A Framework for Interactive and Automatic Refinement of Transfer-based Machine Translation

Ariadna Font Llitjós, Jaime G Carbonell and Alon Lavie

Language Technologies Institute, Carnegie Mellon University
5000 Forbes Ave.
Pittsburgh PA 15213
USA
aria@cs.cmu.edu

Abstract. Most current Machine Translation (MT) systems do not improve with feedback from post-editors beyond the addition of corrected translations to parallel training data (for statistical and example-base MT) or to a memory database. Rule based systems to date improve only via manual debugging. In contrast, we propose a largely automated method for capturing more information from human post-editors, so that corrections may be performed automatically to translation grammar rules and lexical entries. This paper introduces a general framework for incorporating a refinement module into rule-based transfer MT systems. This framework allows for generalizing post-editing efforts in an effective way, by identifying and correcting rules semi-automatically on order to improve coverage and overall translation quality.

1. Introduction

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. Examples of resource poor languages are Quechua and Mapudungun, which contrast with languages that have more economic, and therefore also electronic, resources, such as Spanish and English.

Rule-based transfer MT systems are the only feasible solution for resource-poor scenarios. Developing and expanding such systems manually can, however, prove very costly and time consuming. On the other hand, finding trained computational linguists with knowledge of resource-poor languages is a real challenge. Moreover, if the translation rules are written manually, no matter how many rules there already are, coverage and accuracy can always be increased. If they are automatically learned, they might be either too general or too specific. In both cases, the translation rules can be refined to account for new data. The goal of our research is to generalize post-editing efforts in an effective way, by identifying and correcting rules semi-automatically in order to improve coverage and overall translation quality.

In this paper, we introduce a novel approach that proposes an MT module for automatically refining translation rules based on the feedback provided by bilingual speakers.

There are two main challenges in this approach. First, the elicitation of accurate correction information from non-expert bilingual speakers. Second, the automatic refinement of existing translation rules, given a corrected and word-aligned translation pair, and information about the MT errors.

The approach described in this paper automatically determines the appropriate rule refinement operations that need to be applied to a grammar and a lexicon in order for the system to output the correct translation, as given by the native speaker.

The resulting refinements and extensions can therefore apply not only to the translation instance corrected by the user, but also to other similar cases where the same error would be encountered.

2. Related Work

2.1. On Post-editing to Improve MT

Post-editing has often been defined as the correction of MT output by human linguists or editors. In the case of native and minority languages on which we are working, the editors are actually bilingual speakers with no expertise in linguistics or translation, 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).

The minimal correction method we are proposing for the task of rule refinement involves grammar correctness and fluency, in addition to meaning preservation. Stylistic changes are not considered minimal post-editing.

Some researchers have looked at ways of including user feedback in the MT loop. 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. They 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 parameters 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 case of languages with limited data, such a system is not feasible, though, 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 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 score of an evaluation corpus. In contrast to filtering out incorrect or redundant rules, we propose to actually refine the translation rules themselves, by editing valid but inaccurate rules that might be lacking a constraint, for example.

2.2. On Rule Refinement

The idea of rule adaptation to correct or expand an initial set of rules is an appealing one and researchers have indeed 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.

Brill (1993) introduced 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.

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 the paper.

In sum, even though adaptation has been researched for MT and other natural language processing applications before, to this day, work so far has not attempted to refine the translation rules themselves, and thus the framework described in this paper constitutes an interesting and novel approach to automatically refine and expand MT systems.

3. Automating the Post-editing process

Current solutions to improve MT output are limited to manually correcting the output and, in the best-case scenario, to some post-processing to alleviate the tedious task of manual post-editing by correcting the most frequent errors beforehand (Allen & Hogan, 2000). Currently, there exists no solution to fully automating the post-editing process.

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 correcting the source of the error 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 access to translation rules that can be refined.

Therefore, the approach proposed by this framework 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 these corrections back into the MT system.

4. Elicitation of Translation Correction Information

Even in resource-poor contexts, there is usually at least one resource available, namely, bilingual speakers. Our approach exploits this fact and relies on non-expert bilingual users to extract as much accurate information as possible to determine error location and cause, which can then be used by the Rule Refinement (RR) module.

In order to elicit MT error information from naïve speakers reliably, we designed and implemented a graphic user interface, called the Translation Correction Tool (TCTool), that is intuitive and easy to use and that does not assume any knowledge about translation or linguistics. For details on how the TCTool works to elicit translation error information, please refer to Font-Llitjós and Carbonell (2004).

A set of English-Spanish user studies showed that bilingual speakers with no linguistics or translation skills, are able to use the TCTool to evaluate and correct MT output with 90% and 72% accuracy, respectively (Font-Llitjós and Carbonell 2004).

Given the evidence that non-expert bilingual speaker judgments and corrections of MT output are reliable, automating a rule refinement mechanism based on this information becomes an option.[1]

5. MT Error typology

As part of the initial research mostly based on English to Spanish translation, a preliminary MT error typology was defined, and it is shown in a simplified form in Figure 1.

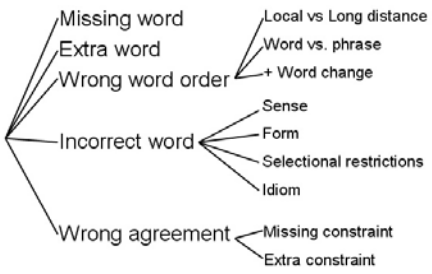


Figure 1: Initial MT Error Typology

[1] To reduce noise, a threshold can be set so that only if 90% of the speakers agree on any particular correction is the information considered reliable.

Not coincidentally, these errors nicely correspond to correction actions that can be performed by bilingual speakers when using the Translation Correction Tool.

6. Rule Refinement Approach

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

6.1. Formalizing Error Information

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 target language (TL) sentences, i.e. translations, as vectors of words from 1 to *n* (sentence length), indexed from 1 to *m* (corpus length) $TL_m = (W_1,...W_i,...W_n)$ and the corrected sentences (TL') as follows:

$TL_m{}' = (W_1,...W_i{}',...W_c,...W_{n'})$ 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 a word that provides a 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 coche roja* (the red car), W_c should be instantiated to *coche*, 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*] *cayó*, where the plural is implied by the conjoined NP).

W_c is not always present and it can be before or after W_i. They can be contiguous or separated by one or more words.

The TCTool is designed so that if such a clue word were present in the sentence, it would be easy for a non-expert bilingual speaker to give us this information.

6.2. 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 correction 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.[2]

We call this the feature delta function and it can be written as $\delta(W_i, W_i')$.

The resulting δ set can be a single 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, this 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(comían,comías) = {person, number}.[3]

If the δ set is empty, this 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) = {Ø}[4], since the lexical entries in our grammar are not marked for animacy.

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

[3] *comían* is 3rd person plural and *comías* 2nd person singular.

[4] In Spanish, *women* and *guitar* are both singular, feminine nouns.

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

6.3. Rule Refinement Schemata

In general, if the new refined rule 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 a general, default rule, R1, and a more specific rule, R2.

Figure 2 illustrates the two types of RR operations that we anticipate being able to deal with our system. If user corrections require a brand new rule not already in the original grammar, this is outside the scope of the framework. In this case, in our MT system, an automatic rule learner (Probst et al., 2002) would be invoked, instead.

In the first refinement schema shown in Figure 2 (RS1), 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.

The second rule refinement schema (RS2) represents the case when the original grammar rule (GR0) bifurcates into a general rule, GR1, which should apply by default, and a more specific rule, GR2, perhaps with a different word order.

In order to prevent the application of the general rule (GR1) to the current translation pair, a blocking constraint is added to it. In the case of a binary constraint, the RR module would assign it value −.

At the same time, the specific rule needs to be applied in the special cases only, and not in the general case, and thus the same binary constraint will also be added to GR2 but with value +.

An instantiation of when it is appropriate to apply this schema can be found in object pronouns in Spanish. Spanish object pronouns often appear in a pre-verbal position (*I saw you* → **vi te* → *te vi*), instead of following the verb like other object NPs, and thus the VP rule ([V $NP_{(pron\ -)}$]) would need to be bifurcated into a rule like the original but with a new constraint to block its application when the object NP is a pronoun (thus decreasing the ambiguity of the refined grammar), and a more specific rule with the order flipped and a constraint enforcing its application to TL sentences where the object is realized with a pronoun (VP → $[NP_{(pron\ +)}$ V]).

The constraint added to the more specific rule (GR2) enforces that the lexical entry be tagged as + (this is done with the use of =c), so that if the lexical entry is underspecified with respect to *constr*, only the general rule (GR1) will apply.

Grammar
Refine
RS1: GR0 → GR1 [=GR0 + constr]
Cov(GR0) > Cov(GR1)
Bifurcate
RS2: GR0 → GR1 [=GR0 + constr = −]
→ GR2 [≈GR0 + constr =c +]
Cov(GR0) ≤ Cov(GR1&GR2)
Lexicon
Refine
RS3: Lex0 → Lex1 [=Lex0 + constr]
Bifurcate
RS4: Lex0 → Lex1 [=Lex0 + constr = −]
→ Lex2 [≠ TLword + constr = +]

RS5: Ø → Lex1

Figure 2: Main types of Refinement Schemata (RS) for grammar rules (GR) and lexical entries (Lex), and their effect on the rule's coverage (Cov).

The reason the coverage (Cov) of GR0 might be smaller than the coverage of GR1 plus GR2 in RS2 is that the modification undergone by GR2 might allow different kinds of TL sentences to be correctly generated.

The first lexical refinement schema is equivalent to the first grammar schema. One possible instantiation of SR3 is when adding a constraint (*feat0* = +) to all animated nouns, such as *woman*, *boy*, *Mary*, and in contrast with *trees*, *book* and *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*).

RS4 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 bifurcates Lex0 into Lex1 and Lex2 and changes the TL side of Lex2 to match the translation proposed by the user. For example, if bilingual speakers 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, RS5 represents the schema 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.

7. Refinement Coverage

In order to determine the refinement space, we organized the rule refinement cases according to the type of action users can perform to correct a sentence using the TCTool (add, delete or modify a word, change word order), and then according to what error information is available to the RR module at refinement time.

The tree in Figure 3 sketches the different Rule Refinement conditions identified so far.

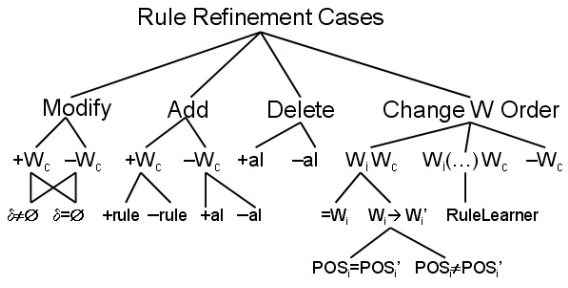


Figure 3: Different Rule Refinement cases, given TCTool correction actions and error information available.

When the user identifies a triggering word, indicated as "+W_c" in Figure 3, 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 further user interaction to be solvable, possibly using Active Learning techniques to minimize the number of translation pairs that the user needs to correct.

When there is an alignment to the corrected word, this is indicated with "+al" in the tree above; "−al" indicates lack of alignment to the corrected word.

8. 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 grammar ambiguity, and to correct rule errors.

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

Given specific user feedback, the RR module will first use the parse tree produced by the transfer engine[5] 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.

The refinement schemata defined in Figure 2 will work when considering user corrections in isolation. However, when users perform more than one correction action allowed by the TCTool to address a single MT error, it

[5] 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.

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 intended to correct 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, 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.

In any case, and as a starting point, we adopt the Occam's razor principle and assume that when affecting the same word, different correction actions are due to the same error.

The generalization power of the Rule Refinement approach is greatest if the refinements involve existing feature constraints (e.g. gender, number, person), 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), only local improvements will be observed. 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 work can be seen as the first step towards semantic correction, 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.

8.1. Batch Mode vs. Interactive Mode

One of the main goals of this framework is to automate the refinement process as much as possible. And since initial examination of the English-Spanish data shows that a significant amount of sentences can be automatically refined with just the correction and error information elicited by the TCTool, we are currently implementing a RR module that operates in batch mode.

In batch mode, however, it is not always possible to automatically decide whether a refinement or a bifurcation of the original rule is the appropriate operation. When no evidence is available to determine that the original rule can never be applied, the system adopts a conservative approach and applies the bifurcate operation, leaving the original unchanged and refining the duplicate rule.

Moreover, 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. Sometimes, the ideal refinement would have been at the POS level. But 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.

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.

An interactive mode of operation allows the RR module to prompt users with new sentences to evaluate at run time, so as to obtain any additional information required to determine the appropriate refinement operation that can be applied reliably.

To minimize further user interaction as much as possible, Active Learning methods can be used to optimize user time by presenting them with most informative sentences first.

9. Refinement simulation

A comprehensive set of end-to-end simulations has been developed to cover all the refinement cases identified in Figure 3. For illustration purposes and to provide with better insight into the refinement process, one simulation is described below.

The simulation example requires a word to be changed into a different position in the TL sentence and to be slightly modified. This corresponds to the first branch of the subtree rooted at "Change W Order" in Figure 3, as

well as the branch rooted at the "Modify" node following by "+W_c" and "δ=∅".

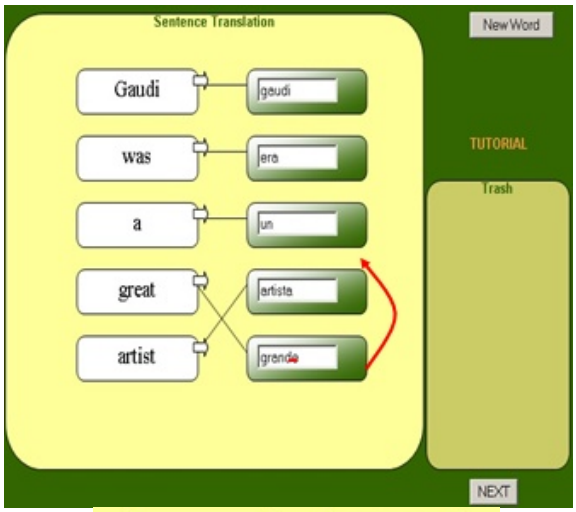
SL: *Gaudí was a great artist*
TL: *Gaudí era un artista grande*
User corrections:
**Gaudí era un artista grande*
Gaudí era un gran artista

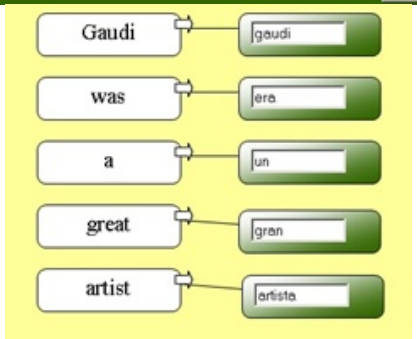
Figure 4: Source language (SL) sentence, translation in Spanish as it is output by the MT system (TL) and correction information given by the user.

Simulation steps

1. Error Information Elicitation

Given the SL and TL sentences in Figure 4, a bilingual speaker will move *grande* before *artista* and change *grande* to *gran*, as can be seen from the two snapshots showing the initial and final screens of the TCTool.





2. Variable Instantiation

All the actions users perform with the TCTool are properly logged, and thus by parsing a session log file, the system can instantiate the error information variables with the logged information. In this example, the log file will contain the following correction actions allowing the three variable instantiations indicated below:

1. Word order change: *grande* is moved in front of *artista*) → W_i = grande
2. Edited *grande* into *gran* → W_i' = gran
3. User selected the following option from a menu: "The word *great* can be translated as *grande* but not in this sentence. The key word in the sentence that indicates this is [*artista*]".

→ W_c = artista

In this case, even if the user had not identified a W_c, the RR module could still refine the grammar and lexicon automatically, but it would not be able to make the refined grammar tighter.

3. Retrieve Relevant Lexical Entries

Assuming there is no entry for [great → gran] in the lexicon, the system will apply RS6 (Figure 2) and duplicate the lexical entry for [great → grande] and change TL side to *gran*: [6]

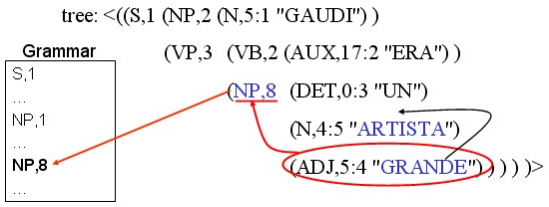
```
ADJ::ADJ |: [great] -> [grande]          ADJ::ADJ |: [great] -> [gran]
((X1::Y1)                                ((X1::Y1)
(...)                                    (...)
((y0 agr num) = sg)                      ((y0 agr num) = sg)
((y0 agr gen) = masc))                   ((y0 agr gen) = masc))
```

4. Finding Triggering Feature(s)

Since the delta set between *grande* and *gran* is empty (δ(grande,gran) = ∅), precisely because their lexical entries have the same features, the system postulates a new binary feature, let's call it *feat1*.[7]

5. Blame assignment

The MT system output provides with all the information required to trace what rules were involved in producing the error:



[6] The Spanish morphological analyzer gives us the information that *grande* and *gran* have the same features.

[7] A more mnemonic name for *feat1* would be *pre-nominal*.

6. Variable Instantiation in the Rules

Since the system has access to the part-of-speech of *grande* (ADJ) through the MT system output shown in the previous step, the RR module can trace what variables refer to *grande* by the position of its POS in the TL-side of the relevant grammar rule.

But first, a few words about the rule formalism used by our MT system. The translation rules include all information necessary for parsing, transfer, and generation, and have 6 components: 1) type, which in most cases corresponds to a syntactic constituent type; 2) part-of-speech/constituent sequence for both the SL (x-side) and the TL (y-side); 3) alignments between the SL constituents and the TL constituents; 4) x-side constraints, which are defined as equality of grammatical features in the SL sentence; 5) y-side constraints, which are defined as equality of grammatical features in the TL sentence, and 6) xy-constraints, which provide information about which feature values or agreements transfer from the source into the target language.

Back to the variable instantiation in the grammar rules, in the NP,8 rule, ADJ is in the third position on the TL side (y-side) and thus the variable that refers to it is y3:

W_i = grande → POS_i = ADJ = Y3, y3
W_c = artista → POS_c = N = Y2, y2

```
{NP,8}                          ;; Y1  Y2  Y3
NP::NP : [DET ADJ N] -> [DET N ADJ]
(  (X1::Y1)  (X2::Y3)  (X3::Y2)
  ((x0 def) = (x1 def))
  (x0 = x3)
  ((y1 agr) = (y2 agr)) ; DET-N agreement
  ((y3 agr) = (y2 agr)) ; DET-N agreement
  (y2 = x3)  )
```

7. Refining Rules

Assuming the system has the information that NP,8 has applied correctly in the past (perhaps because users have evaluated the translation pair *I saw a black bird – vi un pájaro negro* as correct), the RR module proceeds to bifurcate the original rule, following the second rule schema in Figure 2 (SR2). It then modifies the copy (NP,8') by flipping the order of the constituents, as indicated by the user, and by adding the constraint that the Spanish adjective (y2) needs to have the *feat1* with value +:

```
{NP,8'}
NP::NP : [DET ADJ N] -> [DET ADJ N]
(  (X1::Y1)  (X2::Y2)  (X3::Y3)
((x0 def) = (x1 def))
(x0 = x3)
((y1 agr) = (y3 agr)) ; det-noun agreement
((y2 agr) = (y3 agr)) ; adj-noun agreement
(y2 = x3)
((y2 feat1) =c + ))
```

8. Refining Lexical Entries

For this refinement to be effective, the lexical entries need to be expanded with the new feature postulated by the RR module in step 4:

```
ADJ::ADJ |: [great] -> [grande]
((X1::Y1)
((x0 form) = great)
((y0 agr num) = sg)
((y0 agr gen) = masc)
((y0 feat1) = -))

ADJ::ADJ |: [great] -> [gran]
((X1::Y1)
((x0 form) = great)
((y0 agr num) = sg)
((y0 agr gen) = masc)
((y0 feat1) = +))
```

9. Add Blocking Constraints

In addition to this, the system already has the information that *un artista gran* is not a correct sequence in Spanish[8], and thus the grammar can be further refined to also rule out the incorrect translation. This can be done by restricting the application of the general rule (NP,8) to just post-nominal adjectives, which in this example are marked in the lexicon with *feat1* = – .

But can the system also eliminate other incorrect translations automatically? In addition to generating the correct translation, we would also like the RR module to produce a refined grammar that is as tight as possible, given the data that is available.

In this case, the system can only further tighten the grammar if it knows what the clue word is (W_c=artista). If it does, then it can add the constraint *feat* = + to the lexical entry for *artista* and add an agreement constraint

[8] Since instead of just changing *grande* to *gran*, the user proceeded to move it to the pre-nominal position.

between the N position (y2) and the ADJ position (y3) for the original rule (NP,8).

The resulting refined grammar would now correctly translate *a great artist* into *un gran artista* as well as rule out all the incorrect combinations of N and ADJ (**un artista grande*, **un artista gran*, ** un grande artista*)

10.Conclusions and Future Work

Non-expert bilingual speakers can provide us with accurate translation error information, by using the Translation Correction Tool, so that automatic rule refinement becomes an option.

Once the error information is instantiated with the appropriate variables, we can automatically extract the set of feature attributes that triggered a particular correction. Using an error typology, which takes into account the correction action and the error information available to the system, the RR module is then able to automatically determine what refinement operations need to apply. The trace of the MT system is used to automatically perform the blame assignment and determine what rules need to be refined.

When operating in batch mode, given a set of user corrections, the RR module can automatically refine some of the errors by just using the correction and the error information provided by the TCTool.

If extra information is required to automatically determine what triggered the correction, the system will need to present users with other relevant translation pairs at run-time.

Therefore, we are planning to expand the system to also include an interactive mode, which will allow the system to refine a larger set of translations, possibly using Active Learning techniques.

Initial end-to-end simulations indicate that this framework for interactive and automatic refinement of transfer MT systems is appropriate for the task. We are currently developing a first prototype of the RR module and plan to fully test this framework for at least two language pairs: Mapudungun ↔ Spanish and Quechua ↔ Spanish.

11.Acknowledgments

This research was funded in part by NSF grant number IIS-0121-631.

12. References

ALLEN, Jeffrey (2003). Post-editing. ed. Harold Somers. Benjamins Translation Library, 35.

ALLEN, Jeffrey & HOGAN, Christopher. (2000). Toward the Development of a Post editing Module for Raw Machine Translation Output: A Controlled Language Perspective. CLAW.

BRILL, Eric (1993). Automatic Grammar Induction and Parsing Free Text: A Transformation-Based Approach. ACL.

CORSTON-OLIVER, Simon, & GAMON, Michael (2003). Combining decision trees and transformation-based learning to correct transferred linguistic representations. MT Summit 2003.

FONT-LLITJÓS, Ariadna & CARBONELL, Jaime (2004). The Translation Correction Tool: English-Spanish user studies. LREC.

IMAMURA, Kenji, SUMITA, Eiichiro, & MATSUMOTO, Yuji (2003). Feedback cleaning of Machine Translation Rules Using Automatic Evaluation. ACL.

LIN, Yi-Chung, CHIANG, Tung-lIui., & SU, Keh-Yih (1994). Automatic Model Refinement with an application to tagging. COLING-94.

MENEZES, Arul, & RICHARDSON, Stephen 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.

PROBST, Kathrin, LEVIN, Lori, PETERSON, Erik, LAVIE, Alon, & CARBONELL, Jaime (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.

SU, Keh-Yih, CHANG, Jing-Shin, & HSU, Yu-Ling Una (1995). A corpus-based statistics-oriented two-way design for parameterized MT systems: Rationale, Architecture and Training issues. TMI.

YAMADA, Setsuo, NAKAIWA, Hiromi, OGURA, Kentaro, & IKEHARA, Satoru (1995). A Method of Automatically Adapting a MT System to Different Domains. TMI.