherent ambiguity that makes the task more complex and, in fact, still one of the main open questions of machine translation. There are two main forms of ambiguity that add complexity to a natural language: *sentence boundary ambiguity* and *word sense ambiguity*.

2.3.1.Sentence Boundary Ambiguity

When given some input text, it is necessary to determine how to break the text up for translation. In most languages there is some form of sentence-end symbol, such as the period (.) in English. However, that symbol may often be used for other purposes in the language. In English, for example, a period may denote the end of a sentence, the end of an abbreviation (Vol.

3), a person's initials (J. M. Barrie), part of a web URL (google.com), computer code (a.getSize()) or a decimal point ($3.49). Complicating matters further are colloquial uses, such as performer aliases (Will.I.Am) or even incorrect punctuation. To account for the various uses of the end-of-sentence punctuation, often a maximum entropy model is trained on a hand-marked set of documents(Reynar & Ratnaparkhi, 1997) and used to split input into distinct sentences.

2.3.2.Word Sense Ambiguity

Some languages have more inherent ambiguity than others, especially those, such as English, that have little *inflectional morphology.* Inflection is the process of adding *inflectional morphemes* to a word to add some grammatical information, such as number, gender, person, tense, etc. A morpheme is the smallest unit of meaning in a language. Inflectional morphology is simply the process of using inflection when generating a sentence(Van Valin, 2001).

Read
I *read* the news today.
I *read* the news everyday.

Out
He put the cat *out* for the night.
The runner was *out* at first.
I am *out* of sugar.
He threatened to *out* the whistleblower.
She called *out* of the blue.
Roger, over and *out*.

Fig. 3 Word Sense Ambiguity Examples

English, as stated above, is considered a weakly inflected language, having only four verb tenses and very little noun inflection, only pluralization and pronouns. In general, words are not altered to include additional information and, as a result, are ambiguous in their meaning. Additionally, for written language, different pronunciation is not readily apparent in text alone. Figure 3 shows a few examples of such ambiguity in English. Note the tense difference in the two versions of “read”. POS tagging would identify both as verbs, but when translating into an overtly inflectional language that uses morphemes to differentiate tenses, it has to determine which is the correct form of the verb in the target language with identical parse trees.

Chapter 3.System Introduction

3.1.The Language Pair: English-Japanese

The language pair chosen for this work is English and Japanese. There are several reasons for this decision. Firstly, the author has a good degree of fluency in both languages, which is necessarily a requirement to working on translation. Secondly, there is a lot of work yet to be done with this language pair for a number of reasons that we will consider. Thirdly, there is no existing Japanese-English language pair for Apertium, the RBMT system the author chose to employ, giving additional motivation to the project. Lastly, written Japanese has a relatively rigid grammatical structure, with some well-defined exceptions, making it an ideal initial language with which to test automatic rule generation.

It is important to note that exceptions to grammar rules exist in virtually every natural language, so any set of rules that may be generated automatically will likely be incomplete. A simple example of exceptions in English would be generating past tenses of verbs. The typical rule would be adding the suffix '-ed' to verbs or '-d' to verbs ending in '-e', such as with the words "opened" or "closed". We have many exceptions to this rule, and sometimes exceptions for the exceptions. When we

conjugate the verb “to go”, the plain past tense is the irregular form “went”, as in “I/you/he/she/we/they went”. However, when using the past perfect tense, there is an exception to the irregular form, and it becomes “had gone”. In contrast to English, Japanese conjugation of past tense verbs remains constant, adding the suffixes “-mashita” or “-ta” to the stem of the verb. Using the same example in Japanese, “iku”(to go), is conjugated as a past tense verb as “ikimashita, while the past perfect tense is similarly conjugated as “itteimashita”.

3.2.Some Difficulties in Translating Between English and Japanese

There are a number of reasons translating between English and Japanese poses an interesting MT problem. There are many dissimilarities between the two languages that introduce points of failure as they are handled. For each difference in grammar, an additional subsystem in the MT system is required to handle that difference.

3.2.1.Sentence Structure

The sentential structures are different between the languages, English being a *subject-verb-object*(SVO) language and Japanese a

subject-object-verb(SOV). This creates a certain amount of difficulty with word sense and reordering, causing the system to first attempt to break the sentence into discrete parts and alter the ordering of the sentence before translation. This technique, called *preordering*, is defined as the transformation of a source sentence[1] $F = f_1...f_j$ to target sentence $E = e...e_i$).

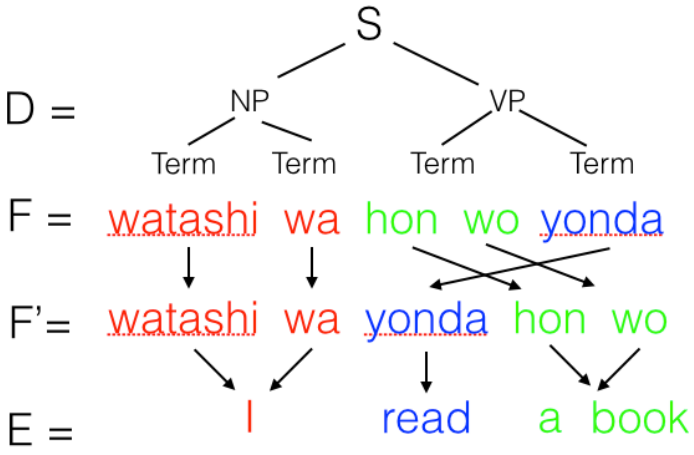


Fig. 4 An example with a source sentence *F* reordered into target order *F'*, and its corresponding target sentence *E*. *D* is one of the permutations that can produce this ordering

The preordering process is generally done in three stages, the first of which is deterministically generating a parse tree from the source sentence from the parallel corpus by permuting the tree and scoring the resulting trees, then deterministically transforming the sentence *F* into *F'*, using the same terms

[1] In MT literature, it is common to refer to sentences in the two languages in a pair as *E* and *F*, as the first published papers were concerned with translating from French to English. Modern usage simply has *F* denoting the *source* language (the language we are translating from) and *E* denoting the *target* language (the language we are translating to). It is also common to refer to individual parts of the sentence not as words, but as *tokens* or *terms*. This is due there often being multiple words from one language that map to a single word in the other language.

contained in *F*, but matching the order of the target language for *E* (Neubig, Watanabe, & Mori, 2012). From there, the sentence is translated from either a dictionary or language model (Figure 4). This transformation in Apertium is handled in a slightly different manner, using three separate dictionary files and some transfer rules, that will be explained further in Section 4, but the same basic idea of transforming the sentential structure to match the target language is used.

3.2.2.Weak Inflection vs Overt Inflection

As discussed in Section 2.3.2, English is a weakly inflected language. Japanese, on the other hand, is strongly, or *overtly inflected*. There are numerous verb tenses that use a variety of suffixes to indicate tense, social status, negation, strength of opinion/conviction and other meanings. While English has 12 basic verb tense conjugations, Japanese has 10 base forms and 154 conjugations (Kamiya, 2012), along with a more expansive use of affixes to verbs and adjectives that English does not use. The complexity of the Japanese grammar system is not what makes English-Japanese translation difficult. It is, instead, the ambiguity introduced by English, that requires additional work to be done to accurately translate to or from Japanese. With a weakly inflected language, a good deal of inference needs to

occur in order to determine the appropriate word sense for the target language (Figure 5). It is very often the case that an accurate translation cannot be done without some form of

Tense

simple common past
simple polite past
common regretful past
polite regretful past
common explicative past
polite explicative past

Fig. 5 An example of an English sentence that could have many viable translations in Japanese, but cannot infer which translation based solely on the source sentence.

translational memory, or storage of a certain amount of previously translated sentences from a longer document to reference, though this is beyond the scope of this thesis. In many systems there is a default, base tense that is used when another, more specific, tense cannot be inferred from the source sentence.

3.3.The Base System

3.3.1.Apertium

For a target system, the RBMT system Apertium was chosen to build a rule set for a number of reasons. Firstly, it is an open-source system, allowing free use of all aspects of the system. There are only a few actively supported open-source

translation systems, and only Apertium is both Rule-Based and not language-specific, allowing us to implement the rules that we will generate. Secondly, the rule set is coded using a modified xml structure, allowing for quick parsing, testing and searchability. Thirdly, there was a need for a base rule set for the English-Japanese language pair in Apertium's resource collection, as none currently exists. The rules and dictionary files for Apertium will be explained in detail in Chapter 4.

3.3.2.Bilingual Corpus and Dictionary

Generating good quality rules requires a well-translated bilingual corpus and a sufficiently detailed bilingual dictionary file. The author used two well-known collections as source texts, the Tanaka Corpus (Tanaka, 2001) and Jim Breen's WWWJDIC online dictionary (Breen, 1995). The Tanaka corpus was compiled by Yasuhito Tanaka of Hyogo University in 2001, and is

```
<entry>
<ent_seq>1380640</ent_seq>
<k_ele>
<keb>製鋼所</keb>
</k_ele>
<r_ele>
<reb>せいこうじょ</reb>
</r_ele>
<r_ele>
<reb>せいこうしょ</reb>
</r_ele>
<sense>
<pos>&n;</pos>
<gloss>steelworks</gloss>
<gloss>steel mill</gloss>
<gloss>steelmaking plant</gloss>
</sense>
</entry>
```

```
A: 何時ですか。    What time do you have?#ID=24596_5059
B: 何時(なんじ) ですか
A: 今日はとても暑い。       It is very hot today.#ID=1689_5092
B: 今日 は 迚も[01]{とても} 暑い
A: 彼は約束を破った。       He broke his word.#ID=304317_5095
B: 彼(かれ)[01] は 約束を破る{約束を破った}
A: 警察を呼んで！  Call the police!#ID=1718_5120
B: 警察 を 呼ぶ{呼んで}
```

Fig 6. Excerpts from WWWJDIC (left) and the Tanaka Corpus (right)

the largest public domain Japanese-English bilingual corpus, containing over 150,000 sentence pairs, and is continually revised and updated. The corpus version we used was 2014-8-16(Ho, 2009). The Jim Breen WWWJDIC dictionary is an open-source dictionary project based on EDICT, an electronic dictionary of over 150,000 entries maintained by the Electronic Dictionary Research and Development Group at Monash University, with almost weekly revisions. The dictionary file was downloaded on 2014-09-18 and is the version used throughout this project.

Chapter 4.Apertium

4.1.Apertium Overview

Apertium (Forcada, 2011) is an open-source, rule-based system for machine translation.

It is comprised of a number of XML-based data files that, at minimum, store for a given language pair, two monolingual dictionary (.dix) files, a bilingual dictionary and some transfer rule (.t#x) files. Larger language pairs may contain additional files to handle complex syntax and other tasks. A full Apertium system is a pipeline of many small modules, each with a specific task that transforms a source language sentence into its target language translation. Figure 7 shows the full Apertium pipeline in detail. The simplified system implemented in this project is designed to test the grammar rules that were generated, so, while functional, it does not contain all of the modules indicated.

4.2.Monolingual Dictionaries

Apertium's monolingual dictionaries are used to build morphological analyzers and generators that obtain all the possible lexical forms for a certain surface form in the source language and generates the surface form in the target language

```
<sdefs>
  <sdef n="n"           c="noun"/>
  <sdef n="sg"          c="singular"/>
  <sdef n="pl"          c="plural"/>
  <sdef n="prn"         c="pronoun"/>
  <sdef n="subj"        c="subject"/>
  <sdef n="obj"         c="object"/>
  <sdef n="p1"          c="first person"/>
 <pardef n="s/ee_vblex">
   <e><p><l>ee</l><r>ee<s n="vblex"/><s n="pres"/></r></p></e>
   <e><p><l>aw</l><r>ee<s n="vblex"/><s n="past"/></r></p></e>
 </pardef>
 <pardef n="prsubj_prn">
   <e><p><l/><r>prpers<s n="prn"/><s n="p1"/><s n="sg"/></r></p></e>
 </pardef>

<section id="main" type="standard">
  <e lm="home"><i>home</i><par n="home_n"/></e>
  <e lm="see"><i>s</i><par n="s/ee_vblex"/></e>
  <e lm="personal subject pronouns"><i/><par n="prsubj_prn"/></e>
```

Fig. 7 Constituent parts of an Apertium .dix file. <sdefs> define the POS tags used in the system, <pardefs> define the paradigms, or grammar rules, of the dictionary entries and the main section is filled with <e> dictionary entries.

from the lexical form of each word and can be read from left to right or right to left, depending on which direction the translation is going. This means that *English->Japanese* or *Japanese->English* translations use the same dictionary files.

The monolingual dictionary files are comprised, minimally, of three sections: *Symbol Definitions* (sdef), *Paradigm Definitions* (pardef) and *Lemmas* (lm), portions of which are shown in Figure 8. The *sdef* section contains definitions and descriptions of the POS tags that are used in the system. These sdefs are used as tags for each word to add information such as tense, amount, gender, and so on to produce accurate translations.

The *pardef* section contains “Paradigm Definitions”, rules that are comprised of sets of entries. Each entry indicates what lexical transformation will occur, based on the associated sdefs. These transformations are essentially rules for altering words due to conjugation, pluralization, etc.

The *lemmas*, base forms of the words contained in the dictionary, are stored in the “main” section of the file. Each lemma entry contains at least three pieces of information: the word itself, the lexical stem of the word and the pardef that particular word will use when being transformed during translation.

When translating, Apertium will take each word (or phrase) and consult the dictionary, looking for the appropriate entry, and transforming it. Using Figure 8 as an example, if the English verb “saw” was sent to the dictionary, Apertium would cycle through the list of lemmas for stems that would match the input (in this case, “s”, “sa” and “saw”). All lemmas that match would then call their pardefs, which would add the appropriate sdef tags. In this case, “saw” would match with the lemma “see”, which would call the “s/ee_vblex” pardef. The stem for “see” is “s”, so the pardef would select the sdef tags that are associated with the suffix ”aw”. In this case, the tags <vblex>,

indicating lexical verb, and <past>, indicating past tense. Once this is all completed, the input word “saw” would be transformed into its tagged base form, “see<vblex><past>”.

When more than one lemma matches an input, all matching lemmas are processed as above. Once this is done, the POS Tagger module is consulted to resolve the ambiguity. The POS Tagger module runs the input sentence through a POS Tagger, then compares its tag with the tags returned from the dictionary and selects the matching lemma. In our example, the POS Tagger would tag “saw” as a verb. It would then check all the returned lemmas for a verb tag and return the lemma that was tagged as a verb. If more than one lemma matches, it means the transformation rules are not correctly written, and Apertium returns nothing.

4.3.Bilingual Dictionary

Once Apertium has sent a word to its monolingual dictionary to be tagged, the word is then sent to the bilingual dictionary, which handles the lexical transfer process. Apertium assigns the target language a certain lexical form that corresponds to each of the source language lexical forms. In addition, the bilingual dictionary, unsurprisingly, maps the individual words between the language pair. The bilingual dictionary itself is identical

to the monolingual dictionaries with the exception of the “main” section. Where the monolingual dictionaries contain lemma entries, the bilingual dictionary contains entries that pair lemmas from each language.

```
<e><p><l>私<s n="prn"/></l><r>I<s n="prn"/></r></p></e>
<e><p><l>家<s n="n"/></l><r>home<s n="n"/></r></p></e>
<e><p><l>見る<s n="vblex"/></l><r>see<s n="vblex"/></r></p></e>
<e><p><l>を<s n="prt"/></l><r><s n="prt"/></r></p></e>
```

Fig. 9 Examples of bidix entries

Looking at Figure 9, we can continue our example from above. Apertium has transformed the English verb “saw” to “see<vblex><past>” and now passes it to the bilingual dictionary, where it cycles through all of the entries looking for a match. It finds the bilingual entry that contains the lemma “see” and matches the <vblex> tag. Now, Apertium transforms “see<vblex><past>” to match the entry’s other side, so we now have:“見る<vblex><past>”.

4.4.Transfer Rules

Through the Apertium system, which is comprised of several modules that pass text stream between them, these dictionary files contain all the necessary rules to generate an initial parse tree for the input sentence, permute that parse tree into all its legal forms based on the target language, and match

lexical forms (phrases) and word matches between the language pairs.

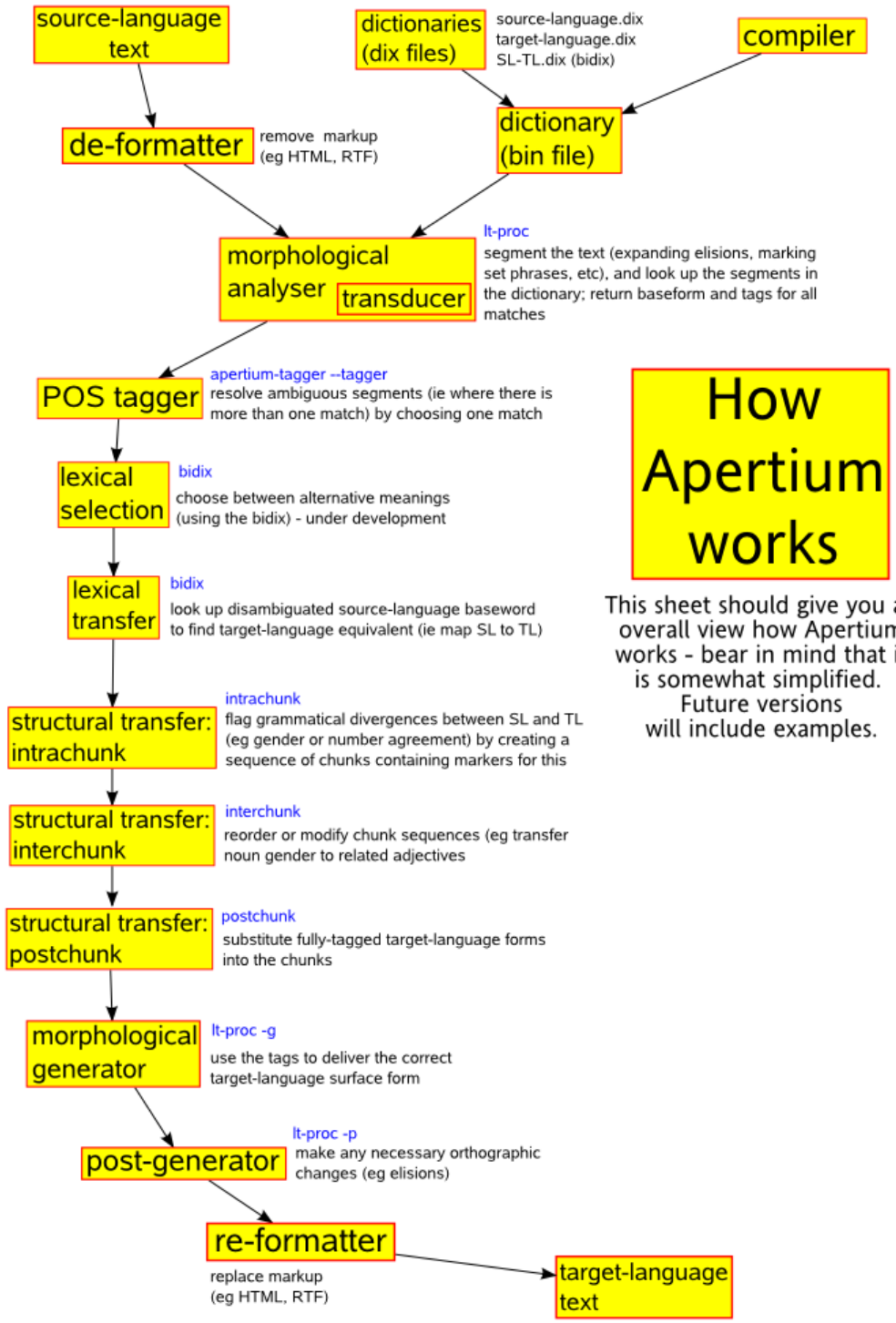


Fig. 8 The Apertium pipeline, taken from the *Apertium for Dummies* section of Apertium's wiki

```
<section-def-cats>
  <def-cat n="c_nom">
    <cat-item tags="n.*"/>
  </def-cat>
  <def-cat n="c_vrb">
    <cat-item tags="vblex.*"/>
  </def-cat>
```

```
<def-attr n="nbr">
  <attr-item tags="sg"/>
  <attr-item tags="pl"/>
</def-attr>
<def-attr n="a_nom">
  <attr-item tags="n"/>
</def-attr>
```

Fig. 10 Category and Attribute definitions in a transfer rule file

Additional rules are needed, however, to govern the transfer inflections, word sense and ordering. There are three separate *structural transfer* rule files that contain such information. These transfer rules determine agreement among tenses and pronouns. They also handle word reordering and anything else needed to transform the source sentence to the target language. The three structural transfer files are separated into the three modules that use them: *Intrachunk* (.t1x), which tag grammatical differences between the source an target languages and create a sequence of marked-up chunks[2]. *Interchunk* (.t2x), which handles reordering of chunk sequences and modifying lexical units (words or phrases) that have been tagged for alteration in the intrachunk phase. *Post chunk* (.t3x), transforms the output of the interchunk phase back into a tagged sentence, breaking up the chunks and assigning the appropriate tags to each lexical unit.

[2] A *chunk* is the label used for phrases in Apertium, and may contain a single word or multiple words, depending on their intended purpose.

All of the transfer rule files use, with some minor exceptions, the same basic structure. Category definitions, <def-cats>, are used to define the chunks that will be worked with in the rules. The use of chunks are primarily to generate agreement between words and do the reordering of a sentence. Attribute definitions, <def-attr>, are used to group similar symbols together as a type. As an example, the symbols for singular and plural (sg and pl, respectively) are be grouped together under a "nbr" attribute. There are optional sections for macros, in this setting generic rules to be applied to other rules and other, more advanced rules, then the rules section itself. Figure 10 shows some examples of these sections, and Figure 11 below shows

```
<rule>
   <pattern>
     <pattern-item n="c_nom"/>
   </pattern>
   <action>
     <out>
       <lu>
         <lit v="det"/>
         <lit-tag v="det"/>
         <clip pos="1" side="tl" part="nbr"/>
       </lu>
       <b/>
       <lu>
         <clip pos="1" side="tl" part="lem"/>
         <clip pos="1" side="tl" part="a_nom"/>
         <clip pos="1" side="tl" part="nbr"/>
       </lu>
     </out>
   </action>
</rule>
```

Fig. 11 A simple transfer rule from an intrachunk file

a simple rule to add a determiner to a noun (Japanese does not normally use determiners, while English often does).

There, of course, could be many rules that apply to nouns, so the example in Figure 14 is just one. The more specific the rule is, the higher priority Apertium assigns to it, and will be applied over more generic rules. Thus a rule that takes the pattern “noun-particle-verb” would match a chunk comprised of that pattern and be applied over a different rule that takes the pattern “noun-particle”, even though “noun-particle” also matches to that chunk. Once matched, these rules perform the commands listed in the <action> section. In the example in Figure 14, the rule matches a single nominal noun from Japanese and outputs a string literal “det” as a placeholder for a determiner and tags it both with the appropriate POS (“det”) and the ‘nbr’ tag from the noun (indicating if it is a singular or plural noun, which would alter the determiner choice), followed by the noun itself with it’s grammar rules, ‘noun’ tag and ‘nbr' tag. In short, it takes a single noun and turns it into a noun phrase that consists of a determiner and noun with matching ‘nbr’ tags.

A simple example would be a chunk that is made up of only the Japanese noun "家<n><sg>", which means "house" or "home" with noun and singular POS tags. This chunk would match the above rule exactly and execute the actions contained in the rule, returning the string "det<sg><b/>家<n><sg>"[3]. A full example of the shallow-transfer[4] Apertium language pair generated using the system and tools described in this thesis is shown in Appendix A.

4.5.Apertium Dictionary Builder

Each module of Apertium relies on the previous module to send properly formatted input to function correctly. And every rule must add (or remove) the appropriate tag(s) to produce a correct translation. As a result, both monolingual dictionaries must have corresponding entries, and the bilingual dictionary must map those lemmas to each other to properly translate. This creates a lot of cross-referencing of dictionary files when building a language pair, and, as the pair grows, it becomes difficult to keep track of every entry as well as time-consuming. Also, a certain knowledge of and comfort level with XML syntax is needed to navigate and create these files,

[3] <b/> is Apertium's tag to indicate a space within a chunk or phrase.

[4] A shallow-transfer pair uses only one transfer rules file, with all expected grammar patterns, including reordering, coded in the single file.

creating a barrier of entry to skilled linguists with little computer experience. We built the Apertium Dictionary Builder tool to address these issues.

The Dictionary Builder is a simple entry tool that allows for the simultaneous creation of all three dictionary files for a language pair (the monolingual dictionaries for each language and the bilingual dictionary) and does this while requiring no knowledge of XML. The interface is intelligent enough to warn when required information is missing and provide some default information in fields as the user inputs data. It also allows importing and exporting of files, so one may add entries to properly formatted existing dictionaries. A full description and instructions on use of the tool is detailed in Appendix C.

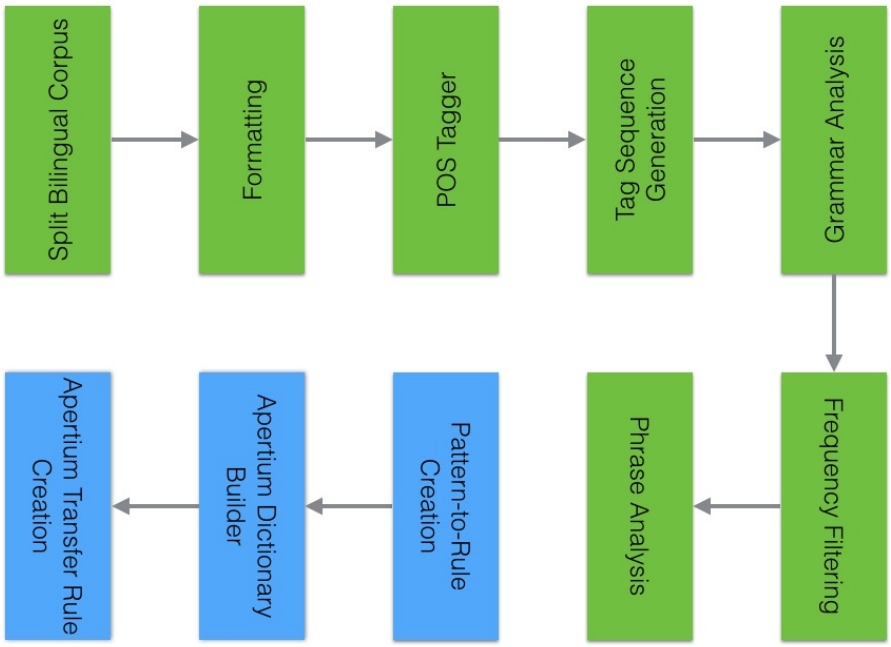


Fig. 12 A System flow diagram of the grammar analysis, rule generation and Apertium system building. The green modules are automatically processed, while the blue modules require manual intervention. The output of this system is a shallow-transfer language pair that may be used by Apertium.

Chapter 5.Generating Rules

5.1.Preprocessing

When generating the grammar rules, some filtering and preprocessing is required. This is partially due to the sheer number of grammar rules in any given language. Linguist David Crystal estimates that the English language, as mentioned earlier a weakly inflected language, has 3500 or so distinct rules (Crystal, 2007), so any attempt at partially automated grammar generation needs to be reduced in scope. First, we split the bilingual corpus into two monolingual corpora, stripping extraneous tags and formatting so we have only monolingual

sentences in each corpus. We then run the corpus containing our target language through a POS tagger to generate tags for every sentence. This gives us a tagged corpus that is ready to be analyzed.

Once the target corpus has been prepared, a multi-pass analysis is done on the sentences, generating *tag sequences*, a representation of the sentences using only POS tags and indexing them against the original sentences. We also keep a record of each tagged word and its frequency for use in building the dictionary files once the rules have been generated. From there, a statistical analysis is done on each bigram and larger phrase in all the tag sequences to find the most commonly occurring grammatical patterns in the corpus. To keep the amount of patterns to a reasonable amount (our target was between 30-60 patterns), these are filtered to those occurring 10,000 times or more. In this project, among the 150,000 sentences, it returned roughly the top 10% most common grammar patterns. After the list is filtered, further analysis is done, larger patterns that are made up of several smaller patterns are removed, as are patterns with only verb tense variations. This filtering generated, for us, 47 unique base grammatical patterns for which to generate rules.

5.2.Parsing rules into Apertium

5.2.1.Interpreting the Rules

Once the preprocessing steps are completed, the results must be interpreted from the POS patterns back into actual grammar rules. To do this, each pattern is mapped back to the ID numbers of the original sentences that contained them. Figure 13, below, shows a sample of that output. The first line denotes the grammar pattern, with POS tags represented by numbers, separated by a '$' character. Below the pattern is a list of ID numbers that map to sentences in the corpus containing the given pattern. Those sentences are then analyzed by hand to determine

```
1$2$4$5
6 7 10 16 19 21 22 24 28 29 30 31 35 37 38 42 43 45 49 52 53 57 60 63 64 65 66 67 71 74 78
79 82 83 86 88 89 90 92 95 96 99 102 104 105 107 108 112 113 114 117 119 122 126 130 131
132 137 139 140 142 143 145 146 148 149 150 151 154 156 159 165 166 167 168 169 170 171
173 176 180 181 182 183 185 191 192 194 201 202 205 206 207 208 209 210 212 215 216 217
222 223 225 226 227 230 231 232 233 234 235 238 239 242 244 245 246 248 251 252 253 256
257 260 263 269 272 274 277 280 281 286 287 289 290 295 296 297 300 301 302 303 306 307
308 309 310 312 313 314 316 320 322 323 325 327 328 329 331 334 337 338 340 341 343 346
351 353 354 355 356 357 359 362 365 367 368 369 375 377 378 379 380 382 384 385 386 388
390 391 395 396 397 399 400 401 402 406 407 410 412 414 415 416 417 418 419 421 424 425
426 428 429 430 432 436 439 441 443 444 445 447 448 451 453 454 455 456 457 458 459 461
464 466 468 470 471 474 476 477 478 481 484 487 488 491 492 493 494 496 497 503 505 506
507 509 510 512 513 514 516 522 523 527 530 532 533 534 536 539 540 543 544 545 546 548
550 552 554 556 560 561 562 564 565 566 569 571 573 576 580 581 582 583 584 590 591 593
597 598 599 607 609 610 611 614 615 617 618 619 620 622 624 625 626 627 628 630 631 636
637 638 643 649 652 654 655 659 661 664 665 673 675 677 678 679 681 682 683 685 687 688
690 692 693 694 695 698 705 709 712 716 717 724 729 731 732 733 735 742 745 746 750 752
753 754 757 759 762 763 764 765 770 771 772 777 778 780 781 782 784 785 788 790 791 792
793 795 796 800 802 803 810 813 820 827 834 837 838 845 846 847 849 852 853 854 855 856
859 861 862 863 866 868 870 873 876 877 881 884 885 887 889 891 895 896 903 908 909 910
915 917 921 923 925 928 932 935 937 938 939 944 946 947 951 952 954 955 956 957 959 971
```

Fig. 13 The pattern 1$2$4$5 (1 = noun, 2 = particle, 3 = verb, 4 = verb suffix) mapped to the ID numbers of all the sentences containing that phrase type in the corpus.

the associated grammatical structure the patterns represented.

Some of them were fairly obvious, such as simple verb conjugations (present tense ichidan verb suffix = masu, for example). Other patterns must be mapped to several rules, as the POS patterns do not, for example, differentiate between types of particles in the most common trigram, noun-particle-verb. While this pattern represents the most basic structure of Japanese, the rules and resulting meanings change depending on the particle (Figure 14). This can also, at times change the verb

Sensei wa tabemasu.	The teacher eats.
Sensei ga tabemasu.	The teacher eats./It is the teacher that eats.
Sensei to tabemasu.	I eat with the teacher.
Sensei mo tabemasu.	The teacher also eats.
Sensei wo tabemasu	I eat the teacher.

Fig 14 Japanese particles altering the meaning of a sentence

form in the target language, as in the above example with eat or eats. Since the personal pronoun is optional in Japanese, the target language verb form must agree with different subjects, depending on the particle used. This requires specific transfer rules for each particle type to determine if a pronoun is to be added to the sentence, where it should be added and if the verb form should be altered. These patterns were gone through again by hand, this time differentiating the particle types and creating additional rules for directional particles, possessive particles, etc.

Once all of this had been done, we combined the results of the initial analysis, the additional patterns for particles with some prerequisite grammar definitions (verb conjugations, pronouns, noun pluralization) and this makes up the final list of rules to be implemented in Apertium.

5.2.2.Formatting the Rules for Apertium

The XML structure of Apertium's data files made it easy to build a template for the needed files and to pull in some basic information, such as dictionary entries, automatically by simply scanning the dictionary and pulling the entries and their POS tags and parsing them into Apertium entries. However, the interconnectedness of the files, the complexity of building some of the resulting rules and the needed linguistic knowledge to put into the rules themselves requires a good deal of hand-coding.

To this end, we decided to generate dictionary files using the Dictionary Builder tool and the most common nouns and verbs discovered during the tagging process. This allows us to add the Symbol Definitions and some of the necessary syntax rules while building the dictionary.

5.3.Evaluation

Once all this was in place, the system was tested with known good sample phrases and sentences. Sentences in Japanese were sent as input one at a time to Apertium, which passed the sentences through its modules, using the dictionaries and transfer rules we coded as its reference, and returned a translation of the phrase or sentence in English. Only words that were known to exist in the dictionaries were used. We evaluated the four steps of our basic Apertium translation process for correct output:

1) Morphological Analysis: Recognizing the dictionary entry for each word and adding it's base tags.
2) POS and Semantic Tagging: Source language pardef entry selection, disambiguation and application.
3) Lexical Transfer: Word reordering and transformations based on transfer rules and translating to base forms of the target language.
4) Final Output: Target language pardef application and tag removal.

If a sentence fails at any of these steps, the translation itself will fail, as each successive step requires a correct input from the previous step. A failure indicates that there is either an error in one of the rules or there is a necessary rule that is missing from the system. Unlike SMT and HMT systems, RBMT systems will always return the proper output if the rules and words required for a given sentence are present and, conversely, not return proper output if there is any unknown or unexpected input, unless a specifically assigned default value is coded.

6.Results

6.1.Overview

In this thesis, we described some tools that were built to infer grammar rules from a bilingual corpus and bilingual dictionary to assist in generating a basic grammar rule structure for the Apertium rule-based translation system. In the next few chapters we will discuss the results of this work, some conclusions and possible future work.

The automatic generation of basic grammatical patterns from the Tanaka corpus, a bilingual text of over 150,000 sentence pairs, and the WWWJDIC dictionary, an electronic Japanese-English dictionary with over 150,000 unique entries, produced 1,761,231 phrasal patterns of two or more POS tags. After filtering out duplicates and all patterns occurring less than 10,000 times in the corpus, there were 115 unique patterns remaining. These remaining patterns were analyzed manually to remove redundant patterns that shared everything but verb tense and longer patterns that were made up of smaller, existing patterns. These remaining base patterns were combined with their constituent POS rules necessary to build larger phrasal rules and yielded a total of 52 basic rule patterns that were chosen to use for rule encoding.

Once the rules were selected, words and phrases containing and/or using these rules were chosen from the corpus to populate the dictionary files. The paradigms for the English monolingual dictionary entries were imported from an existing Apertium dictionary file and altered to match the tag structure of the rest of the system. Symbol names were altered to match those chosen for the Japanese monolingual dictionary. The Japanese monolingual dictionary entries were hand coded, based on the information gathered from rule selection. Finally, the transfer rules, again based on the POS patterns from rule selection, were hand coded into the intrachunk file, as described in Section 4.1.4. The resulting system contained 48 base Japanese words, 49 base English words, 49 base Japanese rules, 141 base English rules and 22 transfer rules that handled lexical agreement and word ordering.

6.2.Test Results

For each of the 52 rules we generated, we ran a test through our newly created Apertium language pair, using only words and phrases known to exist in the pair, and checked those tests against the four metrics explained in Chapter 5. The expectation was that all the tests would pass, as the system was designed

based on the specific rules that were being tested. As mentioned in Chapter 5, RBMT systems will either produce a (mostly) correct translation or, if the input does not match any rules or patterns in the system, it will fail.

The test results, shown in Appendix B, showed the expected output for all 18 test sentences through each of the four modules of the shallow-transfer Apertium system. This is unsurprising, as the tests were limited to the words and rules contained within the language pair. Were we to input some word or grammar pattern not coded into the system, it would fail to output a correct translation. Still, these results show that grammar rules generated by the system described in this thesis can be used to create the foundation for a new language pair that could be expanded and improved upon.

6.Conclusion

6.1.Summary

This work contributes toward the goal of developing a new partially automated method for generating rules for a machine translation system. Automating the generation of rules would be significant, as it would reduce the time and expertise needed to build a new set of rules for a language pair, such as Japanese and English, which does not presently exist. The results of the work are promising, but suggest that there is still significant work to be done. Converting a corpus into the intermediate representation of grammatical patterns and running statistical analysis on the results did somewhat reduce the amount of manual work involved in generating rule structures and required very little upfront grammatical knowledge of the source language. However, once the most basic patterns had been imported, the work of coding the rules, in the Apertium system at least, required a deeper linguistic knowledge of both languages to generate reasonable sentence structures, verb forms and accurate translations. Furthermore, additional rules, as is often the case in interconnected systems, in many cases require altering, or adding to, existing rules. In the end, the overall results of the rule generation and the resultant system did not match initial expectations, either in scope or complexity.

The key contribution of this work is to provide a basic Japanese-English language pair for an open-source RBMT platform that can be improved and expanded upon, where previously none existed. Additionally, the system created for rule generation is not specific to Japanese, and, with some adjustment for formatting and an appropriate POS tagger, could be used to generate grammar patterns and rules for virtually any source language. Finally, the Apertium Dictionary Builder tool is a small, cross-platform application that could enable those with linguistic skill who are not comfortable working with XML to contribute their knowledge to Apertium language pairs.

6.2.Limitations

There is clearly much work that could yet be done to this system. Certainly adding to the dictionary files and rule files of the Apertium system and making it more robust would be useful. Also, lowering the filter to allow larger patterns may yield more complex transfer rule structures that could reduce the overall rule count or handle more complex syntax structures. The work has met the initial goal of this project, which was to, at least partially automatically, generate grammar rules that could be used in any system. However, there are limitations,

such as ambiguity, context and alternate usages that cannot be addressed simply by adding more rules or grammatical patterns.

6.3.Future Work

It now seems likely that investing in an entirely new approach might be worth pursuing. The usefulness of intermediate representation in both the generating of grammar rules and in Apertium's transfer rules could hint at pursuing some form of interlingua as a way to address the limitations from the conclusion of this paper.

Most MT systems (Apertium included) employ some basic form of interlingua, such as POS and grammar tagging, to give additional meaning to written representations of words. In these systems the interlingua is used often only to find the proper syntax or correct grammar. Occasionally it will remove some ambiguity. This is, of course, important, but translation is about conveying the meaning of a sentence from one language to another. Systems such as WordNet group clusters of words that share similar meaning, using numeric ID tags as an interlingua that conveys the broad meaning, but do not offer any grammatical or syntactic information.

A better approach might be to expand the amount of meaning stored in interlingua, and use that intermediate representation to choose words and phrases that carry the same meaning from the source to the target language. Using some knowledge base, perhaps expanding WordNet by increasing the amount of IDs and making the meaning they represent more specific, to map words and phrases to their meanings. A system could use several layers of interlingua: one for meaning, one to attach grammar and syntax information to the interlingua containing the meaning. The system could then employ existing SMT or RBMT techniques to reorder the phrases into target language syntax and grammar. The goal would be to remove the meaning from the mechanics of word ordering and grammatical structure and translate just the intent of the communication from the source language, then apply the appropriate rules to form a grammatically correct sentence in the target language.

Bibliography

Akeel, M., & Mishra, R. (2014). ANN and Rule Based Method for English to Arabic Machine Translation. International Arab Journal of Information Technology (IAJIT), 11(4).

Baker, K., Bloodgood, M., Callison-Burch, C., Dorr, B. J., Filardo, N. W., Levin, L., . . . Piatko, C. (2014). Semantically-informed syntactic machine translation: A tree-grafting approach. arXiv preprint arXiv:1409.7085.

Berger, A. L., Della Pietra, V. J., Della Pietra, S. A. (1996). A Maximum Entropy Approach to Natural Language Processing. Computational Linguistics, 22(1), 39-71.

Bijimol, T., & Abraham, J. T. (2014). A Study of Machine Translation Methods.

Bisbey, R. L., & Kay, M. (1972). The MIND Translation System: A Study in Man-Machine Collaboration.

Breen, J. (1995). Building an electronic Japanese-English dictionary. Paper presented at the Japanese Studies Association of Australia Conference.

Brown, P., Della Pietra, V. J., Della Pietra, S. A., Mercer, R. L. (1993). The Mathematics of Statistical Machine Translation: Parameter Estimation. Computational Linguistics, 19(2), 263-311.

Carnie, A. (2013). Syntax: A generative introduction: John Wiley & Sons.

Charniak, E. (1997). Statistical techniques for natural language parsing. AI magazine, 18(4), 33.

Chiang, D. (2005). A hierarchical phrase-based model for statistical machine translation. Paper presented at the Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics.

Church, K. W. (1988). A stochastic parts program and noun phrase parser for unrestricted text. Paper presented at the Proceedings of the second conference on Applied natural language processing.

Costa-jussá, M., Farrús, M., & Pons, J. (2012). Machine translation in medicine. A quality analysis of statistical machine translation in the medical domain. Advanced Research in Scientific Areas.

Crystal, D. (2007). On counting English grammar. Retrieved from http://david-crystal.blogspot.com/2007/04/on-counting-english-grammar.html

España Bonet, C., Màrquez Villodre, L., Labaka, G., Díaz de Ilarraza Sánchez, A., & Sarasola Gabiola, K. (2011). Hybrid machine translation guided by a rule-based system.

Ethnologue. (2015). Ethnologue-Languages of the World. Retrieved from http://www.ethnologue.com/statistics

Farzi, S. H. F. S. K. J. M. (2013). A Novel Reordering Model for Statistical Machine Translation. Research in Computing Science(54), 51-64. Retrieved from http://www.micai.org/rcs/2013_65/

Forcada, M. L., et al. (2011). Apertium: a free/open-source platform for rule-based machine translation. Machine Translation, 25(2), 127-144.

Foster, G., Isabelle, P., & Plamondon, P. (1997). Target-text mediated interactive machine translation. Machine Translation, 12(1-2), 175-194.

Francis, W. N., Kucera, H. (1964). Brown Corpus. Retrieved from: http://clu.uni.no/icame/manuals/BROWN/INDEX.HTM

Google. (2015). Google Translate-Languages. Retrieved from https://translate.google.com/about/intl/en_ALL/languages.html

Ho, T. (2009). Tatoeba Project Blog.

Hutchins, J. (2007). Machine translation: A concise history. Computer aided translation: Theory and practice.

Kamiya, T. (2012). The Handbook of Japanese Verbs (Rep Blg ed.): Kodansha.

Koehn, P. (2009). Statistical machine translation: Cambridge University Press.

Koehn, P., Och, F. J., & Marcu, D. (2003). Statistical phrase-based translation. Paper presented at the Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1.

Koetsier, J. (2013, 3-1-2013). How Google searches 30 trillion web pages, 100 billion times a month. VBNews. Retrieved from http://venturebeat.com/2013/03/01/how-google-searches-30-trillion-web-pages-100-billion-times-a-month/

Li, H., Zhu, Y., & Jin, Y. (2015). Identifying Verb-Preposition Multi-Category Words in Chinese-English Patent Machine Translation Artificial Life and Computational Intelligence (pp. 409-421): Springer.

Li, J., Marton, Y., Resnik, P., & Daumé III, H. (2014). A unified model for soft linguistic reordering constraints in statistical machine translation. Paper presented at the Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics.

Lu, Y., Wang, L., Wong, D. F., Chao, L. S., Wang, Y., & Oliveira, F. (2014). Domain Adaptation for Medical Text Translation Using Web Re-sources. ACL 2014, 233.

Marcus, M. P., Marcinkiewicz, M. A., & Santorini, B. (1993). Building a large annotated corpus of English: The Penn Treebank. Computational Linguistics, 19(2), 313-330.

Maruyama, H., Watanabe, H., & Ogino, S. (1990). An interactive Japanese parser for machine translation. Paper presented at the Proceedings of the 13th conference on Computational linguistics-Volume 2.

Neubig, G., Watanabe, T., & Mori, S. (2012). Inducing a discriminative parser to optimize machine translation reordering. Paper presented at the Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning.

Och, F. J., & Ney, H. (2002). Discriminative training and maximum entropy models for statistical machine translation. Paper presented at the Proceedings of the 40th Annual Meeting on Association for Computational Linguistics.

Ortiz-Martınez, D., & Casacuberta, F. (2014). The New THOT Toolkit for Fully-Automatic and Interactive Statistical Machine Translation. EACL 2014, 45.

Park, E.-j., Kwon, O.-W., Kim, K., & Kim, Y.-k. A Classification-based Approach for Hybridizing Statistical Machine Translation and Rule-based Machine Translation. ETRI Journal.

Petrov, S., Das, D., & McDonald, R. (2011). A universal part-of-speech tagset. arXiv preprint arXiv:1104.2086.

Princeton. (2014, 2014/8/26). WordNet: a lexical database in English. Retrieved from http://wordnet.princeton.edu/

Reynar, J. C., & Ratnaparkhi, A. (1997). A maximum entropy approach to identifying sentence boundaries. Paper presented at the Proceedings of the fifth conference on Applied natural language processing.

Sánchez Martínez, F., Forcada Zubizarreta, M. L., & Way, A. (2009). Hybrid rule-based-example-based MT: feeding Apertium with sub-sentential translation units.

Sánchez-Cartagena, V., Pérez-Ortiz, J. A., Sánchez-Martínez, F.,. (2014). The UA-Prompsit hybrid machine translation system for the 2014 Workshop on Statistical Machine Translation. ACL 2014, 178-185.

Somers, H. (1999). Review Article: Example-based Machine Translation. Machine Translation, 14(2), 113-157. doi: 10.1023/A:1008109312730

Tanaka, Y. (2001). Compilation of a multilingual parallel corpus. Proceedings of PACLING 2001, 265-268.

Tomita, M. (1985). Feasibility study of personal interactive machine translation systems.

Uchida, H., Zhu, M., & Della Senta, T. (2005). Universal Networking Language. UNDL foundation.

Van Valin, R. D. (2001). An introduction to syntax: Cambridge University Press.

Zens, R., Och, F. J., & Ney, H. (2002). Phrase-Based Statistical Machine Translation. In M. Jarke, G. Lakemeyer, & J. Koehler (Eds.), KI 2002: Advances in Artificial

Intelligence (Vol. 2479, pp. 18-32): Springer Berlin Heidelberg.

Zhang, J., Liu, S., Li, M., Zhou, M., & Zong, C. (2015). Beyond Word-based Language Model in Statistical Machine Translation. arXiv preprint arXiv:1502.01446.

Appendix A.Apertium Files

apertium-en.en.dix

```
<?xml version="1.0" encoding="UTF-8"?>
<dictionary>
<alphabet></alphabet>

<sdefs>
  <sdef n="n"               c="noun"/>
  <sdef n="sg"                   c="singular"/>
  <sdef n="pl"          c="plural"/>
  <sdef n="sgpl"        c="singular or plural"/>
  <sdef n="prn"         c="pronoun"/>
  <sdef n="subj"          c="subject"/>
  <sdef n="obj"         c="object"/>
  <sdef n="p1"          c="first person"/>
  <sdef n="p2"          c="second person"/>
  <sdef n="p3"          c="third person"/>
  <sdef n="m"           c="masculine"/>
  <sdef n="f"           c="feminine"/>
  <sdef n="mf"          c="masculine/feminine"/>
  <sdef n="adv"         c="adverb"/>
  <sdef n="nn"          c="inanimate"/>
  <sdef n="an"          c="animate/inanimate"/>
  <sdef n="aa"          c="animate"/>
  <sdef n="det"         c="determiner"/>
  <sdef n="def"         c="definite article"/>
  <sdef n="ind"         c="indefinite article"/>
  <sdef n="vblex"       c="verb"/>
  <sdef n="iru"         c="verb 'to be'"/>
  <sdef n="pres"        c="present tense"/>
  <sdef n="past"        c="past tense"/>
  <sdef n="pot"         c="potential tense"/>
  <sdef n="neg"                 c="negative"/>
  <sdef n="prt"         c="particle"/>
  <sdef n="n_suff"      c="noun suffix"/>
  <sdef n="poss"        c="possesive"/>
  <sdef n="dir"         c="direction/movement"/>
  <sdef n="adj"         c="adjective"/>
  <sdef n="copula"      c="copula"/>
  <sdef n="dem_adj"     c="demonstrative adjective"/>
  <sdef n="too"         c="too"/>
</sdefs>

<pardefs>
  <pardef n="house_n">
    <e><p><l/><r><s n="n"/><s n="sg"/></r></p></e>
    <e><p><l>s</l><r><s n="n"/><s n="pl"/></r></p></e>
  </pardef>
  <pardef n="pe/rson_n">
    <e><p><l>rson</l><r>rson<s n="n"/><s n="sg"/></r></p></e>
    <e><p><l>ople</l><r>rson<s n="n"/><s n="pl"/></r></p></e>
  </pardef>
  <pardef n="work_n">
    <e><p><l/><r><s n="n"/><s n="sg"/></r></p></e>
    <e><p><l/><r><s n="n"/><s n="pl"/></r></p></e>
  </pardef>
  <pardef n="child_n">
    <e><p><l/><r><s n="n"/><s n="sg"/></r></p></e>
    <e><p><l>ren</l><r><s n="n"/><s n="pl"/></r></p></e>
```

```
  </pardef>

  <pardef n="prsubj_prn">
    <e><p><l>I</l><r>prpers<s n="prn"/><s n="p1"/><s n="sg"/><s n="mf"/></r></p></e>
    <e><p><l>we</l><r>prpers<s n="prn"/><s n="p1"/><s n="pl"/><s n="mf"/></r></p></e>
    <e><p><l>he</l><r>prpers<s n="prn"/><s n="p3"/><s n="sg"/><s n="m"/></r></p></e>
    <e><p><l>she</l><r>prpers<s n="prn"/><s n="p3"/><s n="sg"/><s n="f"/></r></p></e>
    <e><p><l>they</l><r>prpers<s n="prn"/><s n="p3"/><s n="pl"/><s n="mf"/></r></p></
e>
    <e><p><l>it</l><r>prpers<s n="prn"/><s n="p3"/><s n="sg"/><s n="mf"/></r></p></e>
    <e><p><l>you</l><r>prpers<s n="prn"/><s n="p2"/><s n="sg"/><s n="mf"/></r></p></e>
    <e><p><l>you</l><r>prpers<s n="prn"/><s n="p2"/><s n="pl"/><s n="mf"/></r></p></e>
  </pardef>
  <pardef n="poss_prn">
    <e><p><l>my</l><r><s n="poss"/><s n="p1"/><s n="sg"/><s n="mf"/></r></p></e>
    <e><p><l>your</l><r><s n="poss"/><s n="p2"/><s n="sg"/><s n="mf"/></r></p></e>
    <e><p><l>her</l><r><s n="poss"/><s n="p3"/><s n="sg"/><s n="f"/></r></p></e>
    <e><p><l>his</l><r><s n="poss"/><s n="p3"/><s n="sg"/><s n="m"/></r></p></e>
    <e><p><l>our</l><r><s n="poss"/><s n="p1"/><s n="pl"/><s n="mf"/></r></p></e>
  </pardef>
  <pardef n="det_n">
    <e><p><l>a</l><r><s n="det"/><s n="obj"/><s n="sg"/></r></p></e>
    <e><p><l>the</l><r><s n="det"/><s n="subj"/><s n="sg"/></r></p></e>
    <e><p><l>some</l><r><s n="det"/><s n="obj"/><s n="pl"/></r></p></e>
    <e><p><l>there</l><r><s n="det"/><s n="subj"/><s n="sgpl"/><s n="copula"/></r></
p></e>
  </pardef>
  <pardef n="dir">
    <e><p><l></l><r><s n="dir"/></r></p></e>
  </pardef>
  <pardef n="too">
    <e><p><l></l><r><s n="too"/></r></p></e>
  </pardef>


  <pardef n="s/ee_vblex">
    <e><p><l>ee</l><r>ee<s n="vblex"/><s n="pres"/><s n="p1"/></r></p></e>
    <e><p><l>ee</l><r>ee<s n="vblex"/><s n="pres"/><s n="p2"/></r></p></e>
    <e><p><l>ees</l><r>ee<s n="vblex"/><s n="pres"/><s n="p3"/></r></p></e>
    <e><p><l>aw</l><r>ee<s n="vblex"/><s n="past"/><s n="p1"/></r></p></e>
    <e><p><l>aw</l><r>ee<s n="vblex"/><s n="past"/><s n="p2"/></r></p></e>
    <e><p><l>aw</l><r>ee<s n="vblex"/><s n="past"/><s n="p3"/></r></p></e>
  </pardef>
  <pardef n="walk_vblex">
    <e><p><l/><r><s n="vblex"/><s n="pres"/><s n="p1"/></r></p></e>
    <e><p><l/><r><s n="vblex"/><s n="pres"/><s n="p2"/></r></p></e>
    <e><p><l>s</l><r><s n="vblex"/><s n="pres"/><s n="p3"/></r></p></e>
    <e><p><l>ed</l><r><s n="vblex"/><s n="past"/><s n="p1"/></r></p></e>
    <e><p><l>ed</l><r><s n="vblex"/><s n="past"/><s n="p2"/></r></p></e>
    <e><p><l>ed</l><r><s n="vblex"/><s n="past"/><s n="p3"/></r></p></e>
  </pardef>
  <pardef n="hurr/y_vblex">
    <e><p><l>y</l><r>y<s n="vblex"/><s n="pres"/><s n="p1"/></r></p></e>
    <e><p><l>y</l><r>y<s n="vblex"/><s n="pres"/><s n="p2"/></r></p></e>
    <e><p><l>ies</l><r>y<s n="vblex"/><s n="pres"/><s n="p3"/></r></p></e>
    <e><p><l>ied</l><r>y<s n="vblex"/><s n="past"/><s n="p1"/></r></p></e>
    <e><p><l>ied</l><r>y<s n="vblex"/><s n="past"/><s n="p2"/></r></p></e>
    <e><p><l>ied</l><r>y<s n="vblex"/><s n="past"/><s n="p3"/></r></p></e>
  </pardef>
  <pardef n="me/et_vblex">
    <e><p><l>et</l><r>et<s n="vblex"/><s n="pres"/><s n="p1"/></r></p></e>
    <e><p><l>et</l><r>et<s n="vblex"/><s n="pres"/><s n="p2"/></r></p></e>
    <e><p><l>ets</l><r>et<s n="vblex"/><s n="pres"/><s n="p3"/></r></p></e>
    <e><p><l>t</l><r>et<s n="vblex"/><s n="past"/><s n="p1"/></r></p></e>
```

```
    <e><p><l>t</l><r>et<s n="vblex"/><s n="past"/><s n="p2"/></r></p></e>
    <e><p><l>t</l><r>et<s n="vblex"/><s n="past"/><s n="p3"/></r></p></e>
  </pardef>
  <pardef n="d/o_vblex">
    <e><p><l>o</l><r>o<s n="vblex"/><s n="pres"/><s n="p1"/></r></p></e>
    <e><p><l>o</l><r>o<s n="vblex"/><s n="pres"/><s n="p2"/></r></p></e>
    <e><p><l>oes</l><r>o<s n="vblex"/><s n="pres"/><s n="p3"/></r></p></e>
    <e><p><l>id</l><r>o<s n="vblex"/><s n="past"/><s n="p1"/></r></p></e>
    <e><p><l>id</l><r>o<s n="vblex"/><s n="past"/><s n="p2"/></r></p></e>
    <e><p><l>id</l><r>o<s n="vblex"/><s n="past"/><s n="p3"/></r></p></e>
  </pardef>
  <pardef n="go_vblex">
    <e><p><l>go</l><r>go<s n="vblex"/><s n="pres"/><s n="p1"/></r></p></e>
    <e><p><l>go</l><r>go<s n="vblex"/><s n="pres"/><s n="p2"/></r></p></e>
    <e><p><l>goes</l><r>go<s n="vblex"/><s n="pres"/><s n="p3"/></r></p></e>
    <e><p><l>went</l><r>go<s n="vblex"/><s n="past"/><s n="p1"/></r></p></e>
    <e><p><l>went</l><r>go<s n="vblex"/><s n="past"/><s n="p2"/></r></p></e>
    <e><p><l>went</l><r>go<s n="vblex"/><s n="past"/><s n="p3"/></r></p></e>
  </pardef>
  <pardef n="be_copula">
    <e><p><l>am</l><r>be<s n="copula"/><s n="pres"/><s n="p1"/></r></p></e>
    <e><p><l>was</l><r>be<s n="copula"/><s n="past"/><s n="p1"/></r></p></e>
    <e><p><l>are</l><r>be<s n="copula"/><s n="pres"/><s n="p2"/></r></p></e>
    <e><p><l>is</l><r>be<s n="copula"/><s n="pres"/><s n="p3"/></r></p></e>
    <e><p><l>are</l><r>be<s n="copula"/><s n="pres"/><s n="p3"/><s n="pl"/></r></p></
e>
    <e><p><l>were</l><r>be<s n="copula"/><s n="past"/><s n="p2"/></r></p></e>
    <e><p><l>was</l><r>be<s n="copula"/><s n="past"/><s n="p3"/></r></p></e>
    <e><p><l>were</l><r>be<s n="copula"/><s n="past"/><s n="p3"/><s n="pl"/></r></p></
e>
  </pardef>
  <pardef n="is_copula">
    <e><p><l>am</l><r>is<s n="copula"/><s n="pres"/><s n="p1"/></r></p></e>
    <e><p><l>was</l><r>is<s n="copula"/><s n="past"/><s n="p1"/></r></p></e>
    <e><p><l>are</l><r>is<s n="copula"/><s n="pres"/><s n="p2"/></r></p></e>
    <e><p><l>is</l><r>is<s n="copula"/><s n="pres"/><s n="p3"/></r></p></e>
    <e><p><l>are</l><r>is<s n="copula"/><s n="pres"/><s n="p3"/><s n="pl"/></r></p></
e>
    <e><p><l>were</l><r>is<s n="copula"/><s n="past"/><s n="p2"/></r></p></e>
    <e><p><l>was</l><r>is<s n="copula"/><s n="past"/><s n="p3"/></r></p></e>
    <e><p><l>were</l><r>is<s n="copula"/><s n="past"/><s n="p3"/><s n="pl"/></r></p></
e>
  </pardef>
  <pardef n="c/ome_vblex">
    <e><p><l>ome</l><r>ome<s n="vblex"/><s n="pres"/><s n="p1"/></r></p></e>
    <e><p><l>ome</l><r>ome<s n="vblex"/><s n="pres"/><s n="p2"/></r></p></e>
    <e><p><l>omes</l><r>ome<s n="vblex"/><s n="pres"/><s n="p3"/></r></p></e>
    <e><p><l>ame</l><r>ome<s n="vblex"/><s n="past"/><s n="p1"/></r></p></e>
    <e><p><l>ame</l><r>ome<s n="vblex"/><s n="past"/><s n="p2"/></r></p></e>
    <e><p><l>ame</l><r>ome<s n="vblex"/><s n="past"/><s n="p3"/></r></p></e>
  </pardef>
  <pardef n="sa/y_vblex">
    <e><p><l>y</l><r>y<s n="vblex"/><s n="pres"/><s n="p1"/></r></p></e>
    <e><p><l>y</l><r>y<s n="vblex"/><s n="pres"/><s n="p2"/></r></p></e>
    <e><p><l>ys</l><r>y<s n="vblex"/><s n="pres"/><s n="p3"/></r></p></e>
    <e><p><l>id</l><r>y<s n="vblex"/><s n="past"/><s n="p1"/></r></p></e>
    <e><p><l>id</l><r>y<s n="vblex"/><s n="past"/><s n="p2"/></r></p></e>
    <e><p><l>id</l><r>y<s n="vblex"/><s n="past"/><s n="p3"/></r></p></e>
  </pardef>
  <pardef n="can_vblex">
    <e><p><l>can</l><r>ome<s n="vblex"/><s n="pres"/><s n="p1"/></r></p></e>
    <e><p><l>can</l><r>ome<s n="vblex"/><s n="pres"/><s n="p2"/></r></p></e>
    <e><p><l>can</l><r>ome<s n="vblex"/><s n="pres"/><s n="p3"/></r></p></e>
    <e><p><l>could</l><r>ome<s n="vblex"/><s n="past"/><s n="p1"/></r></p></e>
```

```
  <e><p><l>could</l><r>ome<s n="vblex"/><s n="past"/><s n="p2"/></r></p></e>
  <e><p><l>could</l><r>ome<s n="vblex"/><s n="past"/><s n="p3"/></r></p></e>
</pardef>
<pardef n="think_vblex">
  <e><p><l>think</l><r>ome<s n="vblex"/><s n="pres"/><s n="p1"/></r></p></e>
  <e><p><l>think</l><r>ome<s n="vblex"/><s n="pres"/><s n="p2"/></r></p></e>
  <e><p><l>think</l><r>ome<s n="vblex"/><s n="pres"/><s n="p3"/></r></p></e>
  <e><p><l>thought</l><r>ome<s n="vblex"/><s n="past"/><s n="p1"/></r></p></e>
  <e><p><l>thought</l><r>ome<s n="vblex"/><s n="past"/><s n="p2"/></r></p></e>
  <e><p><l>thought</l><r>ome<s n="vblex"/><s n="past"/><s n="p3"/></r></p></e>
</pardef>
<pardef n="kn/ow_vblex">
  <e><p><l>ow</l><r>ome<s n="vblex"/><s n="pres"/><s n="p1"/></r></p></e>
  <e><p><l>ow</l><r>ome<s n="vblex"/><s n="pres"/><s n="p2"/></r></p></e>
  <e><p><l>ows</l><r>ome<s n="vblex"/><s n="pres"/><s n="p3"/></r></p></e>
  <e><p><l>ew</l><r>ome<s n="vblex"/><s n="past"/><s n="p1"/></r></p></e>
  <e><p><l>ew</l><r>ome<s n="vblex"/><s n="past"/><s n="p2"/></r></p></e>
  <e><p><l>ew</l><r>ome<s n="vblex"/><s n="past"/><s n="p3"/></r></p></e>
</pardef>
<pardef n="ha/ve_vblex">
  <e><p><l>ve</l><r>ve<s n="vblex"/><s n="pres"/><s n="p1"/></r></p></e>
  <e><p><l>ve</l><r>ve<s n="vblex"/><s n="pres"/><s n="p2"/></r></p></e>
  <e><p><l>as</l><r>ve<s n="vblex"/><s n="pres"/><s n="p3"/></r></p></e>
  <e><p><l>d</l><r>ve<s n="vblex"/><s n="past"/><s n="p1"/></r></p></e>
  <e><p><l>d</l><r>ve<s n="vblex"/><s n="past"/><s n="p2"/></r></p></e>
  <e><p><l>d</l><r>ve<s n="vblex"/><s n="past"/><s n="p3"/></r></p></e>
</pardef>
<pardef n="hear_vblex">
  <e><p><l/><r><s n="vblex"/><s n="pres"/><s n="p1"/></r></p></e>
  <e><p><l/><r><s n="vblex"/><s n="pres"/><s n="p2"/></r></p></e>
  <e><p><l>s</l><r><s n="vblex"/><s n="pres"/><s n="p3"/></r></p></e>
  <e><p><l>d</l><r><s n="vblex"/><s n="past"/><s n="p1"/></r></p></e>
  <e><p><l>d</l><r><s n="vblex"/><s n="past"/><s n="p2"/></r></p></e>
  <e><p><l>d</l><r><s n="vblex"/><s n="past"/><s n="p3"/></r></p></e>
</pardef>
<pardef n="die_vblex">
  <e><p><l/><r><s n="vblex"/><s n="pres"/><s n="p1"/></r></p></e>
  <e><p><l/><r><s n="vblex"/><s n="pres"/><s n="p2"/></r></p></e>
  <e><p><l>s</l><r><s n="vblex"/><s n="pres"/><s n="p3"/></r></p></e>
  <e><p><l>d</l><r><s n="vblex"/><s n="past"/><s n="p1"/></r></p></e>
  <e><p><l>d</l><r><s n="vblex"/><s n="past"/><s n="p2"/></r></p></e>
  <e><p><l>d</l><r><s n="vblex"/><s n="past"/><s n="p3"/></r></p></e>
</pardef>
<pardef n="sp/eak_vblex">
  <e><p><l>eak</l><r><s n="vblex"/><s n="pres"/><s n="p1"/></r></p></e>
  <e><p><l>eak</l><r><s n="vblex"/><s n="pres"/><s n="p2"/></r></p></e>
  <e><p><l>eak</l><r><s n="vblex"/><s n="pres"/><s n="p3"/></r></p></e>
  <e><p><l>oke</l><r>ome<s n="vblex"/><s n="past"/><s n="p1"/></r></p></e>
  <e><p><l>oke</l><r>ome<s n="vblex"/><s n="past"/><s n="p2"/></r></p></e>
  <e><p><l>oke</l><r>ome<s n="vblex"/><s n="past"/><s n="p3"/></r></p></e>
</pardef>

<pardef n="read_vblex">
  <e><p><l></l><r><s n="vblex"/><s n="pres"/><s n="p1"/></r></p></e>
  <e><p><l></l><r><s n="vblex"/><s n="pres"/><s n="p2"/></r></p></e>
  <e><p><l>s</l><r><s n="vblex"/><s n="pres"/><s n="p3"/></r></p></e>
  <e><p><l></l><r><s n="vblex"/><s n="past"/><s n="p1"/></r></p></e>
  <e><p><l></l><r><s n="vblex"/><s n="past"/><s n="p2"/></r></p></e>
  <e><p><l></l><r><s n="vblex"/><s n="past"/><s n="p3"/></r></p></e>
</pardef>

<pardef n="new_adj">
  <e><p><l></l><r><s n="adj"/></r></p></e>
</pardef>
```

```
  <pardef n="that_adj">
    <e><p><l></l><r><s n="dem_adj"/></r></p></e>
  </pardef>


</pardefs>

<section id="main" type="standard">
  <e lm="personal subject pronouns"><i/><par n="prsubj_prn"/></e>
  <e lm="determiner"><i/><par n="det_n"/></e>
  <e lm="possesive"><i/><par n="poss_prn"/></e>

  <e lm="I"><i>I</i><par n="prsubj_prn"/></e>
  <e lm="my"><i>my</i><par n="poss_prn"/></e>
  <e lm="your"><i>your</i><par n="poss_prn"/></e>
  <e lm="to"><i>to</i><par n="dir"/></e>
  <e lm="from"><i>from</i><par n="dir"/></e>
  <e lm="too"><i>too</i><par n="too"/></e>

  <e lm="house"><i>house</i><par n="house_n"/></e>
  <e lm="book"><i>book</i><par n="house_n"/></e>
  <e lm="person"><i>pe</i><par n="pe/rson_n"/></e>
  <e lm="work"><i>work</i><par n="work_n"/></e>
  <e lm="day"><i>day</i><par n="house_n"/></e>
  <e lm="child"><i>child</i><par n="child_n"/></e>
  <e lm="car"><i>car</i><par n="house_n"/></e>
  <e lm="thing"><i>thing</i><par n="house_n"/></e>
  <e lm="problem"><i>problem</i><par n="house_n"/></e>

  <e lm="see"><i>s</i><par n="s/ee_vblex"/></e>
  <e lm="walk"><i>walk</i><par n="walk_vblex"/></e>
  <e lm="play"><i>play</i><par n="walk_vblex"/></e>
  <e lm="hurry"><i>hurr</i><par n="hurr/y_vblex"/></e>
  <e lm="live"><i>live</i><par n="walk_vblex"/></e>
  <e lm="die"><i>die</i><par n="die_vblex"/></e>
  <e lm="wait"><i>wait</i><par n="walk_vblex"/></e>
  <e lm="meet"><i>me</i><par n="me/et_vblex"/></e>
  <e lm="do"><i>d</i><par n="d/o_vblex"/></e>
  <e lm="be"><i/><par n="be_copula"/></e>
  <e lm="is"><i/><par n="is_copula"/></e>
  <e lm="come"><i>c</i><par n="c/ome_vblex"/></e>
  <e lm="study"><i>stud</i><par n="hurr/y_vblex"/></e>
  <e lm="become"><i>bec</i><par n="c/ome_vblex"/></e>
  <e lm="go"><i/><par n="go_vblex"/></e>
  <e lm="say"><i>sa</i><par n="sa/y_vblex"/></e>
  <e lm="can"><i/><par n="can_vblex"/></e>
  <e lm="think"><i/><par n="think_vblex"/></e>
  <e lm="know"><i>kn</i><par n="kn/ow_vblex"/></e>
  <e lm="have"><i>ha</i><par n="ha/ve_vblex"/></e>
  <e lm="hear"><i>hear</i><par n="hear_vblex"/></e>
  <e lm="speak"><i>sp</i><par n="sp/eak_vblex"/></e>
  <e lm="read"><i>read</i><par n="read_vblex"/></e>
  <e lm="return"><i>return</i><par n="walk_vblex"/></e>

  <e lm="new"><i>new</i><par n="new_adj"/></e>
  <e lm="good"><i>good</i><par n="new_adj"/></e>
  <e lm="bad"><i>bad</i><par n="new_adj"/></e>
  <e lm="tall"><i>tall</i><par n="new_adj"/></e>
  <e lm="early"><i>early</i><par n="new_adj"/></e>
  <e lm="long"><i>long</i><par n="new_adj"/></e>
  <e lm="that"><i>that</i><par n="that_adj"/></e>

</section>
```

```
</dictionary>
```

apertium-jp.jp.dix

```
<?xml version="1.0" encoding="UTF-8"?>
<dictionary>
<alphabet></alphabet>

<sdefs>
  <sdef n="n"       c="noun"/>
  <sdef n="sg"             c="singular"/>
  <sdef n="pl"             c="plural"/>
  <sdef n="prn"            c="pronoun"/>
  <sdef n="subj"    c="subject"/>
  <sdef n="obj"            c="object"/>
  <sdef n="p1"             c="first person"/>
  <sdef n="p2"             c="second person"/>
  <sdef n="p3"             c="third person"/>
  <sdef n="m"         c="masculine"/>
  <sdef n="f"         c="feminine"/>
  <sdef n="mf"    c="masculine/feminine"/>
  <sdef n="adv"            c="adverb"/>
  <sdef n="nn"             c="inanimate"/>
  <sdef n="an"             c="animate/inanimate"/>
  <sdef n="aa"             c="animate"/>
  <sdef n="det"            c="determiner"/>
  <sdef n="def"            c="definite article"/>
  <sdef n="ind"            c="indefinite article"/>
  <sdef n="vblex"   c="verb"/>
  <sdef n="iru"            c="verb 'to be'"/>
  <sdef n="pres"    c="present tense"/>
  <sdef n="past"    c="past tense"/>
  <sdef n="pot"            c="potential tense"/>
  <sdef n="neg"   c="negative"/>
  <sdef n="prt"            c="particle"/>
  <sdef n="n_suff"  c="noun suffix"/>
  <sdef n="prn_suff"  c="pronoun suffix"/>
  <sdef n="poss"    c="possesive"/>
  <sdef n="dir"            c="direction/movement"/>
  <sdef n="adj"   c="adjective"/>
  <sdef n="copula"   c="copula"/>
  <sdef n="dem_adj"   c="demonstrative adjective"/>
  <sdef n="too"       c="too"/>
</sdefs>

<pardefs>
  <pardef n="家_n">
    <e><p><l/><r><s n="n"/><s n="sg"/></r></p></e>
    <e><p><l>達</l><r><s n="n"/><s n="pl"/></r></p></e>
  </pardef>
  <pardef n="を_prt">
    <e><p><l/><r><s n="prt"/><s n="obj"/></r></p></e>
  </pardef>
  <pardef n="に_prt">
    <e><p><l/><r><s n="dir"/></r></p></e>
  </pardef>
  <pardef n="の_prt">
    <e><p><l/><r><s n="poss"/></r></p></e>
  </pardef>
  <pardef n="は_prt">
    <e><p><l/><r><s n="prt"/><s n="subj"/></r></p></e>
  </pardef>
  <pardef n="が_prt">
    <e><p><l/><r><s n="prt"/><s n="subj"/></r></p></e>
```

```
  </pardef>
  <pardef n="も_prt">
    <e><p><l/><r><s n="too"/></r></p></e>
  </pardef>

  <pardef n="私_prn">
    <e><p><l/><r><s n="prn"/><s n="p1"/><s n="sg"/><s n="mf"/></r></p></e>
    <e><p><l>達</l><r><s n="prn"/><s n="p1"/><s n="pl"/><s n="mf"/></r></p></e>
  </pardef>
  <pardef n="彼_prn">
    <e><p><l/><r><s n="prn"/><s n="p3"/><s n="sg"/><s n="m"/></r></p></e>
    <e><p><l>ら</l><r><s n="prn"/><s n="p3"/><s n="pl"/><s n="mf"/></r></p></e>
  </pardef>
  <pardef n="彼女_prn">
    <e><p><l/><r><s n="prn"/><s n="p3"/><s n="sg"/><s n="f"/></r></p></e>
  </pardef>
  <pardef n="あなた_prn">
    <e><p><l/><r><s n="prn"/><s n="p2"/><s n="sg"/><s n="mf"/></r></p></e>
    <e><p><l>達</l><r><s n="prn"/><s n="p2"/><s n="pl"/><s n="mf"/></r></p></e>
  </pardef>
  <pardef n="それ_prn">
    <e><p><l/><r><s n="prn"/><s n="p3"/><s n="sg"/><s n="mf"/></r></p></e>
  </pardef>

<!--Verb Pardefs-->

  <pardef n="見/る_vblex">
    <e><p><l>ます</l><r>る<s n="vblex"/><s n="pres"/></r></p></e>
    <e><p><l>ました</l><r>る<s n="vblex"/><s n="past"/></r></p></e>
  </pardef>
  <pardef n="歩/く_vblex">
    <e><p><l>きます</l><r>く<s n="vblex"/><s n="pres"/></r></p></e>
    <e><p><l>きました</l><r>く<s n="vblex"/><s n="past"/></r></p></e>
  </pardef>
  <pardef n="急/ぐ_vblex">
    <e><p><l>ぎます</l><r>ぐ<s n="vblex"/><s n="pres"/></r></p></e>
    <e><p><l>ぎました</l><r>ぐ<s n="vblex"/><s n="past"/></r></p></e>
  </pardef>
  <pardef n="遊/ぶ_vblex">
    <e><p><l>びます</l><r>ぶ<s n="vblex"/><s n="pres"/></r></p></e>
    <e><p><l>びました</l><r>ぶ<s n="vblex"/><s n="past"/></r></p></e>
  </pardef>
  <pardef n="住/む_vblex">
    <e><p><l>みます</l><r>む<s n="vblex"/><s n="pres"/></r></p></e>
    <e><p><l>みました</l><r>む<s n="vblex"/><s n="past"/></r></p></e>
  </pardef>
  <pardef n="死/ぬ_vblex">
    <e><p><l>にます</l><r>ぬ<s n="vblex"/><s n="pres"/></r></p></e>
    <e><p><l>にました</l><r>ぬ<s n="vblex"/><s n="past"/></r></p></e>
  </pardef>
  <pardef n="帰/る_vblex">
    <e><p><l>ります</l><r>る<s n="vblex"/><s n="pres"/></r></p></e>
    <e><p><l>りました</l><r>る<s n="vblex"/><s n="past"/></r></p></e>
  </pardef>
  <pardef n="話/す_vblex">
    <e><p><l>します</l><r>す<s n="vblex"/><s n="pres"/></r></p></e>
    <e><p><l>しました</l><r>す<s n="vblex"/><s n="past"/></r></p></e>
```

```
  </pardef>
  <pardef n="待/つ_vblex">
    <e><p><l>ちます</l><r>つ<s n="vblex"/><s n="pres"/></r></p></e>
    <e><p><l>ちました</l><r>つ<s n="vblex"/><s n="past"/></r></p></e>
  </pardef>
  <pardef n="会/う_vblex">
    <e><p><l>います</l><r>う<s n="vblex"/><s n="pres"/></r></p></e>
    <e><p><l>いました</l><r>う<s n="vblex"/><s n="past"/></r></p></e>
  </pardef>
  <pardef n="する_vblex">
    <e><p><l>します</l><r>する<s n="vblex"/><s n="pres"/></r></p></e>
    <e><p><l>しました</l><r>する<s n="vblex"/><s n="past"/></r></p></e>
  </pardef>
  <pardef n="ある_copula">
    <e><p><l>あります</l><r>ある<s n="copula"/><s n="pres"/></r></p></e>
    <e><p><l>ありました</l><r>ある<s n="copula"/><s n="past"/></r></p></e>
  </pardef>
  <pardef n="来る_vblex">
    <e><p><l>来ます</l><r>来る<s n="vblex"/><s n="pres"/></r></p></e>
    <e><p><l>来ました</l><r>来る<s n="vblex"/><s n="past"/></r></p></e>
  </pardef>
  <pardef n="です_copula">
    <e><p><l>です</l><r>です<s n="copula"/><s n="pres"/></r></p></e>
    <e><p><l>でした</l><r>です<s n="copula"/><s n="past"/></r></p></e>
  </pardef>
  <pardef n="勉強/する_vblex">
    <e><p><l>します</l><r>する<s n="vblex"/><s n="pres"/></r></p></e>
    <e><p><l>しました</l><r>する<s n="vblex"/><s n="past"/></r></p></e>
  </pardef>

  <pardef n="新し/い_adj">
    <e><p><l>い</l><r>い<s n="adj"/><s n="pres"/></r></p></e>
    <e><p><l>かった</l><r>い<s n="adj"/><s n="past"/></r></p></e>
  </pardef>

  <pardef n="その_adj">
    <e><p><l></l><r><s n="dem_adj"/></r></p></e>
  </pardef>

</pardefs>

<section id="main" type="standard">

  <e lm="私"><i>私</i><par n="私_prn"/></e>
  <e lm="彼"><i>彼</i><par n="彼_prn"/></e>
  <e lm="彼女"><i>彼女</i><par n="彼女_prn"/></e>
  <e lm="あなた"><i>あなた</i><par n="あなた_prn"/></e>
  <e lm="君"><i>君</i><par n="あなた_prn"/></e>
  <e lm="それ"><i>それ</i><par n="それ_prn"/></e>

  <!--Nouns-->

  <e lm="家"><i>家</i><par n="家_n"/></e>
  <e lm="本"><i>本</i><par n="家_n"/></e>
  <e lm="人"><i>人</i><par n="家_n"/></e>
  <e lm="仕事"><i>仕事</i><par n="家_n"/></e>
```

```
  <e lm="日"><i>日</i><par n="家_n"/></e>
  <e lm="子供"><i>子供</i><par n="家_n"/></e>
  <e lm="車"><i>車</i><par n="家_n"/></e>
  <e lm="事"><i>事</i><par n="家_n"/></e>
  <e lm="問題"><i>問題</i><par n="家_n"/></e>

  <e lm="は"><i>は</i><par n="は_prt"/></e>
  <e lm="が"><i>が</i><par n="が_prt"/></e>
  <e lm="を"><i>を</i><par n="を_prt"/></e>
  <e lm="に"><i>に</i><par n="に_prt"/></e>
  <e lm="の"><i>の</i><par n="の_prt"/></e>
  <e lm="も"><i>も</i><par n="も_prt"/></e>
  <e lm="から"><i>から</i><par n="に_prt"/></e>

<!--Verbs-->

  <e lm="歩く"><i>歩</i><par n="歩/く_vblex"/></e>
  <e lm="帰る"><i>帰</i><par n="帰/る_vblex"/></e>
  <e lm="急ぐ"><i>急</i><par n="急/ぐ_vblex"/></e>
  <e lm="遊ぶ"><i>遊</i><par n="遊/ぶ_vblex"/></e>
  <e lm="住む"><i>住</i><par n="住/む_vblex"/></e>
  <e lm="死ぬ"><i>死</i><par n="死/ぬ_vblex"/></e>
  <e lm="待つ"><i>待</i><par n="待/つ_vblex"/></e>
  <e lm="会う"><i>会</i><par n="会/う_vblex"/></e>
  <e lm="見る"><i>見</i><par n="見/る_vblex"/></e>
  <e lm="する"><i/><par n="する_vblex"/></e>
  <e lm="ある"><i/><par n="ある_copula"/></e>
  <e lm="来る"><i/><par n="来る_vblex"/></e>
  <e lm="です"><i/><par n="です_copula"/></e>
  <e lm="勉強する"><i>勉強</i><par n="勉強/する_vblex"/></e>
  <e lm="なる"><i>な</i><par n="帰/る_vblex"/></e>
  <e lm="行く"><i>行</i><par n="歩/く_vblex"/></e>
  <e lm="言う"><i>言</i><par n="会/う_vblex"/></e>
  <e lm="出来る"><i>出来</i><par n="見/る_vblex"/></e>
  <e lm="思う"><i>思</i><par n="会/う_vblex"/></e>
  <e lm="知る"><i>知</i><par n="帰/る_vblex"/></e>
  <e lm="帰る"><i>帰</i><par n="帰/る_vblex"/></e>
  <e lm="持つ"><i>持</i><par n="待/つ_vblex"/></e>
  <e lm="聞く"><i>聞</i><par n="歩/く_vblex"/></e>
  <e lm="話す"><i>話</i><par n="話/す_vblex"/></e>
  <e lm="読む"><i>読</i><par n="住/む_vblex"/></e>

  <!--Adjectives-->

  <e lm="新しい"><i>新し</i><par n="新し/い_adj"/></e>
  <e lm="良い"><i>良</i><par n="新し/い_adj"/></e>
  <e lm="悪い"><i>悪</i><par n="新し/い_adj"/></e>
  <e lm="高い"><i>高</i><par n="新し/い_adj"/></e>
  <e lm="早い"><i>早</i><par n="新し/い_adj"/></e>
  <e lm="長い"><i>長</i><par n="新し/い_adj"/></e>
  <e lm="その"><i>その</i><par n="その_adj"/></e>
</section>
</dictionary>
```

apertium-jp-en.jp-en.dix

```
<?xml version="1.0" encoding="UTF-8"?>
<dictionary>
<alphabet></alphabet>

<sdefs>
 <sdef n="n"    c="noun"/>
  <sdef n="sg"    c="singular"/>
  <sdef n="pl"    c="plural"/>
  <sdef n="prn"   c="pronoun"/>
  <sdef n="subj"  c="subject"/>
  <sdef n="obj"   c="object"/>
  <sdef n="p1"    c="first person"/>
  <sdef n="p2"    c="second person"/>
  <sdef n="p3"    c="third person"/>
  <sdef n="m"     c="masculine"/>
  <sdef n="f"     c="feminine"/>
  <sdef n="mf"    c="masculine/feminine"/>
  <sdef n="adv"   c="adverb"/>
  <sdef n="nn"    c="inanimate"/>
  <sdef n="an"    c="animate/inanimate"/>
  <sdef n="aa"    c="animate"/>
  <sdef n="det"   c="determiner"/>
  <sdef n="def"   c="definite article"/>
  <sdef n="ind"   c="indefinite article"/>
  <sdef n="vblex" c="verb"/>
  <sdef n="iru"   c="verb 'to be'"/>
  <sdef n="pres"  c="present tense"/>
  <sdef n="past"  c="past tense"/>
  <sdef n="pot"   c="potential tense"/>
  <sdef n="neg"   c="negative"/>
  <sdef n="prt"   c="particle"/>
  <sdef n="n_suff"  c="noun suffix"/>
  <sdef n="poss"  c="possesive"/>
  <sdef n="dir"   c="direction/movement"/>
  <sdef n="adj"   c="adjective"/>
  <sdef n="copula"   c="copula"/>
  <sdef n="dem_adj"   c="demonstrative adjective"/>
  <sdef n="too"   c="too"/>
</sdefs>

<section id="main" type="standard">
  <e><p><l>私<s n="prn"/></l><r>prpers<s n="prn"/></r></p></e>
  <e><p><l>彼<s n="prn"/></l><r>prpers<s n="prn"/></r></p></e>
  <e><p><l>彼女<s n="prn"/></l><r>prpers<s n="prn"/></r></p></e>
  <e><p><l>あなた<s n="prn"/></l><r>prpers<s n="prn"/></r></p></e>
  <e><p><l>君<s n="prn"/></l><r>prpers<s n="prn"/></r></p></e>
  <e><p><l>それ<s n="prn"/></l><r>prpers<s n="prn"/></r></p></e>

  <e><p><l>家<s n="n"/></l><r>house<s n="n"/></r></p></e>
  <e><p><l>本<s n="n"/></l><r>book<s n="n"/></r></p></e>
  <e><p><l>人<s n="n"/></l><r>person<s n="n"/></r></p></e>
  <e><p><l>仕事<s n="n"/></l><r>work<s n="n"/></r></p></e>
  <e><p><l>日<s n="n"/></l><r>day<s n="n"/></r></p></e>
  <e><p><l>子供<s n="n"/></l><r>child<s n="n"/></r></p></e>
  <e><p><l>車<s n="n"/></l><r>car<s n="n"/></r></p></e>
  <e><p><l>事<s n="n"/></l><r>thing<s n="n"/></r></p></e>
  <e><p><l>問題<s n="n"/></l><r>problem<s n="n"/></r></p></e>
```

```
  <e><p><l>を<s n="prt"/></l><r><s n="det"/></r></p></e>
  <e><p><l>に<s n="dir"/></l><r>to<s n="dir"/></r></p></e>
  <e><p><l>から<s n="dir"/></l><r>from<s n="dir"/></r></p></e>
  <e><p><l>は<s n="prt"/></l><r><s n="det"/></r></p></e>
  <e><p><l>の<s n="poss"/></l><r><s n="poss"/></r></p></e>
  <e><p><l>が<s n="prt"/></l><r><s n="det"/></r></p></e>
  <e><p><l>も<s n="too"/></l><r>too<s n="too"/></r></p></e>


  <e><p><l>歩く<s n="vblex"/></l><r>walk<s n="vblex"/></r></p></e>
  <e><p><l>急ぐ<s n="vblex"/></l><r>hurry<s n="vblex"/></r></p></e>
  <e><p><l>遊ぶ<s n="vblex"/></l><r>play<s n="vblex"/></r></p></e>
  <e><p><l>住む<s n="vblex"/></l><r>live<s n="vblex"/></r></p></e>
  <e><p><l>死ぬ<s n="vblex"/></l><r>die<s n="vblex"/></r></p></e>
  <e><p><l>待つ<s n="vblex"/></l><r>wait<s n="vblex"/></r></p></e>
  <e><p><l>会う<s n="vblex"/></l><r>meet<s n="vblex"/></r></p></e>
  <e><p><l>見る<s n="vblex"/></l><r>see<s n="vblex"/></r></p></e>
  <e><p><l>する<s n="vblex"/></l><r>do<s n="vblex"/></r></p></e>
  <e><p><l>ある<s n="copula"/></l><r>be<s n="copula"/></r></p></e>
  <e><p><l>来る<s n="vblex"/></l><r>come<s n="vblex"/></r></p></e>
  <e><p><l>です<s n="copula"/></l><r>is<s n="copula"/></r></p></e>
  <e><p><l>勉強する<s n="vblex"/></l><r>study<s n="vblex"/></r></p></e>
  <e><p><l>なる><s n="vblex"/></l><r>become<s n="vblex"/></r></p></e>
  <e><p><l>言う<s n="vblex"/></l><r>say<s n="vblex"/></r></p></e>
  <e><p><l>出来る><s n="vblex"/></l><r>can<s n="vblex"/></r></p></e>
  <e><p><l>思う><s n="vblex"/></l><r>think<s n="vblex"/></r></p></e>
  <e><p><l>知る><s n="vblex"/></l><r>know<s n="vblex"/></r></p></e>
  <e><p><l>持つ<s n="vblex"/></l><r>have<s n="vblex"/></r></p></e>
  <e><p><l>聞く<s n="vblex"/></l><r>hear<s n="vblex"/></r></p></e>
  <e><p><l>話す><s n="vblex"/></l><r>speak<s n="vblex"/></r></p></e>
  <e><p><l>読む<s n="vblex"/></l><r>read<s n="vblex"/></r></p></e>
  <e><p><l>帰る<s n="vblex"/></l><r>return<s n="vblex"/></r></p></e>
  <e><p><l>行く<s n="vblex"/></l><r>go<s n="vblex"/></r></p></e>

  <e><p><l>新しい<s n="adj"/></l><r>new<s n="adj"/></r></p></e>
  <e><p><l>良い<s n="adj"/></l><r>good<s n="adj"/></r></p></e>
  <e><p><l>悪い<s n="adj"/></l><r>bad<s n="adj"/></r></p></e>
  <e><p><l>高い<s n="adj"/></l><r>tall<s n="adj"/></r></p></e>
  <e><p><l>早い<s n="adj"/></l><r>early<s n="adj"/></r></p></e>
 <e><p><l>長い<s n="adj"/></l><r>long<s n="adj"/></r></p></e>
 <e><p><l>その<s n="dem_adj"/></l><r>that<s n="dem_adj"/></r></p></e>


</section>

</dictionary>
```

apertium-jp-en.jp-en.t1x

```
<?xml version="1.0" encoding="UTF-8"?>
<transfer>

<section-def-cats>
  <def-cat n="c_nom">
    <cat-item tags="n.*"/>
  </def-cat>
  <def-cat n="c_vrb">
    <cat-item tags="vblex.*"/>
  </def-cat>
  <def-cat n="c_prpers">
    <cat-item lemma="prpers" tags="prn.*"/>
  </def-cat>
  <def-cat n="c_nom_prt_vrb">
    <cat-item name="c_nom_prt_vrb"/>
  </def-cat>
  <def-cat n="c_nom_dir_vrb">
    <cat-item name="c_nom_dir_vrb"/>
  </def-cat>
  <def-cat n="c_prt">
    <cat-item tags="prt"/>
    <cat-item tags="prt.*"/>
  </def-cat>
  <def-cat n="c_dir">
    <cat-item tags = "dir"/>
    <cat-item tags="dir.*"/>
  </def-cat>
  <def-cat n="c_det">
    <cat-item tags="det"/>
    <cat-item tags="det.*"/>
  </def-cat>
  <def-cat n="c_poss">
    <cat-item tags="poss"/>
    <cat-item tags="poss.*"/>
  </def-cat>
  <def-cat n="c_prn">
    <cat-item tags="prn"/>
    <cat-item tags="prn.*"/>
  </def-cat>
  <def-cat n="c_adj">
    <cat-item tags="adj"/>
    <cat-item tags="adj.*"/>
  </def-cat>
  <def-cat n="c_copula">
    <cat-item tags="copula"/>
    <cat-item tags="copula.*"/>
  </def-cat>
  <def-cat n="c_dem_adj">
    <cat-item tags="dem_adj"/>
    <cat-item tags="dem_adj.*"/>
  </def-cat>
  <def-cat n="c_too">
    <cat-item tags="too"/>
    <cat-item tags="too.*"/>
  </def-cat>
  <def-cat n="c_want">
    <cat-item tags="want"/>
    <cat-item tags="want.*"/>
  </def-cat>

</section-def-cats>
```

```
<section-def-attrs>
  <def-attr n="a_nbr">
    <attr-item tags="sg"/>
    <attr-item tags="pl"/>
  </def-attr>
  <def-attr n="a_nom">
    <attr-item tags="n"/>
  </def-attr>
  <def-attr n="a_prn">
    <attr-item tags="prn"/>
  </def-attr>
  <def-attr n="a_tense">
    <attr-item tags="pres"/>
    <attr-item tags="past"/>
  </def-attr>
  <def-attr n="a_person">
    <attr-item tags="p1"/>
    <attr-item tags="p2"/>
    <attr-item tags="p3"/>
  </def-attr>
  <def-attr n="a_verb">
    <attr-item tags="vblex"/>
  </def-attr>
  <def-attr n="a_types_nom">
    <attr-item tags="subj"/>
    <attr-item tags="obj"/>
  </def-attr>
  <def-attr n="a_negative">
    <attr-item tags="neg"/>
  </def-attr>
  <def-attr n="a_prt">
    <attr-item tags="prt"/>
    <attr-item tags="を_prt"/>
    <attr-item tags="は_prt"/>
    <attr-item tags="が_prt"/>
    <attr-item tags="に_prt"/>
  </def-attr>
  <def-attr n="a_det">
    <attr-item tags="det"/>
  </def-attr>
  <def-attr n="a_poss">
    <attr-item tags="poss"/>
  </def-attr>
  <def-attr n="a_gender">
    <attr-item tags="m"/>
    <attr-item tags="f"/>
    <attr-item tags="mf"/>
  </def-attr>
  <def-attr n="a_adj">
    <attr-item tags="adj"/>
  </def-attr>
  <def-attr n="a_dir">
    <attr-item tags="dir"/>
  </def-attr>
  <def-attr n="a_copula">
    <attr-item tags="copula"/>
  </def-attr>
  <def-attr n="a_dem_adj">
    <attr-item tags="dem_adj"/>
  </def-attr>
  <def-attr n="a_too">
    <attr-item tags="too"/>
```

```
  </def-attr>
  <def-attr n="a_want">
    <attr-item tags="want"/>
  </def-attr>

</section-def-attrs>

<section-def-vars>
  <def-var n="number"/>
</section-def-vars>

<section-def-macros>

</section-def-macros>

<section-rules>

<!--Verb Phrases-->

<rule c="verb">
  <pattern>
    <pattern-item n="c_vrb"/>
  </pattern>
  <action>
    <out>
      <lu>
        <clip pos="1" side="tl" part="lem"/>
        <clip pos="1" side="tl" part="a_verb"/>
        <clip pos="1" side="tl" part="a_tense"/>
        <lit-tag v="p1"/>
       </lu>
    </out>
  </action>
</rule>

<rule c="noun-part-verb">
  <pattern>
    <pattern-item n="c_nom"/>
    <pattern-item n="c_prt"/>
    <pattern-item n="c_vrb"/>
  </pattern>
  <action>
    <out>
      <lu>
        <clip pos="3" side="tl" part="lem"/>
        <clip pos="3" side="tl" part="a_verb"/>
        <clip pos="3" side="tl" part="a_tense"/>
        <lit-tag v="p1"/>
      </lu>
      <b/>
      <lu>
        <clip pos="2" side="tl" part="a_det"/>
        <clip pos="2" side="tl" part="a_types_nom"/>
        <clip pos="1" side="tl" part="a_nbr"/>
      </lu>
      <b/>
      <lu>
        <clip pos="1" side="tl" part="lem"/>
        <clip pos="1" side="tl" part="a_nom"/>
        <clip pos="1" side="tl" part="a_nbr"/>
      </lu>
    </out>
  </action>
</rule>
```

```
<rule c="prn-part-verb">
  <pattern>
    <pattern-item n="c_prn"/>
    <pattern-item n="c_prt"/>
    <pattern-item n="c_vrb"/>
  </pattern>
  <action>
    <out>
      <lu>
        <clip pos="1" side="tl" part="lem"/>
        <clip pos="1" side="tl" part="a_prn"/>
        <clip pos="1" side="tl" part="a_person"/>
        <clip pos="1" side="tl" part="a_nbr"/>
        <clip pos="1" side="tl" part="a_gender"/>
      </lu>
      <b/>
      <lu>
        <clip pos="3" side="tl" part="lem"/>
        <clip pos="3" side="tl" part="a_verb"/>
        <clip pos="3" side="tl" part="a_tense"/>
        <clip pos="1" side="tl" part="a_person"/>
      </lu>
    </out>
  </action>
</rule>

<rule c="noun-dir-verb">
  <pattern>
    <pattern-item n="c_nom"/>
    <pattern-item n="c_dir"/>
    <pattern-item n="c_vrb"/>
  </pattern>
  <action>
    <out>
      <lu>
        <clip pos="3" side="tl" part="lem"/>
        <clip pos="3" side="tl" part="a_verb"/>
        <clip pos="3" side="tl" part="a_tense"/>
        <lit-tag v="p1"/>
      </lu>
      <b/>
      <lu>
        <clip pos="2" side="tl" part="lem"/>
        <clip pos="2" side="tl" part="a_dir"/>
      </lu>
      <b/>
      <lu>
        <clip pos="1" side="tl" part="lem"/>
        <clip pos="1" side="tl" part="a_nom"/>
        <clip pos="1" side="tl" part="a_nbr"/>
      </lu>
    </out>
  </action>
</rule>

<rule c="noun-prt-copula">
  <pattern>
    <pattern-item n="c_nom"/>
    <pattern-item n="c_prt"/>
    <pattern-item n="c_copula"/>
  </pattern>
  <action>
    <out>
```

```
      <lu>
        <lit-tag v="det"/>
        <clip pos="2" side="tl" part="a_types_nom"/>
        <lit-tag v="sgpl"/>
        <clip pos="3" side="tl" part="a_copula"/>
      </lu>
      <b/>
      <lu>
        <clip pos="3" side="tl" part="lem"/>
        <clip pos="3" side="tl" part="a_copula"/>
        <clip pos="3" side="tl" part="a_tense"/>
        <lit-tag v="p3"/>
      </lu>
      <b/>
      <lu>
        <clip pos="2" side="tl" part="a_det"/>
        <lit-tag v="obj"/>
        <clip pos="1" side="tl" part="a_nbr"/>
      </lu>
      <b/>
      <lu>
        <clip pos="1" side="tl" part="lem"/>
        <clip pos="1" side="tl" part="a_nom"/>
        <clip pos="1" side="tl" part="a_nbr"/>
      </lu>
    </out>
  </action>
</rule>

<rule c="prn-prt-adj-copula">
  <pattern>
    <pattern-item n="c_prn"/>
    <pattern-item n="c_prt"/>
    <pattern-item n="c_adj"/>
    <pattern-item n="c_copula"/>
  </pattern>
  <action>
    <out>
      <lu>
        <clip pos="1" side="tl" part="lem"/>
        <clip pos="1" side="tl" part="a_prn"/>
        <clip pos="1" side="tl" part="a_person"/>
        <clip pos="1" side="tl" part="a_nbr"/>
        <clip pos="1" side="tl" part="a_gender"/>
      </lu>
      <b/>
      <lu>
        <clip pos="4" side="tl" part="lem"/>
        <clip pos="4" side="tl" part="a_copula"/>
        <clip pos="3" side="tl" part="a_tense"/>
        <lit-tag v="p3"/>
      </lu>
      <b/>
      <lu>
        <clip pos="3" side="tl" part="lem"/>
        <clip pos="3" side="tl" part="a_adj"/>
      </lu>
    </out>
  </action>
</rule>

<rule c="noun-prt-adj-copula">
  <pattern>
    <pattern-item n="c_nom"/>
```

```
    <pattern-item n="c_prt"/>
    <pattern-item n="c_adj"/>
    <pattern-item n="c_copula"/>
  </pattern>
  <action>
    <out>
      <lu>
        <clip pos="1" side="tl" part="lem"/>
        <clip pos="1" side="tl" part="a_nom"/>
        <clip pos="1" side="tl" part="a_nbr"/>
      </lu>
      <b/>
      <lu>
        <clip pos="4" side="tl" part="lem"/>
        <clip pos="4" side="tl" part="a_copula"/>
        <clip pos="4" side="tl" part="a_tense"/>
        <lit-tag v="p3"/>
      </lu>
      <b/>
      <lu>
        <clip pos="3" side="tl" part="lem"/>
        <clip pos="3" side="tl" part="a_adj"/>
      </lu>
    </out>
  </action>
</rule>

<rule c="adj-copula">
  <pattern>
    <pattern-item n="c_adj"/>
    <pattern-item n="c_copula"/>
  </pattern>
  <action>
    <out>
      <lu>
        <clip pos="2" side="tl" part="lem"/>
        <clip pos="2" side="tl" part="a_copula"/>
        <clip pos="2" side="tl" part="a_tense"/>
        <lit-tag v="p3"/>
      </lu>
      <b/>
      <lu>
        <clip pos="1" side="tl" part="lem"/>
        <clip pos="1" side="tl" part="a_adj"/>
      </lu>
    </out>
  </action>
</rule>

<rule c="pronoun-part-copula">
  <pattern>
    <pattern-item n="c_prn"/>
    <pattern-item n="c_prt"/>
    <pattern-item n="c_copula"/>
  </pattern>
  <action>
    <out>
      <lu>
        <clip pos="1" side="tl" part="lem"/>
        <clip pos="1" side="tl" part="a_prn"/>
        <clip pos="1" side="tl" part="a_person"/>
        <clip pos="1" side="tl" part="a_nbr"/>
        <clip pos="1" side="tl" part="a_gender"/>
     </lu>
```

```
      <b/>
      <lu>
        <clip pos="3" side="tl" part="lem"/>
        <clip pos="3" side="tl" part="a_verb"/>
        <clip pos="3" side="tl" part="a_tense"/>
        <clip pos="3" side="tl" part="a_nbr"/>
      </lu>
    </out>
  </action>
</rule>

<!--Noun Phrases-->

<rule c="pronoun">
  <pattern>
    <pattern-item n="c_prn"/>
  </pattern>
  <action>
    <out>
      <lu>
        <clip pos="1" side="tl" part="lem"/>
        <clip pos="1" side="tl" part="a_prn"/>
        <clip pos="1" side="tl" part="a_person"/>
        <clip pos="1" side="tl" part="a_nbr"/>
        <clip pos="1" side="tl" part="a_gender"/>
      </lu>
    </out>
  </action>
</rule>

<rule c="noun">
  <pattern>
    <pattern-item n="c_nom"/>
  </pattern>
  <action>
    <out>
      <lu>
        <clip pos="1" side="tl" part="lem"/>
        <clip pos="1" side="tl" part="a_nom"/>
        <clip pos="1" side="tl" part="a_nbr"/>
      </lu>
    </out>
  </action>
</rule>

<rule c="prn-poss-noun">
  <pattern>
    <pattern-item n="c_prn"/>
    <pattern-item n="c_poss"/>
    <pattern-item n="c_nom"/>
  </pattern>
    <action>
      <out>
        <lu>
          <lit-tag v="poss"/>
          <clip pos="1" side="tl" part="a_person"/>
          <clip pos="1" side="tl" part="a_nbr"/>
          <clip pos="1" side="tl" part="a_gender"/>
        </lu>
        <b/>
        <lu>
          <clip pos="3" side="tl" part="lem"/>
          <clip pos="3" side="tl" part="a_nom"/>
          <clip pos="3" side="tl" part="a_nbr"/>
```

```
        </lu>
      </out>
    </action>
</rule>

<rule c="noun-dir">
  <pattern>
    <pattern-item n="c_nom"/>
    <pattern-item n="c_dir"/>
  </pattern>
    <action>
      <out>
        <lu>
          <clip pos="2" side="tl" part="lem"/>
          <clip pos="2" side="tl" part="a_dir"/>
          <clip pos="2" side="tl" part="a_types_nom"/>
        </lu>
        <b/>
        <lu>
          <lit-tag v="det"/>
          <lit-tag v="subj"/>
          <clip pos="1" side="tl" part="a_nbr"/>
        </lu>
        <b/>
        <lu>
          <clip pos="1" side="tl" part="lem"/>
          <clip pos="1" side="tl" part="a_nom"/>
          <clip pos="1" side="tl" part="a_nbr"/>
        </lu>
      </out>
    </action>
</rule>

<rule c="adjective-noun">
  <pattern>
    <pattern-item n="c_adj"/>
    <pattern-item n="c_nom"/>
  </pattern>
  <action>
    <out>
      <lu>
        <lit-tag v="det"/>
        <lit-tag v="obj"/>
        <clip pos="2" side="tl" part="a_nbr"/>
      </lu>
      <b/>
      <lu>
        <clip pos="1" side="tl" part="lem"/>
        <clip pos="1" side="tl" part="adj"/>
      </lu>
      <b/>
      <lu>
        <clip pos="2" side="tl" part="lem"/>
        <clip pos="2" side="tl" part="a_nom"/>
        <clip pos="2" side="tl" part="a_nbr"/>
      </lu>
    </out>
  </action>
</rule>


<!--Other Phrases-->

<rule c="adjective">
```

```
  <pattern>
    <pattern-item n="c_adj"/>
  </pattern>
  <action>
    <out>
      <lu>
        <clip pos="1" side="tl" part="lem"/>
        <clip pos="1" side="tl" part="a_adj"/>
      </lu>
    </out>
  </action>
</rule>

<rule c="demonstrative">
  <pattern>
    <pattern-item n="c_dem_adj"/>
  </pattern>
  <action>
    <out>
      <lu>
        <clip pos="1" side="tl" part="lem"/>
        <clip pos="1" side="tl" part="a_dem_adj"/>
      </lu>
    </out>
  </action>
</rule>

<rule c="too">
  <pattern>
    <pattern-item n="c_too"/>
  </pattern>
  <action>
    <out>
      <lu>
        <clip pos="1" side="tl" part="lem"/>
        <clip pos="1" side="tl" part="a_prt"/>
        <clip pos="1" side="tl" part="a_too"/>
      </lu>
    </out>
  </action>
</rule>

<!--Sentence Rules-Testing-->

<rule c="prn-part-noun-part-verb">
  <pattern>
    <pattern-item n="c_prn"/>
    <pattern-item n="c_prt"/>
    <pattern-item n="c_nom"/>
    <pattern-item n="c_prt"/>
    <pattern-item n="c_vrb"/>
  </pattern>
  <action>
    <out>
      <lu>
        <clip pos="1" side="tl" part="lem"/>
        <clip pos="1" side="tl" part="a_prn"/>
        <clip pos="1" side="tl" part="a_person"/>
        <clip pos="1" side="tl" part="a_nbr"/>
        <clip pos="1" side="tl" part="a_gender"/>
      </lu>
      <b/>
      <lu>
        <clip pos="5" side="tl" part="lem"/>
```

```
        <clip pos="5" side="tl" part="a_verb"/>
        <clip pos="5" side="tl" part="a_tense"/>
        <clip pos="1" side="tl" part="a_person"/>
      </lu>
      <b/>
      <lu>
        <clip pos="4" side="tl" part="a_det"/>
        <clip pos="4" side="tl" part="a_types_nom"/>
        <clip pos="3" side="tl" part="a_nbr"/>
      </lu>
      <b/>
      <lu>
        <clip pos="3" side="tl" part="lem"/>
        <clip pos="3" side="tl" part="a_nom"/>
        <clip pos="3" side="tl" part="a_nbr"/>
      </lu>
    </out>
  </action>
</rule>

<rule c="prn-part-adjective-noun-part-verb">
  <pattern>
    <pattern-item n="c_prn"/>
    <pattern-item n="c_prt"/>
    <pattern-item n="c_adj"/>
    <pattern-item n="c_nom"/>
    <pattern-item n="c_prt"/>
    <pattern-item n="c_vrb"/>
  </pattern>
  <action>
    <out>
      <lu>
        <clip pos="1" side="tl" part="lem"/>
        <clip pos="1" side="tl" part="a_prn"/>
        <clip pos="1" side="tl" part="a_person"/>
        <clip pos="1" side="tl" part="a_nbr"/>
        <clip pos="1" side="tl" part="a_gender"/>
      </lu>
      <b/>
      <lu>
        <clip pos="6" side="tl" part="lem"/>
        <clip pos="6" side="tl" part="a_verb"/>
        <clip pos="6" side="tl" part="a_tense"/>
        <clip pos="1" side="tl" part="a_person"/>
      </lu>
      <b/>
      <lu>
        <clip pos="5" side="tl" part="a_det"/>
        <clip pos="5" side="tl" part="a_types_nom"/>
        <clip pos="4" side="tl" part="a_nbr"/>
      </lu>
      <b/>
      <lu>
        <clip pos="3" side="tl" part="lem"/>
        <clip pos="3" side="tl" part="a_adj"/>
      </lu>
      <b/>
      <lu>
        <clip pos="4" side="tl" part="lem"/>
        <clip pos="4" side="tl" part="a_nom"/>
        <clip pos="4" side="tl" part="a_nbr"/>
      </lu>
    </out>
  </action>
```

```
</rule>

<rule c="prn-part-noun-dir-verb">
  <pattern>
    <pattern-item n="c_prn"/>
    <pattern-item n="c_prt"/>
    <pattern-item n="c_nom"/>
    <pattern-item n="c_dir"/>
    <pattern-item n="c_vrb"/>
  </pattern>
  <action>
    <out>
      <lu>
        <clip pos="1" side="tl" part="lem"/>
        <clip pos="1" side="tl" part="a_prn"/>
        <clip pos="1" side="tl" part="a_person"/>
        <clip pos="1" side="tl" part="a_nbr"/>
        <clip pos="1" side="tl" part="a_gender"/>
      </lu>
      <b/>
      <lu>
        <clip pos="5" side="tl" part="lem"/>
        <clip pos="5" side="tl" part="a_verb"/>
        <clip pos="5" side="tl" part="a_tense"/>
        <clip pos="1" side="tl" part="a_person"/>
      </lu>
      <b/>
      <lu>
        <clip pos="4" side="tl" part="lem"/>
        <clip pos="4" side="tl" part="a_dir"/>
        <clip pos="4" side="tl" part="a_types_nom"/>
      </lu>
      <b/>
      <lu>
        <clip pos="3" side="tl" part="lem"/>
        <clip pos="3" side="tl" part="a_nom"/>
        <clip pos="3" side="tl" part="a_nbr"/>
      </lu>
    </out>
  </action>
</rule>

<rule c="prn-prt-adj-noun-copula">
  <pattern>
    <pattern-item n="c_prn"/>
    <pattern-item n="c_prt"/>
    <pattern-item n="c_adj"/>
    <pattern-item n="c_nom"/>
    <pattern-item n="c_copula"/>
  </pattern>
  <action>
    <out>
      <lu>
        <clip pos="1" side="tl" part="lem"/>
        <clip pos="1" side="tl" part="a_prn"/>
        <clip pos="1" side="tl" part="a_person"/>
        <clip pos="1" side="tl" part="a_nbr"/>
        <clip pos="1" side="tl" part="a_gender"/>
      </lu>
      <b/>
      <lu>
        <clip pos="5" side="tl" part="lem"/>
        <clip pos="5" side="tl" part="a_copula"/>
        <clip pos="5" side="tl" part="a_tense"/>
```

```
        <clip pos="1" side="tl" part="a_person"/>
      </lu>
      <b/>
      <lu>
        <lit-tag v="det"/>
        <clip pos="2" side="tl" part="a_types_nom"/>
        <clip pos="1" side="tl" part="a_nbr"/>
      </lu>
      <b/>
      <lu>
        <clip pos="3" side="tl" part="lem"/>
        <clip pos="3" side="tl" part="a_adj"/>
      </lu>
      <b/>
      <lu>
        <clip pos="4" side="tl" part="lem"/>
        <clip pos="4" side="tl" part="a_nom"/>
        <clip pos="1" side="tl" part="a_nbr"/>
      </lu>
    </out>
  </action>
</rule>

<rule c="prn-prt-dem-noun-prt-verb">
  <pattern>
    <pattern-item n="c_prn"/>
    <pattern-item n="c_prt"/>
    <pattern-item n="c_dem_adj"/>
    <pattern-item n="c_nom"/>
    <pattern-item n="c_prt"/>
    <pattern-item n="c_vrb"/>
  </pattern>
  <action>
    <out>
      <lu>
        <clip pos="1" side="tl" part="lem"/>
        <clip pos="1" side="tl" part="a_prn"/>
        <clip pos="1" side="tl" part="a_person"/>
        <clip pos="1" side="tl" part="a_nbr"/>
        <clip pos="1" side="tl" part="a_gender"/>
      </lu>
      <b/>
      <lu>
        <clip pos="6" side="tl" part="lem"/>
        <clip pos="6" side="tl" part="a_verb"/>
        <clip pos="6" side="tl" part="a_tense"/>
        <clip pos="1" side="tl" part="a_person"/>
      </lu>
      <b/>
      <lu>
        <clip pos="3" side="tl" part="lem"/>
        <clip pos="3" side="tl" part="a_adj"/>
      </lu>
      <b/>
      <lu>
        <clip pos="4" side="tl" part="lem"/>
        <clip pos="4" side="tl" part="a_nom"/>
        <clip pos="1" side="tl" part="a_nbr"/>
      </lu>
    </out>
  </action>
</rule>

</section-rules>
```

```
</transfer>
```

Appendix B.Test Results

II.1.Rule List

```
Japanese Grammar Rules
======================
1.は                particle, subject
2.の                particle. possessive
3.に                particle, direction
4.を                particle, object
5.が                particle, subject
6.します            する-irregular verb
7.ある              ある-copula
8.です              です-copula
9.私                personal pronoun, first person, singular
10.彼               personal pronoun, third person, singular, masculine
11.その             adjective, singular
12.彼女             personal pronoun, third person, feminine
13.も               particle, also
14.から             particle, from location
15.達               suffix, plural
16.あなた           personal pronoun, second person, singular
17.君               personal pronoun, second person, singular
18.彼ら             personal pronoun, third person, plural
19.それ             pronoun, third person, singular
20.くる             来る-irregular verb
21.ます             suffix, る-verb present tense
22.います           suffix, う-verb, present tense
23.ました           suffix, る-verb past tense
24.いました         suffix, う-verb past tense
25.ーぶ             ぶ-godan verbs
26.ーつ             つ-godan verbs
27.ーく             く-godan verbs
28.ーぐ             ぐ-godan verbs
29.ーむ             む-godan verbs
30.ーぬ             ぬ-godan verbs
31.ーる             る-godan verbs
32.ーう             う-godan verbs
33.ーる             る-ichidan verbs
34.ーする           する-noun verb
35.新しい           い-adjectives

Transfer Rules
==============
36.Verb
37.Noun
38.Pronoun
39.Adjective
40.Demonstrative Particle
41.Also Particle
42.Noun-Directional-Verb
43.Adjective-Copula
44.Pronoun-Particle-Adjective-Copula
45.Noun-Particle-Adjective-Copula
```

46.Noun-Particle-Verb
47.Noun-Particle-Copula
48.Pronoun-Particle-Verb
49.Pronoun-Particle Copula
50.Pronoun-Possessive-Noun
51.Noun-Directional
52.Adjective-Noun

II.2.Raw Test Output

1. Translating: 私 は 家 に 歩きます

Morphological Analyzer: ^私/私<prn><p1><sg><mf>$ ^は/は<prt><subj>$ ^家/家<n><sg>$ ^に/に<dir>$ ^歩きます/歩く<vblex><pres>$

POS Tagger: ^私<prn><p1><sg><mf>$ ^は<prt><subj>$ ^家<n><sg>$ ^に<dir>$ ^歩く<vblex><pres>$
Lexical Transfer: ^prpers<prn><p1><sg><mf>$ ^walk<vblex><pres><p1>$ ^to<dir>$ ^<det><subj><sg>$ ^house<n><sg>$
Output: I walk to the house

2. Translating: あなた の 本

Morphological Analyzer: ^あなた/あなた<prn><p2><sg><mf>$ ^の/の<poss>$ ^本/本<n><sg>$

POS Tagger: ^あなた<prn><p2><sg><mf>$ ^の<poss>$ ^本<n><sg>$
Lexical Transfer: ^<poss><p2><sg><mf>$ ^book<n><sg>$
Output: your book

3. Translating: 彼 は 高い 家 を 見ました

Morphological Analyzer: ^彼/彼<prn><p3><sg><m>$ ^は/は<prt><subj>$ ^高い/高い<adj><pres>$ ^家/家<n><sg>$ ^を/を<prt><obj>$ ^見ました/見る<vblex><past>$

POS Tagger: ^彼<prn><p3><sg><m>$ ^は<prt><subj>$ ^高い<adj><pres>$ ^家<n><sg>$ ^を<prt><obj>$ ^見る<vblex><past>$
Lexical Transfer: ^prpers<prn><p3><sg><m>$ ^see<vblex><past><p3>$ ^<det><obj><sg>$ ^tall<adj>$ ^house<n><sg>$
Output: he saw a tall house

4. Translating: 私 が 子供 です

Morphological Analyzer: ^私/私<prn><p1><sg><mf>$ ^が/が<prt><subj>$ ^子供/子供<n><sg>$ ^です/です<copula><pres>$

POS Tagger: ^私<prn><p1><sg><mf>$ ^が<prt><subj>$ ^子供<n><sg>$ ^です<copula><pres>$
Lexical Transfer: ^prpers<prn><p1><sg><mf>$ ^is<copula><pres><p1>$ ^<det><subj><sg>$ ^child<n><sg>$
Output: I am the child

5. Translating: 私達 は 勉強します

Morphological Analyzer: ^私達/私<prn><p1><pl><mf>$ ^は/は<prt><subj>$ ^勉強します/勉強する<vblex><pres>$

POS Tagger: ^私<prn><p1><pl><mf>$ ^は<prt><subj>$ ^勉強する<vblex><pres>$
Lexical Transfer: ^prpers<prn><p1><pl><mf>$ ^study<vblex><pres>$
Output: we study

6. Translating: 本 が ありますMorphological Analyzer: ^本/本<n><sg>$ ^が/が<prt><subj>$ ^あります/ある<copula><pres>$

POS Tagger: ^本<n><sg>$ ^が<prt><subj>$ ^ある<copula><pres>$
Lexical Transfer: ^<det><subj><sgpl><copula>$ ^be<copula><pres><p3>$ ^<det><obj><sg>$ ^book<n><sg>$
Output: there is a book

7. Translating: その 仕事 は 悪い です

Morphological Analyzer: ^その/その<dem_adj>$ ^仕事/仕事<n><sg>$ ^は/は<prt><subj>$ ^悪い/悪い<adj><pres>$ ^です/です<copula><pres>$

POS Tagger: ^その<dem_adj>$ ^仕事<n><sg>$ ^は<prt><subj>$ ^悪い<adj><pres>$ ^です<copula><pres>$
Lexical Transfer: ^that<dem_adj>$ ^work<n><sg>$ ^is<copula><pres><p3>$ ^bad<adj>$
Output: that work is bad

8. Translating: 私 も 遊びます

Morphological Analyzer: ^私/私<prn><p1><sg><mf>$ ^も/も<too>$ ^遊びます/遊ぶ<vblex><pres>$

POS Tagger: ^私<prn><p1><sg><mf>$ ^も<too>$ ^遊ぶ<vblex><pres>$
Lexical Transfer: ^prpers<prn><p1><sg><mf>$ ^too<too>$ ^play<vblex><pres><p1>$
Output: I too play

9. Translating: 私 は 仕事 に 急ぎます

Morphological Analyzer: ^私/私<prn><p1><sg><mf>$ ^は/は<prt><subj>$ ^仕事/仕事<n><sg>$ ^に/に<dir>$ ^急ぎます/急ぐ<vblex><pres>$

POS Tagger: ^私<prn><p1><sg><mf>$ ^は<prt><subj>$ ^仕事<n><sg>$ ^に<dir>$ ^急ぐ<vblex><pres>$
Lexical Transfer: ^prpers<prn><p1><sg><mf>$ ^hurry<vblex><pres><p1>$ ^to<dir>$ ^work<n><sg>$
Output: I hurry to work

10. Translating: 私達 は 仕事 に 急ぎます

Morphological Analyzer: ^私達/私<prn><p1><pl><mf>$ ^は/は<prt><subj>$ ^仕事/仕事<n><sg>$ ^に/に<dir>$ ^急ぎます/急ぐ<vblex><pres>$

POS Tagger: ^私<prn><p1><pl><mf>$ ^は<prt><subj>$ ^仕事<n><sg>$ ^に<dir>$ ^急ぐ<vblex><pres>$

Lexical Transfer: ^prpers<prn><p1><pl><mf>$ ^hurry<vblex><pres><p1>$ ^to<dir>$ ^work<n><sg>$
Output: we hurry to work

11. Translating: 私 は 仕事 に 帰ります

Morphological Analyzer: ^私/私<prn><p1><sg><mf>$ ^は/は<prt><subj>$ ^仕事/仕事<n><sg>$ ^に/に<dir>$ ^帰ります/帰る<vblex><pres>$

POS Tagger: ^私<prn><p1><sg><mf>$ ^は<prt><subj>$ ^仕事<n><sg>$ ^に<dir>$ ^帰る<vblex><pres>$
Lexical Transfer: ^prpers<prn><p1><sg><mf>$ ^return<vblex><pres><p1>$ ^to<dir>$ ^work<n><sg>$
Output: I return to work

12. Translating: 君達 は 良い 人 です

Morphological Analyzer: ^君達/君<prn><p2><pl><mf>$ ^は/は<prt><subj>$ ^良い/良い<adj><pres>$ ^人/人<n><sg>$ ^です/です<copula><pres>$

POS Tagger: ^君<prn><p2><pl><mf>$ ^は<prt><subj>$ ^良い<adj><pres>$ ^人<n><sg>$ ^です<copula><pres>$
Lexical Transfer: ^prpers<prn><p2><pl><mf>$ ^is<copula><pres><p2>$ ^<det><obj><pl>$ ^good<adj>$ ^person<n><pl>$
Output: you are some good people

13. Translating: 彼ら は 車 を 持ちました

Morphological Analyzer: ^彼ら/彼<prn><p3><pl><mf>$ ^は/は<prt><subj>$ ^車/車<n><sg>$ ^を/を<prt><obj>$ ^持ちました/持つ<vblex><past>$

POS Tagger: ^彼<prn><p3><pl><mf>$ ^は<prt><subj>$ ^車<n><sg>$ ^を<prt><obj>$ ^持つ<vblex><past>$
Lexical Transfer: ^prpers<prn><p3><pl><mf>$ ^have<vblex><past><p3>$ ^<det><obj><sg>$ ^car<n><sg>$
Output: they had a car

14. Translating: それ は 良かった です

Morphological Analyzer: ^それ/それ<prn><p3><sg><mf>$ ^は/は<prt><subj>$ ^良かった/良い<adj><past>$ ^です/です<copula><pres>$

POS Tagger: ^それ<prn><p3><sg><mf>$ ^は<prt><subj>$ ^良い<adj><past>$ ^です<copula><pres>$
Lexical Transfer: ^prpers<prn><p3><sg><mf>$ ^is<copula><past><p3>$ ^good<adj>$
Output: it was good

15. Translating: 私 は 家 から 来ました

Morphological Analyzer: ^私/私<prn><p1><sg><mf>$ ^は/は<prt><subj>$ ^家/家<n><sg>$ ^から/から<dir>$ ^来ました/来る<vblex><past>$

POS Tagger: ^私<prn><p1><sg><mf>$ ^は<prt><subj>$ ^家<n><sg>$ ^から<dir>$ ^来る<vblex><past>$
Lexical Transfer: ^prpers<prn><p1><sg><mf>$ ^come<vblex><past><p1>$ ^from<dir>$ ^<det><subj><sg>$ ^house<n><sg>$

Output: I came from the house

16. Translating: 彼 は 本 を 読みます

Morphological Analyzer: ^彼/彼<prn><p3><sg><m>$ ^は/は<prt><subj>$ ^本/本<n><sg>$ ^を/を<prt><obj>$ ^読みます/読む<vblex><pres>$

POS Tagger: ^彼<prn><p3><sg><m>$ ^は<prt><subj>$ ^本<n><sg>$ ^を<prt><obj>$ ^読む<vblex><pres>$
Lexical Transfer: ^prpers<prn><p3><sg><m>$ ^read<vblex><pres><p3>$ ^<det><obj><sg>$ ^book<n><sg>$
Output: he reads a book

17. Translating: 彼 は 死にました

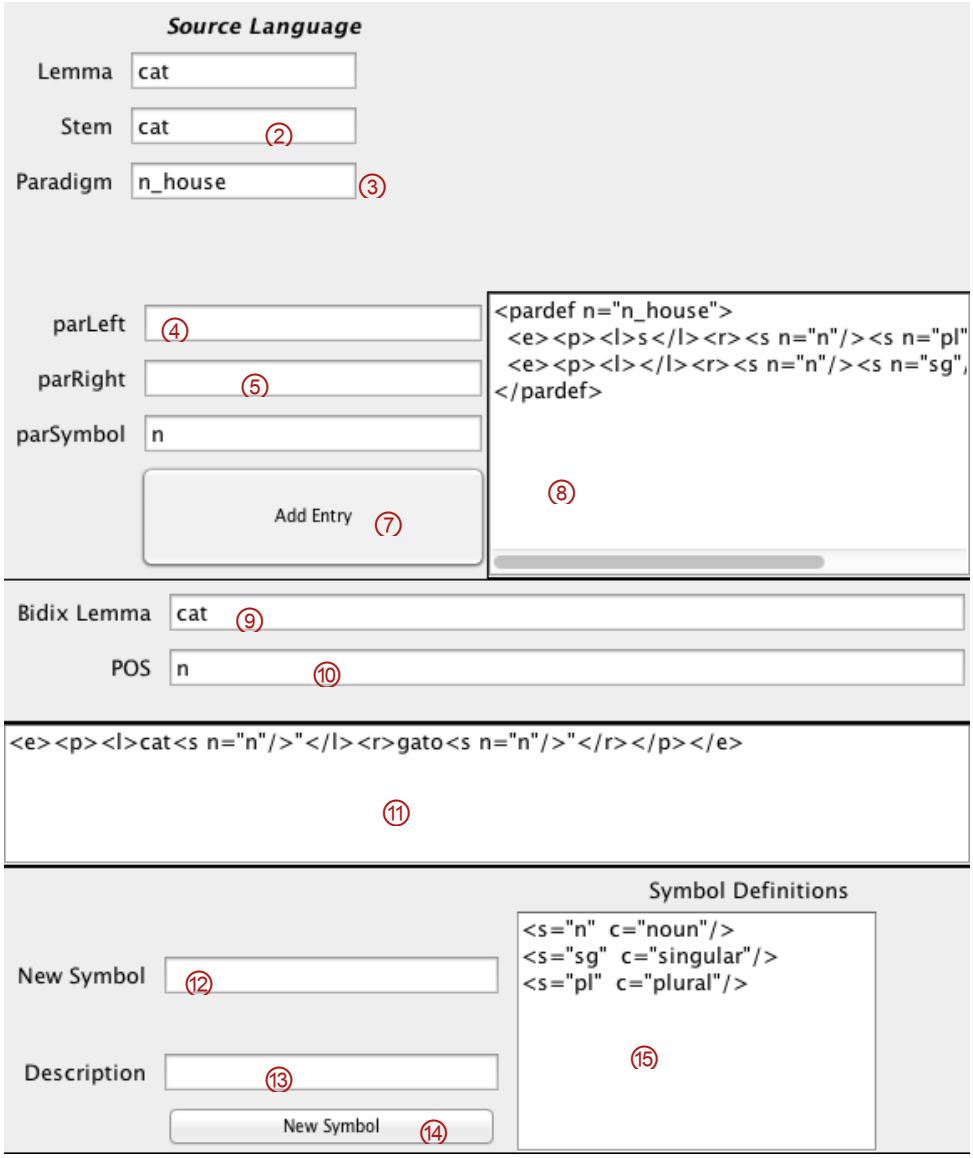
Morphological Analyzer: ^彼/彼<prn><p3><sg><m>$ ^は/は<prt><subj>$ ^死にました/死ぬ<vblex><past>$

POS Tagger: ^彼<prn><p3><sg><m>$ ^は<prt><subj>$ ^死ぬ<vblex><past>$
Lexical Transfer: ^prpers<prn><p3><sg><m>$ ^die<vblex><past><p3>$
Output: he died

18. Translating: 彼 は その 事 を 言いました

Morphological Analyzer: ^彼/彼<prn><p3><sg><m>$ ^は/は<prt><subj>$ ^その/その<dem_adj>$ ^事/事<n><sg>$ ^を/を<prt><obj>$ ^言いました/言う<vblex><past>$

POS Tagger: ^彼<prn><p3><sg><m>$ ^は<prt><subj>$ ^その<dem_adj>$ ^事<n><sg>$ ^を<prt><obj>$ ^言う<vblex><past>$
Lexical Transfer: ^prpers<prn><p3><sg><m>$ ^say<vblex><past><p3>$ ^that$ ^thing<n><sg>$
Output: he said that thing



Appendix C.Apertium Dictionary Builder

Synopsis
========
The Apertium Dictionary Builder is a simple tool designed to allow a user to simultaneously create the three required dictionaries for a language pair in the rule-based machine translation system, Apertium. It requires no knowledge of the underlying XML structure of Apertium, only a basic working knowledge of the Apertium dictionary file structure and concepts.

Requirements

==========
Dictionary Builder was developed and tested using Java 1.8, though it may run on 1.7 or 1.6.

Installation
==========
Dictionary Builder is a small, standalone runnable .jar file that can be run on any platform. Simply download it to the location of your choice, and double-click the icon to run the application.

Usage
=====
This tool is designed to be used with the Apertium machine translation system (https://www.apertium.org). Definitions and explanations of the system and its components can be found at the Apertium Wiki (http://wiki.apertium.org/wiki/Main_Page). This documentation assumes a basic knowledge of Apertium and it's data files.

Dictionary Builder allows the user to create dictionary entries for both Apertium monolingual dictionaries, the bilingual dictionary and their corresponding pardefs and sdefs all at the same time. Dictionaries built with this application can be exported and used with the Apertium system without alteration. Currently macros and special sections are not supported, but these sections may be inserted into a file after export.

Below are two screenshots, one of each half of the application's main window with explanations of the various features:

(1) Source Language Lemma Field. Here you will enter the name of the lemma you would like to add to the source language. Once the lemma is entered here, it will automatically populate in (9)

(2) Source Language Stem Field. Here you will enter the stem (the <i> field for Apertium's monolingual dictionaries) of the lemma you are adding to the source language. For example, if your word is the English noun "house", the stem would also be 'house', as there is no morphological transformation that is done to the base form. If your word is the English verb "see", the stem would be 's', as the base form transforms depending on the conjugation (see, saw, sees, seen, seeing, etc) and the only part of the base that remains constant is 's'. If your word is the English verb "go", the stem would be empty, as the

base form transforms to irregular forms (go, goes, gone, went) that contain no part of the base form.

(3) Source Language Paradigm Field: Here you will enter the Paradigm Definition (pardef) that the lemma will reference. Once entered here, any entries added in the Paradigm Section below will be added to this pardef.

(4) Source Language Paradigm Entry Left Field: Here you will enter the <l> field for the current pardef entry. This will be the suffix appended to the stem of the lemma that references this pardef.

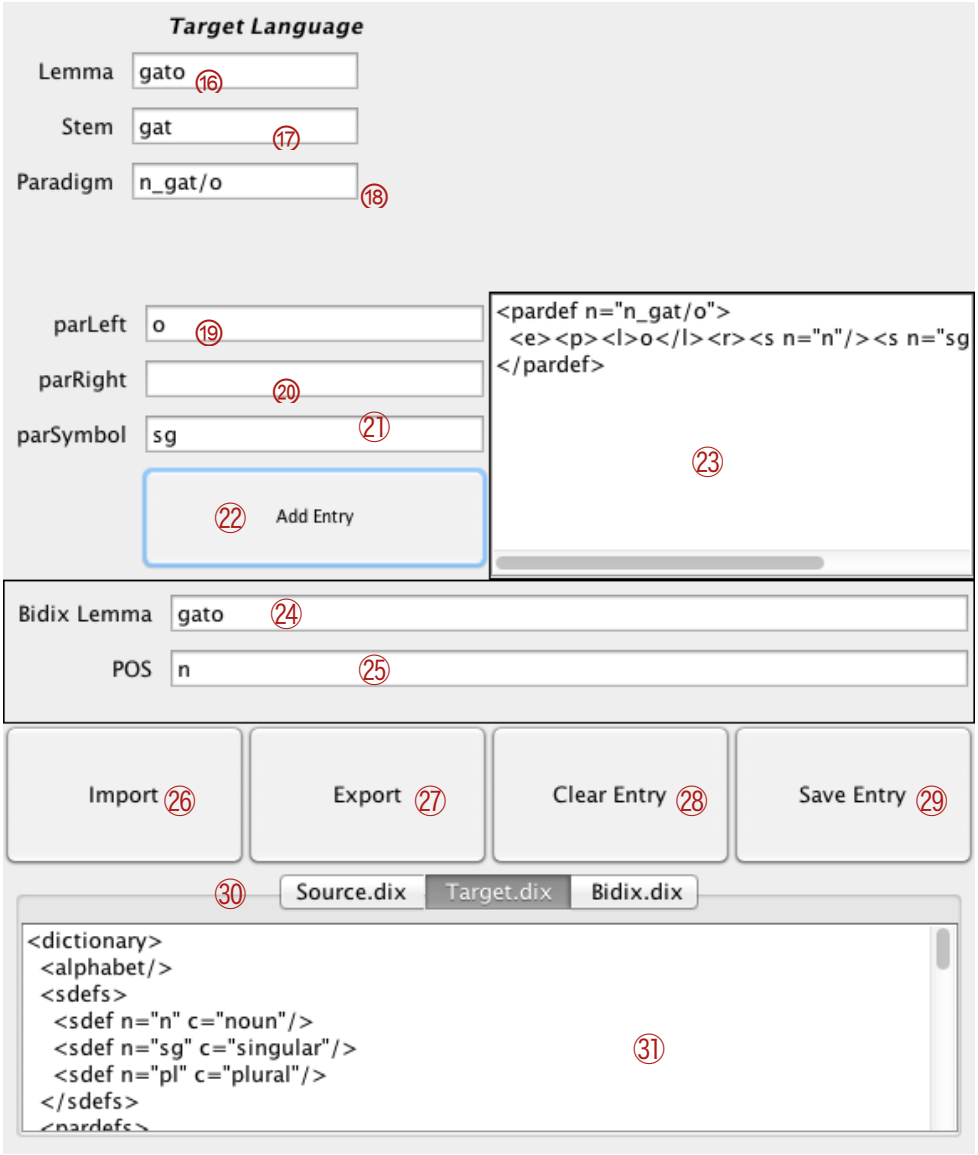
(5) Source Language Paradigm Entry Right Field: Here you will enter the <r> field for the current pardef entry. This will be the suffix appended to the stem of the lemma that references this pardef. Note that by default, the direction of transformation is L->R. The R->L annotation is not currently supported in Dictionary Builder.

(6) Source Language Paradigm Entry Symbol Field: Here you will enter the Symbol Definitions (sdef) that will be added to the lemma that references this pardef. When you add an sdef to the current pardef entry, it will appear in the pardef preview window (8). You may add as many sdefs as are needed to an entry and continue to add more at any time. A symbol must be defined and appear in the Symbol Definition List (15) before you can enter it here.

(7) Source Language Add Paradigm Entry Button: This button adds to the current pardef in field (3) the data contained in fields (4), (5) and (6). Duplicate entries will be ignored by the program. Dictionary Builder currently does not support removal of a harder entry without clearing the entire global entry.

(8) Source Language Paradigm Definition Preview Pane: Here you will see a preview of the current pardef you are woking with. It is updated every time you add a pardef entry using (7).

(9) Bilingual Dictionary Source Lemma Field: Here you will enter the source language lemma to map to a target language lemma in the bilingual dictionary. This field is automatically populated after the source lemma (1) is entered.



(10) Bilingual Dictionary Source Symbol Field: Here you will enter the sdef that correlates to the lemma in (9). A symbol must be defined and appear in the Symbol Definition List (15) before you can enter it here.

(11) Bilingual Dictionary Preview Pane: Once fields (9), (10), (24) and (25) have been filled, a preview of the bilingual dictionary entry for the source lemma (1) and the target lemma (16) will be displayed here.

(12) New Symbol Field: Here you will enter a new Symbol to be defined and added to the Symbol Definition List (15). Duplicate sdefs are not allowed.

(13) New Symbol Description Field: Here you can enter the optional ‘c” attribute for the sdef describing its Symbol.

(14) Add New Symbol Button: Once the New Symbol Field (12) contains a Symbol (and optionally the New Symbol Description Field (13) is filled, as well), clicking this button will add the sdef to the Symbol Definition List (15). Duplicate sdefs will be rejected.

(15) Symbol Definition List: This pane will display all the sdefs that are currently defined for the dictionaries you are building. The list is global and persistent, so it is used by all three dictionaries and remains while the application is running.

(16) Target Language Lemma Field. Here you will enter the name of the lemma you would like to add to the target language. Once the lemma is entered here, it will automatically populate in (24)

(17) Target Language Stem Field. Here you will enter the stem (the <i> field for Apertium’s monolingual dictionaries) of the lemma you are adding to the target language. See the description of Field (2) for more details.

(18) Target Language Paradigm Field: Here you will enter the Paradigm Definition (pardef) that the lemma will reference. Once entered here, any entries added in the Paradigm Section below will be added to this pardef.

(19) Target Language Paradigm Entry Left Field: Here you will enter the <l> field for the current pardef entry. This will be the suffix appended to the stem of the lemma that references this pardef.

(20) Target Language Paradigm Entry Right Field: Here you will enter the <r> field for the current pardef entry. This will be the suffix appended to the stem of the lemma that references this pardef. Note that by default, the direction of transformation is L->R. The R->L annotation is not currently supported in Dictionary Builder.

(21) Target Language Paradigm Entry Symbol Field: Here you will enter the Symbol Definitions (sdef) that will be added to the lemma that references this pardef. When you add an sdef to the current pardef entry, it will appear in the pardef preview

window (8). You may add as many sdefs as are needed to an entry and continue to add more at any time. A symbol must be defined and appear in the Symbol Definition List (15) before you can enter it here.

(22) Target Language Add Paradigm Entry Button: This button adds to the current pardef in field (3) the data contained in fields (4), (5) and (6). Duplicate entries will be ignored by the program. Dictionary Builder currently does not support removal of a harder entry without clearing the entire global entry.

(23) Target Language Paradigm Definition Preview Pane: Here you will see a preview of the current pardef you are woking with. It is updated every time you add a pardef entry using (7).

(24) Bilingual Dictionary Target Lemma Field: Here you will enter the source language lemma to map to a target language lemma in the bilingual dictionary. This field is automatically populated after the source lemma (1) is entered.

(25) Bilingual Dictionary Target Symbol Field: Here you will enter the sdef that correlates to the lemma in (9). A symbol must be defined and appear in the Symbol Definition List (15) before you can enter it here.

(26) Import Dictionaries Button: Clicking this button imports dictionary files that reside in the SAME DIRECTORY as Dictionary Builder into memory. The Symbol Definition List (15) will populate, and you will see previews of the files in the Dictionaries Preview Pane (31). These dictionary files additionally must be named “source.dix”, “target.dix” and “bilingual.dix” when working with Dictionary Builder.

(27) Export Dictionaries Button: Clicking this button will export all the data stored in memory to the three appropriate Apertium dictionary files: “source.dix”, “target.dix” and “bilingual.dix”. The files will be exported into the SAME DIRECTORY as the application. Furthermore, they will overwrite any identically named files in that directory. If you need to retain older versions of dictionary files with the same name that exist in the same directory, they must be moved or renamed before exporting.

(28) Clear Entry Button: Clicking this button clears all the fields of the app. It does not clear the Symbol Definition

List, it does not delete any pardef entries or global entries that have been previously saved.

(29) Save Entry Button: Clicking this button saves all fields and any new pardefs and pardef entries that have been added since the last time this button was clicked. In order for the data to be saved, all fields that are not optional must be filled. Duplicate entries will be rejected. You must save an entry before it can be exported to a file.

(30) Dictionary File Preview Selection Pane: Here you can click on any of the three buttons to change the view in the Dictionary Preview Pane (31) to the corresponding dictionary file.

(31) Dictionary Preview Pane: Here you can view the current dictionary files that are saved on your drive.