Elicitation-Based Learning of Syntactic Transfer Rules for Machine Translation for Resource-Poor Languages

Katharina Probst, Lori Levin, Erik Peterson, Alon Lavie, Jaime Carbonell
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA, U.S.A.

January 17, 2003

Abstract. Elicitation-based Machine Translation is an approach to MT that can be applied to language pairs where neither large bilingual corpora nor other NLP resources are available. In our approach, a bilingual speaker translates and aligns a small elicitation corpus that serves as training data for a rule learning system. The elicitation corpus is designed to cover major linguistic phenomena, so as to make the training corpus as diverse as possible. We assume a resource-rich elicitation language for which resources such as a grammar and a parser are available. The rule learning system then uses the syntactic analysis of the sentence in the resource-rich language, as well as any information given in the other language, to infer syntactic *seed* transfer rules. The seed rules are generalized to increase their coverage of unseen examples, using the bilingual elicitation corpus for validation. Rule generalization is accomplished by a learning mechanism, Seeded Version Space Learning, which similar in nature to Version Space Learning. The approach bases on a partial ordering of transfer rules, which is defined in terms of their generality. The seed rules provide an informed starting point, thus encouraging fast convergence.

Keywords: elicitation, learning, syntactic transfer rules, minority languages

1. Introduction and Motivation

In recent years, much of machine translation research has focused on two issues: portability to new languages, and developing machine translation systems rapidly. Since human expertise on uncommon languages may be in short supply, and human development time can be lengthy, automated learning of statistics or rules has been critical to both language portability and rapid development. Automated methods have typically been trained on large, uncontrolled parallel corpora (Brown et al., 1990; Brown et al., 1993; Brown, 1997; Och and Ney, 2002; Yamada and Knight, 2001; Papineni et al., 1998; Vogel and Tribble, 2002). However, a minority of projects (Nirenburg, 1998; Sherematyeva and Nirenburg, 2000; Jones and Havrilla, 1998) have addressed automated learning of translation rules from a controlled corpus of carefully elicited sentences.

This article presents an elicitation-based approach to learning transfer rules for machine translation. The input to the rule learning mechanism is a parallel corpus of elicited sentences, where each sentence pair is word-aligned (allowing multiple and zero alignments). The learning mechanism is a new technique called Seeded Version Space Learning (SVSL). SVSL is based on Version Space Learning (Mitchell, 1982), a machine learning method that is designed to find the most general hypothesis that is consistent with the traning data, in a hypothesis space that is partially ordered by an approporiate generalization function. In the case of translation rules, hypotheses are annotated with many feature constraints (such as 'the subject of the sentence is in the singular'), which constrain their applicability to specific sentences. More general hypotheses, generalize the feature constraints in an effort to make the rule applicable to a large set of sentences. For example, a more general constraint may be that the subject and main verb of a sentence agree in number, regardless of what the number is. The goal of the learning process is to discover the appropriate level of generalization for hypothesized transfer rules. Rules that are too general will mis-translate sentences to which they should not apply. Rules that are too specific will not apply to many sentence examples that were not observed in training, resulting in poor translation performance. To make the approach more tractable, we use seed rules as a starting point. A seed rule is created automatically from a bilingual word-aligned phrase or sentence and a lexicon. Generalizations are hypothesized by comparing seed rules and are confirmed by checking the translations of the elicitation sentences to which the rules apply.

The work described here is part of the AVENUE project at Carnegie Mellon University.[1] The overall goal of AVENUE is to research methods for machine translation that are suitable for languages with scarce electronic resources, and possibly also a scarcity of humans trained in linguistic analysis. AVENUE aims to develop "omnivorous" machine translation techniques that can take advantage of whatever resources are available. This includes using large unconstrained corpora when they are available and eliciting controlled corpora for training data.

This article is organized as follows: First, we present the transfer rule formalism and briefly the transfer engine. Also, we describe and show the interface that is used for eliciting data. We continue by discussing the general design of the elicitation corpus, followed by an in-depth treatment of our rule learning algorithms, including preliminary evaluation results. Further, we discuss advantages and disadvantages of learning from a controlled, elicited corpus and list several challenges

[1] NSF grant number IIS-0121-631

that we face in our work. We conclude by mentioning several areas of future investigation.

2. The Transfer Rule Formalism and Interpreter

As was already mentioned above, our system uses a bilingual word-aligned corpus as training data. The output of the SVSL system is a set of unification-based syntactic transfer rules for machine translation. This section describes the transfer rule formalism and transfer rule engine. Transfer engines typically have separate rules for the three traditional stages of MT: analysis, transfer, and generation. The AVENUE rule formalism combines the information for all three into a single rule representation. The AVENUE engine consequently is similar to the 'modified' transfer approach used in the early METAL system (Hutchins, 1992).

Like most modern translation sytems, the AVENUE transfer engine maintains a strict separation between the transfer algorithms and linguistic data. The design of the transfer rule formalism itself was guided by the consideration that the rules must be simple enough to be learned by an automatic process, but also powerful enough to allow manually-crafted rule additions and changes to improve the automatically learned rules.

The following list summarizes the components of a transfer rule. In general, the x-side of a transfer rules refers to the source language (SL), whereas the y-side refers to the target language (TL).

- **Type information:** This identifies the type of the transfer rule and in most cases corresponds to a syntactic constituent type. Sentence rules are of type `S`, noun phrase rules of type `NP`, etc. The formalism also allows for SL and TL type information to be different.

- **Part-of speech/constituent information:** For both SL and TL, we list a linear sequence of components that constitute an instance of the rule type. These can be viewed as the 'right-hand sides' of context-free grammar rules for both source and target language grammars. The elements of the list can be lexical categories, lexical items, and/or phrasal categories.

- **Alignments:** Explicit annotations in the rule describe how the set of source language components in the rule align and transfer to the set of target language components. Zero alignments and many-to-many alignments are allowed.

```
{S,2}  ; Rule identifier
S::S : [NP VP MA] -> [AUX NP VP]
(
   (x1::y2) ; set constituent alignments
   (x2::y3)

   ; Build Chinese sentence f-structure
   ((x0 subj) = x1) ; NP supplies subj. f-structure
   ((x0 subj case) = nom)
   ((x0 act) = quest) ; speech act is question
   (x0 = x2)

   ((y1 form) = do) ; base form of AUX is "do"

   ((y3 vform) =c inf) ; verb must be infinitive
   ((y1 agr) = (y2 agr)) ; "do" must agree with subj.
)
```

Figure 1. Sample Transfer Rule

- **X-side constraints:** The x-side constraints provide information about features and their values in the source language sentence. 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 guide and constrain the generation of the target language sentence.

- **XY-constraints:** The xy-constraints provide information about which feature values transfer from the source into the target language. Specific TL words can obtain feature values from the source language sentence.

For illustration purposes, Figure 1 shows an example of a transfer rule for translating Chinese *yes/no*-questions into English. Chinese *yes/no*-questions use the same word order as declarative sentences, but end in the particle *ma*. The transfer rule shown in Figure 1 adds the auxiliary verb *do*, makes it agree with the subject, and constrains the `VP` to be infinitive.

In Figure 1, the notation `S::S` indicates that a sentence (dominated by an `S` node) on the source side translates into another sentence in the TL. The production rules (constituent sequences) are `[NP VP MA] ->`

[AUX NP VP]. The remaining portion of the transfer rule specifies alignments and constraints. Following a basic unification-based approach, we assume that each constituent structure node may have a corresponding feature structure. The feature structure holds syntactic features (such as number, person, and tense) so that they can be checked for agreement and other constraints. The feature structure may also specify the grammatical relations that hold between the nodes.

The feature unification equations used in the rules follow the formalism from the Generalized LR Parser/Compiler (Tomita, 1988). The x and y variables used in the alignments and constraints correspond to the feature structures of the elements of the production rules. x0 is the feature structure of the source language parent node S. y0 is the feature structure of the target language parent node S. The source constituents are referred to with x followed by the constituent index, starting at one. Thus, NP is referenced as x1, VP as x2, and MA is referenced as x3. On the target side the production rule order is AUX NP VP, referred to with y1, y2, and y3, respectively.

The body of the rule begins with one or more constituent alignments that describe which constituents on the source side should transfer to which constituents on the target side. Not all constituents need to have an alignment to the other side. A double-quote-enclosed string may also be used on the target side, directing the transfer engine to always insert this exact word into the translation in relative position to the rule's other target constituents.

The constituent alignments are followed by a set of feature unification constraints, to be used in each of the analysis, transfer, and generation stages. Transfer unification equations are generally divided into three types: x-side constraints, which refer to source language constituents and are used during analysis; xy-constraints, used during transfer to selectively move feature structures from the source to the target language structures; and y-side constraints, used during generation to distribute target language feature structures and to check agreement constraints between target constituents.

In a further distinction, x-side and y-side constraints can be either *value constraints* or *agreement constraints*. Value constraints are of the format $(X_n\ attribute) = value))$ or $(Y_n\ attribute) = value))$, while agreement constraints are of the format $((X_n\ attribute) = (Y_m\ attribute))$. Agreement constraints can also be between two x-incides (xx-constraints) or two y-indices (yy-constraints). An example of a value constraint is ((X2 AGR) = *3PLU), where an agreement constraint example is ((X2 AGR) = (X3 AGR)). This distinction is used during learning. Initial seed hypotheses contain value constraints, which are then generalized to agreement constraints where possible.

Since the transfer rules are comprehensive (in the sense that they incorporate parsing, transfer, and generation information all in the rules), the set of rules that results from the learning process in effect represents a transfer grammar. The transfer engine uses it exclusively to translate, and no additional parsing and/or generation grammars are available. Hence, the transfer rules contain a myriad of information, including feature information about its constituents.

In many systems, transfer rules contain merely the mapping, i.e., alignment, between the source and target language constituents. This is the obvious choice if an additional parsing and an additional generation grammar are available. Our system, however, learns the parsing, transfer, and generation grammars at once, encoding all the necessary information in each rule. There are two main reasons for this. First, the rules are learned from bilingual sentence pairs, so that it makes intuitive sense to build a transfer rule for each sentence pair. The second reason is the discrepancy between the amount of information available on the source language and the target language sides. The less information is available in the TL, the more assumptions are made, using the SL information. For instance, if no number information is available for the TL, we assume that it is the same as in the SL. Our transfer rules are thus specifically designed for the circumstances as well as for the learning task at hand.

The fact that each transfer rule contains parsing, transfer, and generation information should however not be confused with the way translation is implemented. The transfer engine does indeed first parse the sentence, then transfer and generation are achieved in an integrated pass. In other words, while each transfer rule jointly represents parsing, transfer, and generation constraints, the transfer engine keeps a strict distinction between them (Peterson, 2002).

3. Elicitation Interface

AVENUE's machine learning mechanism takes as input pairs of word-aligned sentences in the source and target languages. In order to obtain this information, a bilingual informant is presented with a sentence in English or Spanish (or another resource-rich language) and is then asked to translate the sentence to the language of interest using the Elitication Interface. The interface contains a context field, providing the possibility to include additional information about the sentence (such as the gender of one of the participants) that may influence the translation. After translating the sentence, the informant then indicates for each word in the elicited sentence the corresponding word(s)

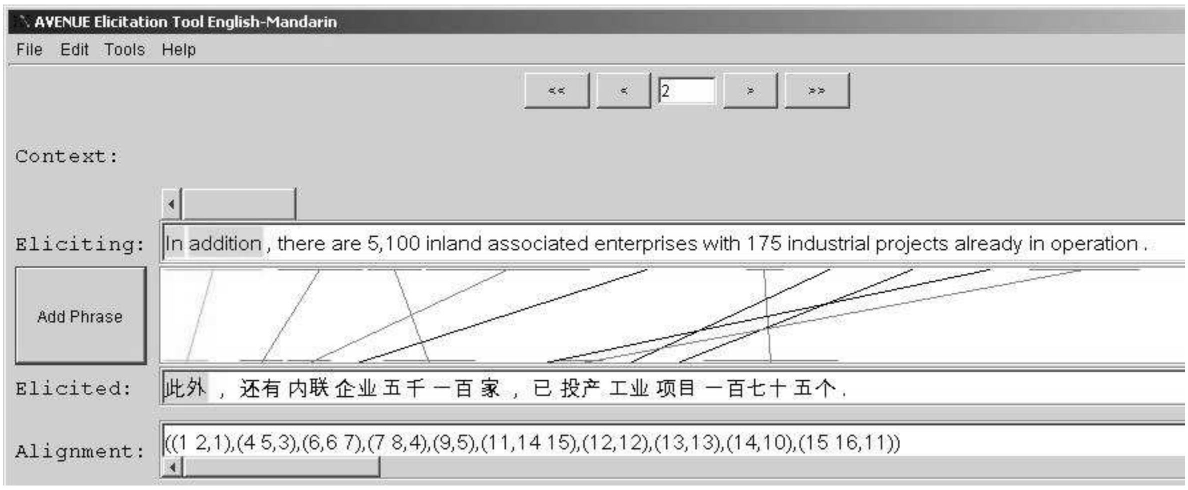

Figure 2. Elicitation Interface

in the elicitation language. In some cases, the word may not have a corresponding word in the matching sentence, or may correspond to multiple words.

If the informant wants to include some information or clarifications about the translation or the alignment of the sentence pair, the interface provides a comment field in which to write these comments. Once the informant has finished with one sentence pair, clicking on the ">" button will bring up the next sentence to translate and align. Users can also go back to change previous sentences or type in the index number for a particular sentence to jump directly to it.

Alignments are annotated by using the mouse to click on a word in one sentence and then the corresponding word in the other sentence. A link is automatically added, which is visually indicated with a line between the two words and color-highlighting of the newly aligned words. Deletion of an existing alignment pair is equally straightforward: the informant clicks on the first word of the existing pair and then clicks on the second word. The link is then deleted. In the case where two or more (possibly non-contiguous) words act as a unit, the informant can hold down the Control key while clicking on the words with the mouse. In this way, one-to-many and many-to-many word alignments are possible.

The language in the menus and buttons of the interface, as well as the help file, can be easily changed without recompiling the entire program. Currently, the interface has been localized for English and Spanish.

Beside the basic function of translating and word-aligning sentences, the elicitation interface also has several helpful support tools. From the main menu the user can display a bilingual glossary built from existing word alignments. Users can also easily search for words in the eliciting or elicited sentences. The search results include the sentence the word was found in, along with any aligned words. Clicking on a result links directly to the sentence. This makes it easier for users to be consistent in their alignment work.

4. General Design of the Elicitation Corpus

The elicitation corpus is a list of sentences in a major language such as English or Spanish. It is designed to cover major linguistic phenomena in typologically diverse languages. In realizing that it is a hard undertaking to be typologically complete, we start by eliciting simple structures that are likely to be present in a wide variety of languages, such as noun phrases and basic transitive and intransitive sentences. Currently, the source language is a major language such as English or Spanish. The target language can in principle be any language.

The current pilot elicitation corpus has about 2,000 sentences (in English, some of which have been translated into Spanish), though we expect it to grow to at least the number of sentences in (Bouquiaux and Thomas, 1992), which includes around 6,000 to 10,000 sentences, and to ultimately cover most of the phenomena in the Comrie and Smith (Comrie and Smith, 1977) checklist for descriptive grammars. The phenomena covered in the pilot corpus include number, person, and definiteness of noun phrases; case marking of subject and object; agreement with subjects and objects; differences in sentence structure due to animacy or definiteness of subjects and objects; possessive noun phrases; and some basic verb subcategorization classes.

The elicitation corpus has two organizational properties: it is organized into minimal pairs; and it is compositional, i.e. starting with smaller phrases and combining them into larger phrases. In the future, we also plan for it to be dynamic, so that the elicitation takes a different path depending on what features have been found so far.

A minimal pair can be defined as two sentences that differ only in one linguistic feature. Each sentence can be associated with a feature vector. Consider, for instance, the following three sentences:

1. The man saw the young girl.
2. The men saw the young girl.
3. The woman saw the young girl.

The first and the second sentence differ merely in one feature, namely the number of the subject. Thus, they represent a minimal pair. However, sentence one and three also form minimal pair. They differ only in the gender of the subject.

The reason that we decided to organize the corpus in minimal pairs is specific to our use of the elicited data. One component of the transfer rule learning is feature detection. If we define linguistic phenomena (such as number, gender, subject-verb agreement, etc.) as a set of features, we want to determine what subset of these features is instantiated in any given language. This is achieved by defining the features such that they can be tested using a minimal pair of sentences. For instance, if we want to find out whether target language verbs inflect depending on the number of the subject, we can use sentences 1 and 2 above: they differ only in the number of their subject, while all other features are kept constant. An indication of whether the target language verbs agree in number with their subjects can then be found by to comparing the verbs in the elicited target language versions of sentences 1 and 2.

However, we have to careful to make the opposite conjecture. If the two verbs do not differ, then this could be an indication that target language verbs do not agree with their subjects in number. However, it could also mean that the verb used for comparison represents an exception to a more common rule. We approach this problem in two ways. First, we never rule out the existence of a feature based on only one example (i.e., one minimal pair). Second, the results from feature detection are interpreted not as absolutes, but as tendencies, which guide our search through the space of possible transfer rules and give more preference to those rules that use features that were detected over those that use features that were not detected. Feature detection is not yet integrated with the rest of the learning module. A detailed discussion is beyond the scope of this paper, but can be found in (Probst et al., 2001). The second design principle, compositional build-up, was chosen in order to aid the learning module. The system first learns flat, unstructured rules. Structure can be added by using previously learned rules for simpler constituents. This principle will be discussed in detail below.

5. The Rule Learning System

In this section we introduce the rule learning system for learning syntactic transfer rules from elicitated examples. Commonly, rule-based systems are developed by human experts who write grammar rules that map structures from the source onto the target language. Our goal is to automate this process, to whatever extent possible. The reader is reminded that in the current state of the system, the source language is the resource-rich language (such as English), whereas the target language is the resource-poor language. We have begun investigating the reversal of translation direction with some success. The results will be reported in a future publication.

The system architecture (Figure 3) is divided into a learning module and a run-time translation module. The elicitation process, drawing from user input as well as the elicitation corpus, falls into the learning module. The other major part of the learning module is the rule learner itself, using the output of the elicitation process as training data. The rule learner interacts frequently with the transfer engine, as will become clearer in the discussion on how the rules are learned. The output of the rule learner is a set of rules which during run-time serves as the transfer grammar.

At run time, the translation module translates a source language sentence into a target language sentence. This is achieved by the transfer engine using the output of the rule learning module, the learned transfer grammar.

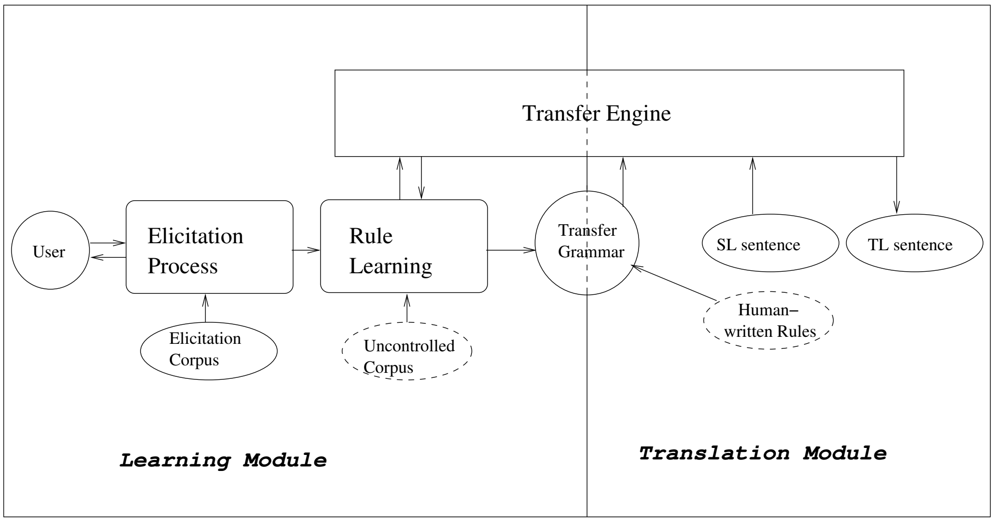

Figure 3. AVENUE's rule learning and translation system architecture

Figure 3 further shows two optional sources of information. During the learning phase, the rule learning system can also use an uncontrolled corpus. Further, the learned transfer grammar can at any point be supplemented by human-written rules.

Learning from elicited data proceeds in three main stages: the first phase, Seed Generation, produces initial 'guesses' at transfer rules. The rules that result from Seed Generation are 'flat'—they transfer linear sequences of parts of speech. This is clearly not desirable for syntactic transfer rules. For this purpose, the second phase, Compositionality Learning, adds structure using previously learned rules. For instance, in order to learn a rule for a PP, it is useful to recognize the NP that is contained in it, and to reuse NP rules that were learned previously. The third stage of learning, Seeded Version Space Learning, attempts to increase the generality of the rules. The rules that are learned during Seed Generation and structuralized during Compositionality Learning are specific to the sentence pair from which they were derived. In order to make the rules applicable to a wider variety of unseen examples at run-time, Seeded Version Space Learning attempts to automatically generalize them. The following sections explain all three stages in detail.

5.1. 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 SL sentences are parsed and disambiguated in advance, so that the system has access to a correct feature structure (f-structure) and a correct phrase or constituent (c-)structure for each source 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. The other extreme is the absence of any information, in which case the system makes assumptions about what linguistic feature values can be assumed to transfer from the source language to the target language. In such a case we assume, for example, that a plural noun will be translated as a plural noun.

With the information available, how can we construct a transfer rule of the format described in Section 2? 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, while 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. 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.

For y-side constraints, we make a similar assumption as with parts of speech: in absence 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. Some features, like gender (e.g., masculine, feminine, neuter) and noun class (e.g., Bantu nominal classifiers or Japanese numerical classifiers), will not generally have the same values in different languages. The Japanese classification of a window as a sheet, like paper, will not be relevant to a Bantu language that classifies it as a manmade artifact, or to French, which classifies it as feminine. Other features may tend to have the same values across languages. For example, if a language has plural nouns in common usage, it will probably use one in the translation of *The children were playing*, because using a singular noun (*child*) would change the meaning. The decision of which features to transfer is necessarily based on heuristics that are less than perfect. For example, we will transfer past tense for a sentence like *The rock fell* because it is part of the meaning, even though we are aware that tense systems vary widely and that English past tense will not always correspond to past tense in another language.

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 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. This arises from the fact that the seed rules are quite specific to the sentence pair that they are produced from. In some cases, a y-side constraint could equivalently also be expressed as an xy-constraint. The difference does, however, become apparent during Seeded Version Space Learning. From a procedural point of view, y-side constraints are more specific than the equivalent xy-constraints. For instance, the constraints `((x1 num) = pl) and ((y2 num) = pl)` are, on the surface, equivalent to `((x1 num) = pl) ((y2 num) = (x1 num))`. There is, however, a difference: in the first case, changing `((x1 num) = pl))` to `((x1 num) = (*OR* sg pl))` does not have any impact on the value of `y2`'s number, but in the second case it does. As Seeded Version Space Learning manipulates constraints in such a fashion, there is actually a difference between the two forms. As the goal is to start Seeded Version

Space Learning at a very specific level, we produce y-side constraints rather than xy-constraints. An example of a seed rule can be found in the left-most column in Table I.

After a seed rule is produced, it is added to a tentative translation grammar. Using this rule, the transfer engine then tries to translate the bilingual sentence pair that the rule was derived from. Only if the rule can be used to translate the pair correctly (if the rule *covers* the sentence pair) is the rule acceptable and is passed on to further learning steps.

5.2. Compositionality

In order to scale to more complex examples, the system next learns compositional rules: when producing a rule e.g. for a sentence, we can make use of sub-constituent transfer rules already learned.

Compositional rules are learned by first producing a flat rule in Seed Generation. Then the system traverses the c-structure parse of the SL sentence. Each node in the parse is annotated with a label such as `NP`, which is the root of a subtree. The system then checks if it has learned a lower-level transfer rule that can be used to correctly translate the words in the source language subtree.

The left-most column in Table I contains the flat seed rule for the bilingual sentence pair *The highly qualified applicant did not accept the offer.—Der äußerst qualifizierte Bewerber nahm das Angebot nicht an.*. In order to add structure to the flat rule, the system performs a top-down traversal of the c-structure tree. For each node, such as `S`, `NP`, `NBAR`, etc., it extracts the part of the sentence that is rooted at this node. For example, in the following part of a c-structure tree, the top node `NP` covers the chunk *the highly qualified applicant*.

```
(<NP> (<DET> ((ROOT *THE)) THE)
(<NBAR>
(<ADJP> (<ADVP> (<ADV> ((ROOT *HIGHLY)) HIGHLY))
(<ADJP> (<ADJ> ((ROOT *QUALIFIED)) QUALIFIED)))
(<NBAR> (<N> ((ROOT *APPLICANT)) APPLICANT))))
```

The question is now if there already exists a learned transfer rule that can correctly translate this chunk. Before this can be determined, the system first needs to

1. find the TL reference translation that corresponds to the SL chunk. For this information, the system consults the user-specified alignments and the given target language translation.
2. find the category of the rule. This is obtained simply from the label of the top node of the sentence chunk. In this example, the

category would be NP. Compositionality is only added to the rule if a previously learned rule *of the desired type*, (here NP) can account for the bilingual chunk. This is done in order to preserve the integrity of the original c-structure.

The next step is to call the transfer engine to translate the chunk *the highly qualified applicant*. The transfer engine returns zero or more translations together with the f-structures that are associated with each translation. The f-structures are stored as constraints of the same format as the y-side constraints in the transfer rules. In case there exists a transfer rule that can translate the sentence chunk in question, the system takes note of this and traverses the rest of the c-structure, excluding the subtree that corresponds to the chunk. The goal of traversing the c-structure is to find the *maximal* subtrees that can be translated by lower-level rules, hence the top-down traversal.

After traversal is completed and one or more compositional sub-constituents have been found, the initial flat seed rule must be revised. Table I presents an example of how the flat rule is modified to reflect the existence of an NP rule that can translate the chunk *the highly qualified applicant*. The flat transfer rule (left-most column) and the lower-level rule (middle column) are combined to form the structured rule that can be seen in the right-most column. Several adjustments are made to the flat seed rule: First, the part-of-speech sequence from the flat rule is turned into a constituent sequence on both the SL and the TL sides, where those chunks that are translatable by lower-level rules are represented by the category information of the lower-level rule, in this case NP. The alignments are adjusted to the new sequences. Lastly, the constraints must be changed. The x-side constraints are mostly retained (with the indices adjusted to the new sequences). However, those constraints that pertain to the sentence/phrase part that is accounted for by the lower-level rule are eliminated, as can be seen in Table I. Note that for the rules in the two rightmost colums no specific sentences can be given since the rules are compositional and thus must have been learned from a combination of at least two training examples.

Finally, the y-side constraints are adjusted. For each new sub-constituent that was detected, the transfer engine may return multiple translations, some of which are correct and some of which are wrong. The compositionality module compares the f-structures (constraints) of the correct translation and the incorrect translations. This is done so as to determine what constraints need to be added to the higher level rule in order to produce the correct translation in context. For each constraint in the correct translation, the system checks if this constraint appears in *all* other translations. If this is not the case, a new constraint is

Table I. Example of Compositionality

Uncompositional rule	**Lower-level Rule**	**Compositional Rule**
;;SL: *The highly qualified applicant*		
did not accept the offer.		
;;TL: *Der äußerst qualifizierte*		
Bewerber nahm das Angebot nicht an.		
S::S	NP::NP	S::S
[det adv adj n aux neg v det n]	[det ADJP n]	[NP aux neg v det n]
[det adv adj n v det n neg vpart]	[det ADJP n]	[NP v det n neg vpart]
(;;alignments:	;;alignments:	;;alignments:
(x1:y1)	(x1::y1)	(x1::y1)
(x2::y2)	(x2::y2)	(x3::y5)
(x3::y3)	(x3::y3)	(x4::y2)
(x4::y4)	(x4::y4)	(x4::y6)
(x6::y8)		(x5::y3)
(x7::y5)		(x6::y4)
(x7::y9)		
(x8::y6)		
(x9::y7)		
;;x-side constraints:	;;x-side constraints:	;;x-side constraints:
((x1 def) = *+)	((x3 agr = *3-sing))	((x2 tense) = *past)
((x4 agr) = *3-sing)		
((x5 tense) = *past)		
;;y-side constraints:	;;y-side constraints:	;;y-side constraints:
((y1 def) = *+)	((y3 agr) = *3-sing)	((y1 def) = *+)
((y3 case) = *nom)		((y1 case) = *nom))
((y4 agr) = *3-sing))		

constructed and inserted into the compositional rule. Before simply inserting the constraint, however, the indices need to be adjusted to the higher-level constituent sequence.

5.3. Seeded Version Space Learning

Seeded Version Space Learning follows the construction of compositional rules. The learning goal is to transform the set of seed rules into more general transfer rules. This 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. When two rules are merged into one, their CSets are also 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.

5.3.1. *Grouping Similar Rules*

In order to minimize the search space, the learning algorithm should only merge two rules if there is a chance that the resulting rule will correctly cover the CSets of both unmerged rules. We group together rules that have the same 1) type information, 2) constituent sequences, and 3) alignments. The rules in a group may have different feature constraints, for example some may have singular nouns and some may have plural nouns. Merges are only attempted within a group, thus simplifying the merging process.

From a theoretical standpoint, each such group of rules effectively defines a version space. Each version space has a boundary of specificness (the specific boundary) marked by the seed rules, and a boundary of generality (the general boundary), marked by a virtual rule with the same part of speech sequences, alignments, and type information as the other rules in the group, but no constraints. The goal of version space learning is to pinpoint the right level of generality between the specific boundary and the general boundary.

5.3.2. *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 partial order is defined almost identically to the definition of subsumption in formalisms such as HPSG (Shieber, 1986). The goal of the Seeded Version Space Learning is to adjust the generality of rules. If the process is successful, the resulting rules should be 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. However, the seed rules already present a generalization over sentences, generalizing to the part of speech or constituent 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. The issue is left for future investigation.

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

Transfer Rule 1:	*Transfer Rule 2:*
...	...
$((X1\ AGR) = *3SING)$	$((X1\ AGR) = *3SING)$
$((X2\ DEF) = *DEF)$	
...	...

Here, 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
   $(X_n\ attribute) = value))$ or $(Y_n\ attribute) = value)) \rightarrow NULL$
2. Deletion of agreement constraint
   $((X_n\ attribute) = (Y_m\ attribute)) \rightarrow NULL$
   (similarly for an xx-agreement constraint or a yy-agreement constraint.)
3. Merging of two value constraints into one agreement constraint. Two value constraints can be merged if they are of the following format:
   $(X_n\ attribute) = value1))\ (X_m\ attribute) = value2))\ ((X_n\ attribute) = (X_m\ attribute))$
   (similarly for an x- and a y-constraint or two y constraints.)
   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)).

5.3.3. *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 both 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 5.3.2 above) that are possible.

3. Repeat step 1.

After performing the merge, the system uses the new transfer rule, tr_3 to translate the CSets of tr_1 and tr_2. The merged rule, tr_3, is accepted if it can correctly tranlate all of the items in both CSets. In the example in Table II, the system can verify that this is the case.[2] 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 to the example in Table II, the result of a merging operation can be seen. The first two columns contain the unmerged rules, the last column the merged rule.

5.3.4. *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. Cross-validation as an alternative evaluation metric is currently under investigation.

5.4. Preliminary Experimental Results

The above algorithms have been implemented and run on a corpus of 141 noun phrases and sentences. The training set consists of structurally relatively simple noun phrases and short sentences. Two examples from the training set are "the very tall man" and "The family ate dinner." The training corpus was translated into German and word-aligned by hand. A fully inflected target language dictionary as described above was used. No other target language information was given. This section presents preliminary results that indicate that Seed

[2] Note that in the example, the source and target language sentences are not given with the rule in the middle column. The reason again is that the rules are compositional, so that they must have been learned from at least two training examples. However, the example is given in reference to the previous example, where are rule was learned from *The highly qualified applicant did not accept the offer*, which translates into *Der äußerst qualifizierte Bewerber nahm das Angebot nicht an.*.

Table II. Example of Version Space Generalization

Seed Rule 1	**Seed Rule 2**	**Generalized Rule**
S::S	S::S	S::S
[NP aux neg v det n] →	[NP aux neg v det n] →	[NP aux neg v det n] →
[NP v det n neg vpart]	[NP v det n neg vpart]	[NP n det n neg vpart]
(;;alignments:	(;;alignments:	(;;alignments:
(x1::y1)	(x1::y1)	(x1::y1)
(x3::y5)	(x3::y5)	(x3::y5)
(x4::y2)	(x4::y2)	(x4::y2)
(x4::y6)	(x4::y6)	(x4::y6)
(x5::y3)	(x5::y3)	(x5::y3)
(x6::y4)	(x6::y4)	(x6::y4)
;;constraints:	;;constraints:	;;constraints:
((x2 tense) = *past) ...	...	...
((x2 tense) = *past)	((x2 tense) = *past)	((x2 tense) = *past)
((y1 def) = *+)	((y1 def) = *+)	((y1 def) = *+)
((y1 case) = *nom)	((y1 case) = *nom)	((y1 case) = *nom)
...	...	...
((y4 agr) = *3-sing))	((y4 agr) = *3-plu))	((y4 agr) = (y3 agr)))

Generation, Compositionality, and Seeded Version Space Learning can indeed effectively be used to infer transfer rules, and the resulting transfer rules can be used to translate, albeit not without making mistakes.

We report results for both training set accuracy (i.e. the system was trained on all 141 examples and evaluated on all of them), and for each split of a 10-fold cross-validation. To evaluate translation power, we used an automatic evaluation metric. Namely, we simply compared the resulting translations against the reference translation. If the grammar could be used to produce the exact reference translation, the sentence counts as correctly translated. No partial matching was done. Note that this is a very harsh evaluation metric - it is well known that a sentence can correctly be translated into many different surface forms in the target language. For this reason, MT evaluation generally performs user studies or matches partial outputs to reference translations. In this test, our system was graded on a harder scale, while in future evaluations, we will consider other metrics as well.

Further, our system considers a sentence as correctly translated if one of the possible translations matches the reference translation. This set accuracy measure should be judged in conjunction with the number

Table III. Translation Set Accuracy and Number of Possible Translations for the Training Set as well as 10 Cross-validation Sets.

Evaluation Set	**Translation Accuracy**	**Ave Num Possible Translations**
Training	0.716	4.936
TestAve	0.625	4.460
CVSplit1	0.4	2.533
CVSplit2	0.714	7
CVSplit3	0.714	3.286
CVSplit4	0.643	6.643
CVSplit5	0.714	2.357
CVSplit6	0.786	4.857
CVSplit7	0.786	3.214
CVSplit8	0.5	5.643
CVSplit9	0.429	2.929
CVSplit10	0.571	6.143

of possible translations per sentence (see Table III). For instance, if the grammar can produce the reference translation, but only as one of a hundred possible translations, then the system may still perform poorly, as it may not pick as its top translation the perfect translation. The current version of the transfer engine does not rank outputs. Therefore, it is important to report the average number of possible translations per rule in order to put into perspective the translation (set) accuracy. The above are the two metrics with the highest priority: the system will always be measured by its translation accuracy (which is desired to be high) and by the average number of possible translations per sentence (which is desired to be low). The results of this evaluation can be seen in Table III below.

Because of the simplicity and small size of the training set, it is clear that much work remains to be done, both in refining the algorithms and making them scale to more complex examples and to bigger sets of data. The initial results, however, are promising and suggest that future exploration will very much be worthwhile.

6. Controlled Elicitation vs. Learning from Uncontrolled Corpora?

In our work, we are interested in what we call a 'resource-rich/resource-poor situation': for one of the languages in the translation pair, the resource-rich language, linguistic and language-technological tools are easily available. Such tools include monolingual dictionaries, part-of-speech taggers, morphological analyzers, parsers and grammars. For the other language, such resources are not available, or only to a limited extent. This situation is not uncommon: while extensive language technologies are available for languages such as English, Spanish, and Japanese, other languages such as Croatian, Bulgarian, or indigenous languages have been neglected. Thus, when translating between a resource-rich language such as English and a resource-poor language such as Mapudungun (an indigenous language spoken in Southern Chile and Argentina (Levin et al., 2002)), dealing with the imbalance between the information available on each language side becomes the highest priority.

The decision of how to build an MT system in such a situation will depend on the training data that is available. In particular, if a large bilingual corpus is available, one could use it to infer statistics, example-bases, or, as in our approach, transfer rules. If no such corpus is available (a common case for unusual language pairs like Spanish-Mapudungun), we can elicit a small, controlled corpus and use it as structured training data. Both forms of training data have their advantages and disadvantages. In the above sections, we have already described how learning is facilitated in our system if the training data is a small, but structured, corpus. Ideally, our system will be trained on both unstructured and structured data. The two types of data are sufficiently different, so that we hope that the learned rules will supplement each other.

The existence of a large bilingual corpus brings about a number of advantages. One appeal of an unstructured training corpus is that no elicitation corpus needs to be designed if it does not yet exist. The design of a corpus that covers major typological phenomena is a non-trivial and time-consuming task. However, once the elicitation corpus is designed, it can easily be applied to new language pairs.

Learning from a large, unstructured training corpus has the further advantage that quantitative statements about the language can be made. The elicitation corpus aims at covering all kinds of linguistic phenomena, while keeping the corpus small. Therefore, each single phenomenon does not receive attention beyond a few sentences. If one of the sentences provides an exception to a rule, the system will learn two different transfer rules, but it has no way of knowing which rule should be the default. In other words, the design of the elicitation corpus inherently violates the distribution of rules in a given language pair. If learning from a large corpus, the distribution is preserved, and can possibly be extracted and used by the learning algorithm.

In the absence of a large bilingual corpus, we have chosen to obtain training data via controlled elicitation. This means that we obtain translated text from a bilingual user – however, we carefully construct the sentences that the user will translate in order to optimize the training data for our system. In particular, the corpus can aim at covering major typological phenomena or even at being typologically complete. Any unstructured training corpus will lack examples of various linguistic phenomena, with the problem becoming more prevalent the smaller the corpus is. This serious data sparseness problem can be overcome with a linguistically inspired corpus.

The elicitation corpus is further designed to be applicable for any language. This makes it a reusable resource that facilitates the quick ramp-up of an MT system for a new language pair, given that one of the languages is a language that the elicitation corpus is available in. (This is in line with our assumption of a resource-rich/resource-poor situation). Not only can we reuse the sentences and phrases that the corpus consists of, we can also parse and disambiguate them once and then reuse this information for any new resource-poor language. The above technique results in time savings for porting to a new language pair. Furthermore, the information that was encoded in the corpus on the resource-rich language side is less noisy than if it was automatically obtained (such as by parsing unstructured training data). As was said above, we parse all training elements in the corpus once and hand-disambiguate them, so that the system has access to the correct parse. We also hand-encode into the corpus the type of the training example (e.g. NP, S). While this is a time-intensive operation, it again creates reusable information, so that for new language pairs the work does not have to repeated and the system has access to (relatively) noise-free data.

In creating an elicitation corpus, we have the opportunity to design it to 1) fit the learning task at hand, SVSL in our case, and 2) use control strategies in order to extract as much information from as little training data as possible.

In order to fit the SVSL learning task, we have designed the elicitation corpus compositionally. As was described above, the learning algorithms aim at learning rules that are built from rules for smaller structures. Thus, it makes sense to first elicit simple structures (such as noun phrases) and learn rules for them. Later, we elicit more complex structures (such as complex sentences that contain a number of noun phrases) and learn compositional rules that take advantage of the learned NP rules.

The corpus is also designed to be consistent with a control strategy that allows the bilingual informant to navigate through it efficiently. Basic linguistic features and phenomena are elicited first, followed by less basic features. For instance, the corpus covers noun phrases and simple intransitive and transitive sentences first, whereas relative clauses are not elicited until later. Therefore, the ratio between the size of the training corpus and the linguistic coverage of the system can be much smaller than when learning from unstructured training data. This design also implies that the time of the bilingual informant can be used most effectively: even if he or she only translates part of the corpus, the resulting learned rules cover common linguistic features and constructions. The more data translated, the more sophisitcated the resulting transfer grammar will become.

Another way in which a controlled corpus can be useful is that it uses knowledge of language typology to determine what needs to be learned and what does not apply to a given language. This process was described briefly above as feature detection. For example, once we determine that a language does not mark plural number on nouns, we do not need to check whether it marks dual number. Conversely, if we know that a language encodes plural differently than dual, the system will not attempt to learn a transfer rule that fits both dual and plural. It will not consider it noise if the predicted transfer rule for a dual noun phrase does not match a transfer rule for plural that has already been learned.

To summarize, both learning from an unstructured and a structured corpus have their advantages. Our system will adapt to a given situation, depending simply on the availability of large amounts of sentence-aligned bitext versus the availability of a native informant who can translate the elicitation corpus. Ideally, of course, both types of training data can be combined in order to get the 'best of both worlds'. Currently, we have not yet developed sophisticated techniques to apply our rule learning algorithm to unstructured text. However, future versions of our system will provide this technology.

7. Some Challenges of Elicitation Based MT

Although elicitation is a promising solution to MT in a resource-rich/resource poor environment, it is not common because it is difficult. In this section we will outline some of the challenges facing elicitation based approach to MT. Some of the challenges may be specific to our project, whereas others may apply to any elicitation-based system.

7.1. Challenge Number 1: The bilingual informant

Before we can design a controlled corpus, we need to define clearly what the characteristics of a typical user of the system would be. Of high priority in our work is not to expect the bilingual informants to know and understand linguistic terminology. However, we must expect that the bilingual speaker has a good intuition about the complexity of language and knows that the elicited data will be analyzed automatically rather than by a human. For instance, if we present a user with a minimal pair of sentences, ideally the translation will again be a minimal pair. If no minimal pair translation is possible, this should be pointed out by the user or else detected automatically by the system.

In our system, an ideal user is furthermore able to make reliable grammaticality judgments via a Translation Correction Tool. This tool can propose a translation using a transfer rule that is not completely confirmed. The translation is then presented to the user for a grammaticality judgment. If the user can act as an oracle for the system, and, even more ideally, correct the translation to the closest possible fit, automated learning will be much facilitated.

7.2. Challenge Number 2: Alignment issues

As has been pointed out in the literature (e.g. (Melamed, 1998)), human-specified alignment is not noise-free. There is some degree of disagreement between people who align the same sentences, and even one and the same person will sometimes not be completely consistent. This provides a problem for a system that automatically elicits data and relies heavily on informant-specified alignments. Informants can be given intuitive instructions for alignment - for instance, we will ask the informant to align a word to only those words in the target language that actually have the root with the same meaning. Such guidelines have proven useful, as in (Melamed, 1998). Yet, the learning process must still be tolerant to noise in alignments, and we must also be prepared to accept non-optimal rules that are learned from noisy data. It should be noted, however, that while human alignments are not error-free, automatic alignment will also suffer from similar noise problems. Therefore, alignment issues must be dealt with whether learning from the elicitation corpus or from an unstructured corpus.

7.3. Challenge Number 3: Orthographic and segmentation issues

Many of the languages Avenue is working with have no pre-established writing system, and thus if we ask more than one bilingual informant to translate the elicitation corpus, we can not only get different ways of saying the same thing or the use of different words for the same object, but also different spellings of the same word. Naturally, if the same word is written with two or three different spellings throughout the elicitation corpus, a computational system will not take them to be the same word, and will create different alignment pairs as well as different dictionary entries. In addition, in the case of polysynthetic languages, different informants might segment words differently, and since the alignments are very sensitive to different word segmentations, the system might create two separate rules for the same phenomenon.

7.4. Challenge Number 4: Learning morphology

In addition to learning which inflectional features are present in a language, we must learn which sequences of characters or perturbations of stems are associated with each inflectional feature. We are currently investigating learning morphology from an untagged, monolingual training corpus, as has been done by other groups, e.g. (Gaussier, 1999) and (Goldsmith, 2001). Of course segmented or aligned corpora (such as our elicitation corpus) are useful, if they are available, for learning morphemes.

7.5. Challenge number 5: Elicitation of non-compositional data

In certain cases, we can expect a priori that the source and target language sentences will differ widely in their grammatical construction. Many fixed expressions and special constructions contribute to *machine translation mismatches*, meanings that are expressed with different syntax in the source and target languages. A typical example of a special construction is *Why not* `VP-infinitive` (e.g., *Why not go to the conference?* as a way of performing the speech act of suggesting in English. This construction poses a problem for our compositionally learned transfer rules in two ways: first, it does not follow normal English syntax in that it does not contain a tensed verb or a subject. Second, it does not translate literally into other languages using the translation equivalents of *why*, *not*, and infinitive. For example, conventional ways of saying the same thing in Japanese are *Kaigi ni ittara, doo?* (How (is it) if you go to the conference?) or *Kaigi ni itemittara, doo?* (How (is it) if you try going to the conference?)

In contrast to the literature on machine translation mismatches, which seems to assume that mismatches are exceptions to an otherwise compositional process, we believe that conventional, non-compositional constructions are basic and on an equal footing with compositional sentences (as in Construction Grammar). In fact, in our other projects, we have capitalized on the inherent non-compositionality of some types of sentences, such as those found in task oriented dialogues (Levin et al., 1998).

Our plans for non-compositional constructions are based on the observation that some types of meaning are more often a source of mismatches than others. Specific sections of the elicitation corpus will be devoted to eliciting these meanings including, modalities such as obligation and eventuality; evidentiality; comparatives; potentials, etc. Of course, there may be some compositional sub-components of non-compositional constructions such as the `VP` structure of *go to the confrerence* in *Why not go to the conference?*.

7.6. Challenge Number 6: Verb Phrases

We have identified the elicitation of verb phrases as a particular challenge. The main reason for this is that verbs do not agree in their subcategorizations across languages (see e.g. (Trujillo, 1999), p. 124). For example, in the English construction *He declared him Prime Minister*, the verb *declare* subcategorizes for two noun phrases, an indirect object (*him*), and a direct object (*Prime Minister*). The same construction in German would be:

Er	*ernannte*	*ihn*	*zum*	*Premierminister.*
He	*declare-3SG-PAST*	*him-ACC*	*to+the-DAT-SG*	*prime minister*

In German, the verb 'ernennen' ('to declare') subcategorizes for a direct object ('ihn' - 'him'), as well as a prepositional phrase introduced by the preposition 'zu' ('to'). Subcategorization mismatches are common, and may be random or systematic according to verb classes (Dorr, 1992). Thus our aim will be lexical transfer rules for individual verbs or verb classes as opposed to lexically independent verb phrase rules.

7.7. Challenge Number 7: Limits of the elicitation language

A problem always faced by field workers is that elicitation may be biased by the wording of the elicitation language, or by the lack of of discourse context. A language that has word order determined by the structure of given and new information may be difficult to elicit using only isolated English sentences. Furthermore there are many categories such as evidentiality (whether the speaker has direct or indirect evidence for a proposition) which can be expressed in English (e.g., *Apparently, it is raining* or *It must be raining* are expressions of indirect evidence) but are not an inherent part of the grammar as they are in other languages. In order to elicit phenomena that are not found in the elicitation language, it may be necessary to include discourse scenarios in addition to isolated elicitation sentences.

8. Conclusions and Future Work

We have presented a novel approach to learning syntactic transfer rules for machine translation. We emphasize again that the learning algorithms are not designed to work exclusively for structured data. However, learning is facilitated in a variety of ways when the translated and aligned elicitation corpus is available, for example because we can create relatively noise-free data on the resource-rich language side.

Our long-term plan is to expand the existing elicitation corpus to have reasonable coverage of linguistic phenomena that occur across languages and language families. In practice, we first have to focus on the more common features, so that the informant's time is used as efficiently as possible and a preliminary rough translation can be produced.

The elicitation corpus as well as the learning system are continually being expanded to handle more structures and more structurally complex sentences. As the elicitation corpus and the leearning algorithms are developed in conjunction, scaling them up is also achieved in parallel.

Acknowledgements

The authors would like to thank Ralf Brown, Ariadna Font Llitjós, and Christian Monson for their support and their useful comments.

References

Bouquiaux, Luc and Thomas, Jacqueline M.C. Studying and Describing Unwritten Languages. The Summer Institute of Linguistics, Dallas, TX, 1992.

Brown, Peter and Cocke, John and Della Pietra, Vincent J. and Della Pietra, Stehen A. and Jelinek, Fredrick and Lafferty, John and Mercer, Robert L. and Roossin, Paul S. A statistical approach to Machine Translation. In: Computational Linguistics, Vol. 16, No. 2, pages 79–85, 1990

Brown, Peter and Della Pietra, Stephen A. and Della Pietra, Vincent J., and Mercer, Robert L. The mathematics of statistical Machine Translation: Parameter estimation In: Computational Linguistics, Vol. 19, No. 2, pages 263–311, 1993

Brown, Ralf. Automated Dictionary Extraction for 'Knowledge-Free' Example-Based Translation In: Proceedings of the 7th International Conference on Theoretical and Methodological Issues in Machine Translation (TMI-97), 1997.

Calendino, P. Francisco. Selección de verbos mapuches. Estructura general del verbo mapuche. Instituto Superior Juan XXIII, Bahía Blanca, Argentina, 2001.

Carbonell, Jaime and Probst, Katharina and Peterson, Erik and Monson, Christian and Lavie, Alon and Brown, Ralf and Levin, Lori Automatic Rule Learning for Resource-Limited MT. In: Proceedings of the 5th Biennial Conference of the Association for Machine Translation in the Americas (AMTA-02), 2002.

Comrie, Bernard. Language Universals & Linguistic Typology. The University Chicago Press, Chigaco, IL, 1981, 2nd edition.

Comrie, Bernard and Smith, Noah. Lingua Descriptive Series: Questionnaire. In: Lingua, 42, pages 1–72, 1977.

Dorr, Bonnie. Machine Translation: A View from the Lexicon. Cambridge, MA: MIT Press, 1992.

Eskenazi, Maxine. Using Automatic Speech Processing For Foreign Language Pronunciation Tutoring: Some Issues and a Prototype. In: Language Learning & Technology, Vol. 2, No. 2, pages 62–69, 1999.

Gaussier, Eric. Unsupervised Learning of Derivational Morphology from Inflectional Lexicons. In: Proceedings of the Workshop on Unsupervised Learning in Natural Language Processing at the 37th Annual Meeting of the Assosication for Computational Linguistics (ACL-99), 1999.

Goldsmith, John. Unsupervised Learning of the Morphology of a Natural Language. In: Computational Linguistics, Vol. 27, Np. 2, pages 153–198, 2001.

Joseph H. Greenberg. Universals of language Cambridge, MA: MIT Press, 1966, 2nd edition.

Han, Benjamin and Lavie, Alon UKernel: A Unification Kernel. Available at http://www-2.cs.cmu.edu/˜ benhdj/c_n_s.html. 2001.

Hutchins, W. John and Somers, Harold L. *An Introduction to Machine Translation.* Academic Press, London. 1992.

Jones, Douglas and Havrilla, R. Twisted Pair Grammar: Support for Rapid Development of Machine Translation for Low Density Languages. In: Proceedings of the Third Conference of the Association for Machine Translation in the Americas (AMTA-98), 1998.

Levin, Lori and Gates, Donna and Lavie, Alon and Waibel, Alex. An Interlingua Based on Domain Actions for Machine Translation of Task-Oriented Dialogues. In: Proceedings of the 5th International Conference on Spoken Language Processing (ICSLP-98), Vol. 4, pages 1155–1158, 1998.

Levin, Lori and Vega, Rodolfo and Carbonell, Jaime and Brown, Ralf and Lavie, Alon and Cañulef, Eliseo and Huenchullan, Carolina. Data Collection and Language Technologies for Mapudungun. In: Proceedings of the Third International Conferences on Language Resources and Evaluation (LREC-02), 2002.

Melamed, Dan I. Manual Annotation of Translational Equivalence: The Blinker Project. IRCS Technical Report, Num. 98-07, 1998.

Mitchell, Tom. Generalization as Search In: *Artificial Intelligence*, Vol. 18, pages 203–226, 1982.

Nirenburg, Sergei. Project Boas: A Linguist in the Box as a Multi-Purpose Language. In: Proceedings of the First International Conference on Language Resources and Evaluation (LREC-98), 1998.

Nirenburg, Sergei and Raskin, V. Universal Grammar and Lexis for Quick Ramp-Up of MT Systems. In: Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and the 17th International Conference on Computation Linguistics (COLING-ACL-98), 1998.

Och, Franz Josef and Ney, Hermann. Discriminative Training and Maximum Entropy Models for Statistical Machine Translation. In: Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics (ACL-02), 2002.

Papineni, Kishore and Roukos, Salim and Ward, Todd. Maximum Likelihood and Discriminative Training of Direct Translation Models. In: Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP-98), 1998, 189-192.

Peterson, Erik. Adapting a Transfer Engine for Rapid Machine Translation Development. Masters Research Paper, Georgetown University. 2002.

Probst, Katharina and Brown, Ralf and Carbonell, Jaime and Lavie, Alon and Levin, Lori and Peterson, Erik Design and Implementation of Controlled Elicitation for Machine Translation of Low-density Languages. Workshop MT2010 at Machine Translation Summit VIII, 2001.

Sherematyeva, Svetlana and Nirenburg, Sergei. Towards a Unversal Tool for NLP Resource Acquisition. In: Proceedings of the 2nd International Conference on Language Resources and Evaluation (LREC-00), 2000.

Shieber, Stuart M. An Introduction to Unification-Based Approaches to Grammar. Center for the Study of Language and Information, Lecture Notes Number 4, 1986.

Tomita, Masaru, Editor and Mitamura, Teruko and Musha, Hiroyuki and Kee, Marion. The Generalized LR Parser/Compiler Version 8.1: User's Guide. CMU Center for Machine Translation Technical Report. April, 1988.

Trujillo, Arturo. *Translation Engines.* Springer Verlag, 1999.

Wheeler, Alvaro. *Dictionary of Siona.* unpublished, 1981?.

Vogel, Stephan and Tribble, Alicia. Improving Statistical Machine Translation for a Speech-to-Speech Translation Task. In: Proceedings of the 7th International Conferenc on Spoken Language Processing (ICSLP-02).

Yamada, Kenji and Knight, Kevin. A Syntax-Based Statistical Translation Model. In: Proceedings of the 39th Anniversary Meeting of the Association for Computational Linguistics (ACL-01).