Towards Interactive and Automatic Refinement of Translation Rules

Ariadna Font Llitjós
aria@cs.cmu.edu
Language Technologies Institute
School of Computer Science
Carnegie Mellon University

Thesis Proposal

December 21, 2004

Abstract

Although Machine Translation (MT) has advanced recently for language pairs with large amounts of parallel data, translation quality has not yet reached satisfactory levels, especially not for resource-poor languages with little if any parallel text to train statistical or example-based MT systems.

Rule-based transfer MT systems are the only feasible solution for resource-poor scenarios. However it can prove very costly and time consuming to refine and extend translation rule sets manually by trained computational linguists with knowledge of both languages. If the translation rules are written manually, no matter how many rules there are, coverage and accuracy can always be increased. If they are automatically learned, they might be either too general or too specific. Either way, in the face of unseen examples, the translation rules will need to be refined to account for new data. Thus, the goal of this thesis is to generalize post-edition efforts in an effective way, by identifying and correcting rules semi-automatically to improve coverage and overall translation quality, especially for resource-poor languages.

This proposal describes a plan for developing a novel approach for automatically refining translation rules from bilingual speakers' feedback and presents initial progress towards this goal. The main challenge is the development of the Rule Refinement module. Given a corrected and word-aligned translation pair as well as some information about the MT errors, the proposed approach determines the appropriate Rule Refinement operations that can be applied to the grammar and lexicon in order to extend and refine the existing translation rules and directly fix the errors. The resulting refinements and extensions apply not only to the translation instance corrected by the user, but also for other similar cases where the same error would be manifest.

One practical application of this research is extending and refining automatically-learned translation grammars for resource-poor languages, such as Mapudungun and Quechua, into a major language, such as English or Spanish. An automatic rule refinement module for at least one such MT system will be developed with the ultimate goal of making the rule refinement method as language-independent as possible.

Contents

1 Introduction

Machine Translation (MT) has been an around for more than fifty years and researchers have come a long way since Weaver's memo in 1947, when inspired by successes in cryptography, he suggested treating translation as a decoding problem.[1] However, in spite of recent advances in the field that have made MT systems available for new language pairs in just a few months, a few important challenges remain unsolved. The two unsolved challenges this thesis is most concerned with are how to automatically build MT systems for resource-poor languages with acceptable translation quality, and how to semi-automatically improve Transfer-based MT systems in general.

There are several factors that impede successful construction of an MT system. Besides the traditional cost and time factors, the lack of parallel data available in electronic format for a language pair has become the limiting factor. The growth of parallel corpora resulted in a proliferation of shallow MT systems, such as Statistical MT systems (SMT) and Example-Based MT systems (EBMT), that have reached comparable translation accuracy rates with those of traditional Transfer-based MT systems developed over many years by a large number of people. Indeed, the general feeling in the community is that large quantities of parallel data is a great substitute for linguists and even computational linguists.

However, such statistical systems are only an option for major language pairs where there are large amounts of parallel data. When there are little or no electronic data available, only the traditional way to build MT systems applies, namely, hiring linguists or technical expert native speakers to write, test, refine and expand translation rules.

When dealing with resource-poor languages, such as Mapudungun, Aymara or Quechua, in addition to little or no electronic data available, there might be very few or no native speakers who are also linguists or technical experts, so the traditional MT techniques are not practical. Such resource-poor scenarios require a different approach. Probst et al. (2002) introduced an automatic rule learning technique to build a translation grammar that only requires the cooperation of a non-expert native speaker to translate and align a previously designed Elicitation corpus (on the order of magnitude of a few thousand sentences) and a computer scientist to train the system. This is a major advance in the field, since even when large amounts of parallel data and native technical experts are lacking, an MT system can be automatically built for a new language pair. However, the translation quality of such a system is still below that of hand-crafted MT systems (Lavie et al., 2003). In this context, automatic refinement of translation rules becomes a crucial component of MT.

In general, most MT systems have failed to incorporate post-editing efforts beyond the addition of corrected translations to the parallel training data for

[1] "One naturally wonders if the problem of translation could conceivably be treated as a problem in cryptography. When I look at an article in Russian, I say: 'This is really written in English, but it has been coded in some strange symbols. I will now proceed to decode'." Warren Weaver, March 1947.

SMT and EBMT or to a translation memory database. And thus, a largely automated method that uses post-editor feedback to automatically improve translation rules would constitute a great advance in the field. If an MT-produced translation is incorrect, a bilingual speaker can easily diagnose the presence of an error, and with a bit more work give us some information pertinent to the type of error. But the informant cannot diagnose which complex translation rules produced the error, and even less determine how to improve the rules. The objective of the research proposed in this thesis is to automate the Rule Refinement process based on just *error-locus* and *error-type* information from the bilingual speaker, relying on blame assignment, compositional rule backtrace and on Active Learning, treating the bilingual speaker as an oracle, and measuring the consequent improvement in MT accuracy.

In this proposal, I motivate and describe a research plan for developing a new method to improve existing Transfer-based MT systems. This method is expected to be particularly useful in resource-poor scenarios, where statistical systems are not an option and where there might be no experts with knowledge of the resource-poor language, but need not be limited to them. Specific emphasis is placed on the challenge of accurately extracting information about MT errors from bilingual native speakers that is of particular interest and utility for the purpose of automatic Rule Refinement. Locally-validated corrections will be tested over a regression set of translation pairs, and kept if they do not introduce errors in stored past translations.

The online graphical user interface (GUI) developed as a result of the proposed approach, the Translation Correction Tool (TCTool), will be evaluated on its ability to achieve high measures of precision on a specific set of MT errors identified as critical for the task of automatic Rule Refinement. Given a corrected translation and some basic information about the errors from the user, the automatic application of Rule Refinement operations necessary to improve the translation grammar and lexicon will be implemented. An example of MT error is `Gaudí was a great artist - *Gaudí era un artista grande`, which given the corrected translation `Gaudí era un gran artista` by a bilingual user, can then be used to refine the noun phrase rule for general noun-adjective order in Spanish so as to also cover the exception of prenominal adjectives, such as `gran`. To see how this can be done with the Rule Refinement module, see the last simulation described in 5.5.

For cases where the error information is incomplete, I will also explore the use of interactive learning techniques to detect the set of necessary refinements or the appropriate level of granularity of such refinements, at run time. In such cases, we need to present users with other relevant sentences (minimal pairs), so that the Rule Refinement (RR) module gathers enough information to detect what is causing the user correction. For the example mentioned above, `Gaudí was a great artist - Gaudí era un gran artista`, in order to determine that this is an exception to the general `NP` rule for Spanish (`N ADJ`), the RR module would present similar relevant sentences to the user, such as `Gaudí was an innovative artist - Gaudí era un artista innovador`, and detect that the original `NP` rule is correct, but that a specific rule (`ADJ N`)

needs to be added for some cases (namely the cases with pre-nominal adjectives).

The main challenge of the work described in this proposal is the development of an automatic RR module that, given a corrected and aligned translation pair as well as some MT error information, needs to perform blame assignment to determine what rules are to be corrected and to apply the appropriate set of Rule Refinements required to correct the original translation. The RR module will be evaluated on its ability to improve coverage and overall translation quality as measured by a sensible measure. Initial experiments have shown that both BLEU (Papineni et al., 2001) and METEOR (Lavie et al., 2004) are sensible enough to automatically distinguish between raw MT output and corrected MT output, even for a small set of sentences (Section 7).

The overriding technical objective of the preliminary research was to develop a system capable of eliciting just the right information from a non-linguist and to use that information to the appropriate set of Rule Refinement operations that can be applied fully automatically on the existing translation rules. The proposed research will deal with implementing such set of refinement operations to be able to generate refinement hypotheses and explore ways to validate such hypotheses with and without further user interaction.

For a specification of what input and output is expected and produced by each of the two main modules proposed in this thesis, namely the Translation Correction Tool and the Automatic Rule Refinement module, see Figure 1.

After the refinement process is completed, the RR module has to make sure that, given the original source language sentence (sl), the MT system with the refined grammar and lexicon produces the translation as corrected by the user, and that the changes in the translation rules increase translation quality over a large enough test, or at least do not reduce it.

TCTool

INPUT
sl: Source_Language_Sentence (e.g. `I see the girl`)
tl: Target_Language_Sentence_1 (e.g. `veo la muchacha`)
al: Word-to-word_alignments_for_tl1 (e.g. ((1,1),(2,1),(3,2),(4,3)))
tl: Target_Language_Sentence_2 (e.g. `yo veo la muchacha`)
al: Word-to-word_alignments_for_tl2 (e.g. ((1,1),(2,2),(3,3),(4,4)))
...

OUTPUT
best tl selected from all translation candidates
good or bad, if bad:
c-tl: Corrected_Target_Language_Sentence_selected_by_user
(e.g. `veo a la muchacha`))
c-al: Corrected_Word-to-word_alignments_for_c-tl (e.g. ((1,1),(2,1),(3,3),(4,4)))
+ log file with user actions (e.g. addition of "a" between "veo" and "la")

Automatic RR module

INPUT from TCTool
sl: Source_Language_Sentence (e.g. `I see the girl`)
tl: Target_Language_Sentence_selected_by_user (e.g. `veo la muchacha`)
al: Word-to-word_alignments_for_tl (e.g. ((1,1),(2,1),(3,2),(4,3)))
if tl is corrected by user:
c-tl: Corrected_Target_Language_Sentence_selected_by_user
(e.g. `veo a la muchacha`))
c-al: Corrected_Word-to-word_alignments_for_c-tl (e.g. ((1,1),(2,1),(3,3),(4,4)))
+ log file with user actions (e.g. addition of "a" between "veo" and "la")

INPUT from MT system
constituency parse tree for Target Language side with rule ids
(e.g. (S,0 (VP,3 (VB,1 (V,7 "VEO")) (NP,3 (DET,7 "LA") (N,10 "MUCHACHA")))))
original grammar
original lexicon

OUTPUT
refined grammar
refined lexicon

Figure 1: Input and output specifications or two main modules proposed.

2 Thesis Statement and Scope

Thesis Statement

Given a rule-based Transfer MT system, one can extract useful information from non-expert bilingual speakers about the corrections required to make a Machine Translation output acceptable and about the MT errors present in an easy and structured way.
Furthermore, we can automatically refine transfer rules, given corrected and aligned translation pairs and error information, performing blame assignment, feature detection and, when necessary, resorting to further user interaction, so as to improve coverage and overall Machine Translation quality.

The following issues will be directly addressed in the thesis:

- **Translation Correction Tool:** Designing and implementing an online GUI to elicit information from non-expert bilingual speakers biased towards high precision rule corrections, complementing the high-recall property of the original Rule Learner. The TCTool will allow users to easily correct machine translations, by adding, deleting or modifying words, or changing their order in the translation, and to give as much information as possible about the errors, given a set of non-technical questions, the source language sentence and, optionally, context information.
- **MT Error Information:** Identifying an appropriate set of non-technical questions about MT errors with the goal of achieving high classification precision.
- **User Studies:** Designing and running user studies with non-expert users to validate the online GUI-based methods and methods to extract MT error information for the purpose of automatic rule refinement. This will result in data collection and analysis of MT error classification and detection.
- **Automatic Rule Refinement:** Designing and implementing a set of refinement operations to modify inaccurate translation rules in both learned and hand-crafted grammars and lexicons to reflect user corrections. Each correcting action allowed by the TCTool goes hand-in-hand with a set of refining operations that perform the appropriate changes in the translation rules, according to the error information available.
- **Interactive Learning:** Exploring interactive learning methods to find relevant minimal pair translation sentences to be presented to users to determine the appropriate level of generalization (word or POS) of the constraints that have been identified by the Rule Refinement module as

necessary to fix a specific translation rule or, if crucial error information is missing, to discover the set of features that triggered a specific correction, possibly using Active Learning methods to optimize user time.

- **Evaluation:** Testing the improvement on coverage (both in terms of number of words and phenomena) and translation accuracy of the rule refinement operations, as measured by one or more sensible measures. Establishing human oracle scores to obtain an upper-bound on how well any automated method can do given specific user feedback.

The following items fall outside the scope of this thesis and will not be directly addressed in my work. However, they are being developed in parallel by other researchers in the context of the AVENUE project:

- **Data collection:** The existence of an Elicitation corpus with wide linguistic coverage used to collect diverse data and of a fully inflected dictionary for at least one resource-poor language is assumed.

- **Morphology:** For the purpose of this thesis, I do not rely on the existence of a morphology module or the resource-poor language; even though I might run some experiments with it if a working morphology learning system is available before 2006 (Monson et al., 2004).

- **Transfer engine:** I assume a transfer engine capable of producing candidate translations using a lexicon and learned or hand-crafted rules, with back-tracing in rule space to determine which rule(s) are potentially responsible for identified errors (Peterson, 2002).

- **Decoder:** The decoder scores the partial translations and finds their most likely combination.

- **Rule Learning:** Part of this thesis focuses on improving the output of an automatic Rule Learner, especially designed to build a transfer-based MT system from languages with scarce resources into a major language. However, it is beyond the scope of this thesis to address the issue of non-interactive rule learning. For more details about this line of research, see Probst et al. (2002)

3 Related Work

3.1 Post-editing to Improve MT Systems

Post-editing has been defined as "the correction of MT output by human linguists/editors" (Veale & Way, 1997). For the task of automatic rule refinement, though, the editors are actually bilingual speakers with no expertise in linguistics, translation or computers, and their goal is to evaluate and minimally correct MT output, in a way that is similar to what has been referred to as *minimal post-editing* in the literature.

Allen (2003) defines the guiding objective of minimal post-editing to be the smallest number of changes possible for producing an understandable working document, rather than an optimal quality document. This implies that minimal post-editing is often associated with lower MT output quality. The minimal correction method we are proposing for the task of rule refinement should not compromise final translation quality, and what we consider minimal changes for this task also involves grammar correctness and fluency, in addition to meaning. Stylistic changes, however, are not considered minimal post-editing for this task.

Some researchers have looked at ways of including user feedback in the MT loop. Callison-Burch et al. (2004) developed a statistical MT system that can be improved by dynamically learning the correct translation of new phrases, through simple editing, as well as by allowing advanced users to correct misaligned sentence pairs from training data. One of the nicest things about their approach is the demystification of statistical MT as a black box. In their system, users can find out which way the MT system translated a sentence and which phrases are wrong, and change the behavior of the system by altering the underlying representations, i.e. the alignments.

Allen and Hogan (2000) proposed an automated post-editing (APE) prototype module which is meant to automatically fix up the highly frequent, repetitive errors in raw MT output before such texts are given to human post-editors, in order to speed up their work. The inspiration behind this work is very similar to ours in that they also react to the observation that if an MT system makes a particular error when translating a document, it is very likely to commit the same error each time the same set of conditions are presented. And if the error is fixed in a similar way, then it is possible to capture these modifications and to implement them automatically so that such repetitive errors can be reduced in MT output (Allen, 2003). The advantage of automatic post-editing is that it is system independent. In comparison with the Rule Refinement system proposed in this thesis, though, if two rules are incorrect but are easy to fix automatically, the combinatorics of the interaction of such rules might result into hundreds or thousands of sentences that need to be post-edited.

Su et al. (1995) have explored the possibility of using feedback for a corpus-based MT system to adjust the system parameters so that the user style could be respected in the translation output. Su et al. proposed that the distance between the translation output of the system and the translation preferred by the user should be proportional to the amount of adjustment to the parame-

ters involved in the score evaluation function, and should be minimized over time. We could not find, however, any papers reporting testing of these ideas. In the context of data sparseness, such a system is not feasible, since there is not enough data to estimate and train system parameters. Moreover, we are interested in improving the translation rules themselves, which in the case of automatically learned grammars will typically lack some of the feature constraints required for the correct application of the rule, rather than just tweaking the evaluation parameters, which in their system are conditional probabilities and their weights.

Menezes and Richardson (2001) and Imamura et al. (2003) have proposed the use of reference translations to "clean" incorrect or redundant rules after automatic acquisition. The method of Imamura et al. consists of selecting or removing translation rules to increase the BLEU scores of an evaluation corpus. The idea of using already existing reference translations to automatically refine MT rules is extremely appealing, and, time permitting, we plan to run some experiments to see how well this works in our system. In contrast to filtering out incorrect or redundant rules, though, we propose to actually refine the translation rules themselves, by editing valid but inaccurate rules that might be lacking a constraint, for example.

3.2 MT Error Information

To automatically refine a grammar, we need users to tell the system as much as possible about the MT errors that are present in the original MT output, in addition to correcting the MT output. Hence, in order for non-expert bilingual speakers to classify MT errors reliably and with high accuracy, we need to devise an intuitive MT error classification which does not rely on technical terminology, or knowledge of linguistics.

There are several ways to classify MT errors, but most MT evaluation methods described in the literature are designed to be used by either potential MT users, and thus they focus on system comparison and on ways to measure translation quality from an end-user viewpoint (Flanagan, 1994), or by developers to be used as a reference for manually modifying the grammar (White et al., 1994) or tweaking a few system parameters.

Much like White's approach, our MT evaluation method needs to capture translation adequacy, fluency and informativeness. However, our approach is based on a radically different method to extract MT error information. Instead of having developers or translation experts make judgments on the translation quality of the system, we have non-expert bilingual speakers correcting the translations and narrowing down the cause of the error by answering a predefined set of non-technical questions designed to allow automatic improvement of translation grammars and lexicons. Thus, we hope that the result from the error information elicitation process proposed in this thesis will shed a new light on MT evaluation, providing information that is missing from existing evaluation methods.

General Motors, in their project on Controlled Automotive Service Language, used minimal post-editing following the Society for Automotive Engineering (SAE) J2450 standard metric for translation quality. This standard specifies the following categories of errors, which are rated as unacceptable in translated texts (Allen, 2003): wrong term, syntactic error, omission, word-structure or agreement error, misspelling, punctuation error, miscellaneous error.

Niessen et al. (2000) present a tool to facilitate access to an MT evaluation database. Their MT evaluation is based on edit distance (number of insertions, deletions and substitutions) as well as a predefined set of error classes (missing, syntax, meaning, other), and it the applicatons of different evaluation metrics automatically. This tool makes MT evaluation more consistent over time and it can also be used to predict human evaluation judgment.

3.3 Rule Refinement

The idea of rule adaptation to correct or expand an initial set of rules is an appealing one. Researchers have looked at rule adaptation for several natural language processing applications.

Lin et al. (1994) report research on automatically refining models to decrease the error rate of part-of-speech tagging.

Lehman (1989) worked on adaptive parsing, and more specifically on gradual augmentation of a kernel grammar to include each user's preferred forms of expression, when communicating with a computer. In her work, the existing grammar is assumed to be correct and it is subsequently expanded through interactions with users to learn their idiosyncratic style.

Brill (2003) introduces a new technique for parsing free text: a transformational grammar is automatically learned that is capable of accurately parsing text into binary-branching syntactic trees with non-terminals unlabeled. The system learns a set of simple structural transformations that can be applied to reduce error. Brill's method can be used to obtain high parsing accuracy with a very small training set. Although small, the learning algorithm does need the training corpus to be partially bracketed and annotated with part-of-speech information, which is a scarce resource for minority languages. Even if we had such a small initial annotated corpus, transforming translation rules is non-trivial and cannot be done with simple patterns like the ones proposed in Brill's method. The rule refinement algorithm proposed here needs to deal with the lexicon, the syntax and the feature constraints in the rules.

Corston-Oliver and Gamon (2003) learned linguistic representations for the target language with transformation-based learning (Brill style) and used decision trees to correct binary features describing a node in the logical form to reduce noise.

In the same spirit as the research proposed in this thesis, Gavaldà (2000) provided a mechanism that enables a non-expert end-user to dynamically extend the coverage of a natural language understanding (NLU) module, just by answering simple clarification questions. Gavaldà relied on non-expert users to automatically learn new semantic mappings for his NLU system.

Yamada et al. (1995) use structural comparison (parse tree) between machine translations and manual translations in a bilingual corpus to adapt a rule-based MT system to different domains. In order for this method to work, though, a parser for the target language (TL) needs to be readily available, which is typically not the case for resource-poor languages. Moreover, such a parser must have coverage for the manually-corrected output as well as the incorrect MT output to compute the differences. The actual adaptation technique is not described in this paper.

In her thesis, Naruedomkul (2001) proposes a basic word-to-word MT system, called the Generate and Repair MT system, that *repairs* a non-acceptable translation if it has a different meaning from the SL sentence. In order to repair a translation candidate, the system outputs an HPSG-like semantic representation for both the SL and the TL, detects the part of the TL that causes the mis-translation and replaces it with the corresponding, appropriate SL semantic representation. The system iterates until the semantic information of the SL and the TL are acceptably similar. In a final step, the word ordering module makes sure that the syntax is correct. All the examples given illustrate corrections of sense errors.

In sum, even though adaptation has been researched for MT and other natural language processing applications before, to my knowledge, none has attempted to refine the translation rules themselves, and thus this thesis proposes and interesting an novel approach to automatically refine and expand MT systems. Even though the approach proposed is not completely system independent, we believe that it can be easily adaptable to other Transfer-based MT systems.

4 Interactive Elicitation of MT Error Information

Even in resource-poor contexts, there is usually one resource available: bilingual speakers. The research proposed in this document exploits this fact maximally and relies on non-expert bilingual users to extract as much accurate information possible to determine error location and cause for use by the Rule Refinement (RR) module. In order to elicit MT error information from naive users reliably, we propose a user-friendly GUI that is intuitive and very easy to use and that does not assume any knowledge about translation, linguistics or computers.

4.1 Interface Design and Implementation

The Translation Correction Tool (TCTool) is a user-friendly online GUI interface designed to accurately evaluate MT output and to obtain as much information about MT errors as possible from non-expert bilingual speakers.

A first version of the TCTool has already been implemented and tested with a first set of English-Spanish user studies (Font-Llitjós & Carbonell, 2004). The current version of the TCTool presents users with a sentence in the source

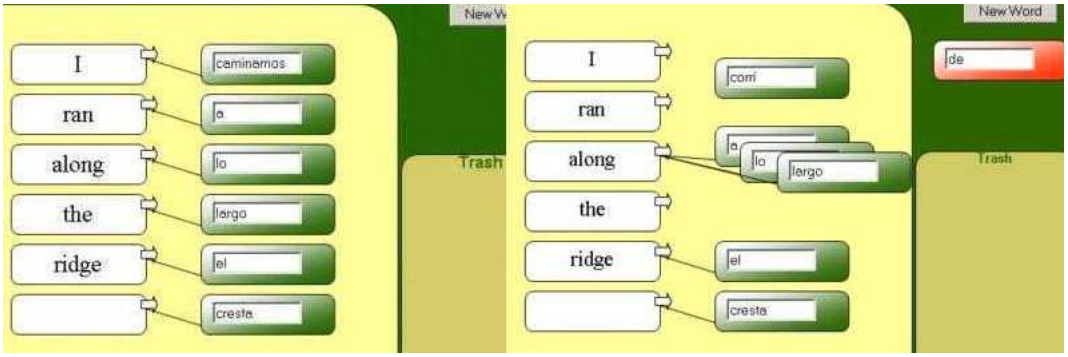

Figure 2: TCTool example of initial screen with incorrect translation (left), and the same example screen with sentence in the process of being corrected (right). Note that alignments, words and word order may be corrected.

language with up to five translations in the target language (TL), with an initial word alignment produced by the translation rules that applied, and asks them to check all the correct translations. Alternatively, if none of the alternative translations are correct, the TCTool asks users to fix the best translation with the least number of corrections possible. In this context, we tell users that the best incorrect translation is the one requiring the least number of changes to render the same meaning as the original sentence, in a grammatically correct and fluent TL sentence, but not necessarily in the most ideal form of expression.

This has been referred to as *minimal post-editing* in the literature, and the main problem is how to quantify the amount of post-editing changes that must be made to raw MT output text.

The most important aspect of the TCTool is that user feedback is not only used to improve the translation at hand, but it is critical for the refinement of the translation rules and will be used to improve the MT system at its core. For this reason, we emphasize to users the great importance of only correcting what is strictly necessary to obtain a correct translation of the original sentence from the given translation. There are three parts to the translations: the words in the TL and the alignments from the source language (SL) to the TL and the order of the words. The alignments indicate the word-to-word correspondence, namely what word in the SL translates as a word in the translation sentence. The current implementation of the TCTool makes the assumption that if a translation is correct, the alignments for that translation are also correct.

To illustrate what it means to correct a translation minimally, as well as how the TCTool can be used, I wrote a 23-page long TCTool tutorial[2], The tutorial shows the possible actions to correct a translation with 4 example sentences. Figure 2 shows the TCTool interface before and in the middle of the correction of the first sentence in the tutorial.

[2]http://avenue.lti.cs.cmu.edu/~aria/spanish/tutorial.html

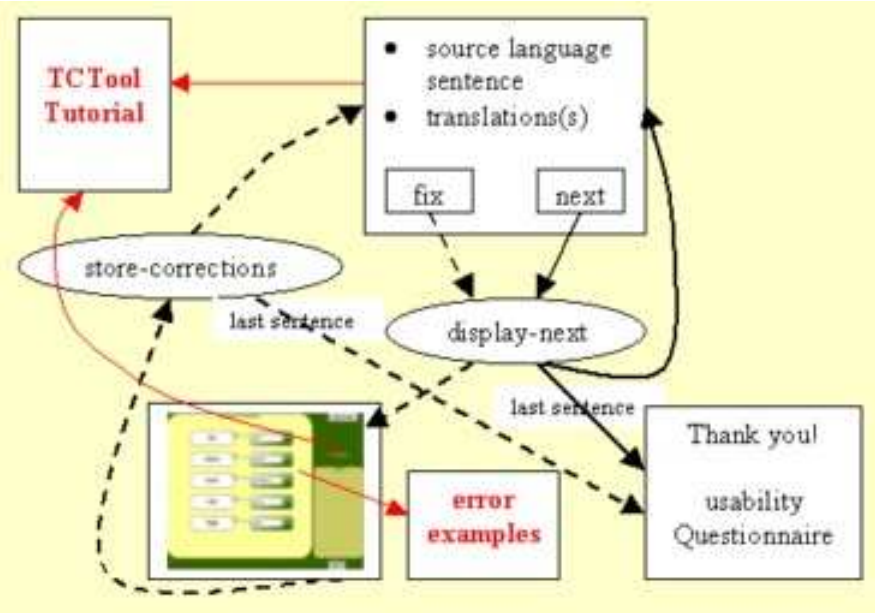

Figure 3: TCTool simplified data flow diagram. Discontinuous arrows represent the flow if the translation is not correct and user decides to fix it; continuous black arrows show the flow if translation is acceptable; everything in red refers to help pages.

The TCTool interface is designed to abstract away as much as possible from what is happening inside the MT system, and to allow users to correct errors at a high level. Since we are aware of the intrinsic difficulty of the task at hand, we tried to choose an interface that is easy and fun to use, and that will guide users on how to correct a translation and, at the same time, will give them enough flexibility.

The simplified data flow diagram in Figure 3 shows how the core of the TCTool works. Pages presented to users are represented in squares and CGI scripts are represented in ovals. The initial page is displayed in the upper center and shows the SL sentence and its translations and asks the user to pick the best translation. Discontinuous arrows represent the flow if the translation is not correct and user decides to fix it. In this case, users are provided with the drag and drop GUI shown in Figure 2. This page is written in JavaScript to allow users to easily fix translations just by clicking on the words that they want to modify and by dragging and dropping words when they want to change their order. From this page, users have two help options available: they can either go to the online tutorial or to a page with examples for all the error types. The

CGI script stores the log file of the correction actions performed by the user and, if the user is not fixing the last sentence, it loops back to the initial page which displays the next sentence pair. Continuous black arrows show the flow if translation is acceptable.

As can be seen from Figure 2, when correcting a sentence, the user is presented with two columns of blocks, each block containing a word. The SL sentence is displayed on the left column, and the TL sentence is displayed on the right. The word alignments between source and target sides are represented as lines that connect boxes containing SL words to boxes containing TL words. The white arrows on the SL side are used to create an alignment when dragged to a TL word.

*** Note that the TCTool does not need to know the innards of the MT system that produces the output. It only requires the SL sentence, the translation candidates produced by the MT system and the word alignment specification (e.g. ((1,1),(2,1),(3,2))). ***

For the AVENUE MT system, the alignments are directly extracted from the translation rules that generated the TL sentence.

There are six basic operations users can do to correct a sentence:

- modify a word
- add a word
- delete a word
- drag a word into a different position (change word order)
- add an alignment
- delete an alignment

For the first set of user studies, "modify a word" has a set of error types associated with it, and the user is asked to pick all the one(s) considered to be the cause of the error that they are correcting. Figure 4 shows the screen that pops up when user edits a word. Since we expect many users not to know what these types of errors stand for, users can click on each error type for a brief explanation with one or two examples.[3]

The other five operations already give us subtantial information about the error type (e.g. add a word *rightarrow* word missing), and thus users are not asked to further specify their action.

4.2 MT Error Typology

As part of this proposal, I defined an MT error typology mostly based on the MT error types that occur when translating from English to Spanish[4] with some indication as to what Rule Refinement operations, abbreviated below as RR, seem appropriate for each relevant type. Hence RR below refers to what the

[3] http://avenue.lti.cs.cmu.edu/~aria/spanish/error-examples.html.

[4] Spanish is used here for broader understandability only.

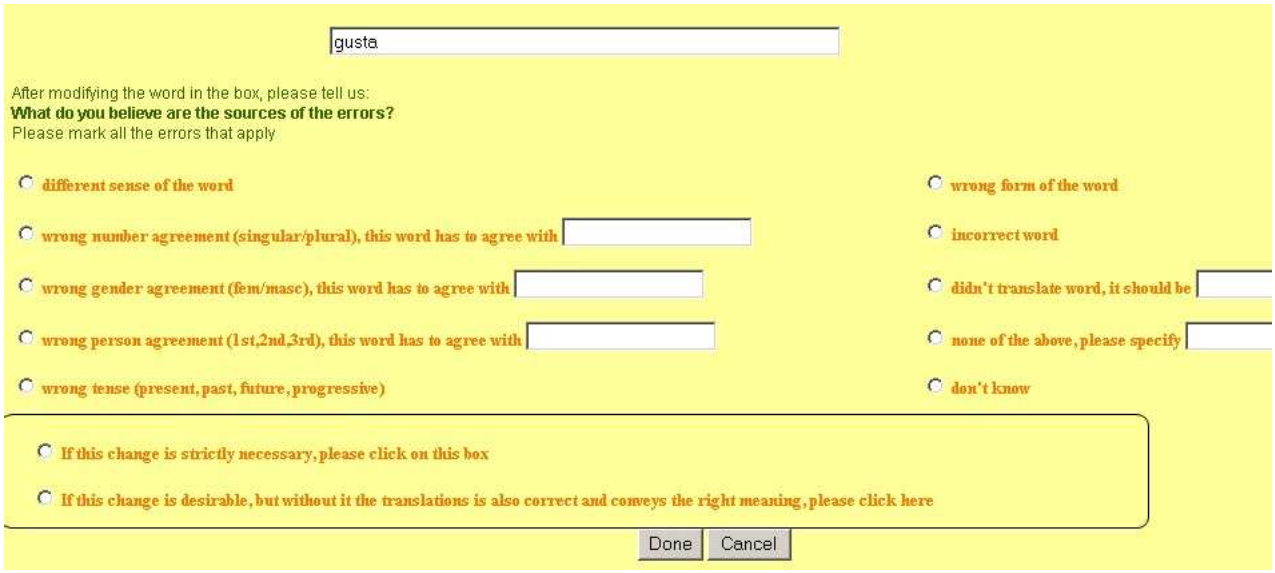

Figure 4: Error types associated with modifying a word.

Rule Refinement module needs to do in each case, not to what users must do or tell the system.

The first-level *types* are usually real error types, however, the second-level *types* tend to denote more the cause of the error, and third-level *types* tend to further specify features or constraints for that error/cause.

1. **Misspelling**
   RR: correct spelling in the lexicon (given by user).

2. **No translation**
   RR: add correct translation to lexicon (given by user).

3. **Missing word**

   3.0 **Translation not in the lexicon**
   Example: `Mary and John fell`
   `-> *Maria y Juan cayeron`
   `-> Maria y Juan SE cayeron`
   RR: Add new translation to the lexicon by copying existing lexical entry and updating TL side (`[fell] -> [cayeron] + [fell] -> [se cayeron]`) .

   3.1 **Part of another entry** (lexical expression)
   3.1.1 **Original lexical entry is wrong/incomplete**
   Example: `Me gustaria ir de viaje`

-> *I would like travel
-> I would like TO travel
RR: If user aligns the new word to an already aligned SL word and it's contiguous to the aligned TL word, add the new word to an existing lexical entry ([would like TO] -> [me gustaria]) or else add a lexicalized rule ([V to V] -> [V V]).

3.2 **Preposition missing**
Example: I am proud of you
-> *estoy orgullosa tu
-> estoy orgullosa DE ti
RR: add information about the preposition required to the ADJ(orgulloso).

3.3 **Syntactic restrictions apply**
Example: I saw the woman -> *vi la mujer -> vi A la mujer
RR: add directly to the rule as a terminal ([V NP]->[V ''a'' NP]) and if no feature exists for animacy, create one and add it as a constraint to the relevant rules and lexical entries.

4. **Extra word**

4.1 **Literal translation of a lexicalized form**
Example: I would like to show you my home
-> *me gustaria QUE mostrar tu mi hogar
-> me gustaria mostrarte mi hogar
RR: add SL word that generated spurious translation as part of an existing lexical entry ([would like]->[would like to]).

4.2 **Overgeneralization**
Example: John read the book
-> *A Juan leyo el libro
-> Juan leyo el libro
RR: add constraint that restricts the application of the rule to the right context, in this case, [NP]->[''a'' NP] can only apply in an object position.

5. **Word order**

5.1 **Local - within a constituent**
5.1.1 **Single word**
5.1.1.1 **No word change**
Example: John held me with his arm
-> *Juan sujeto ME con su brazo

`-> Juan ME sujeto con el brazo`
RR: create a new rule with the order flipped and
add a constraint specifying the features of the word moved,
in this case, that `me` is a pronoun.

5.1.1.2 **Word change**
Example: `Gaudi is a great artist`
`-> *Gaudi es un artista GRANDE`
`-> Gaudi es un GRAN artista`
RR: create a new rule with the order flipped and restrict its
application to prenominal adjectives.

5.1.2 **Multiple words**

5.1.2.2 **Word change**
Example: `I will help him fix the car`
`-> *Ayudare A EL arreglar el auto`
`-> LE ayudare a arreglar el auto`
RR: create a new rule with the order flipped and
add a constraint specifying the features of the word moved,
in this case, that `le` is a pronoun.

5.2 **Long distance** - across constituents
Example: `Where are you from?`
`-> *Donde eres tu DE?`
`-> DE donde eres tu?`
RR: Feed back to the Rule Learning module to learn a new rule.

6. **Incorrect word**

6.1 **Sense - semantic restrictions apply**
Example: `Wally plays the guitar`
`-> *Wally JUEGA la guitarra`
`-> Wally TOCA la guitarra`
RR: Create a binary feature to distinguish between the
two senses of the verb `play` in Spanish; add this
feature to the verb as well as the noun that triggers
the right sense, in this case `guitar`.

6.2 **Form - semantic restrictions apply**
Example: `*There were some flowers and MUCH tree`s
`-> There were some flowers and MANY trees`
RR: Create a binary feature to distinguish between
countable and uncountable nouns and mark quantifiers
as being able to quantify one or the other noun type

6.3 **Sense/Form - syntactic restrictions apply**
Example: `I'm playing chess`
`-> *Soy jugando al ajedrez`
`-> ESTOY jugando al ajedrez`
RR: Duplicate the general `VP` rule (`[AUX V]`)
and add a constraint to the specific rule so that, when
followed by a gerundive verb, the verb `to be`
translates as `estar` instead of `ser`.

6.4 **Selectional restrictions**
Example: `Today we are eating fish`
`-> *Hoy vamos a comer PEZ`
`-> Hoy vamos a comer pescado`
RR: create a feature to distinguish between `pez` and `pescado`
and add the right constraint to `comer` so that it only combines
with pescado.

6.5 **Preposition**
Example: `I was worried about you`
`-> *Estaba preocupada SOBRE ti`
`-> Estaba preocupada por ti`
RR: Add the preposition (or word in question) as part of the
lexical entry (`[worry]->[worry about]`) with the
appropriate translation (`[preocupada por]`), or add
a constraint that enforces the preposition following `preocupado/a`
to have the right form.

6.6 **Idiom - lexicalized expression**
Example: `Paul bit the dust`
`-> *Pablo mordio el polvo`
`-> Pablo se murio`
RR: Given correct user alignments, enter in the lexicon as a unit
(`[bite the dust]->[morir se]`).

6.7 **Form - morphology restrictions apply**

6.7.1 **Overgeneralization**
Example: `*wifes -> wives`
RR: correct in lexicon

6.8 **Form - phonetic restrictions apply**
Example: `*an wife -> a wife`
RR: create a binary feature that has value `+` for `wife`
(and eventually all other words starting with a consonant), and
add the appropriate constraints so that `a` can only combine with
such words, and `an` can't.

6.9 **Other - wrong translation**
Example: `I drove to the movies`
`-> *Conduje al cine`
`-> Fui al cine (en coche)`
RR: Add `ir` as an alternative translation for `drive`.

6.10 **Translation not in lexicon**
Example: `I'm tired of you`
`-> *Estoy cansada de tu`
`-> Estoy cansada de TI`
RR: If the right translation (according to the feature constraints) is not in the lexicon (`ti`), the transfer engine picks the first existing translation in the lexicon, in this case `tu`.
Need to add it to the lexicon, as indicated in 3.1.0.

7. **Agreement** (number, gender, person and tense)

7.1 **Right form not in the lexicon**
Example: `They climb the mountain`
`-> *Ellos ESCALA la montana`
`-> Ellos ESCALAN la montana`
RR: If the right form cannot be generated by the morphology module, add the correct from to the lexicon and make sure appropriate agreement constraints are in place.

7.2 **Missing agreement constraint**

7.2.1 **Within a constituent**
Example: `The tall chair was red`
`-> *El silla ALTO era roja`
`-> La silla ALTA era roja`
RR: Add a gender agreement constraint to the `NP` rule between `N(silla)` and the `ADJ(alto)`.

7.2.2 **Across constituents**
Example: `The chairs were very tall`
`-> *Las sillas son muy ALTO`
`-> Las sillas son muy ALTAS`
RR: Add a gender agreement constraint to the mother node of the constituent containing `sillas` and the constituent containing `alto`.

7.3 **Extra agreement constraint - Overgeneralization**
Example: `John protects animals`
`-> *Juan protege al ANIMAL`

> `-> Juan protege a los ANIMALES`
> RR: Eliminate overly general constraint. In this case, the `VP` rule `[V NP]` has a number agreement constraint between the verb and the direct object.

This error typology is by no means complete and is still under development. I expect it to continue evolving during the time I take to finish the research proposed here.

The actual RR formalization is described in section 5.4 and the question whether the refinement operations suggested for each error type can be done in a fully automatic way is addressed in section 9.3.

4.3 Evaluation: English-Spanish User Studies

A preliminary usability evaluation presented users with 32 simple English-Spanish translations. The SL sentences were chosen from the AVENUE Elicitation corpus, which was designed to cover a variety of linguistic phenomena (Probst et al., 2001), so as to expose users with a wide range of different MT errors.

The MT system used for this user study consisted of a very small manually written English to Spanish grammar containing 12 rules (2 S rules, 7 NP rules and 3 VP rules) and 442 lexical entries, designed to translate the first 400 sentences of the Elicitation corpus.

The purpose of this first user study was twofold, to evaluate the TCTool interface, as well as the initial MT error classification. For this task, we need to think of MT error classification in a completely different way, and we need to find a balance between simplicity and informativeness. Users of the TCTool, i.e. non-expert bilingual speakers, have to be able to understand the different error types and classify them accurately, and, at the same time, we have to obtain the most information about errors possible, in order to be able to automatically refine translation rules.

For this TCTool user study, we used an MT error classification based on standard linguistic distinctions with 9 error types: wrong word order, wrong sense, agreement error (which can be with respect to number, person, gender or tense), wrong form, incorrect word and no translation. We found that some of these distinctions are actually not relevant for the purpose of Rule Refinement (i.e. wrong sense and wrong form), but contribute to the task complexity faced by the user. Thus, a less traditional error classification seems more appropriate for this task, and is proposed in Section 9.2.1 below for the next version of the TCTool.

There were 29 native speakers of Spanish with good knowledge of English who completed the evaluation. Most users were from Spain (83%). Two thirds of the users did not have any background in Linguistics, 75% had a graduate degree and 25% of the users had a Bachelor's degree. On average, users took an hour and a half to evaluate the 32 translation pairs and fix 26.6 translations, about 3 minutes per translation pair. But there was a significant variance

among users, the duration range being [28min-4:18hours]. For more details, see Font-Llitjós and Carbonell (2004).

4.4 Data Analysis

To measure accuracy a gold standard was established, and 10 users were selected according to demographics (to reduce possible dialectal differences) as well as levels of education in order to represent as many different education levels as possible. For these 10 users, we analyzed a total of 300 log files in detail, and manually counted the times that the errors detected by users and the error type associated with each error coincided with the gold standard. This count was very strict and good corrections that were different from what the gold standard indicated, were counted as not being 100% accurate. This might seem more strict than necessary; however, I was trying to simulate what the process would be for a language pair I am not familiar with, where I cannot make any judgments.

We found that users can detect translation errors with reasonably high accuracy (90%), but have a harder time determining what type of error it is. Given our MT error classification, users identified the error correctly 72% of the time (Table 1).

	precision	recall	F1
error detection	0.896	0.894	0.895
error classification	0.724	0.715	0.719

Table 1: Average accuracy measures for 10 users and 32 sentences (300 log files)

From the precision-recall viewpoint, we are interested in having users correct and classify errors with high precision, even at the cost of lower recall. If users do not detect or classify all the actual errors, this does not have a great impact on the Rule Refinement module. However, when they do detect and classify errors, we need them to do so as precisely as possible, so that the translation grammar and lexicon are not refined incorrectly. User corrections were not always consistent with each other; however, most of the time, when the final translations differed from the gold standard, they were still correct. On average, users only produced 2.5 translations that were worse than the gold standard (out of 26.6 that they corrected). Users got most alignments correct.

5 Automatic Refinement of Translation Rules

As stated in the related work section (3), current solutions to improve MT output are limited to manually correcting the output (Allen, 2003) and, in the best case scenario, to doing some post-processing to alleviate the tedious task of manual post-editing by correcting the most frequent errors beforehand (Allen &

Hogan, 2000). To this date, however, there exists no solution to fully automate the post-editing process.

Thus, one main goal of this thesis is to improve MT output as automatically as possible. There are at least two different approaches one could take in order to do that. First, one could try to learn post-editing rules automatically from concrete corrections, which has the advantage of being system independent. With this approach, however, one cannot generalize over specific corrections to correct the same structural error with a different word, say; furthermore, several thousands of sentences would need to be corrected for the same error.

Alternatively, we could go to the root of the problem, and try to avoid getting the error in the first place by refining existing translation rules automatically. This way, by just fixing one or two translation rules, we can avoid the generation of a structural error that would otherwise creep in thousands of sentences. This approach naturally requires the existence of translation rules that can be refined.

Therefore, the approach proposed in this thesis is to attack the core of the problem and refine the incorrect translation rules themselves guided by user corrections. In other words, we propose to automate post-editing efforts by recycling them back into the MT system.

(Callison-Burch et al., 2004) also proposes to fix the underlying representation of an MT system to reduce error rate, but his is a Statistical system and thus the underlying representation is just the word-to-word alignments. Another difference with the proposed research is that his method involves expert users manually fixing the alignments, whereas we are proposing to go one step further by not requiring expert users and by automating the correction process.

5.1 A Framework for Rule Adaptation

After eliciting the *error-locus*, namely the location of the error in the translated sentence, and *error-type* information from non-expert bilingual speakers, as illustrated in the previous section (figures 2 and 4), we are half way towards being able to automatically refine the translation rules that generated each specific error. The other half involves using the error information available to trace back the incorrect translation rules and fix them automatically so as to improve coverage and translation quality.

More formally, the general goal of my thesis is to maximize coverage (C) and translation quality (TQ) of MT output given the following information: the source language sentence (SL), the original target language sentence (TL), the target language sentence corrected by users (TL′), a grammar (G) and lexicon (L),[5] the parser information (P) and a set of Rule Refinement operations (RR).

As a starting point, I have a G and a L, the SL and corresponding TL and the P, hence the general goal can be divided into two sub-goals:

Sub-goal 1: obtain user corrections + error information (TL′) [This is done with the TCTool as shown in the previous section]

[5] One of the necessary steps to build an initial MT system is to build a lexicon, and thus even when dealing with resource-poor scenarios, I assume the existence of a basic lexicon or glossary.

Sub-goal 2: find optimal set of RR operations and apply to current G and L, so that $TQ_2+C_2 > TQ_1+C_1$:

find best (RR operations |(G,L,SL,TL,TL′,P)) such that $TQ_2+C_2 > TQ_1+C_1$

Sometimes, TL′ (in addition to TL, SL, P) is enough to define an appropriate set of RR operations, possibly with a small amount of Active Learning to be able to make the refinement more robust through further user interaction. However, some other times, TL′ is not enough, and in order to define the appropriate RR operations we would need to make extensive use of Active Learning methods. In such cases, the RR module would need to generate several relevant minimal pairs to be evaluated by users to narrow down the cause of the error and the nature of the fix. For concrete examples of these two cases, see Section 9.3.2.

I am interested in having in place both interactive and batch modes for the Rule Refinement module. However, since I anticipate situations where only the batch mode is possible (for example, when user availability is limited or when we need to compute refinements with all data gathered so far), some reasonable defaults can be set (such as always adding the constraint at the most specific level), so that the rule refinement process can take place even when no further user interaction is possible.

In the rest of the document, I will clarify the distinction between cases that are only solvable through further user interaction, and cases that can be refined on the sole basis of static information (which include initial user corrections).

In addition to having user corrections trigger a particular RR operation, at run time (interactive mode), once I define a set of RR, I can also apply the whole set to the existing G and L, so as to maximize TQ of MT input (batch mode):

$$\max \text{TQ+C } (\text{TL}|(\text{TL}',\text{SL},\text{P},\text{RR}(\text{G},\text{L}))$$

For this, we would need to find a way to measure TQ and C automatically and run a program to try all the possible RR combinations (order matters), until it found the RR operation sequence that maximizes TQ and/or C given a particular input and a particular G and L.

In order for BLEU or any other automatic measure to be able to asses a real improvement in TQ, we would need to pick RR so as to maximize TQ directly over a corpus ($\text{CP}_{Language}$):

$$\max \text{TQ+C } (\text{CP}_{TL}|(\text{CP}_{TL'},\text{CP}_{SL},\text{G, L,RR}))$$

For a resource-poor language, I would need a set of reference translations from the Elicitation corpus, which currently has about a thousand sentences and is expanding.

In sum, the main goal of this thesis is to find a computable minimal extension of an original grammar (and lexicon) that is consistent with user corrections, namely that is able to translate the SL sentence into the correct TL sentence, as indicated by users.

```
a      {NP,9}                ;; Rule identifier
b       NP::NP : [ADJ N] -> [N ADJ]
      ; x0  y0    x1 x2      y1 y2
c      ( (x1::y2)  (x2::y1)  ; alignments
d      ((x0 mod) = x1)       ; the adjective is the modifier
d      (x0 = x2)             ; the noun is the head
e      (y2 == (y0 mod))
e      (y1 = y0)
e      ((y2 agr) = (y1 agr)))
```

Figure 5: English-Spanish translation rule for NP; x here means source and y, target.

Before going into details about the formalization of the Rule Refinement operations, let me briefly describe the formalism used for the translation rules.

5.2 The Transfer Rule Formalism

In the AVENUE MT system, translation rules can be written by hand, or can be acquired automatically by the Rule Learning module. The Rule Learner automatically infers hierarchical syntactic transfer rules, which encode how constituent structures in the source language (SL) transfer to the target language (TL). Automatic structure learning is done in two main steps: Seed Generation and Compositionality. For more details, see Probst et al. (2002).

For both types of grammar, the translation rules are comprehensive in the sense that they include all information that is necessary for parsing, transfer, and generation, similar to the modified transfer approach used in the early METAL system (Hutchins & Somers, 1992). In this regard, they differ from "traditional" transfer rules that exclude parsing and generation information.

In our system, translation rules have 6 components: a) the type information, which in most cases corresponds to a syntactic constituent type; b) part-of-speech/constituent sequence for both the SL (`x`-side) and the TL (`y`-side); c) alignments between the SL constituents and the TL constituents; d) `x`-side constraints, which are defined as equality of grammatical features in the SL sentence; e) `y`-side constraints, which are defined as equality of grammatical features in the TL sentence, and f) `xy`-constraints, which provide information about which feature values or agreements transfer from the source into the target language.

Figure 5 shows an example of an English to Spanish translation rule for noun-phrases containing a noun and an adjective. This translation rule swaps the original English word order, from adjective-noun to noun-adjective, and enforces their agreement in Spanish. In this case, there are no `xy`-side constraints specified by the rule.

In Figure 5, the notation `NP::NP` indicates that a noun phrase (dominated by an `NP` node) on the SL side translates into another noun phrase on the TL side. The constituent sequences are `[ADJ N] - > [N ADJ]`. 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 a typical unification grammar formalism. For more details about the rule formalism, see (Probst et al., 2002).

5.3 Formalizing Error Information

The MT error typology described above (section 4.2) provides us with a first approximation to the different refinement operations that we might be able to perform to improve a translation grammar and lexicon. But how does the system know when to apply which operation automatically?

In order for any system to apply refinement operations efficiently in an automatic way, we need to formalize the different kinds of user corrections and the refinement operations that they should trigger.

We represent TL sentences as vectors of words from 1 to n (n=sentence length), indexed from 1 to m (m=corpus length) $\overrightarrow{TL_m}$=(W_1,...W_i...,W_n),

and the corrected sentence TL_m' as follows:

1. $\overrightarrow{TL'_m}$**=(**$\mathbf{W}_1$**,...,**$\mathbf{W}_i$**',...,**$\mathbf{W}_c$**,...,**$\mathbf{W}'_n$**)** where W_i represents the error, namely the word that needs to be modified, deleted or dragged into a different position by the user in order for the sentence to be correct; and W_i' represents the correction, namely the user modification of W_i or the word that needs to be added by the user in order for the sentence to be correct. W_c represents the word that gives away the clue with respect to what triggered the correction, namely the cause of the error.

For example, in the case of lack of agreement between a noun and the adjective that modifies it, as in `*el auto roja` (the red car), W_c is `auto`, namely the word that gives us the clue about what the gender agreement feature value of W_i, `roja`, should be. W_c can also be a phrase or constituent like a plural subject (eg. *[`Juan y Maria`] `cai`, where the plural is implied by the conjoined NP).

W_c is not always present and it can be before or after W_i ($i > c$ or $i < c$). They can be contiguous or separated by one or more words.

Finding Triggering Features

After users correct a word W_i, the RR module can compare W_i and its correction, W_i', at the feature level and try to find out which is the triggering feature, namely what feature attribute (or set of attributes, in cases where a corrections fixes two errors) has a different value in W_i and W_i'.

For RR purposes, we define the difference between an incorrect word and its

correction as the set of feature attributes for which they have different values. We can extract the set of features and their values from the lexicon.[6] We call this the feature delta function (δ) and it can be written as follows:

$$\delta(\ \mathrm{W}_i,\ \mathrm{W}_i\text{'})$$

The resulting δ set can be one feature attribute, a set of feature attributes, which are all responsible for the correction, or the empty set.

If the δ set has one or more elements, that indicates that there is a missing feature constraint for all the attributes in the set. Examples of this can be found when comparing Spanish variations for red δ(rojo,roja)={gender} and eat δ(comimos,comia)={person,number}. If the δ set is empty, that indicates that the existing feature set is insufficient to explain the difference between the error and the correction and therefore a new binary feature is postulated by the RR module, `feat_1`, say. An example of two words that would not have any attribute with a differing value is δ(mujer,guitarra)={$\emptyset$}, since the lexical entries in our grammars are not marked for animacy.

Once the RR module has determined the triggering features, and assuming the user was able to identify a W_c, it proceeds to refine the relevant grammar and lexical rules by adding the appropriate feature constraints between W_i and W_c.

The next section outlines all the Rule Refinement cases taking into account what information is available to the RR module at each time.

5.4 Rule Refinement Operations

In general, if the new refined rule (R') needs to translate the same sentences as before plus the corrected sentence, the original rule R is substituted by the refined rule R'. However, if the refined rule should only apply to the corrected sentence, then R bifurcates into R1, whose application needs to be restricted so as not to apply to the corrected sentence but still apply to the original sentences, and R2, the refined rule that applies to the corrected sentences.

Figure 6 shows the main types of Rule Refinement operations that we anticipate being able to implement with the RR module. If user corrections require a brand new rule not already in the original grammar, this falls outside the scope of this thesis. Instead, the Rule Learner (Probst et al., 2002) would be invoked.

In the first Rule Refinement operation shown in Figure 6 (RR1), the original grammar rule (GR0) bifurcates, and the Rule Refinement module leaves the original as is, but modifies the new copy of the rule, maybe by changing the word order and by making it more specific, so that it applies on the new case but not necessarily on the old. An example of this can be found in object pronouns in Spanish, which instead of following the verb like object NPs, often appear in a pre-verbal position (I saw you → *Vi te → Te vi), and thus the VP rule ([V NPobj]) would need to be bifurcated and a specific one created that only

[6] If the lexicon contains roots, some kind of morphological analyzer is needed to extract the features for each word.

Grammar

RR1:	GR0 → GR0 + GR1 [=GR0' + constr]	C[GR0] ≤ C[GR1]
RR2:	GR0 → GR1 [=GR0 + constr]	C[GR0] > C[GR1]
RR3:	GR0 → GR1[=GR0 + constr = -]	C[GR0] ≤ C[GR1,GR2]
	→ GR2[=GR0' + constr $=_c$ +]	

Lexicon

RR4: Lex0 → Lex0 + Lex1[=Lex0 + constr]
RR5: Lex0 → Lex1[=Lex0 + constr]
RR6: Lex0 → Lex0 + Lex1[≈Lex0 + ≠ TLword]
RR7: ∅ → Lex1 (adding lexical item)

Figure 6: Main types of Rule Refinement operations (RR) for grammar rules (GR) and lexical entries (Lex), and their effect on the rule's coverage (C).

applied to sentences where the object is realized with a pronoun with the order flipped ([NPobj_pron V]). The reason the coverage (C) of GR0 might be less than the coverage of GR0 + GR1 is that the modification undergone by GR1 (GR0') might allow different kinds of TL sentences to be correctly generated.

In the second Rule Refinement operation (RR2), the original rule is not tight enough, and needs to be made more specific for all instances of such rule application. A good example of this is if the NP rule was missing number and gender agreement constraints in Spanish; the noun, adjective and determiner always need to agree. This requires adding a constraint equation.

If a rule needs to be bifurcated, the RR module will generate the right translation by adding the appropriate constraints to the specific rule, but it will also attempt to decrease ambiguity by blocking the application of the general rule, since its already known it should not apply in this case. Therefore, in addition to modifying the more specific rule in a way very similar to the one described in RR1, RR3 refines the general rule with the addition of a blocking constraint (GR1), so that it applies to the earlier cases (where it translated correctly) and not to the new case (where it erred).

To go back to the pre-verbal object pronouns in Spanish, if the RR module has information consistent with the fact that they cannot appear in a post-verbal position (probably only possible through further user interaction), then it can block the application of the general rule (R1) to object pronoun, and thus a constraint requiring the NP not to be of type PRON (constr = -) should be added. The triggering constraint that gets added to the more specific rule (GR2) is made to constrain that the lexical entry is indeed tagged as + ($=_c$), so that if the lexical entry is underspecified, only the general rule will apply.

The first two lexical RR operations are the equivalent of the first two grammar RR operations. An example of RR4 can be seen in Mary and Anna fell → *Maria y Ana cayeron → Maria y Ana se cayeron, where "se cayeron" needs to be added to the lexicon with a constraint to distinguish it from "cayeron",

which is a perfect translation for "fell" in a different context[7].

One possible example of RR5 is adding a constraint (constr1 = +) to all animated nouns, such as woman, boy, Mary, and in contrast with trees, book, feather, which basically distinguishes nouns with animate referents from nouns with inanimate referents. The reason we might want to do something like this, is that in Spanish animacy is marked explicitly in the sentence in front of the object NP (e.g. I saw Mary → Vi **a** Maria).

RR6 adds a missing sense to the lexicon. Namely, the translation of an SL word required for a sentence is not the one in the lexicon, but a different one. In this case, the RR module, duplicates Lex0 and changes the TL side to match the translation proposed by the user. For example, if users were given Wally plays guitar → *Wally juega guitarra, they would correct the translation of "plays" and change "juega" into "toca", which is the right sense for play + instrument in Spanish. If the lexicon only had an entry for [plays]→[juega], then RR6 would apply and generate a new entry ([plays]→[toca]) with the same feature constraints, but with the TL word modified.

Finally, RR7 represents the operation required for out-of-vocabulary words, i.e. there is no lexical entry for the SL word aligned to it, and thus the system does not output a translation for it.

As an attempt to formalize most of the different Rule Refinements cases, I organized them according to the type of action users can perform to correct a sentence using the TCTool, and then according to what error information is available to the RR module. For each RR operation, I indicate what type from the error typology specified above (Section 4.2) they belong to.

The first two types from the MT error typology above (misspelling and missing translation) are trivial and thus are not described here. Cases where one correction involves more than two TCTool actions, such as the example given for type 5.1.2 above, are harder to detect as being part of the same error correction, and thus might not allow for automatic refinement.

The following typology sketches the different specific Rule Refinement cases identified so far. When the user identifies a triggering word (indicated as "+ W_c" below), there usually is a fully automatic way to refine the appropriate rules, even though further interaction with users might make the refinement more robust. Most cases where the user did not identify a triggering word ("- W_c") will require some amount of Active Learning to be solvable, and this is explicitly indicated below with something like "find a MP", where MP stands for minimal pair.

For complete refinement simulations see next section.

1. **Modify a word: $\mathbf{W}_i \rightarrow \mathbf{W}'_i$**[8]

   If modified word is in the dictionary:

   1.a **+ $\mathbf{W}_c$**

[7] Cayeron billetes del cielo

[8] Assume there is no change in POS unless otherwise indicated.

1.a.1. $\delta(W_i,W'_i) \neq \{\emptyset\}$
Error type: 7.2.1 (agreement)
RR: add constraints for all the features in the δ set between W_i and W_c (assuming they have the same mother)

1.a.2. $\delta(W_i,W'_i) = \{\emptyset\}$
Error type: 6 (incorrect word: sense, form, etc.)
RR: postulate new binary feature ($feat_n$) and add a constraint between W_i and W_c

1.b − $\mathbf{W}_c$[9]

1.b.1. $\delta(W_i,W'_i) \neq \{\emptyset\}$
Error type: 7 (agreement)
RR: use δ set to find all potential W_c (look for other words which have the same value as W'_i for the features in the δ set), change the value of the δ set for one potential W_c at a time to be the same as in W_i, and ask users to correct new sentence (TL"). If they change W'_i back to W_i, we've identified the W_c. Proceed as in 1.a.1 above.

1.b.2. $\delta(W_i,W'_i) = \{\emptyset\}$
Error type: 6 (incorrect word: sense, form, etc.)
RR: Postulate a new binary feature and add with a + to the lexical entry for W'_i and with − to the lexical entry for W_i. Add the feature constraint with value + for Y_i to the appropriate rule.

If the W_i is not in the dictionary as the translation of SLW_i, first add it and then apply 1.b.2.

2. **Add a word: $\emptyset \rightarrow \mathbf{W}_i$'**

If aligned to an unaligned SL word, check lexicon for $[W_{SL}] \rightarrow [W_i']$, if not there, add to lexicon (Error type: 3.2).

Look for POS of W_i' (POS_i')
2.a + $\mathbf{W}_c$
2.a.1. rule with the right POS sequence exists (including POS_i)
Error type: 3 (missing word)
RR: Postulate a new binary feature and add it to W_c and W_i (with value +) and a feature constraint to the rules to enforce the agreement between them (Y_c and Y_i).

[9]Either because the user did not identify it, or because the triggering context is not localized in one word, but rather in the syntactic structure.

That might be too specific, and we might need to adjust it by coping the general rule and just adding the constraint to the duplicate.

2.a.2. no rule including POS_i' in the right position
Error type: 3 (missing word)
RR: 2 options: make a general rule including POS_i' or add "W_i' " directly to the rule (active learning). The W_c will indicate which rule needs to be modified; W_i'|POS_i' will be added to the mother of W_c in the position indicated by the user (ith). Proceed as in 2.a.1 for POS-level. For word-level refinement, add feature constraint to Y_c.

2.b − $\mathbf{W}_c$[10]

2.b.1. Alignment added to W_i
Error type: 3 (missing word)
Add lexical entry with the SL word(s) aligned to W_i' in addition to previous TL words that were part of the lexical entry, if any
($[SL_W i] \rightarrow [(TL_W i-1)\ W_i\text{'}\ (TL_W i+1)]$).

2.b.2. No alignment (or W_i aligned to discontinuous SL words)
Error type: 3.3 (missing word)
RR: create a specific, lexicalized rule, or give to RL.
This will most likely not be an optimal solution.

3. **Delete a word: $\mathbf{W}_i \rightarrow \emptyset$**

3.a + **alignment ($\rightarrow \mathbf{W}_c$)**
Error type: 4.1 (extra word)
If user aligns the SL word(s), which used to be aligned to W_i, to another TL word, we consider it to be W_c.
Make adjustments to lexicon accordingly so that the newly aligned SL word together with the SL words that were previously aligned to W_c form a lexical entry.
($[SL\_word(s)] \rightarrow [\emptyset_i\ W_c]$).

3.b − **alignment ($\rightarrow -\ \mathbf{W}_c$)**
Error type: 4.2 (extra word)
RR: Look for similar examples in the corpus, where the rule that generated W_i applied correctly, and try to figure out the relevant difference with active learning (i.e. by creating MPs and asking users to correct them), or send back to RL.

[10]Either because user did not identify it, or because the triggering context is not localized in one word, but rather it's the syntactic structure.

4. **Change word order: $\mathbf{W}_i$ (...) $\mathbf{W}_c \rightarrow \mathbf{W}_c$ (...) $\mathbf{W}_i$**

4.a **contiguous $\mathbf{W}_i$ $\mathbf{W}_c$**
4.a.1. **= $\mathbf{W}_i$**
Error type: 5.1.1.1
RR: If the rule with the final order in the RHS doesn't exist, create one (duplicate + order flipped: POS_c POS_i).
Find a MP where the original rule (POS_i POS_c) applies successfully and query the user to try to find the triggering context. Add appropriate constraints. Alternatively, send back to RL.

4.a.2. **$\mathbf{W}_i \rightarrow \mathbf{W}'_i$**
4.a.2.1. **$\mathbf{POS}_i = \mathbf{POS}'_i$**
Error type: 5.1.1.2 (single word order change)
RR: Flip POS_i and POS_c. Solve as in 1.b.2.

4.a.2.2. **$\mathbf{POS}_i \neq \mathbf{POS}'_i$**
Error type: 5.1.2.2 (single (local) word order change)
RR: If the δ function contains only `pos`, we assume the different POS is responsible for the change of order, so we add a new feature constraint to the modified rule, where `((`y_i `pos) =c` POS'_i`)`

4.b **non-contiguous $\mathbf{W}_i$ (...) $\mathbf{W}_c$**
Error type: 5.2 (long distance word order)
RR: Send back to RL

4.c **– $\mathbf{W}_c$ (contiguous)**
Error type: 5.1 (local word order change)
If the rule with the final order doesn't exist, create one (duplicate + order flipped: POS_j POS_i). Find a MP where the original rule (POS_i POS_j) applied successfully, calculate the δ function (at the sentence level, Section 9.3.3) to find the triggering context. Add appropriate constraints. Alternatively, send back to RL.

All refinement operations can be done at the word level or at the POS level. Without further information, we don't know which level is most appropriate for each case. However, we can set some reasonable defaults when in batch mode, and use some Active Learning methods to determine what is the right level of granularity for each case when in interactive mode.

5.5 Rule Refinement Simulations

This section illustrates the actual refinement operations by going through a few different refinement examples for each correcting action that users can perform with the online Translation Correction Tool.

Modifying a word
When users modify a word, the TCTool displays a window to elicit more information about what triggered the correction (W_c). Such information is then used for refinement purposes.

a) Lexical refinement. [**Error type 2**] If a word is not in the translation lexicon, the MT system leaves the SL word untranslated as is, and the user will need to translate it and indicate that it did not translate. In this case, the RR module will just need to augment the bilingual lexicon with the new lexical entry.

[**Error type 6.1 / RR op. 1.a.2**] If a word has a different sense from what the translation indicates, users will need to correct it and indicate what the word is in the TL sentence that hints at this. If the correct sense is missing in the lexicon, the RR module will add it. Otherwise, it will need to create a value constraint that forces the right sense to co-occur with the triggering word once added to all the appropriate lexical entries, as well as add an agreement constraint to the right grammar rule. For example, consider `they told me to fire a seller` and its translation `me dijeron que disparara un vendedor`. Bilingual users can easily detect that even though `disparar` would be the correct translation of `fire` in a different context (`they told me to fire a gun`), in this sentence, it should be translated as `despidiera`.

Since there is no feature in the grammar that makes this distinction,[11] the RR module will postulate a new binary feature, `feat_3`, and add it to the appropriate VP rule (`VP::VP : [V NP] -> [V NP] ... ((y1 feat_3) = (y2 feat_3))`) as well as to the relevant lexical entries to allow for only the right verb-direct_object combinations:

```
fire-despedir(feat_3 = +) fire-disparar(feat_3 = -)
seller-vendedor(feat_3 = +) gun-pistola(feat_3 = -)
```

b) Syntactic refinement: [**Error type 7.2 / RR op. 1.a.1**] If a word needs to agree with some other word in the sentence, but it does not, users will detect this and correct the appropriate word. For example, if users are given `the chairs were very high - las sillas son muy alto`, they will probably change `alto` to be `altas` so that it agrees with `sillas`, and will indicate that it is `sillas` that gives them the clue about the error.

Let's examine a fully fleshed-out **refinement simulation** for this example, describing the steps taken by the RR module, assuming that users correctly change `alto` into `altas` and indicate that `altas` has to agree with `sillas`.

[11] δ(disparara,despidiera) = $\{\emptyset\}$.

Step 0: since the delta function of the incorrect word and the corrected word is a set of attributes (δ(`alto,altas`) = {`number,gender`}), we take them to all be triggering attributes.

Step 1: make sure `altas` appears as the translation of `high` in the lexicon with the right POS, otherwise, create a duplicate lexical entry and change the values for the attributes `(y0 agr num)` `(y0 agr gen)`. In this case, both `alto` and `altas` are in the lexicon:

```
ADJ::ADJ |: [high] -> [alto]
((X1::Y1)
((x0 form) = high)
((y0 agr num) = sg)
((y0 agr gen) = masc))
```

```
ADJ::ADJ |: [high] -> [altas]
((X1::Y1)
((x0 form) = high)
((y0 agr num) = pl)
((y0 agr gen) = fem))
```

Step 2: look at the tree output of the MT system and check whether `sillas` and `alto` are subsumed by a common node:

```
tree:  <((S,1  (NP,3 (DET,9:1 "LAS") (N,70:2 "SILLAS"))
               (VP,1 (V,88:3 "SON")) ))>
       <(ADV,5:4 "MUY")> <(ADJ,8:5 "ALTO")>
```

a. if they do (suppose we had an S rule which spanned the whole sentence: `<(s (np (det las) (n sillas)) (vp (v son) (adjp (adv muy) (adj altas))))>`), add an agreement constraint to the rule rooted at the parent node between the modified word and the triggering word. In this case, `((y2 daughter_2 agr) = (y1 agr))`:

```
{S,2}
S::S : [NP VP] -> [NP VP]
( (X1::Y1) (X2::Y2)
  ((x1 case) = nom)
  ((x0 subj) = x1)
  (x0 = x2)
  (y1 == (y0 subj))
  ((y0 agr) = (y1 agr))  ; passing the head agr features up to mother
  ((y2 agr) = (y1 agr))              ; subj-verb agreement
  ((y2 daughter_2 agr) = (y1 agr)))  ; subj-predadj agreement
```

```
{VP,4'}
VP::VP : [V ADJP] -> [V ADJP]
( (X1::Y1)
  (X2::Y2)
  (x0 = x1)
  ((y0 daughter_2 agr) = (y2 agr)))
```

Additionally, the RR module needs to make sure the `VP` with `[V ADJP]` passes the `agr` features of the second daughter (`ADJP`) up to the head as `daughter_2 agr` (as shown in rule {`VP,4`} above).

A linguist would have probably labeled `daughter_2` as `pred` to indicate that the predicative adjective needs to agree with the subject of the sentence. Assigning more mnemonic labels can always be done a posteriori, if desired, but it is impossible to do automatically, and it makes no difference (beyond readability) in the resulting rules.

b. if they do not have the same mother (i.e. there is no rule that spans over the modified word and the triggering word), the RR module feeds the SL sentence - TL sentence pair back into the Rule Learning module as a new training example.

If we are running the system in interactive mode, once the refinement is completed, the MT system is run with the refined grammar and lexicon and the new translation for the SL sentence is presented to the user again for confirmation.

Adding a word
[Error type 3.3 / RR op. 2] Given the Spanish translation `viste la mujer` from the English sentence `you saw the woman`, users will insert the word `a` in front of `la mujer` yielding `viste a la mujer`, and thus we instantiate W_i to W_2=a, and its POS is preposition (PREP). The reason for this correction is that in Spanish, direct objects with animate referent are marked with `a` (`viste a la mujer` vs. `viste la pluma`).

If users mark `mujer` as being the word in the translation responsible for the correction, then we have instantiated W_c to W_4=mujer (RR op. 2.a). In the manually written grammar, there is no rule with the sequences V PP, V PREP NP or V "a" NP, and thus, we arrive at 2.a.2 in the RR typology above.

If the user aligned the word `a` to one or more words in the SL sentence, we would have an indication that a lexical refinement is needed, however there is no word in the SL that translates into `a`, thus most users will leave it unaligned. Hence, we conclude that the refinement needs to be in the grammar.

Now, the RR module has two choices: 1) it can either directly add an `a` in the 2nd position of the right hand side of the rule that should contain both `a` and `mujer`, in this case it is the VP rule `[V NP]` → `[V NP]`, so as to obtain `[V NP]`→ `[V ‘‘a’’ NP]`, or 2) it can add a `PREP` in the same position and the same rule to yield: `[V NP]`→ `[V PREP NP]`.

In this case, only the first choice is correct, and adding a `PREP` results in an overgeneralization (it would yield to arbitrary productions in the TL). The RR module can not know this a priori, and thus at this point, it can follow different strategies, depending on the mode of operation and the default settings.

If the system is running in batch mode, we set the default to always adding the most specific constraint possible (or multiple feature constraints if $|delta| > 1$), namely if users added or modified a word, it does not infer that they would have done the same with a different word just because it has the same POS. In this case, this strategy would result in the right refinement and thus no further user interaction would be needed.

If the system is running in interactive mode, to determine whether any preposition could be added in the translation or just `a`, the RR module will need to find sentences that differ minimally with the TL sentence (called minimal pairs) to give to users.

In any case, once we have created a new rule to accommodate for the correction, some constraints need to be added to the rules so as to restrict the generation of an `a` only in the right context. Again, if we're in batch mode, the best strategy is to just add the most specific constraint, to make sure we're not introducing incorrect generalizations.

As stipulated in RR op. 2.a.1, The RR module postulates a new binary

feature, `feat_1`, to automatically extend the feature language. Then, `feat_1 = +` is added to the lexical entry for `mujer`, and to the appropriate rules. Given the output of the MT system and the user correction specified below, the RR module will bifurcate the appropriate grammar rule (VP,3) by making a copy of it, renaming it with the next VP-rule index available (VP,5), and adding "a" in the appropriate position on the right hand side of the context-free backbone, as you can see below:

```
N::N |: [woman] -> [mujer]
( (X1::Y1)
  ((x0 form) = woman)
  ((x0 agr pers) = 3)
  ((x0 agr num) = sg)
  ((y0 agr gen) = fem)
  ((y0 feat_1) = +) )
```

```
{VP,3} ;; general rule
VP::VP : [VP NP] -> [VP NP]
( (X1::Y1)  (X2::Y2)
  ((x2 case) = acc)
  ((x0 obj) = x2)
  ((x0 agr) = (x1 agr))
  (y2 == (y0 obj))
  ((y0 tense) = (x0 tense))
  ((y0 agr) = (y1 agr)) )
```

```
{VP,5} ;; (I) saw the woman -> viste a la mujer
     VP::VP : [VP NP] -> [VP ``a'' NP]
     ( (X1::Y1)  (X2::Y3)
       ((x2 case) = acc)
       ((x0 obj) = x2)
       ((x0 agr) = (x1 agr))
       ((y3 feat_1) =c +)
       (y3 == (y0 obj))
       ((y0 tense) = (x0 tense))
       ((y0 agr) = (y1 agr)) )
```

The `=c`[12] part of the equation ensures that lexical entries underspecified wrt. `feat_1` do not unify with it, and thus it prevents rule {`VP,5`} to apply for nouns with no value for the `feat_1` constraint. Since adding a word or a constituent to a rule changes the alignment specifications required, the RR module has to make sure that all instances of `y2` are now changed to be `y3`. But there is still something missing, for the constraint in rule `VP,5` to have any effect, the RR module still has to make sure that the value of `feat_1` percolates up from the lexical entry `mujer` to its mother, the NP rule:

```
{NP,3}
NP::NP : [DET N] -> [DET N]
( (X1::Y1) (X2::Y2)
  ((x0 def) = (x1 def))
  ((x0 det) = x1)
  (x0 = x2)
  (y1 == (y0 det))
  ((y0 feat_1) = (y2 feat_1))
  ((y1 agr) = (y2 agr)) )
```

If we are running the interactive mode, we need to find out whether `a` is added because the noun of the object NP is `mujer` or because it belongs to a larger class of nouns that require this behavior (nouns whose referent is animated).

[12] Constraint equations use '=c' to enforce that the right and left hand-side values are not empty. Both values must exist and be equal for a constraint equation to unify.

And so first the RR module needs to look for instances of "X saw(in any form of the verb to see) Y", where Y is different from la mujer, in the Elicitation corpus and put it in the right form to create a minimal pair ("I saw Y - vi a Y", where Y = la mesa, el árbol, los muchachos,...). In this case, the Elicitation corpus already contains I saw the feather, which is translated by the MT system as vi la pluma, and users evaluate as being correct.

Given the corrected TL sentence, vi a la mujer (T1'), and its minimal pair T2, vi la pluma, the RR module can calculate the delta function to extract the differing features between them[13]. In this case, the δ set is empty: δ(mujer,pluma) = $\{\emptyset\}$, since all the feature attributes for the two words are the same. In other words, there is no existing feature in our feature language that can be used to discriminate between these two words, and hence between T1 and T2.

At this point, the RR module would postulate a new feature, feat_1, just like in batch mode. However, in this case, the new feature is used to discriminate between NP[_ mujer] ((y0 feat_1) = +)) and NP[_ pluma] ((y0 feat_1) = -)), and the rest of the refinement operations are as described above.

Note that, if users do not identify a W_c in the translation, and a relevant minimal pair did not already exist in the Elicitation corpus or as already corrected data, the only way to perform the appropriate refinements would be if we could run the system in interactive mode, and query users with a few more sentences to determine W_c and the triggering context. This is a form of Active Learning where the system strives for minimal number of interactions to obtain the desired information.

Deleting a word

Most corrections can lead to refinements at two levels: the lexical and the syntactic level. Very often refinements need to be performed at both levels.

a) Lexical refinement. [**Error type 4.1 / RR op. 3**] Given the SL sentence I would like to show you my home and its translation Me gustaría *que* mostrarte mi hogar, users will detect that the translation of to is not required in the Spanish translation and in fact it makes the sentence incorrect. In the TCTool, all the SL words that are translated into gustaría should also be aligned to it, indicating that they are part of the same lexical rule. Hence, if users align to to gustaría (RR op. 3.a), the RR module just needs to substitute [would like] → [gustaría] by [would like to] → [gustaría] in the lexicon to account for the user correction.

b) Syntactic refinement. [**Error type 4.2 / RR op. 3.b**] Suppose the Rule Learning module acquired an NP rule from the translation pair Books are interesting - Los libros son interesantes, which adds a definite article in front of a noun in Spanish ([N] → [DET N]). When translating the new sentence I want books, the MT system will output Yo quiero los libros. In this case, users will realize that the correct translation of books in this sentence

[13]The delta function can be straightforwardly calculated between minimal pair words, or even minimal pair sentences, if they have the same length and structure, see Section 9.3.3 for details.

is not `los libros`, but rather `libros`, and will delete `los` with the TCTool. Since, unlike the previous example (`Books are interesting - Los libros son interesantes`), in this case, the Spanish object NP has a determiner or not depending on whether the English has one. In the object position, the presence of a determiner has a different meaning (`+the/los` → specific books; `-the/los` → books in general).

The appropriate NP rule (`[N]` → `[N]`) has probably already been learned before, so what the RR module needs to do in this case is restrict the application of `[N]` → `[DET N]` to apply only in subject positions. One way to do this is by adding a new feature, `feat_2`, with `+` as value for the NPs that are in subject position, by adding `((y0 feat_2) = +)` to the mother node of the NP rule (`NP::NP [N]` → `[DET N]`) and to the S rule that contains an NP in the subject position. Now, for the refinement to be effective for the example under consideration, a constraint is required to block the application of the VP rule, so that the NP above never applies in the object position:

```
{VP,3} ;; general rule
VP::VP : [VP NP] -> [VP NP]
( (X1::Y1)  (X2::Y2)
  ((x2 case) = acc)
  ((x0 obj) = x2)
  ((x0 agr) = (x1 agr))
  (y2 == (y0 obj))
  ((y2 feat_2) = -)
  ((y0 tense) = (x0 tense))
  ((y0 agr) = (y1 agr)) )
```

The only problem is, how does the RR module know that this is the right thing to do? Namely, how can the system know from the deletion of the determiner that it is because it's part of the object of the sentence, and not the subject.

Realistically, this type of corrections can only be resolved with further non-trivial user interaction. In batch mode, there would be little chance for a system to figure out automatically what the triggering context is and where the constraint needs to be added.

Changing word order

[Error type 5.1.1.2 / RR op. 4.a.2.1] If we are given the English sentence `Juan is a great friend`, and the translation grammar has the following general rule for noun phrases `NP: [ADJ N]` → `[N ADJ]` (see Figure 5 in Section 5.2 above for the whole rule), the translation output from the MT system will be `Juan es un amigo grande`, since that is what the general rule for nouns and adjectives in Spanish dictates. However, this sentence instantiates an exception to the general rule. Thus, given that the user will most likely correct the sentence into `Juan es un gran amigo`, we need to refine the grammar so that it also accounts for prenominal adjectives, in addition to the most common type of noun-adjective constructions.

If we have a general translation rule that accounts for the most common `N ADJ` order in Spanish (`Juan es un chico simpático - Juan is a friendly`

guy) and are given a corrected TL sentence with a prenominal adjective (ex: Juan es un gran amigo, i.e. an exception to the general rule NP,9 in Figure 5), the RR module will swap the POS in the y-side of the duplicate rule (updating the alignments appropriately) and will add a new feature, feat_4 with value + to gran and with value - to grande. Now, the feature constraint ((feat_4) =c +) also needs to be added to Yi, namely the prenominal ADJ. The resulting refined rule will look like this:

```
{NP,10} ; copy of NP,9. Juan is a great friend -> Juan es un gran amigo
NP::NP : [ADJ N] -> [ADJ N]
( (x1::y1) (x2::y2)       ; constituent alignments
  ((x0 mod) = x1)         ; ADJ is the modifier
  (x0 = x2)               ; N is the head
  ((y1 feat_4) =c +)      ; only ADJs marked as (feat_4=+) can go before N
  (y1 == (y0 mod))
  (y2 = y0)
  ((y1 agr) = (y2 agr)))  ; ADJ agrees with N
```

As before, for completion, in order to keep gran from firing with the general rule, the constraint ((feat_4) = -) also needs to be added to rule NP,9.

6 Comparing grammars for the purpose of RR

Ultimately, we would like to be able to develop a set of universal refinement operations that are user and language independent and that apply to any grammar. However, this is unrealistic.

I ran an experiment to determine the main differences between refinements required to improve a manually written MT grammar and those needed to fix an automatically learned grammar, assuming they both follow the formalism described in Section 5.2 above. From this experiment, I learned that the refinements necessary to fix a manual grammar tend to involve bifurcating a rule to encode an exception, whereas a large number of the refinements required to refine a learned grammar involve adding or deleting agreement constraints. A summary of this experiment is reported in Font-Llitjós et al. (2004).

The manually written grammar used for this experiment was the same one for the English-Spanish user studies mentioned in section 4.3 (12 rules: 2 S rules, 7 NP rules and 3 VP rules). Two example rules are give below.

```
{S,0}
  S::S : [NP VP] -> [VP]
; x0 y0   x1 x2      y1
((X2::Y1) ; set constituent alignments
 ((x1 agr pers) = (*OR* 1 2))
 ((x1 agr num) = sg)
 ((x1 case) = nom)
 (x0 = x2)
 ((y1 agr) = (x1 agr)))
```

```
{VP,3}
 VP::VP : [VP NP] -> [VP NP]
 ((X1::Y1)  (X2::Y2)
  ((x2 case) = acc)
  ((x0 obj) = x2)
  ((x0 agr) = (x1 agr))
  (y2 == (y0 obj))
  ((y0 tense) = (x0 tense))
  ((y0 agr) = (y1 agr)))
```

The automatically learned grammars were built with the Avenue Rule Learning (RL) module (Probst et al., 2002) on the same data set used to write

the manual grammar for the English-Spanish user study. This training set contains the first 200 sentences from the AVENUE Elicitation corpus.

For this experiment, both MT systems used a lexicon with 442 entries, developed semi-automatically seeking to also cover the test corpus (next 200 sentences in the Elicitation corpus), so that we can measure the effectiveness of the rules and abstract away from lexical gaps in the translation system.

The RL module can produce a basic grammar fully automatically from word-aligned translation pairs, namely the ones from the Elicitation corpus examples.

The grammar learned by the RL is enhanced by traversing each English parse tree from the top down. For each internal node, i.e. nodes that do not pertain to only one specific word, the enhancement algorithm extracts the subtree rooted at this node. Then, using the word alignments provided by the bilingual user, it forms a new training example for the English and Spanish chunks covered by the subtree.

For this experiment, the English training data was parsed using the Charniak parser (Charniak, 2000) and two different grammars were built, both with the basic and enhanced settings. The first grammar was built with the current version of the RL module, which does not learn feature constraints, and the second one included a few feature constraints (gender, number, person and tense) extracted from morphological analyzers for both English and Spanish .

The enhanced version of the first grammar (with no constraints), learned from 214 English-Spanish sentence pairs, contains 92 translation rules: 54 S rules, 16 NP rules, 19 VP rules, 1 PP rule and 2 ADVP rules. The following four rules are examples of the first grammar.

```
{S,8}
S::S [NP VP] -> [VP NP]
((X1::Y2) (X2::Y1))

{NP,9}
NP::NP [DET ADJ N] -> [DET N ADJ
((X1::Y1) (X2::Y3) (X3::Y2))

{VP,16}
VP::VP [AUX V NP] -> [AUX V NP]
((X1::Y1) (X2::Y2) (X3::Y3))

{VP,18}
VP::VP [AUX "PICKING" "UP" NP] ->
       [AUX "RECOGIENDO" NP]
((X1::Y1) (X4::Y3))
```

The enhanced version of the second grammar, the one with with constraints, contains 316 translation rules (194 S, 43 NP, 78 VP and 1 PP), 223 more than the learned grammar with no constraints. Examples from the second grammar learned are shown below.

```
{S,90}
S::S [NP VP] -> [NP VP]
((X1::Y1)(X2::Y2)
((X1 def) = +)
((X2 def) = -)
((X2 agr num) = (X1 agr num))
((X2 tense) = past)
((Y2 agr gen) = (Y1 agr gen))
((Y2 agr num) = (Y1 agr num)))

{VP,46}
VP::VP [V NP] -> [V NP]
((X1::Y1)(X2::Y2)
((X2 def) = -)
((X2 agr num) = sg)
((Y1 agr pers) = 2)
((Y2 agr gen) = masc)
((Y2 agr num) = (Y1 agr num)))
```

Since there is no fair way to compare the rules from the first grammar with manually written rules and their effect on translation quality, I take the second

grammar, the one with feature constraints, to be the final learned grammar and use it for further comparison.

The reason there are many more rules in the learned grammars is that the current implementation of the RL module does not throw away rules that are too specific and that are subsumed by rules which have achieved a higher level of generalization during the learning process. Note that the RL module is still under development.

When running our transfer system on a test set of 32 sentences, it was observed that the manually written grammar results in less ambiguity (on average 1.6 different translations per sentence) than the automatically learned grammar (18.6 different translations per sentence). At the same time, the final version of the learned grammar results in less ambiguity than the initial learned grammar with no constraints. This is to be expected, since relevant constraints will restrict the application of general rules to the appropriate cases.

Ambiguity is less of a serious problem than it may appear, since the goal of the transfer engine is to produce the most likely output first, i.e. with the highest rank. In experiments reported elsewhere (Lavie et al., 2003), we have used a statistical decoder with a TL language model to disambiguate between different translation hypotheses. We are currently investigating methods to prioritize rules and partial analyses within the transfer engine, so that we can rank translation hypotheses also when no TL language model is available.

While this work is under investigation, I emulated this module with a simple reordering of the grammar. I reordered three rules (2 NP and 1 S rule) that had a high level of generalization (namely containing agreement constraints instead of the more specific value constraints) to be at the beginning of the grammar. This in effect gives higher priority to the translations produced with these rules, since it outputs them first, which has a direct impact in the comparison results reported below.

6.1 Comparing Grammar Output

For the purpose of rule refinement, we only looked at the first five translations in the output of the learned grammar, since this is what users of the TCTool are asked to correct. Ultimately, given a correction, the RR module will first check through the whole list of the alternative translations, and if the corrected translation was already generated by our MT system, it will just trace back to the right rules and add the appropriate constraints. Once in place, this mechanism will work as a kind of Reinforcement Learning.

Most of the translation errors produced by the manual grammar can be classified into lack of subj-pred agreement, wrong word order of object pronouns (clitic), wrong preposition and wrong form (case) and out-of-vocabulary word. On top of the errors produced by the manual grammar, the current version of the learned grammar also had errors of the following types: missing agreement constraints, missing preposition and overgeneralization.

An example of differences between the errors produced by the two types of grammar can be seen in the translation of `John and Mary fell`. The manual

grammar translates it as `*Juan y Maria cayeron`, whereas the learned grammar translated it as `**Juan y Maria cai`. The learned grammar does have an NP rule that covers `[Juan y Maria]`, however it lacks the number constraint that indicates that the number of an NP with this constituent sequence (`[NP CONJ NP]`) has to be plural. The translation produced by the manual grammar, `*Juan y Maria cayeron`, is also ungrammatical, but the translation error in this case is a bit more subtle and thus much harder to detect. The correct translation is `Juan y Maria` *`se`* `cayeron`.

Another example to illustrate this is the translations of `John held me with his arm`. The MT system with the manual grammar outputs `*Juan sujetó me con su brazo`, whereas the one with the learned grammar outputs `**A Juan sujetó me con su brazo`.

Sometimes the output from the learned grammar is actually better than the manual grammar output. An example of this can be seen in the translations of `Mother, can you help us?`. The manual grammar translated this as `**Madre, puedo ayudar nos?` and the learned grammar translates is as `*Madre, puedes tu ayudar nos?`. The number of corrections required to fix both translations is the same, but the one produced by the learned grammar is clearly better, from a fluency prespective.

6.2 Error Analysis

The 32 sentences used in the English-Spanish user study described in Section 4.3 were translated using both grammars described above. We looked at the five first translations only, and found that both grammars translated the same four SL sentences correctly. Even though the final number of SL sentences correctly translated by the two grammars is the same, overall, the quality of the translations produced by the learned grammar was worse.

	number of sentences (over 32)
same translation	17
manual grammar better	10
learned grammar better	2
different error type	3

Table 2: Error analysis result of manual inspection of the output from the learned grammar and the manually written grammar. See next section for examples.

We looked at the best translation for each source language sentence in the user study for one grammar and compared it with the best translation for the other grammar. The results of manual inspection can be seen in Table 2.

We also created a hypothesis file with the output for each one of the grammars and calculated the standard automatic evaluation metrics, taking user corrections as reference. The results shown in Table 3 are consistent with those

obtained from manual inspection, in that even though the manual grammar performs better when compared to user corrections, the learned grammar does not do terribly worse. See Section 7 for a brief description of what these automatic MT evaluation metrics measure.

	NIST	BLEU	METEOR
manual grammar	4.3	0.16	0.6
learned grammar	3.7	0.14	0.55

Table 3: Automatic MT evaluation results for both types of grammars.

6.3 Rule Refinements Required for Each Type of Grammar

Since 15 out of 32 sentences contained different translation errors for each grammar, the refinement operations required by the translations rules of the two grammars might not be the same for a given SL sentence. Following are a few examples of the cases listed in Table 2.

Same translation

In many cases, the sentences were translated in the same way by the two grammars, and thus they need the same rule refinement operations to be corrected. Sentences that only differ in the optional subject pronoun are included in this category, since both alternatives are correct in Spanish. These are two examples from the test set that illustrate this:

SL: I saw you yesterday - TL: *(Yo) vi *tu* ayer → CTL: Te vi ayer [**RR op. 4.a.2.1**]

SL: It was a small ball - TL: eso era un balón pequeño

CTL stands for correct target language sentence, i.e. correct translation.

Different Translation

For 10 sentences in the test set, even though both translations are wrong, the manual grammar output requires fewer corrections. TL_l is the TL sentence output by the automatically learned grammar and TL_m is the one output by the manually written grammar:

SL: I'm proud of you → CTL: Yo estoy orgullo de ti

TL_l: **Yo estoy orgulloso *tu* [**RR op. 2.1 (+de) + 1.a.1 (tu → ti)**]

TL_m: *Yo estoy orgulloso de *tu* [**RR op. 1.a.1 (tu → ti)**]

Besides changing the `tu` into `ti` in both cases, the learned grammar needs an extra refinement, the addition of `de` in front of the second person pronoun. In fact, it is the presence of the preposition that enforces the oblique case.

In addition to the example given section 6.1, the learned grammar output requires less amount of refinement for the following SL sentence:

SL: I like you → CTL: Me gustas tu

TL_l: **Yo* me gustas tu [**RR op. 3 (yo → ∅)**]

TL_m: **Me *gusta* tu [**RR op. 1.a.1 (gusta → gustas)**]

In a few cases, both grammars output equally bad translations, but with different kinds of errors:

SL: He looked at me → CTL: él me miró

TL_l: *él *miraron* me

[RR op. 4.a.1 (me miraron) + 1.a.1 (miraron → miró)]

TL_m: * él miró *en me* **[RR op. 3.a (en → ∅) + 4.a.1 (me miró)]**

This experiment reveals a couple of main interesting differences between the refinements required by hand-crafted grammars and automatically learned grammars. Several rule refinements required to correct sentences translated by the hand-crafted grammar involve bifurcating a rule to encode an exception, whereas a larger portion of the refinements necessary to refine the learned grammar involve adding or deleting agreement constraints.

In general, the biggest differences between the hand-crafted and the learned grammar is that the learned grammar has a higher level of lexicalization and, currently, it can make good use of the Rule Refinement module to make adjustments to the feature and agreement constraints to achieve the appropriate level of generalization.

7 Automatic MT Evaluation

To determine the upper-bound on the improvement that can be expected from any RR module under ideal circumstances, and abstracting away from the TC-Tool, I set up an oracle experiment where raw MT output was compared with manually corrected MT output (to simulate the best output possible produced by an ideally refined grammar), via automatic MT evaluation metrics,.

As explained by Zhang et al. (2004), the central idea of automatic evaluation is to use a weighted average of variable length phrase matches against the reference translations. This view gives rise to a family of metrics using various weighting schemes. Based on these principles, the IBM MT research group proposed the BLEU metric. BLEU averages the precision for unigram, bigram and up to 4-grams and applies a length penalty if the generated sentence is shorter than the best matching (in length) reference translation (Papineni et al., 2001). A variant of BLEU has been adopted by NIST for its MT effort. The NIST metric is derived from the BLEU evaluation criterion but differs in one fundamental aspect: instead of n-gram precision the information gain from each n-gram is taken into account. The idea behind this is to give more credit if a system gets an n-gram match that is difficult, but to give less credit for an n-gram match which is easy (NIST, 2002).

Recent research has shown that a balanced harmonic mean (F1 measure) of unigram precision and recall outperforms the widely used BLEU and NIST metrics for Machine Translation evaluation in terms of correlation with human judgments of translation quality. METEOR (Metric for Evaluation of Translation with Explicit word Ordering) (Lavie et al., 2004) performs a maximal-cardinality match between translations and references, and uses the match to compute a coherence-based penalty. This computation is done by assessing the

extent to which the matched words between translation and reference constitute well ordered coherent "chunks". METEOR assigns most of the weight to recall, instead of precision and uses stemming.

An initial feasibility test shows that these automatic MT evaluation metrics can detect improvements on the raw MT output with statistical significance, even for a very small test set of 32 sentence, using just two sets of reference translations.

	NIST	BLEU	METEOR
raw MT output	4.3	0.16	0.6
manually corrected	5.6	0.5	0.8

Table 4: Oracle experiment results with automatic evaluation metrics.

NIST is the least discriminative of the metrics, since for bigrams that appear only once (very small set), the system does not get any credit, but even the NIST scores are statistically significant in this case.

8 Discussion

*** The Translation Correction Tool (TCTool) is system independent, and thus error detection and correction can be performed on any MT output. The only requirements are that the MT system provides the TCTool with the SL sentence, the TL sentences and the word-to-word alignments between SL and TL (e.g.: ((1,1),(2,3),(3,2))). This kind of information is typically available for Statistical MT systems, EBMT systems as well as Transfer-based systems.

The approach for Automatic Rule Refinement (RR) proposed here should work for any transfer-based system with a constraint-unification mechanism and a feature specification language for the grammar and the lexicon. The RR module will be implemented in a modular way. However, in order for the RR module to apply useful refinements, it needs to have access to the parse tree for the TL-side (generation) and to the actual grammar and lexical rules involved in the output parse tree. The RR module takes the parse tree, which is a constituency tree, and the original translation rules as input and outputs the refined translation rules.

If the actual translation rules are not accessible to the system, all the RR module can do is partially address a subset of MT errors, the correction of which does not require access to the body of the rules (such as adding a word, deleting a word or changing word order), and propose a new POS sequence that would need to be integrated to the grammar by the MT system itself.

For example, given a different transfer engine and a GPSG rule syntax, the MT system would need to produce a constituency parse tree for the TL-side and convert ID rules and linear precedence as well as feature information into CF translation rules, which include dominance and linear precedence information, as well as feature constraints.***

An English-Spanish user study with the online Translation Correction Tool shows that users can detect errors with high accuracy (90%), but have a harder time classifying error given a linguistically-motivated MT error classification (72%). Asking users to locate the error and indicate if there is any word in the sentence that acts as a clue is a much simpler task and, at the same time, is much more useful for automatic rule refinement purposes. The main challenge of this part of the research proposed is to discover the appropriate MT error classification (corresponding to a set of non-technical questions) that will maximize error type classification accuracy by non-expert users.

Given the difficulty inherent in manually building a translation grammar, especially for resource-poor languages, these results are promising and indicate that building an automatic rule refinement module that takes as input such user feedback is a path worth pursuing.

Some types of errors defined in the general MT error typology (Section 4.2) cannot be distinguished automatically, and thus for the RR module to be effective and as automatic as possible, Rule Refinement operations should be defined not only based on the true nature or cause of the errors, but also on the relevant error information available to the RR module.

Given user corrections through the Translation Correction Tool, the Rule Refinement module can apply a set of well-defined RR operations to the grammar and lexicon to improve both hand-crafted and automatically learned translation grammars.

The addition of feature constraints to the automatically learned grammar brings it closer to manually written grammars. However, with the current implementation of the Rule Learner, the Rule Refinement module will still give the most leverage when combined with an automatically learned grammar.

In general, manually-written grammar rules will need to be refined to encode exceptions, whereas automatically-learned grammars will need to be refined so that they achieve the right level of generalization.

9 Proposed Research

Given the preliminary work described above, the research I propose encompasses 6 main objectives:

- Developing an effective method to elicit error information from non-expert bilingual users
- Validating the general MT error typology as well as the classification used by the Translation Correction Tool, by obtaining higher classification accuracy rates.
- Developing an automatic approach to refine translation rules from existing grammars and lexicons (both hand-crafted and automatically learned), based on elicited error information and the MT error typology. First English-to-Spanish and then Mapudungun/Quechua-to-Spanish.

- Developing interactive learning methods to find relevant minimal pairs in order to determine the appropriate rule refinement operation, the level of granularity for the new feature constraints, and in some cases the triggering context.
- Exploring Active Learning methods to optimize user time and present users with most informative minimal pairs first.
- Evaluating the performance of the resulting refined rules and lexical entries using mostly automatic evaluation metrics such as BLEU and METEOR, as well as parsimony and coverage.
- Assessing the language, user and system independence of the rule refinement operations

9.1 Training and test data

The current data sets available for English to Spanish are the typological Elicitation Corpus, which has 791 sentences and a vocabulary size of 332, and a third of the structural Elicitation Corpus, which has 104 phrases and a vocabulary size of 241. In total, there are 895 sentences and phrases already available and the size of the vocabulary combining these two corpora is 505.

From this data, I will create a development suite of about 400 sentences for which the transfer engine outputs a complete parse (S), categorizing them by error type, so that I can then split them into two sets in an error-balanced way: the true development set (dev set) and a first test set (test1), which I'm not going to look at while developing the RR system. This way I can validate the effect of the RR operations not only on the same dev test used but also on a test set that is comparable and that should allow for the same RR operations to apply.

*** This way, I will make sure that each error type I am proposing to solve in this document has at least two different examples both in the development and test sets.***

A few comparable sentences not already in the Elicitation corpora (EC) might also need to be created, to better test the effectiveness of the refinements, since the goal of the EC is to minimize repetition of structures and linguistic phenomena, rather than providing training data for MT rule refinement.

For the development stages, I will pick the sentences that are part of the user studies, since that allows me to concentrate on sentences for which no good translation is output by the MT system, and thus we can make better use of user's time.

The sentences for the next user studies will be chosen from the dev set, and will be picked according to the MT errors that they exhibit, so that the RR operations can be fully tested. To select sentences for user studies for language pairs I am not familiar with, I will use an automatic mechanism, possibly using some sort of randomization, possibly augmented with suggestions from native speakers.

Data summary:

manual grammar (50-100 rules)
automatically learned grammar (300 rules)
lexicon (10,000 entries) extracted from a large full from lexicon

development suite (400) [categorized by error]
- dev set (200) – used for development and user studies
- test1 (200) – reserved for validation

- test2 (100) – blind set, for final evaluation

Since the test set from the development suite will not really be a blind test, I will also create a wild test set (test2) with about 100 naturally occurring sentences. For testing on wild corpora, the initial requirement will be that all the sentences in the corpus are parsable (even if incorrectly) by the translation grammar, so that the different refinement operations can be tested. During the final stage of the thesis, however, testing on wild corpora without putting any restriction on the sentences (such as having to be parsable) will allow us to determine if our approach can also improve overall translation quality when not all the SL sentences are covered by the translation grammar, which is what happens in a real world situation. *** I expect to show significantly more improvement on the tame test set than on the wild test.***

The final English to Spanish vocabulary size will be of about 10,000 inflected words, which will include the 505 words from the multiple elicitation corpora, and which I will complement with high frequent Spanish words from naturally occurring newspaper corpora.

For my work, I will abstract away from morphology as much as possible, and thus both the vocabulary and the translation lexicon will contain fully inflected forms. For the language pairs where there already is a morphology component in place, I will do some experiments to assess how much further user interaction can be spared if morphological information is available. However, I will not assume that a morphology component is always available.

I will do most of the RR module development in English to Spanish, and will also conduct final experiments with a resource-poor language into Spanish. Likely choices for the resource-poor language are Mapudungun and Quechua, since an MT system is expected to be built for them into Spanish within the AVENUE project. Either of these two languages will allow me to determine whether the RR module is relatively language independent, for they are both agglutinative languages, hence significantly different from Spanish and English. Spanish is chosen as the target language since both Mapudungun and Quechua

speakers tend to speak Spanish but do not speak English.

While creating the development suite, I will be expanding the manual grammar to cover all the basic linguistic phenomena so that all sentences are fully parsable. I expect the manual grammar prior to refinement to have between 50 and 100 rules (the initial grammar had 12 rules).

I will initially focus on refining the manual grammar, and once the Rule Learner is fully implemented (with feature constraints), I will run additional experiments with the automatically learned grammar.

On top of using the manual grammar for the final evaluation described in Section 9.3.5, I will also use subsets of the manual grammar for debugging and to illustrate all the types of MT errors that the RR module can correct and as a comprehensiveness test. The results of this other kind of evaluation will also be reported as a proof-of-concept.

9.2 Elicitation Method for MT Error User Feedback

In the course of preliminary investigations described in the previous sections, I observed that there are a couple of aspects of the current TCTool implementation that can be improved to allow non-expert users to detect and classify MT errors more accurately and reliably. I plan to investigate and implement the features needed to achieve higher precision with respect to MT error detection and classification.

9.2.1 TCTool V.0n

MT Error Information

The final version of the TCTool will allow users to give us more information about MT errors without having to know any linguistics terminology (such as sense and form). In particular, the TCTool will allow users to indicate if there is any word in the target sentence that acts as a clue for that correction.

This method targets errors affecting target language (y-side) constraints, such as subject and predicate agreement, and gender agreement between nouns and their complements (determiners and adjectives) within an NP.

It is not uncommon that agreement constraints change from language to language. Examples of this are the constraint agreement on gender (in addition to number and person) between the subject and the verb in Hebrew, which only transfers to Spanish for the past participle forms, and which does not transfer for any situation to English.

Hence, instead of a linguistically-motivated MT error classification, like the one used for the first version of the TCTool, the new approach will request the user to identify clue words, e.g.: "Wrong agreement, this word has to agree with [drop-down-menu with all other words in the TL sentence].", "The word W_i is in the wrong from, but it is related to W_i'." or "The word W_i is simply and incorrect translation for SL_W (aligned to W_i) and it has no relation to it.".

For example, for the SL sentence `I believe in you` and the translation `creo en tu`, instead of asking users if they changed `tu` into `ti` because it had

the wrong form or sense, the new version of the TCTool will display the following "the word `you` can be translated as `tu`, but not in this sentence. The key word that indicated this is: [drop-down-menu with creo and en]".

Non-expert users can easily indicate that the reason `you` translates into `ti` in this sentence is the presence of the Spanish preposition `en`. This is not only a much simpler task for non-expert users but, at the same time, is much more useful for automatic rule refinement purposes. It may also prove that users can perform this task more consistently (with higher precision), though this remains to be measured.

Given the user correction `you -> tu` and the clue word (`en`), the RR module can now automatically generate a feature constraint between the appropriate constituents in the right-hand side of the grammar rule, the Y-side.

Non-expert users would have a much harder time classifying it as wrong sense vs wrong form, and even if they finally did choose between these two error types, classification accuracy would be lower. Thus, we expect the next version of the TCTool will allow users to achieve higher error classification accuracy or at least higher precision.

9.2.2 Evaluation of TCTool

The final version of the TCTool will be evaluated on how well it performs at error detection and classification.

Since the way to elicit information about MT errors will be different than in our preliminary study, we will report comparable accuracy scores which will indicate whether the new approach is more suitable for the task.

In order to allow for fair comparison, at least one more set of English-Spanish user studies need to be run (preferably both by old and new users of the TC-Tool), to be able to factor out any advantages of the final implementation due to familiarity with the task and the GUI design.

Once there is a working MT system for a resource-poor language (Mapudungun, say) and Spanish, we will also conduct user studies for the full range of corrections, based on the actual error distribution from the learned resource-poor language into Spanish. This might lead to revisions of the GUI design (e.g. to handle multi-morphemic "words").

For each new set of user studies that we run, a gold standard will be developed specifying what errors each TL sentence has as well as any extra error information that can be elicited via our improved TCTool GUI. For user studies involving a resource-poor language, external help will probably be required to develop such a gold standard.

For automatic evaluation of the Rule Refinement module, we assume that user corrections are the gold standard; these become the reference translations, and thus we abstract away from the problem of users not being true linguistic oracles.

There are effective ways to reduce user idiosyncracies when we have corrections from enough users for the same set of sentences; for example, by only taking as reliable any correction supported by 90% of the users.

9.3 Automatic Rule Refinement Module

The philosophy behind the RR module is to extend the grammar to account for exceptions not originally encoded in the translation rules, to make overly general rules more specific so as to reduce arbitrary grammar ambiguity, and to correct rule errors (e.g. constituent or word order).

In the case of automatically learned grammars, the Rule Refinement module also has the role of adding missing constraints to the context-free rules that need them.

The grammar is expected to grow as a result of the Rule Refinement process, especially when running in batch mode, due to the creation of some overly specific rules, especially when applying RR1 and RR3 (Section 5.4). However, once more data is available, the Rule Refinement module should be able to make safe generalizations and merge equivalent specific rules into one general rule.

The generalization power of the Rule Refinement approach is greatest if the refinements involve existing feature constraints (e.g. agreement constraints), since all the relevant lexical entries will already be appropriately tagged for the correct rule to apply. If the RR module needs to postulate a new binary feature attribute (to distinguish between two different senses of a word, say), however, only local improvements will be observed, which can become a bit more general, as new similar data is evaluated by users. The problem is that newly hypothesized features would not populate lexical entries, and in the absence of a generalization mechanism, this process would require one-by-one addition.

This thesis can be seen as the first step for semantic correction, however, in the sense that it annotates the specific examples corrected by users in the appropriate way, which may be used later by a system with Wordnet to make the appropriate generalizations.

9.3.1 Batch mode

In previous work on estimating user accuracy when detecting and classifying MT errors (Font-Llitjós & Carbonell, 2004), we found that an automatic Rule Refinement approach can be made sufficiently robust, even though it relies upon non-expert user feedback. Therefore, I propose to implement the set of Rule Refinement operations identified in Section 5.4 and automatically apply those for which we have all the necessary error information available.

The RR operations defined above will work when considering user corrections in isolation. However, when users perform more than two corrective actions allowed by the TCTool (add, delete, modify a word or change word order) to address a single MT error, it becomes very hard to automatically detect whether such actions are part of one single correction and should be considered together by the RR module, or are in service of correcting multiple errors. For

example when a user deletes a word and then adds a different word in a different position, it could be that the user modified a word and that the modification caused it to have to move in a different position (an example of this can be seen in Section 4.2, error type: 5.1.2.2), or it could be that s/he performed two independent corrections. The appropriate way to show that there is a correlation with the TCTool is to drag and drop the word to the right position and then modify the dragged word, but there is no guarantee that users will always do it that way.

Whenever operating in batch mode, we will take a "Tetris" approach to Rule Refinement and first fix the simpler errors, then run the RR module over the test set and see if some more errors can be fixed on a second pass.

Given specific user feedback, the RR module will first use the parse tree produced by the transfer engine[14] to trace back to the rules and lexical entries that applied and, if it has all the information required, it will determine the type of refinement required to fix the rules. If it needs to add a feature constraint between two positions, a rule covering those positions must already exist in the grammar. If such a rule does not exist in the grammar, the RR module cannot perform any refinements and just feeds the user-corrected SL-TL pair back to the Rule Learner as a new training example.

When the system is running in batch mode, the default settings are to add the constraints at the most specific level possible, namely the word. Often times, the ideal refinement would have been at the POS level, possibly with further constraints required by the triggering context. However, further refinements and generalizations on the specific constraints can only be made automatically at a later stage, when the system has more labeled examples, or when it can interact with the user in Active Learning mode.

The diagram in Figure 7 shows the set of Rule Refinement operations as given in Section 5.4. The colored nodes give an idea of which RR operations can be implemented with the batch mode and which operations require further user interaction to be able to apply. The types that appear in regular font are the ones which I believe can be refined fully automatically with the RR module in batch mode and no further user interaction is required. The types in SMALL CAPS require further user interaction, but sometimes reasonable default settings can be defined when operating in batch mode.

The types that appear underlined might require a significant amount of further user interaction and cannot be solved unless operating in interactive mode. The types in **bold** are missing crucial error information, which cannot be elicited with a reasonable amount of user interaction, and thus cannot be automatically refined under any circumstances. For such cases, the SL-TL pair is sent back to the Rule Learner as a new training example.

[14]The transfer engine may produce multiple translations, possibly generated by different parse trees, as a function of the ambiguity inherent in the grammar and the lexicon; however, users pick which sentence to correct and thus the corresponding parse tree is retrieved for RR purposes.

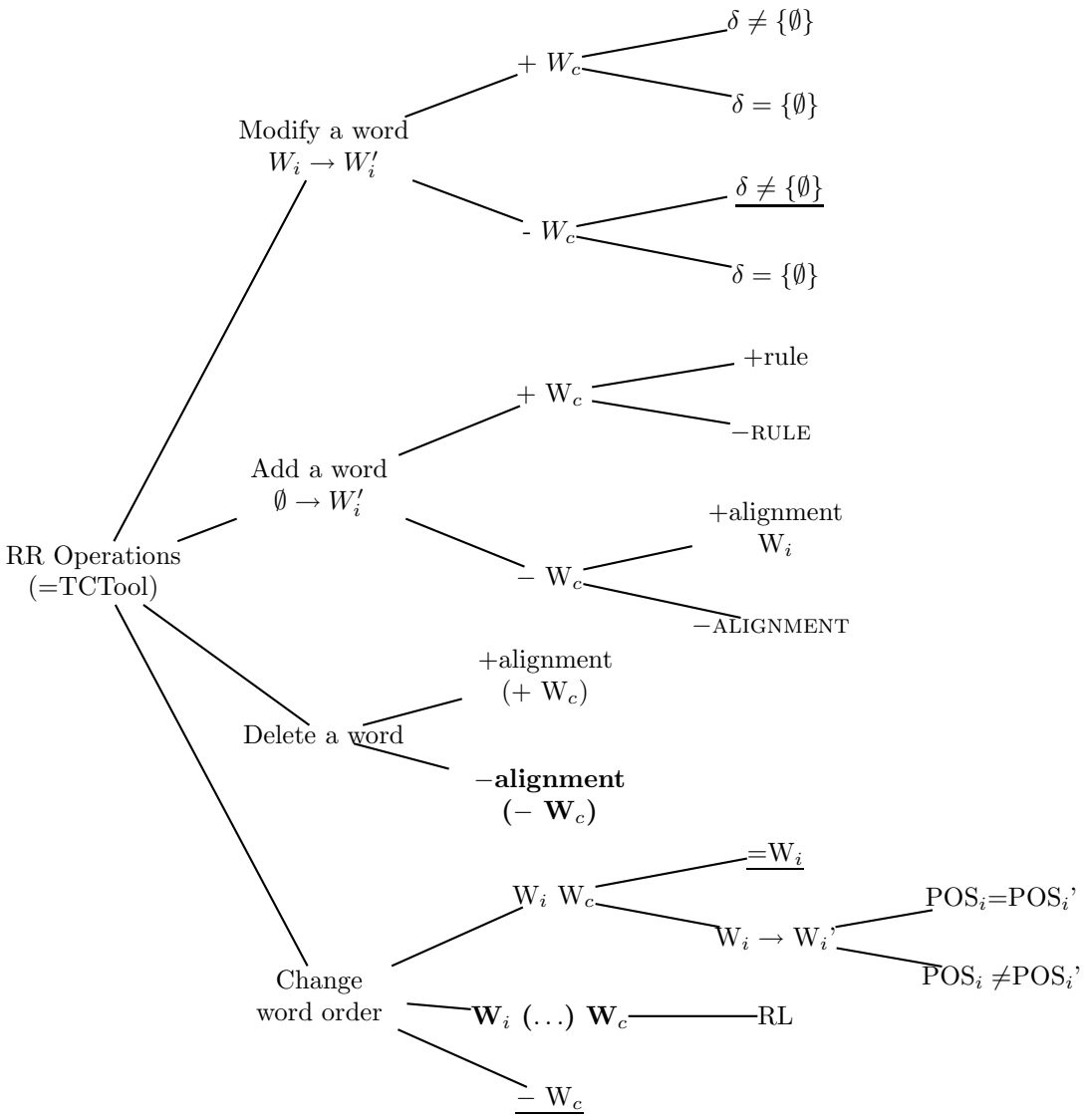

Figure 7: Diagram showing RR operations typology

After having extended the manual grammar so that it fully parses all 32 sentences in the first user study, I did a detailed analysis to find out what percentage of the errors can be dealt with fully automatically with just correction information (and in batch mode), and found that about 68%[15] could probably be refined fully automatically without extra error information or further user interaction.

For all the types of RR, there is a tradeoff between safety (most conservative, specific refinements) and generality (most general refinement, possibly too general). As a first step, I'll focus on those errors which can be corrected fully automatically with just correction information from the user, so that I can test the RR module with the simplest mode of operation first, namely batch mode. For this, I anticipate formalizing some reasonable assumptions into default settings, such as bifurcate the original (general) rule and make any changes to the duplicate (specific) rule.

Without being able to validate RR module refining hypotheses, we need to take a conservative approach and assume the original rule that needs to be refined, may apply to other sentences, and thus leave it unchanged (schemas RR1 and RR3 in Figure 6).

Ultimately, I will explore the tradeoff bewteen "safest" vs. "most general" refinement and compare both modes of operation in terms of their relationship with this tradeoff. In particular, how much does further user interaction add (in safety) to the most general batch settings, and whether there is an optimal balance.

The next section describes interactive learning methods and indicates which might be appropriate for the current task and which can be performed (types in SMALL CAPS and underlined) when running the system in interactive mode. The thesis, however, will mainly focus on the types for which all the error information is available from the initial user corrections, with the exception of user interactions that can be limited to a couple turns, so as not to exhaust the user. An example of such interactions are the ones to determine the appropriate level of granularity of a newly found constraint (e.g. word vs POS).

Hence, I propose to fully implement RR operations that appear in regular font and SMALL CAPS in the tree above and to explore the the ones that appear underlined, but the types that appear in **bold** fall outside the scope of this thesis.

Beyond the core of the RR module functionality, and once the resource-poor language to Spanish system is in place, MT error analysis will determine whether the Rule Refining operations defined in the first version of the RR module are general across more than one language, or more likely, require basic linguistic knowledge based on typology for the resource-poor language. Time permitting, we propose to incorporate typology information to reduce the potentially tedious interaction with the native informant and to guide the search over the potential rule-corrections space.

[15] Some of these errors required correct alignment information, or that a relevant minimal pair be given.

Finally, I plan to continue expanding the MT error typology described in section 4.2 and at the same time determining which error types require what kinds of rule refinement operations, and whether all or only part of such operations can be dealt with by the RR module.

9.3.2 Interactive mode

After the batch mode implementation is completed, I will proceed to implement an interactive mode of operation, which will allow the RR module to prompt users with new sentences to evaluate (and correct, if necessary) at run time, so as to obtain any additional information required to determine what is the appropriate refinement operation to be applied.

The idea is that while processing all the available information, the RR module might detect that it is missing some crucial information about the error or the type of rule refinement operation required to fix the error, and that this crucial information could be retrieved by having other minimal-pair sentences evaluated and corrected.

Since the ultimate goal is to minimize further user interaction, so as not to place a heavier burden on users, I will also explore Active Learning methods to try to optimize user time by presenting them with most informative sentences first, namely relevant minimal pairs, and will only initiate further user interaction if the RR module cannot reliably hypothesize a course of action.

Active Learning

Active Learning has been described as a method to minimize the number of examples a human annotator must label (Cohn et al., 1994) usually by processing examples in order of usefulness. Lewis and Catlett (1994) used uncertainty as a measure of usefulness. Recently, Callison-Burch (2003) has proposed Active Learning to reduce the cost of creating a corpus of labeled training examples for Statistical MT.

For our task, Active Learning allows us to make further user interaction more efficient, and asks users to evaluate (and correct if necessary) just the example sentences that will be most informative in finding out what triggered the correction, what is the RR operation suitable to fix a specific error or whether the general rules needs to either bifurcate or be made more specific. In this context, the most informative sentences to reveal the right level of generalization, such as adding constraints at the POS level or at the word level, are the ones that we call minimal pairs.

Minimal pair differences occur at the feature level, which can be calculated with the delta function as described in Section 9.3.3 below.

More formally, if EC is the Elicitation corpus, EC_c the part which is correctly translated, then for the translation pair under consideration ($x = (s, t)$), select $x_c = (s_c, t_s) \epsilon\ \mathrm{EC}_c$ so that they are minimal pairs, namely so that the feature vectors ($\overrightarrow{x}$, $\overrightarrow{x_c}$) differ minimally, ideally in just one feature.

In the future, the typological Elicitation corpus will contain feature vectors

for all the sentences. So, if there already exists a minimal pair in the Elicitation corpus, the RR module will be able to use the feature vectors that will tag each sentence pair to select the relevant minimal pair, namely the one that can shed some light on what triggered the user correction.

If, for a given sentence that is missing some crucial information, the development set does not already contain a relevant minimal pair, refining it automatically falls outside the scope of this thesis.

The following four cases illustrate the range of possible strategies that can be followed, depending on whether a minimal pair (MP) is required and available. They correspond to the four different colors in the diagram showing the RR typology (Figure 5).

1. No need for a MP evaluation [RR types in regular font]: for `I want the red car`, if users modify `*Quiero el auto roja` into `Quiero el auto rojo`, the user has already given all the information that is needed for the RR module to be able to apply the appropriate set of refinements. See the subsection about modifying a word in Section 5.5 for how this is done.

2. Relevant MP required [RR types in SMALL CAPS**]:** sometimes the correction itself does not contain all the error information required to perform the appropriate refinement automatically. In such cases, the RR module will need to search for a relevant minimal pair in the dev set, and then search the feature space to find the triggering features (next Section).

This would be the example of `I see them` (`*veo los -> los veo`), for which the RR module needs to find a minimal pair, such as `I see cars - veo autos`, and restrict the specific VP rule, with the order flipped (`[NP(obj) VP]` instead of `[VP NP(obj)]`), so that it only applies to sentences with a pronoun as their object.

For these cases, Active Learning methods can be used to explore the space of possible triggering contexts, so as to present users with the least number of minimal pair candidates for evaluation (see below). Even so, we expect this process to be an iterative one, which is likely to place a heavier burden on users.

In some of these cases, though, if running in batch mode, adopting the default settings, of assuming the most specific level is appropriate, will work as well.

3. MP not in the dev set [underlined RR types]: if there is no minimal pair for the SL-TL pair in the dev set (say, in the previous example, there were no sentence like `I see cars - veo autos` in the dev set), the RR module would need to create a feature vector from the lexical entries and grammar rules that translated the originally produced TL sentence, and instantiate different parts of the sentence differently (preferably word by word), according to this feature definition. Due to time constraints, I will not be able tackle the problem of minimal-pair generation in my thesis.

4. Relevant pair required ($\neq$ MP) [RR types in bold]: in some cases, what triggers a specific correction is the syntactic context, which cannot be expressed in terms of minimal pairs, strictly speaking, since the relevant TL sentence to allow automatic discovery of the triggering context might have as little as one word in common with the original translation.

This is the case of the overgeneralization example given in Section 4.2 for error type 4.2, `John read the book -> *A Juan leyo el libro -> Juan leyo el libro`, which corresponds to deleting a word without an alignment in the RR diagram above (Figure 7). In order to be able to discover that in Spanish `a` is only added to NPs in object positions (`Le doy el libro [a Juan] - I give [Juan] the book`), but not to subject positions (`[Juan] lee el libro - [Juan] reads the book`), the RR module would need to find an example where `Juan` appears in the object position and which contains as many words in common with the original TL sentence as possible. Sentences like `Ellos invitaron [a Juan] (They invited Juan)` and `Anna aplaudio [a Juan] (Anna applauded Juan)` might be fine candidates for a linguist to figure out that what is causing the `a` to be able to appear in front of an NP is it's grammatical role. However, such sentences have too few words in common with the original TL sentence, `A Juan leyo el libro`, for an automatic system to be able to hypothesize the right cause of the MT error, namely not taking into account the grammatical role.

On the other hand, since in this case the RR module is shooting in the dark, with respect to what is the cause of the MT error and what might be potential relevant pairs, it might take quite a few sentences until the system presents a relevant sentence pair to the user.

Therefore, this thesis will focus on the first case and explore the second case, but will not attempt to solve the last two cases.

9.3.3 Searching the Feature Space

The Elicitation corpus used in the Avenue MT system to automatically learn a translation grammar is designed to discover the relevant features underlying a language. However it is not infallible, and it will often be the case that we will not know all the relevant features for a language just from the elicited data.

When the error information available to the RR module is not enough to determine which feature, or set of features, is responsible for a particular correction, the first goal of the RR module becomes to search the feature space (of about 200 features) to determine which set of features triggered a particular correction. To do this, the RR module will need to rely on the interactive learning methods sketched above to try to discover the underlying principles and constraints of the TL language grammar, which were not elicited at the first stage of the development process, or which were not learned by the Rule Learning module.

Once the RR module has the corrected TL sentence, T1', and a minimal pair extracted from the Elicitation corpus or generated automatically, T2, it compares them to see how they differ.

For RR purposes, we define the difference between two TL minimal pair sentences as the intersection of the set of feature attributes for which they have different values, not taking into account the user correction we are seeking to account for.

This is a natural extension of the feature delta function introduced in Section 5.3, and in the general case, it can be written as follows:

$$\delta(\text{T1, T2}) = \bigcap_{i=1}^{n} \delta(\text{wT1}_i,\text{wT2}_i)$$

for all the relevant words that are different in T1 and T2 ($\text{wT1}_i \neq \text{wT2}_i$)[16] and where `n` is the length of the sentence. Note that this definition of the delta function only works for TL sentences of the same length[17]. As stated in the previous section, corrections whose triggering context requires minimal or relevant pairs of a different length, namely with a bit more structural modification, fall outside the scope of this thesis, and need to be fed back to the Rule Learner as a new training example.

The resulting delta set is used as a guidance for exploring the feature space of potentially relevant attributes, until the RR module finds the ones that triggered the correction, and can add the appropriate constraints to the relevant rules.

We propose the following algorithm to explore the relevant feature space to find the attribute(s) that trigger user corrections:

```
1. Given the user corrected TL sentence T1', look for a minimal pair, T2, in the
   dev set that users evaluate as being correct.
2. Define the delta function for all relevant words that differ between T1 and
   T2,
   a. if the delta set is non-empty, take the first attribute in the delta set
      (a1) and modify minimally T2 to obtain T2* so that the feature value for
      a1 is the same as in T1.
   b. if the delta set is empty, postulate a new binary feature attribute feat_i
      and add it to the list of triggering features. i++. STOP.
3. Ask users to evaluate the modified version of T2, T2*, and if it requires the
   same correction as T1, add a1 to the list of trigger features.
4. T2 = T2*. Go back to 2.
```

Note that each iteration through step 2.a. will result in deleting an attribute of the δ set until it is empty (2.b), since its value is changed in T2 to be the same as in T1, and thus the next time the δ function is defined, the first attribute will not have differing values any more, and will not be part of the δ set.

The ultimate goal is to automate this process as much as possible. To generate valid minimal pair examples (step 1), further user interaction will be needed, probably through more than one iteration, to make sure T2 is a minimal pair which illustrates the linguistic phenomenon that accounts for the MT error.

Since we are comparing minimal pairs (step 2), the delta function is reduced to comparing just the words that are different between the two TL sentences but are aligned to the same SL word, or are in the same position, and we do not need to calculate the delta functions for differing words that are not relevant,

[16] The delta function is reduced to comparing just the words that are different between the two TL sentences but are aligned to the same SL word, or are in the same position, and we do not need to calculate the delta functions for differing words that are not relevant, such as δ(`la,casa`)

[17] If we add or delete W_i in T2 , then we don't have a counterpart in T1 to compare it with.

such as δ(la,casa). In some cases, the fact that two words do not have the same POS is actually the cause of the error and thus we need to be able to account for that as well (e.g. vi autos vs. los vi). A pre-processing step is added to the lexicon where, for each entry, the POS is added as a y-side feature value constraint (e.g. ((y0 pos) = pron).

An example to illustrate minimal pair comparison and feature space exploration using the algorithm outlined above follows. If the TL sentence Juan vio los muchachos (for the SL sentence John saw the boys) is given to users with the TCTool, users will say that in order for the translation (T1) to be correct, an a is missing in front of los muchachos; and thus the RR module will get the following corrected TL sentence Juan vio a los muchachos (T1').

(1) Now, we need to look for a minimal pair that users will evaluate as being correct. In this case, the Elicitation corpus (briefly described in section 4.3) already contains a good candidate for T2, namely Juan vio la casa, and we are spared extra Active Learning.

(2) Next, the RR module needs to discover what triggered the user correction. Thus, as a first step, it calculates the delta function of the two sentences evaluated by users. Juan vio los muchachos (T1) and Juan vio la casa (T2) have the following delta function:

$$\delta(\texttt{(Juan vio los muchachos),(Juan vio la casa)}) =$$
$$= \bigcap(\delta\texttt{(los,la)}, \delta\texttt{(muchachos,casa)}) = \{\texttt{gender, number}\}$$

Note that since the value of both these deltas is {gender, number}, their intersection is also {gender, number}.

(2.a) Next, the RR module needs to change the differing words in T2 to eliminate feature attributes from the delta set, one by one, so that it becomes clear which feature attribute is responsible for the different behavior. This is done by making their value the same for the two TL sentences. And thus, the RR module picks the first feature attribute from the delta set, in this case gender, and changes T2 minimally to ensure that the value for that attribute is the same for all words that are different between the two TL sentences: Juan vio los muchachos (T1) and Juan vio el árbol (T2*).

Often several modifications (MPs) are possible, and finding the right one, might take a couple of interactions with the user, until the RR module finds a sentence with the same structure and the most identical words but which still needs not be corrected.

(3) At this point, we present T2* to the users to check whether they make the same correction than in T1. In this case users evaluate T2* as being correct, and no correction is made. Thus, the algorithm does nothing and goes to the next step (4), which takes us back to 2.

(2.a) Now the delta set is reduced to having just one attribute, since the new sentence pair has the same gender: δ((los muchachos),(el árbol)) = {number} At this point, the RR module needs to further modify T2* to make its object have the same number as the direct object in T1: Juan vio los

muchachos (T1) and Juan vio los árboles (T2*), which again, is evaluated by users as being correct (steps 3 & 4).

(2.b) This time the delta function results in an empty delta set: δ(muchachos,árboles) = {}.

If one of these features had accounted for the presence or absence of "a" in the translation, we would bifurcate the rule based on this feature. Alas, since it did not, the RR module determines that the existing feature set is insufficient to explain the difference (of when to use "a") and therefore postulates a new feature, let's call it feat_n, and stops.

As shown in Section 5.5 for "Adding a word", once the RR module has determined the triggering features, it proceeds to refine the relevant grammar and lexical rules by adding the appropriate feature constraints. In this case, the RR module bifurcates the rule, one with value + and the other with value - for this new feature. The lexicon will later need to code for the new feature as well (btw. feat_1 corresponds to animacy, though the system does not learn external meanings of features).

***

9.3.4 Adaptation to Resoruce-poor Language

Even though, extra effort will be put to implement the proposed approach in the most general way, I expect some minor adaptations will be required to both the TCTool and the RR module when moving to a new language pair.

Resource-poor language does not mean that we assume there is no resource whatsoever for that language. In fact, for my approach, I assume there already is a basic, initial lexicon with some feature information. When moving to an MT system to translate a Resource-poor language into Spanish, then I will need to build such a lexicon. For initial, smaller experiments, a small manual lexicon with features will be built with the help of bilingual users. For final experiments, I expect to use a larger lexicon, which will be automatically learned using the AVENUE Rule Learner (Probst et al., 2002), as a starting point for the Refinement Module.

The error typology described in Section 4.2 is somewhat hardwired into the methods implemented and proposed. However, given new error types, I believe that it will be fairly straightforward to extend the error typology to incorporate them. For example, if the translation direction was Spanish → Mapudungun, say, several words in the SL would need to be merged into one word in the TL. Right now, even though the TCTool does not have a merge operation, this information can be extracted from the log files.[18]

An illustration of this follows. if the SL sentence was Yo vi al muchacho en la casa (I saw the boy in the house), the system's translation in Mapudungun was *Iñche*-I *pefiñ*-saw *pichi*-small *wentru*-man *meu*-in *ruka*-house,

[18]The RR module has access to the original translation with alignment information as well as the final correction, with the alignments fixed, as well as the correction history.

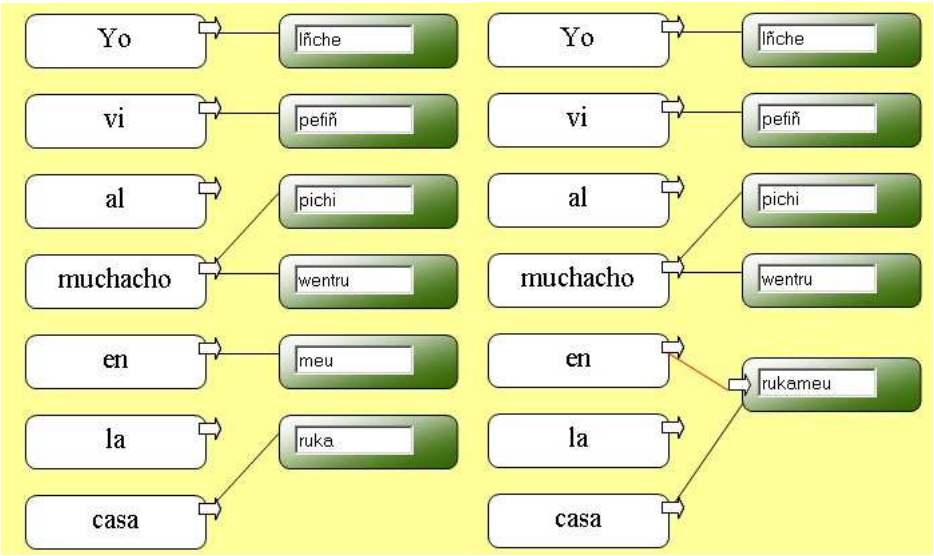

Figure 8: Left - TCTool screen shot showing initial alignments for the Spanish sentence `I saw the boy in the house` translated into Mapudungun. Right - TCTool screen shot showing the final corrected sentence and alignments for the same sentence pair.

the original alignments were as shown on the left in Figure 8, and the user edited *ruka* into *rukameu*, dragged *meu* to the trash, and fixed the alignments to be as shown on the right in Figure 8, where both `casa` and `en` are aligned to *rukameu*, this would effectively give us the same information than if a merge operation was already in place.

Naturally, if this becomes a frequent action required to correct between a language pair and direction I am working on, the best solution is to add a merge operation to the online GUI, and that way the information required for the appropriate refinement, would be given directly by the users action.

In general, given more than one correction per sentence, we assume that different corrections to different words, are due to different errors. This allows us to take a Tetris approach to automatic rule refinement and deal with one error at a time.

This approach can probably not handle structural divergences well, since typically for such cases the MT error spreads out throughout the SL sentence (see example below). However, it is compatible with a divergence module and provides with manually corrected information.

An example of structual divergence can be seen in:
SL: `He danced her out of the room`
TL: * *La*-`her` *bailó*-`he-danced` *fuera*-`out` *de*-`of` *la*-`the` *habitación*-`room`
TL′: *La*-`her` *sacó*-`he-took-out` *de*-`of` *la*-`the` *habitación* `-room` *bailando*-`dancing`.

Where *fuera*(PREP) becomes *sacó*(V) and *bailó*(V), *bailando*(ADV), and

unless the user drags *fuera* to the second position and edits it to *sacó*, and then drags *bailó* to the last position and edits it to *bailando*, there is no way to even tell that *fuera* and *sacó* are related. Even if users did "the right thing", there would still be no way to automatically tell that the two corrections (*fuera-sacó* and *bailó-bailando*) are related.

Therefore, in the case of structural divergences, performing refinement operations individually for each part of the error would most likely bring the overall translation quality of the test set down, which would prevent the refinement operations from applying. Even if for such cases we hypothesized that there is a divergence, the solution would be the same; namely, to feed the corrected translation pair back to the Rule Learner as a new training example. This effectively provides our system with a high quality, manual translation that is impossible to obtain fully automatically. Note that the result of this would be comparable to other structural divergences modules, which detect structural divergences and then mark them for further processing.

The example of agglutinative morphology described above (*ruka+me*), however, would be easy to detect given the original and fixed alignments, and could probably be refined automatically, either with a lexicalized translation rule, or more appropriately by creating a post-processing rule, which would attach *meu* to the preceding N, when it appeared in a sentence. If a morphology module were available, that information could be fed into it.

***

9.3.5 Evaluation of RR Module

MT errors can come from different sources (grammar, lexicon, decoder, etc.), however, I will only tackle the ones concerning the grammar and the lexicon. The goal of the RR evaluation is to focus on sentences for which the MT system does not output a correct translation (anywhere in the final list of translations) so that after RR, a correct translation does appear somewhere in the final list, without growing the size of the final list exponentially. Thus, the underlying goal is to increase translation quality without adding more ambiguity than strictly necessary.

The improvement will be measured with the number of correct translations that make it to the final list of translations (for which the grammar found a complete parse), but will not try to improve decoder choices. If the correct translation is already one of the alternative translations output by the MT system, the RR module will do nothing. It is not the RR module's goal to try to change constraints or the order of the rules in the grammar to boost up the translation in the final candidate list.

A second measure will be the size of the final translation candidate list. If the application of any given RR operation significantly increases the final list (and/or other final lists for previously seen examples), then there probably needs to be some further refinement to constrain its application to allow for the ambiguity that is strictly necessary to generate the set of correct sentences only.

In sum, the RR module will be evaluated in terms of translation accuracy, coverage and parsimony.

Regarding the first two evaluation measures, the oracle experiment described in Section 7 shows that existing automatic MT evaluation metrics are sensitive enough to discriminate between raw MT output and corrected MT output, even for a small test set (32 sentences) and two reference translations.

Thus, I propose using such automatic evaluation metrics to determine improvements in translation accuracy and coverage. Since the goal is to directly optimize the RR module to achieve higher translation quality, I need to detect improvement (or degradation) automatically. In order to do that, I need to automatically generate a hypothesis file for each system and a set of reference translations.

For the raw MT output, the best translation hypothesis is already picked by users, since their task is to pick the translation that is correct or that requires the least amount of corrections. For the refined MT output, we can use the sentence-level METEOR scores to pick best translation candidate from the list.

On the other hand, we can take user corrections to be the translation references, since they are precisely what we are trying to approximate with the RR module. For each user, each sentence evaluated as correct and all the corrected sentences will immediately be part of one set of reference translations.

Once we have the two hypothesis files, one per system, we can run all automatic evaluation metrics (BLEU, METEOR and NIST) using as many reference translation sets (user corrections) as we have available. For a brief description of how these metrics work, see Section 7.

Automatic evaluation metrics take care of both accuracy (precision) and coverage (recall), but say nothing about parsimony. In order to measure parsimony, we need to monitor the size of the translation candidate list, the goal being to either decrease or at least not increase the final candidate list by more than strictly necessary.

Sometimes, there might be an agreement constraint lacking, and thus all the different alternatives in the lexicon will be generated and make it to the final candidate list. For example, if we were translating "the red car" and the NP rule did not have any gender agreement constraints, the final candidate list would contain the correct translation plus all other possible lexical combinations (`el auto rojo`, `*el auto roja`, `**la auto roja`, `*la auto rojo`). In this case, the goal of the RR module is not to increase translation quality, but rather to reduce ambiguity and the size of the list of translation candidates by tightening the grammar.

In other cases, a general rule will need to bifurcate and the duplicate be made more specific. Refining the specific rule will result into higher translation accuracy but will also effectively double the size of the final candidate list, since now both the general rule and the specific rule will apply (schema RR1 in Figure 6). Thus, in this case, the RR module also needs to make sure to add a blocking constraint to the general rule, so that it does not apply to cases where we already know it should not apply (schema RR3). This way we are increasing accuracy but not just at the expense of increasing ambiguity.

Finally, even though all final experiments will be run with real users, I will also initially run experiments with a super user who does all the appropriate corrections in the way expected by the TCTool and the RR module, so that I can have an upper-bound for how well the system can perform under ideal circumstances.

10 Research plan

I intend to complete the work proposed above according to the following timeline:

- November 2004: Presentation of this proposal
- November 2004: Finish creating development suit and expand eng2spa manual grammar and lexicon
- November - December 2004: English-Spanish user studies for TCTool v.02
- December 2004: Finalize TCTool design and implementation and user studies data analysis
- January - March 2005: Implement 1st prototype of RR module for English-Spanish (batch mode)
- April 2005: Evaluate RR module wrt. user corrections from eng2spa user studies
- May - June 2005: Investigate Interactive and Active Learning methods to find relevant minimal pairs
- July - August 2005: Develop interactive RR module prototype
- September 2005: Elaborate Mapudungun/Quechua[19]-Spanish data sets and correction gold standard (might need to subcontract)
- October - November 2005: If no MT system exists for a resource-poor language to Spanish, write a small manual grammar and create a lexicon.
- November 2005: Design and set-up user study for Mapudungun/Quechua-Spanish
- December 2005: Run Mapudungun/Quechua-Spanish user study for a smaller number of users
- January 2006: User study data analysis
- February - March 2006: Adapt RR module to new language pair
- February - April 2006: Wrap up and thesis writing
- May 2006: Thesis defense

[19] Or other resource-poor language for which we have an MT system working

11 Resulting contributions

As a result of the work I have proposed, I expect to make the following contribution to research in Machine Translation, post-editing and natural language processing:

- An efficient online GUI to display translations and alignments and solicit pinpoint fixes from non-expert bilingual users.
- An MT error typology and mapping between user corrections and rule repair operators.
- An expandable set of rule-refinement operators, triggered by user corrections.
- A repair hypotheses management system, able to keep different candidate rule repairs for later confirmation or rejection.
- An analysis of the effects of automatic rule refinements on different types of grammars.
- A mechanism to automatically evaluate automatic RR wrt. user corrections.

References

Allen, J. (2003). *Post-editing.* ed. Harold Somers. Benjamins Translation Library, 35.

Allen, J., & Hogan, C. (2000). Toward the Development of a Postediting Module for Raw Machine Translation Output: A Controlled Language Prespective. *Third International Controlled Language Applications CLAW*, 62–71.

Brill, E. (2003). Automatic Grammar Induction and Parsing Free Text: A Transformation-Based Approach. *ACL*, 259–265.

Callison-Burch, C. (2003). Active Learning for Statistical Machine Translation. PhD Thesis Proposal, Institute for Communicating and Collaborative Systems, Shcool of Informatics, University of Edinburgh.

Callison-Burch, C., Bannard, C., & Schroeder, J. (2004). Improving statistical translation through editing. *European Association for Machine Translation (EAMT) Workshop.*

Charniak, E. (2000). A Maximum-Entropy-Inspired Parser. *North American chapter of the Association for Computational Linguistics (NAACL).*

Cohn, D., Atlas, L., & Ladner, R. (1994). Improving generalization with active learning. *Machine Learning, 15(2)*, 201–221.

Corston-Oliver, S., & Gamon, M. (2003). Combining decision trees and transformation-based learning to correct transferred linguistic representations. *Proceedings of MT Summit 2003.*

Flanagan, M. (1994). Error Classification for MT Evaluation. *Proceedings of AMTA 94*, 65–72.

Font-Llitjós, A., & Carbonell, J. (2004). The Translation Correction Tool: English-Spanish user studies. *Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC).*

Font-Llitjós, A., Probst, K., & Carbonell, J. (2004). Error Analysis of Two Types of Grammar for the Purpose of Automatic Rule Refinement. *Proceedings of the Association of Machine Translation of the Americas (AMTA).*

Gavaldà, M. (2000). *Growing Semantic Grammars.* Doctoral dissertation, Carnegie Mellon University.

Hutchins, J. W., & Somers, H. L. (1992). An Introduction to Machine Translation. *Academic Press, London.*

Imamura, K., Sumita, E., & Matsumoto, Y. (2003). Feedback cleaning of Machine Translation Rules Using Automatic Evaluation. *ACL-03: 41st Annual Meeting of the Association for Computational Lingusitics*, 447–454.

Lavie, A., Sagae, K., & Jayaraman, S. (2004). The Significance of Recall in Automatic Metrics for MT Evaluation. *Proceedings of the Association of Machine Translation of the Americas (AMTA).*

Lavie, A., Vogel, S., Peterson, E., Probst, K., Font-Llitjós, A., Reynolds, R., Carbonell, J., & Cohen, R. (2003). Experiments with a Hindi-to-English Transfer-based MT System under a Miserly Data Scenario. *ACM Transactions on Asian Language Information Processing (TALIP), 2.*

Lehman, J. (1989). *Adaptive Parsing: Self-Extending Natural Language Interfaces.* Doctoral dissertation, Carnegie Mellon University.

Lewis, D., & Catlett, J. (1994). Heterogeneous uncertainty sampling for supervised learning. *In Proceedings of the Eleventh International Conference on Machine Learning (ICML-94)*, 148–156.

Lin, Y.-C., Chiang, T.-H., & Su, K.-Y. (1994). Automatic Model Refinement - with an application to tagging. *COLING-94, 15th International Conference on Computational Linguistics*, 148–152.

Menezes, A., & Richardson, S. D. (2001). A best-first alignment algorithm for automatic extraction of transfer mappings from bilingual corpora. *Workshop on Example-Based Machine Translation, in MT Summit VIII*, 35–42.

Monson, C., Lavie, A., Carbonell, J., & Levin, L. (2004). Unsupervised Induction of Natural Language Morphology Inflection Classes. *Proceedings of the Workshop of the ACL Special Interest Group in Computational Phonology (SIGPHON).*

Naruedomkul, K. (2001). *Generate and repair machine translation.* Doctoral dissertation, University of Regina, Canada. PhD Thesis.

Niessen, S., Och, F. J., Leusch, G., & Ney, H. (2000). An Evaluation Tool for Machine Translation: Fast Evaluation for MT Research. *LREC*, 39–45.

NIST (2002). Automatic Evaluation of Machine Translation Quality Using N-gram Co-Occurrence Statistics. *NIST REPORT http://www.nist.gov/speech/tests/mt/doc/ngramstudy.*

Papineni, K., Roukos, S., Ward, T., & Zhu, W.-J. (2001). BLEU: A Method for Automatic Evaluation of Machine Translation. *IBM Research Report RC22176 (W0109-022).*

Peterson, E. (2002). Adapting a Transfer Engine for Rapid Machine Translation Development. *Master's thesis.*

Probst, K., Brown, R., Carbonell, J., Lavie, A., Levin, L., & Peterson, E. (2001). Design and Implementation of Controlled Elicitation for Machine Translation of Low-density Languages. *Proceedings of the MT2010 workshop at MT Summit 2001.*

Probst, K., Levin, L., Peterson, E., Lavie, A., & Carbonell, J. (2002). MT for Resource-Poor Languages Using Elicitation-Based Learning of Syntactic Transfer Rules. *Machine Translation Journal, vol. 17, No. 4. Special Issue on Embedded MT Systems. Part 3: Elicitation MT*, 245–270.

Su, K.-Y., Chang, J.-S., & Hsu, Y.-L. U. (1995). A corpus-based statistics-oriented two-way design for parameterized MT systems: Rationale, Architecture and Training issues. *TMI-95, 6th Theoretical and Methodological Issues in Machine Translation*, 334–353.

Veale, T., & Way, A. (1997). Gaijin: A Bootstrapping Approach to Example-Based Machine Translation. *Recent Advances in Natural Language International Conference*, 239–244.

White, J., O'Connell, T., & O'Mara, F. (1994). The ARPA MT Evaluation Methodologies: Evaluation, Lessons, and Future Approaches. *Proceedings of AMTA 94*, 193–205.

Yamada, S., Nakaiwa, H., Ogura, K., & Ikehara, S. (1995). A Method of Automatically Adapting a MT System to Different Domains. *TMI-95: Sixth International Conference on Theoretical and Methodological Issues in Machine Translation*, 303–310.

Zhang, Y., Vogel, S., & Waibel, A. (2004). Interpreting Bleu/NIST scores: How much improvement do we need to have a better system? *Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC).*