Flexible finite-state lexical selection for rule-based machine translation

Francis M. Tyers, Felipe Sánchez-Martínez, Mikel L. Forcada
Departament de Llenguatges i Sistemes Informàtics
Universitat d’Alacant
E-03071 Alacant
{ftyers,fsanchez,mlf}@dlsi.ua.es

Abstract
In this paper we describe a module (rule formalism, rule compiler and rule processor) designed to provide flexible support for lexical selection in rule-based machine translation. The motivation and implementation for the system is outlined and an efficient algorithm to compute the best coverage of lexical-selection rules over an ambiguous input sentence is described. We provide a demonstration of the module by learning rules for it on a typical training corpus and evaluating against other possible lexical-selection strategies. The inclusion of the module, along with rules learnt from the parallel corpus provides a small, but consistent and statistically-significant improvement over either using the highest-scoring translation according to a target-language model or using the most frequent aligned translation in the parallel corpus which is also found in the system’s bilingual dictionaries.

1 Introduction
This paper presents a module for lexical selection to be used in rule-based machine translation (RBMT). The module consists of an XML-based formalism for specifying lexical-selection rules in the form of constraints, a compiler which converts the rules written in this format to a finite-state transducer, and a processor which applies the rule transducer to ambiguous input sentences. The paper also presents a method of learning lexical-selection rules from a parallel corpus.

Lexical selection is the task of choosing, given several source-language (SL) translations with the same part-of-speech (POS), the most adequate translation among them in the target language (TL). The task is related to the task of word-sense disambiguation (Ide and Véronis, 1998). The difference is that its aim is to find the most adequate translation, not the most adequate sense. Thus, it is not necessary to choose between a series of fine-grained senses if all these senses result in the same final translation.

The dominant approach to MT for language pairs with sufficient training data is phrase-based statistical machine translation; in this approach, lexical selection is performed by a combination of cooccurrence in the phrase table, and score from the target-language model (Koehn, 2010). There have however been attempts to improve on this by looking at global lexical selection over the whole sentence, see e.g. (Venkatapathy and Bangalore, 2007; Carpuat and Wu, 2007).

In order to test different approaches to lexical selection for RBMT, we use the Apertium (Forcada et al., 2011) platform. This free/open-source platform includes 30 language pairs (as of February 2012).

Sánchez-Martínez et al. (2007) describe a method to perform lexical selection in Apertium based on training a source-language bag-of-words model using TL cooccurrence statistics. This approach was tested, but abandoned as it produced less adequate translations than using the translation marked as default by a linguist in the bilingual dictionary.

Other possible solutions would be to generate all possible combinations of translations, and score them on a language model of the target language. This approach is taken in the METIS-II system (Melero et al., 2007). This has the benefit of being easy to implement, and only requiring a bilingual dictionary and a monolingual target language corpus. It has the drawbacks of being both slow – many
translations must be performed – and not very cus-
tomisable – control over the final translation is left
to the TL model.

Another possible solution, and one that is already
used in some Apertium language pairs (Brandt et
al., 2011; Wiechetek et al., 2010) is to use con-
straint grammar (Karlsson et al., 1995) rules to
choose between possible alternative translations.
An advantage of this is that the constraint grammar
formalism is well known, and powerful, allowing
context searches of unlimited size. However, it is
too slow to be able to be used for production sys-
tems, as the speed is in the order of a few hundred
words per second as opposed to thousands of words
per second for the slowest Apertium module.

Another approach not requiring a parallel cor-
pus is presented by Dagan and Itai (1994). They
first parse the SL sentence and extract syntactic re-
lations, such as verb + object, they then translate
these with a bilingual dictionary and use colloca-
tion statistics from a TL corpus to choose the most
adequate translation. While this method does not
rely on the existence of a parallel corpus, it does
deck on some way of identifying SL syntactic
relations – which may not be available in all RBMT
systems.

The rest of the paper is laid out as follows: Sec-
tion 2 presents some design decisions that were
made in the development of the module. Section 3
describes in detail the rule formalism, the represen-
tation of rules as a finite-state transducer, and the
algorithm for applying the rules to an ambiguous
input sentence. Section 4 shows how rules for the
module may be learnt from a parallel corpus, and
then evaluated on a standard test set for MT. Finally,
section 6 offers some concluding remarks and ideas
for future work.

2  Lexical selection in Apertium

Apertium is an free/open-source platform for cre-
aturing shallow-transfer RBMT systems. The plat-
form is being widely used to build MT systems
for a variety of language pairs, especially in those
cases (mainly with related-language pairs) where
shallow transfer suffices to produce good quality
translations. It has, however, also proven useful
in assimilation scenarios with more distant pairs
involved.

The platform is designed to be: fast, in the order
of thousands of words per second on a normal desk-
top computer; easy to develop; and standalone, no
need for existing data or large parallel corpora to
build a system.

Apertium uses a Unix pipeline architecture (see
Figure 1) to perform translation: text is first stripped
of format and morphologically analysed, then mor-
phologically disambiguated. Then the unambigu-
ous analyses are passed through lexical and struc-
tural transfer and finally morphological generation.
This translation strategy is very similar to other
transfer-based MT systems.

The Apertium platform does not currently have
a specific module for lexical selection. Some trans-
lation ambiguity can be handled using multi-word
expressions (MWEs) encoded in the dictionaries of
the system, but the status quo is that for any given
SL word, the most frequent, or most general trans-
lation is given. This poses a translation problem, as
often it may be difficult to choose the most frequent
or the single most adequate translation of a word,
or the selection strongly depends on the context.

2.1 Requirements

The requirements of a lexical selection module are:

- It should be efficient and fast, that is, it should
  process thousands of words per second on a
  normal desktop computer. For rule sets of tens
  of thousands of rules.

- It should not require any advanced resources,
  such as parallel corpora, but should be able to
  take advantage of them if available.

- The functioning of the module should be trace-
  able. In any given translation, it should be
  possible to identify the rules used.

- The rules should be in a form suitable for read-
  ing and writing by human beings so that users
  can immediately change or add rules.

In the next section we describe a lexical selection
module which fulfils these requirements.

In order to accommodate the new lexical selection
module, a minor change was made to the pipeline
(Figure 1). Where previously lexical transfer was
performed at the same time as structural transfer,
now lexical transfer is performed as a separate pro-
cess before the structural transfer stage.

3  Methodology

3.1 Rule formalism

The rule formalism is based on context rules, con-
taining a sequence of the following features,

- A pattern matching a single SL lexical form
A pattern matching a single TL lexical form

- One of the following operations:
  
  - **select**: chooses the TL translation which matches the lexical-form pattern and removes all translations which do not match.
  
  - **remove**: removes the TL translation which match the given lexical-form pattern; and
  
  - **skip**: makes no changes and passes all the translations through unchanged; this is used when specifying the context of the rule.

The features are expressed by regular expressions, which may match any part of the input word string (e.g. either the lemma, the tags or a combination of both). As with the rest of the modules in the Apertium platform, the rules are written in an XML-based format, which is processable by both humans and machines.

Figure 2 presents some examples of rules written in this formalism. Each rule is enclosed in a *rule* element, with an optional `c` attribute for comments. The *rule* tag may have one or more *match* elements which describe sequences of SL context. Each *match* element may have either a `lemma` or a `tags` attribute, neither (in which case it will match any word) or both.

A *match* element may also contain a lexical selection operation, *select* or *remove*, the default one being *skip*.

The rules can be written by hand to solve specific translation issues with a given context, for example, given the Spanish word *estación* ‘station, season’ with a default translation of ‘station’, we may write rules (see Figure 2) which say that we want to translate the word as ‘season’ if it is followed by an adjective such as *seca* ‘dry’ or *lluviosa* ‘rainy’, or if it is followed by the preposition *de* ‘of’, a determiner (e.g. *el* ‘the’), and the noun *año* ‘year’.

A weak point of the formalism is that rules can only take into account fixed-length, ordered contexts, so it is not possible to e.g. make a rule which selects a given translation based on a given word at any position in the sentence (e.g. treating the sentence, or part of it, as a bag of words). However, a strength is that the rules may be compiled into a compact finite-state transducer, which is traceable; for each translation, it is possible to know exactly which rules were called.

### 3.2 Rule compilation

The set of rules $R$ expressed in XML is not processed directly; they are compiled into a finite-state transducer (see Figure 3). In this transducer, each transition is labelled with a symbol representing an SL pattern and a symbol representing an operation on a TL pattern. Both SL and TL patterns are compiled into regular expressions (finite-state recognisers), and stored in a lookup table.

The transducer is defined as $\langle Q, V, \delta, q_0, q_F \rangle$, where $Q$ is the set of states, $V = \Sigma \times \Gamma$ is the alphabet of transition labels, where $\Sigma$ is the set of input symbols and $\Gamma$ is the set of output symbols, $\delta : Q \times V \rightarrow Q$ is the transition function, $q_0$ is the initial state (nothing matched); and $q_F$ is the final state indicating that a complete pattern has been matched. Rules in $R$ are paths from $q_0$ to $q_F$.

### 3.3 Rule application

In order to apply the rules on an input sentence, we use a variant of the best coverage algorithm described by Sánchez-Martínez et al. (2009). We try to cover the maximum number of words of each SL sentence by using the longest possible rules; the motivation for this is that the longer the rules, the more accurate their decisions may be expected to be because they integrate more context.

To compute the best coverage a dynamic-programming algorithm (Alg. 1) is applied, which starts a new search in the automaton at every new word in the sentence to be translated, and uses a
An example of the rules written by hand in the XML formalism for describing lexical selection rules. The formalism is the same for both hand-written and learnt rules. The order of rules is only important in calculating the rule number for tracing.

A finite-state transducer representing four lexical selection rules; each arc is a transition between a pattern matching an SL lexical form, and an operation with a pattern matching a TL lexical form.
set of alive states $A$ in the automaton and a map $M$ that, for each word in the sentence, returns the best coverage up to that word together with its score.

Algorithm 1 uses four external procedures: $\text{WORDCOUNT}(s)$ returns the number of words in the string $s$; $\text{RULELENGTH}(c)$ returns the number of words of the rule matched by state $c$; $\text{NEWCOVERAGE}(\text{cov}, c)$ computes a new coverage by adding to coverage $\text{cov}$ the rule recognised by state $c$; finally, $\text{BESTCOVERAGE}(a, b)$ receives two coverages and returns the one using the least possible number of rules.

In the current implementation, if two different coverages use the same number of rules, then the former is overwritten. This may not be the most adequate approach to dealing with the problem, and we intend to study other approaches.

## 4 Experiment

In order to test the flexibility of the module, we decided to learn rules from an existing knowledge source, i.e. a parallel corpus, and test the module on a well-known task for the evaluation of MT.

The experimental setup follows the training of the baseline system in the shared task on MT at WMT11 (Callison-Burch et al., 2011), with the following differences: In place of the default Moses perl-based tokeniser, tokenisation was done using the Apertium morphological analyser (Cortés-Vaíllo and Ortiz-Rojas, 2011). The corpus was also not lowercased; instead the case of known words was changed to the dictionary case as found in the Apertium monolingual dictionary.

We use version 6.0 of the EuroParl corpus (Koehn, 2005), and take the first 1.4 million lines for training. We used the Apertium English to Spanish pair apertium-en-es as it is one of the few pairs that has dictionaries with more than one alternative translation per word.

### 4.1 Learning lexical selection rules from a parallel corpus

The procedure to learn rules from a parallel corpus is as follows: We first morphologically analyse and disambiguate for part-of-speech both the SL and TL sides of the corpus. These are then word-aligned with GIZA++ (Och and Ney, 2003).

We then pass the SL side of the corpus through the lexical-transfer stage of the MT system we are learning the rules for; this gives three sets of sentences: the tagged SL sentences, the tagged TL sentences and the possible translations of the SL words into the TL yielded by the bilingual dictionary.

We take these three sets, and extract from the parallel corpus those sentence pairs for which at least one lexically ambiguous SL word is aligned to a word in the TL which is also found in the bilingual dictionary. This step is necessary as in order to be translated by the rest of the system, the alternative translation must appear in the bilingual dictionary. After extracting these sentence pairs we have 332,525 sentences for training, that is around 24% of them.

For each of these extracted sentences, we extract $n$-grams (trigrams and five-grams) of context around the ambiguous SL word(s) which belong to the categories of adjective, noun and verb. We then count up how many times we see this context appearing along with each of the translations in the TL. If a given possible translation appears aligned to a word in a given context more frequently than other possible translations, then we generate a rule which selects the aligned translation in that same context over other translations in that context.

### 4.2 Systems

To evaluate the lexical selection module, and our method for obtaining rules from a parallel corpus, we compare it against four baseline systems:

- **freq**: Frequency defaults; the MT system is tested with rules that select the most frequent translation in the TL corpus. This is equivalent to a unigram TL model.
- **align**: The TL word which is most frequently aligned to the given SL word is chosen. This correspondence must also appear in the bilingual dictionary of the MT system.
- **ling**: The linguistic defaults, here the translations considered ‘most adequate’ by the human linguist who wrote the system, are selected.
- **tlm**: The highest scoring translation out of the possible translations for the whole sentence as chosen by a 5-gram language model of the Spanish side of the EuroParl corpus trained with IRSTLM (Federico et al., 2008).
We also tested three different sets of rules in our lexical-selection module:

- **all**: No filtering. All of the generated rules are included.
- **filt1**: The rules where contexts which only appear once in the training corpus are removed.
- **filt2**: Rules which include the tags for subordinating conjunction and full stop are excluded as well as rules where the translation selected is under half of the total frequency of the word. So for example if a word has three translations with frequency 10 and one translation with frequency 15, the rule selecting this translation would be excluded as $15 < (45 / 2)$ even though it is the most frequent.

The motivation for excluding rules which contain subordinating conjunctions and full stops is that they are likely to be noisy. The motivation for excluding rules with under half of the total frequency of the word is to try and keep only those rules that we are really sure will improve translation quality overall. These are rather coarse heuristics, and the subject of rule filtering merits further investigation (see section 6).

### 5 Evaluation

To evaluate the systems, we extracted the set of sentences from the 2,489-sentence News Commentary corpus which contained at least one ambiguous open-category word in the SL aligned with a TL word in the reference translation which could be generated by the MT system. The alignments between SL and TL words in the corpus were obtained by adding it to a separate copy of the EuroParl corpus to the one used for training, and running GIZA++ again.

In total, this gave 434 sentences (9,463 tokens) to be evaluated (approximately 17%). The average number of translations per word was 1.08. We performed two evaluation tasks, the first was the error rate of the lexical selection module, and the second was a full translation task.

For the first, we made a labelled corpus (similar to that in (Vickrey et al., 2005)) by disambiguating the lexical transfer output using the reference translation. Out of the 434 sentences this gave us a total of 604 disambiguated words. This could be considered an oracle, that is the best result the MT system could get if it just chose the translation looking at the reference translation. The column **Error** in Table 2 gives the lexical-selection error rate over this test corpus, that is the number of times the given system chooses a translation which is not equivalent to what the oracle would choose.

The second task was to compare the systems using the common evaluation metrics **BLEU** (Papineni et al., 2002) and **Word error rate** (WER), based on the Levenshtein distance (Levenshtein, 1965).

This second task is not ideal for evaluating the task of a lexical selection module as the perfor-

---

**Algorithm 1** **OPTIMALCOVERAGE**: Algorithm to compute the best coverage of an input sentence.

**Require**: $s$: SL sentence to translate

$A ← \{q_0\}$

$i ← 1$

while ($i ≤ \text{WORDCOUNT}(s)$) do

$M[i] ← \emptyset$

for all $q ∈ A$ do

for all $c ∈ Q : \exists t : δ(q, (s[i] : t) = c)$ do

$A ← A ∪ \{c\}$

if $c = q_F$ then

$M[i] ← \text{BESTCOVERAGE}(M[i], \text{NEWCOVERAGE}(M[i−\text{RULELENGTH}(c)], c))$

end if

end for

$A ← A − \{q\}$

end for

$i ← i + 1$

$A ← A ∪ \{q_0\}$ /* To start a new search from the next word */

end while

return $M[i−1]$
Table 1: Translation of segment #56 in the News Commentary corpus by two of the systems.

Table 2 reports the 95% confidence interval for the BLEU, WER and ERROR scores achieved on the test set by the seven systems. Confidence intervals were calculated through the bootstrap resampling (Efron and Tibshirani, 1994) method as described by (Koehn, 2004; Zhang and Vogel, 2004). Bootstrap resampling was carried out for 1,000 iterations.

Given the small differences in score between the individual systems, we also performed pair bootstrap resampling between the two highest scoring systems (alig and filt2) to see if the difference was statistically significant. Over 1,000 iterations, the filt2 system was shown to offer an improved translation 95% of the time for both the BLEU and ERROR scores.

6 Concluding remarks

We have presented a lexical-selection module suitable for inclusion in a RBMT system, and shown how the rules it uses may be learnt from a parallel corpus. In pair bootstrap resampling, the system offers a statistically significant improvement in translation quality over the next highest scoring system.

In the future we would like to investigate the following: The first is the possibility of learning the rules without any parallel corpus. We aim to follow the same principles as (Sánchez-Martínez et al., 2008) where a monolingual TL corpus was used to improve the performance of an HMM part-of-speech tagger. Some initial experiments have already been conducted to this effect, however the observed performance of the TL model in choosing between different translations from an RBMT system gives an indication of the difficulty of improving over the ‘linguistic default’ baseline.

While the learning from parallel corpora is only a demonstration, we would like to look into methods to address the problem of filtering/pruning the generated rules to remove those which do not offer an improvement in translation quality, as it would also apply to learning rules without parallel corpora.

The system has also been built with the possibility of weighted rules, we would like to investigate the possibility of automatically assigning rule weights to more reliable rules.

Acknowledgements

We are thankful for the support of the Spanish Ministry of Science and Innovation through project TIN2009-14009-C02-01, and the Universitat d’Alacant through project GRE11-20. We also thank Sergio Ortiz Rojas for his constructive comments and ideas on the development of the system, and the anonymous reviewers for comments on the manuscript.

References


<table>
<thead>
<tr>
<th>System</th>
<th>Total rules</th>
<th>Called</th>
<th>Error</th>
<th>BLEU</th>
<th>WER</th>
</tr>
</thead>
<tbody>
<tr>
<td>freq</td>
<td>-</td>
<td>-</td>
<td>[42.8, 50.3]</td>
<td>[0.1687, 0.1794]</td>
<td>[0.712, 0.725]</td>
</tr>
<tr>
<td>ling</td>
<td>667</td>
<td>473</td>
<td>[25.4, 30.7]</td>
<td>[0.1772, 0.1879]</td>
<td>[0.710, 0.723]</td>
</tr>
<tr>
<td>alig</td>
<td>600</td>
<td>533</td>
<td>[19.3, 25.8]</td>
<td>[0.1786, 0.1892]</td>
<td>[0.709, 0.723]</td>
</tr>
<tr>
<td>tlm</td>
<td>-</td>
<td>-</td>
<td>[37.0, 44.9]</td>
<td>[0.1708, 0.1817]</td>
<td>[0.714, 0.727]</td>
</tr>
<tr>
<td>all</td>
<td>77,077</td>
<td>503</td>
<td>[21.3, 28.2]</td>
<td>[0.1779, 0.1885]</td>
<td>[0.710, 0.723]</td>
</tr>
<tr>
<td>filt1</td>
<td>9,978</td>
<td>503</td>
<td>[20.3, 26.9]</td>
<td>[0.1782, 0.1889]</td>
<td>[0.710, 0.723]</td>
</tr>
<tr>
<td>filt2</td>
<td>2,661</td>
<td>532</td>
<td>[17.9, 24.7]</td>
<td>[0.1789, 0.1896]</td>
<td>[0.709, 0.723]</td>
</tr>
</tbody>
</table>

Table 2: Evaluation results for the seven systems on the news commentary test corpus.


