# Semi-Automatic Learning of Transfer Rules for Machine Translation of Low-Density Languages

Katharina Probst
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA, USA, 15213
kathrin@cs.cmu.edu

Abstract. Data-driven approaches to machine translation often rely heavily on large training corpora. We are developing a translation system targeted specifically at minority languages for which such large corpora are not usually available. Instead, we elicit a controlled corpus from a native speaker. The elicited sentences are used to learn transfer rules using a seeded version space learning algorithm. We describe the theoretical framework of this approach, and provide some preliminary solutions.

## 1 Introduction

In recent years, data-driven methods have gained much attention in the Machine Translation community. Example-based machine translation (e.g. (Brown 1996)) and statistical machine translation (e.g. (Brown, Cocke, Pietra, Pietra, Jelinek, Lafferty, Mercer, and Roossin 1990)) are attractive because they require little human interaction in development and provide good portability. This stands in contrast to traditional interlingua and transfer rule approaches that often require years of human development time. Data-driven methods, however, rely heavily on the existence of large bilingual corpora. The more bilingual data is available, the better these systems are likely to perform. But such corpora may not be available for a given language pair, especially when dealing with low-density languages. In recent years, research groups have started to address MT for low-density languages. For instance, (Frederking, Rudnicky, and Hogan 1997) describe training a multi-engine MT system for speech-to-speech translation of minority languages such as Haïtian-Creole. (Somers 1997) presents an overview of the issues involved in expanding MT to minority languages. One approach to deal with the absence of a large corpus is to collect a comparatively small,



Proceedings of the Seventh ESSLLI Student Session
Malvina Nissim (editor)

but controlled corpus. This approach was taken before us for example by (Nirenburg and Raskin 1998; Sherematyeva and Nirenburg 2000; Nirenburg 1998). The authors describe a corpus that is built based on linguistic features across languages. A bilingual speaker serves as an informant on the target language. Among other things, this informant provides information about what values a linguistic feature can take on in the target language. Our approach is similar in that it uses an elicitation corpus that is based on typological knowledge. A bilingual speaker then translates the corpus, and the system is trained on it, making use of the contolled structure of the sentences in the corpus and of the corpus itself. The goal of the training is to learn a set of transfer rules that can be used for a machine translation system between a major language (English or Spanish) and the target language, which can be a minority language. In our current approach, the source language is always the major language. The minority languages we are currently targeting are Mapudungun (spoken in Chile) and Inupiaq (spoken in Alaska).

The focus of this paper is to describe how transfer rules are learned from controlled bilingual word-aligned sentences. We present how initial rules are constructed for each sentence using an algorithm called seed generation, and how these rules are then generalized using a seeded version space algorithm to form a set of transfer rules that can be used in a translation system. (Tenni, Lehtola, Bounsaythip, and Jaaranen 1999) describe a similar approach where translation rules by first hypothesizing specific rules and then generalizing over them. Unlike our system, however, they use semantically tagged data, so that the generalized rules represent contexts with constraints over semantic categories such as 'color'. Also, their system targets a controlled language and a limited domain. Similar to our approach, rule generalization is done by removing constraints such as the part of speech of the following word. As in our system, rules are evaluated by checking their translation power.

## 2 The Elicitation Corpus

Learning of transfer rules occurs based on a controlled bilingual corpus. The corpus consists of sentences that we design prior to learning, and once for all languages. The reason for a controlled corpus is twofold: When using a relatively small, controlled corpus, we can 'enforce' perfect knowledge of the source language. In particular, we can parse all the sentences in the corpus in advance, disambiguate them, and ensure that the parses are complete and correct. This will help reduce errors and will make the training data less noisy. As mentioned above, the second motivation for a controlled corpus is that there are no large corpora available for the target languages. By designing a corpus, we can maximize coverage while minimizing the size of the

corpus by targeting specific constructions. Currently, the elicitation corpus contains around 800 sentences, but we expect it to grow to several thousand sentences. Due to its expected large size, we have developed methods based on Implicational Universals that allow the informant to translate only part of the corpus without serious loss of information. For more detail on the elicitation process, please refer to (Probst and Levin 2002).

The sentences in the elicitation corpus are based on linguistic principles. We elicit as much information as possible about typologically prevalent features (such as number) and their possible values in the target language (such as singluar, dual, plural). To this end, we rely on research done in field linguistics, and use lists that were designed for the task, such as (Comrie and Smith 1977) and (Bouquiaux and Thomas 1992).

Elicitation is divided into two stages. The first stage is feature detection. During feature detection, the system tries to learn as much as possible about the linguistic features that the target language marks for, on what parts of speech they are marked, etc. The second stage is the elicitation of rules that feed into the seeded version space learning algorithm, as will be described below. Currently, the results of feature detection are not utilized effectively in the seeded version space learning algorithm. Therefore, we will not discuss this part of the system in detail here. The interested reader should refer to (Probst, Brown, Carbonell, Lavie, Levin, and Peterson 2001).

## 3 The Transfer Rule Formalism

The transfer rules are learned from sentence pairs. Thus, while there is an association between the rules and the sentences that were used to produce them, the actual lexical sequences are not part of the transfer rule. Rather, the rules represent abstractions over the lexical level.

A transfer rule then consists of the following parts:

- **Alignments:** In a simple setting, these are the word alignments as they were specified by the user. In a compositional setting, where we learn higher-level transfer rules (such as sentence rules) using previously learned phrase rules (such as NP rules), these alignments specify how the components of the sentences, such as noun phrases, align. Compositionality is currently under investigation.
- **Part-of speech/phrase information:** In a 'flat', i.e. non-compositional setting, each transfer rule encodes the sequence of parts of speech of the words in the source and target sentences. When learning a compositional rule, this sequence can encode types of entire constituents, such as noun phrases.
- **Type information:** This component encodes the type of the current transfer rule. Sentence rules are of type S, noun phrase rules of type NP, etc. The formalism also allows for type information on the target language side, although this is not currently used.
- **X-side constraints:** The x-side constraints provide information about features and their values pertaining to the sentence. The x-side is the source language, which is the major language and thus currently English. Each word in the sentence can

be associated with certain features and values for these features. These constraints are used at run-time to determine whether a transfer rule applies to a sentence.

- **Y-side constraints:** The y-side constraints are similar in concept to the x-side constraints, but they pertain to the target language. At run-time, y-side constraints serve to generate the TL sentence.
- **XY-constraints:** The xy-constraints provide information about what feature values will transfer from the source into the target language. Specific TL words can obtain feature values from the source language sentence.

For illustation purposes, examples of a simple transfer rule can be seen in table 1.1. During development, we have been working with major languages such as English and German, so that all examples in this paper will be for this language pair.

Table 1.1: Seed Rules and Generalized Transfer Rule

| **SeedRule1** | **SeedRule2** | **Generalized Rule** |
|---|---|---|
| ;;SL: THE MAN | ;;SL: THE WOMAN | ;;SL: |
| ;;TL: DER MANN | ;;TL: DIE FRAU | ;;TL: |
| NP::NP [DET N] → [DET N] | NP::NP [DET N] → [DET N] | NP::NP [DET N] → [DET N] |
| (;;alignments: | (;;alignments: | (;;alignments: |
| (X1::Y1) | (X1::Y1) | (X1::Y1) |
| (X2::Y2) | (X2::Y2) | (X2::Y2) |
| ;;x-side constraints: | ;;x-side constraints: | ;;x-side constraints: |
| ((X1 AGR) = *3-SING) | ((X1 AGR) = *3-SING) | ((X1 AGR) = *3-SING) |
| ((X1 DEF) = *DEF) | ((X1 DEF) = *DEF) | ((X1 DEF) = *DEF) |
| ((X2 AGR) = *3-SING) | ((X2 AGR) = *3-SING) | ((X2 AGR) = *3-SING) |
| ((X2 COUNT) = +) | ((X2 COUNT) = +) | ((X2 COUNT) = +) |
| ;;y-side constraints: | ;;y-side constraints: | ;;y-side constraints: |
| ((Y1 AGR) = *3-SING) | ((Y1 AGR) = *3-SING) | ((Y1 AGR) = *3-SING) |
| ((Y1 CASE) = *NOM) | ((Y1 CASE) = (*NOT* *GEN *DAT)) | |
| ((Y1 DEF) = *DEF) | ((Y1 DEF) = *DEF) | ((Y1 DEF) = *DEF) |
| ((Y1 GENDER) = *M) | ((Y1 GENDER) = *F) | |
| ((Y2 AGR) = *3-SING) | ((Y2 AGR) = *3-SING) | ((Y2 AGR) = *3-SING) |
| ((Y2 CASE) = *NOM) | | |
| ((Y2 GENDER) = *M) | ((Y2 GENDER) = *F) | ((Y2 GENDER) = (Y1 GENDER)) |

X-side and y-side constraints can be either *value constraints* or *agreement constraints*. Value constraints are of the format $((\Phi_i \gamma_m\ \alpha_j) = \beta_{kj})$, while agreement constraints are of the format $((\Phi_i \gamma_m\ \alpha_j) = (\Phi_i \gamma_n\ \alpha_j))$, where

- $\Phi_i \in \{X, Y\}$
- $\gamma_i \in \begin{cases} \{0 \ldots number\ of\ SL\ words\} & if \Phi_i = X \\ \{0 \ldots number\ of\ TL\ words\} & if \Phi_i = Y \end{cases}$
- $\alpha_i \in \{linguistic\ features\ such\ as\ number, definiteness, agreement, tense, \ldots\}$,
- $\beta_{ki} \in \{x | x\ is\ a\ valid\ value\ of\ feature\ \alpha_i\}$

An example of a value constraint is ((X2 AGR) = *3PLU), where an agreement constraint example is ((X2 AGR) = (X3 AGR)).

## 4 Seed Generation

The first part of the automated learning process is seed generation, i.e. the production of preliminary transfer rules that will be refined during seeded version space learning. The seed generation algorithm takes as input a pair of parallel, word-aligned sentences. The English sentences have been parsed and disambiguated in advance, so that the system has access to a correct feature (or for short f-)structure and a correct phrase or constituent (c-)structure for each sentence. The seed generation algorithm can proceed in different settings depending on how much information is available about the target language. In one extreme case, we assume that we have a fully inflected target language dictionary, complete with feature-value annotations. An example of a typical entry in such a target language dictionary look as follows (the entry is German for 'black' in one inflection): (SCHWARZER ((CAT *ADJ)(ROOT SCHWARZ)(GENDER *M)(CASE *NOM)(DEF *INDEF)(AGR *3-SING))). Naturally, it is expensive and difficult to build such a dictionary of any reasonable size. Therefore, we have begun to build a morphology learning module, which is intended to eventually eliminate the need for a fully inflected and tagged target language dictionary. For the time being, however, we hand-build fully-inflected target language dictionaries to cover our examples.

The other extreme is the absence of any information, in which case the system makes assumptions about what linguistic features' values can be assumed to transfer into the target language. In such a case we assume, for example, that a plural noun will be translated as a plural noun. One field of future work is the explicit incorporation of the information gathered during feature detection, so that more educated guesses can be made.

With the information available, how can we construct a transfer rule of the format described above? All of the x-side information can be gathered directly from the English f- and c-structures. This includes part of speech information, x-side constraints, and type information. The word alignment is given directly by the user. However, the system also needs to construct constraints for the target language, the y-side. By default, the target language part of speech sequence is determined by assuming that English words transfer into words of the same part of speech on the target side. The sequence is then obtained using the word alignments. Note that the assumption that two aligned words have the same part of speech is not always correct. However, it is the best guess in absence of further information. If more information is available about the target language, e.g. a dictionary with part of speech information, this information overrides the assumption that parts of speech transfer directly.

What y-side constraints are produced by the seed generation algorithm will again depend on the amount of information available about the target language. We make a similar assumption as with parts of speech: in ab-

sence of any information, we assume that linguistic features and their values transfer from SL words to the corresponding TL words. However, we are more selective about this type of projection: based on linguistic principles, we have designed a list of features that will likely project their features onto any given target language, such as definiteness. It is clear that this assumption will sometimes fail, but it is the best assumption we can make in this situation. If an inflected TL dictionary is available, the y-side constraints are determined by extracting all the information about the given TL words from the dictionary, and disambiguating and unfying if necessary, i.e. if TL words have more than one entry in the target language dictionary. For example, the definite article DER in German can be either masculine nominative or else feminine dative, so that the appropriate entry can only be determined by considering the other words in the sentence.

Lastly, no xy-constraints are constructed during seed generation.

To refer back to our previous example, the first two rules in table 1.1 represent actual seed rules as learned by our system.

# 5 Seeded Version Space learning

After producing a seed rule for each sentence in the training corpus, the seeded version space learning algorithm is executed. The learning goal is to transform a set of seed rules into a set of final transfer rules. The transformation is done by merging rules, which in effect represents a generalization. Each rule is associated with a set of training sentences that it 'covers', i.e. is able to translate correctly. This is called the CSet of a rule. After seed generation, each transfer rule's CSet contains exactly one element, namely the index of the sentence that it was produced from. Note that it can happen that two (or more) sentences result in the same seed rule. In this case, only one copy of the rule is retained, and the CSet of the rule will now contain the indices of the two (or more) sentences that produced it. Merging rules results not only in a generalization, but also in a merging of the two CSets of the rules that are merged.

The algorithm can be divided into two steps: the first step groups sets of similar seed rules, where each group effectively defines a separate version space. The second step is then run on each version space separately. It involves exploring the search space that is delimited by a set of rules. The following sections explain both steps in detail.

## 5.1 Grouping Similar Rules

In order to minimize the search space, the learning algorithm should only (attempt to) merge two rules if there is a chance that they can be merged and produce a rule that will correctly cover the CSets of both unmerged rules. Otherwise, the algorithm is faced with an extremely large search space as

defined by part of speech combinations, feature combinations, etc. given in the training sentence rules.[1] Grouping rules into subsets limits this search space. There are a number of ways to do this. We chose to cluster the seed rules by three criteria: 1) the part of speech sequences, 2) the alignments, and 3) their type information. This type of grouping will ensure a high parallelity between the rules in each group, which itself simplifies merging.

From a theoretical standpoint, each such group of rules effectively defines a version space. The space's specific boundary is marked by all rules in the group. The general boundary can be seen as marked by a virtual transfer rule with the same part of speech sequences, alignments, and type information as the other rules in the group. Although this rule is never actually produced, it marks the most general rule that can be generalized to in this version space, as the seeded version space learning algorithm targets merely the x-, y-, and xy-constraints.

In the example in table 1.1, the two seed rules for 'THE MAN' and 'THE WOMAN' are grouped into one version space, as they share their part-of-speech sequences, alignments, their type.

## 5.2 Exploring the search space

Our seeded version space learning algorithm is based loosely on (Mitchell 1982). We view the space of transfer rules as organized in a partial ordering, where some rules are strictly more general than others, but not all rules can be compared directly. The goal of the seeded version space learning is to adjust the generality of rules. In principle, the seed rules are rules that are designed with one particular sentence in mind. The seeded version space learning algorithm, on the other hand, adjusts the rules so that they apply to as many sentences as possible without overgeneralizing. The goal of this process is to produce rules much like those a human grammar-writer would design.

At the moment, we view the rules that the seed generation algorithm produced as the most specific possible rules. Note, however, that this is not necessarily the case. The seed rules already present a generalization over sentences. In particular, they generalize to the part-of speech level. It can happen that words that are of the same part of speech behave differently. For instance, some French adjectives appear before, others after the noun they modify. In these cases, the algorithm will have to explore the space 'below' the seed rules, i.e. the more specific space. However, these issues are left for future investigation, and the current algorithm explores only the space that is more general than the seed rules.

The major goal of the seeded version space algorithm is then to generalize over specific feature values if this is possible. Consider the following

[1] Note that the search space is theoretically infinitely large, but is in practice restricted by the training sentences.

examples:

*Transfer Rule 1:*

...
$((X1\ AGR) = *3SING)$
$((X2\ DEF) = *DEF)$
...

*Transfer Rule 2:*

...
$((X1\ AGR) = *3SING)$
...

The above example illustrates that the second tranfer rule is more general than the first, as the first has more restrictive constraints. We shall now formalize the notion of partial ordering of transfer rules in our framework.

> **Definition:** A transfer rule $tr_1$ is strictly more general than another transfer rule $tr_2$ if all f-structures that are satisfied by $tr_2$ are also satisfied by $tr_1$. The two rules are equivalent if and only if all f-structures that are satisfied by $tr_1$ are also satisfied by $tr_2$.

Based on this definition, we can define operations that will turn a transfer rule $tr_1$ into a strictly more general transfer rule $tr_2$. More precisely, we have defined three operations:

1. Deletion of value constraint
   $((\Phi_i \gamma_m\ \alpha_j) = \beta_{kj}) \rightarrow NULL$
2. Deletion of agreement constraint
   $((\Phi_i \gamma_m\ \alpha_j) = (\Phi_i \gamma_n\ \alpha_j)) \rightarrow NULL$
3. Merging of two value constraints into one agreement constraint. Two value constraints can be merged if they are of the following format:
   $((\Phi_i\ \alpha_k) = \beta_{kl})$
   $((\Phi_j\ \alpha_k) = \beta_{kl})$
   $\rightarrow ((\Phi_i\ \alpha_k) = (\Phi_j\ \alpha_k))$
   For example, the two value constraints ((X1 AGR) = *3PLU) and ((X2 AGR) = *3PLU) can be generalized to an agreement constraint of the form ((X1 AGR) = (X2 AGR)).

Our version space learning algorithm is based on these three generalization operations. It explores the version space described by a set of rules.

## 5.3 Merging Two Transfer Rules

At the heart of the learning algorithm is the merging of two rules. Generalization is achieved by merging, which in turn is done based on the three generalization operations defined above.

Suppose we wish to merge two rules $tr_1$ and $tr_2$ to produce the most specific generalization of the two, stored in $tr_3$. The algorithm proceeds in three steps:

1. Insert all constraints that appear in $tr_1$ and $tr_2$ into $tr_3$ and subsequently eliminate them from $tr_1$ and $tr_2$.
2. Consider $tr_1$ and $tr_2$ separately. Perform all instances of operation 3 (see section 5.2 above) that are possible.
3. Repeat step 1.

Given the way the partial order is defined, this sequence of steps ensures that $tr_3$ does not have any constraints that would violate this partial order, and that it lacks no constraints. To return again to the example in table 1.1, the result of a merging operation can be seen. The first two colums contain the unmerged rules, the last column the merged rule.

The seeded version space learning algorithm is based on the merging of rules. It should be noted that any two rules in a given cluster can be merged syntactically using the elimination operations. So why not simply eliminate all constraints? The reason is that rules can become overly general. For instance, it can happen that the generation procedure needs a specific feature value to produce the TL sentence, and this value is not given in the rule. Therefore, a merge is only acceptable if the merged rule's CSet is a superset of the CSets of both unmerged rules. In the example in table 1.1, the system can verify that this is the case.

Furthermore, a merged rule should be checked for overgeneralization, meaning that the rules should not be so general that they produce spurios translations. In other words, it should be checked whether it produces only translations that are acceptable in some context. This is not currently done, but we plan to incorporate this idea using the bilingual informant.

### 5.4 The learning algorithm and evaluation of merged rules

The ultimate goal of the learning algorithm is to maximize coverage of a rule set with respect to the language pair, meaning that a rule set can correctly translate as many unseen sentences as possible. Currently, we estimate the coverage of a rule set by the generality of the rules. Since merges always provide a generalization of the two unmerged rules, we assume that they will increase coverage with respect to the language pair. Therefore, the learning algorithm seeks to minimize the size of the rule set. It iteratively merges two rules, until no more acceptable merges can be found and the rule set is as general as possible. Note again that a merge is only acceptable if there is no loss in coverage over the more specific rules, and if it produces no incorrect translations. When no more merges are found, the final rule set is output.

Another way of estimating the coverage of a rule set is cross-validation. In the context of cross-validation, it will be possible to relax the constraint that a merged rule has to cover all sentences that the unmerged rules cover, especially if it performs well over a test set.

## 6 Conclusions and Future Work

We have presented a novel approach to inferring transfer rules for a machine translation system. Our approach takes advantage of any knowledge that we have about the two languages, while we assume sufficient knowledge of the source language. We have implemented a prototype of the work described in previous sections. The prototype produces seed rules for elicited sentences. It then clusters the rules and performs the version space operations. The system is fully integrated with a specially engineered transfer engine, so that rules can be checked by actually translating them.

The previous sections have already hinted at some of our future work. In addition to the mentioned improvements, our short-term efforts will focus on the issue of compositionality. We wish to produce transfer rules that make use of previously produced transfer rules. For instance, a sentence-level transfer rule should be able to take advantage of the existence of a NP rule that covers its subject. We are currently exploring methods to incorporate such information in the seed generation. If we want our algorithm to scale to a more general setting, the addition of compositionality is of utmost importance.

We will then start to evaluate the performance of our system. We have run the system on some examples for the language pair English-German (noun phrases, adjective phrases, adverb phrases, sentences) with promising results. However, a more thorough and extensive evaluation is called for. We will evaluate the transfer rules themselves, and we will collect human judgements for translations that are produced by the transfer engine using the learned transfer rules. In particular, we first plan to evaluate the system on the high-density language pair English-German, and then to compare the performance to the pair English-Mapudungun. It will be interesting to see what the difference in performance will be when less information is available on the target language side. We will then explore methods to overcome this lack of information. The evaluation process itself will involve human judgements of test data translations.

Additionally, we wish to explore ways to refine the version space learning algorithm. In particular, we want to be able to specialize already created rules if this is found necessary. Our current algorithm does not provide the means to specialize from the seed rule towards the actual lexical items in the sentence.

## Bibliography


Bouquiaux, L. and J. M. Thomas (1992). *Studying and Describing Unwritten Languages*. xi. Dallas, TX, USA: The Summer Institute of Linguistics.

Brown, P. F., J. Cocke, S. D. Pietra, V. D. Pietra, F. Jelinek, J. Lafferty, R. Mercer, and P. Roossin (1990). A statistical approach to Machine Translation. *Computational Linguistics 16*(2), 79–85.

Brown, R. (1996). Example-based Machine Translation in the PANGLOSS system. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING-96)*, Copenhagen, Denmark, pp. 169–174.

Comrie, B. and N. Smith (1977). Lingua descriptive series: Questionnaire. *Lingua 42*, 1–72.

Frederking, R., A. Rudnicky, and C. Hogan (1997). Interactive Speech Translation in the DIPLOMAT project. In *Presented at the Spoken Language Translation workshop at the 35th Meeting of the Association for Computational Linguistics (ACL-97)*, Madrid, Spain.

Mitchell, T. (1982). Generalization as search. *Artificial Intelligence 18*, 202–226.

Nirenburg, S. (1998). Project Boas: A 'linguist in the box' as a multi-purpose language. In *Proceedings of the first Language Resources and Evaluation Conference (LREC-98)*, Granada, Spain.

Nirenburg, S. and V. Raskin (1998). Universal grammar and lexis for quick ramp-up of MT systems. In *Proceedings of the 17th International Conference on Computational Linguistics and the 36th Meeting of the Association for Computational Linguistics (COLING-ACL '98)*, Montréal, Canada, pp. 975–979.

Probst, K., R. Brown, J. Carbonell, A. Lavie, L. Levin, and E. Peterson (2001). Design and implementation of controlled elicitation for Machine Translation of low-density languages. In *Workshop MT2010, Machine Translation Summit VIII (MT-Summit-2001)*, Santiago de Compostela, Spain.

Probst, K. and L. Levin (2002). Challenges in automated elicitation of a controlled bilingual corpus. In *Proceedings of the 9th Interna-*

*tional Conference on Theoretical and Methodological Issues in Machine Translation (TMI-02)*, Kyoto, Japan, pp. 157–167.

Sherematyeva, S. and S. Nirenburg (2000). Towards a universal tool for NLP resource acquisition. In *The Second International Conference on Language Resources and Evaluation (LREC-00)*, Athens, Greece.

Somers, H. L. (1997). Machine Translation and minority languages. In *Translating and the Computer 19: Papers from the Aslib conference*, London, UK.

Tenni, J., A. Lehtola, C. Bounsaythip, and K. Jaaranen (1999). Machine learning of language translation rules. In *Proceedings of the 1999 IEEE Systems, Man and Cybernetics Conference (IEEE-SMC-99)*, Tokyo, Japan.