2006

Automating Post-Editing to Improve MT Systems

Ariadna Font Llitjós
Carnegie Mellon University

Jaime G. Carbonell
Carnegie Mellon University

Automating Post-Editing To Improve MT Systems

Ariadna Font Llitjós
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, 15217
aria@cs.cmu.edu

Jaime G. Carbonell
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, 15217
jgc@cs.cmu.edu

Abstract

Beyond manual and automated post-editing, we describe an approach that takes post-editing information to automatically improve the underlying rules and lexical entries of a transfer-based Machine Translation (MT) system. This process can be divided into two main steps. In the first step, an online post-editing tool allows for easy error diagnosis and implicit error categorization. In the second step, an Automatic Rule Refiner performs error remediation, by tracking errors and suggesting repairs that are mostly lexical and morpho-syntactic in nature (such as word-order or incorrect agreement in transfer rules). This approach directly improves the intelligibility of corrected MT output and, more interestingly, it generalizes over unseen data, providing improved MT output for similar sentences that have not been corrected. Hence our approach is an alternative to fully-automated Post-Editing.

1 Introduction

Achieving high translation quality remains the biggest challenge for Machine Translation researchers. Currently, the two main directions towards this objective are improving the MT system itself or improving the result of the system, namely the MT output, commonly known as post-editing.

The first direction is by far the most popular, and involves different solutions depending on the different MT approaches. Statistical MT systems, after starting to plateau by adding more training data, are now turning to ways of incorporating syntactic knowledge to guide or inform their translation algorithms.

Traditional solutions to improve Transfer-Based MT systems are costly and time-consuming, since they involve the work of many computational linguists over extended periods of time to develop new rules and refine old ones. Moreover, in any MT system, out-of-vocabulary words are constantly jeopardizing translation quality.

On the other hand, improving MT output has been limited to manually correcting the output and, in the best-case scenario, to some automated post-processing to alleviate the tedious task of manual post-editing by correcting the most frequent errors beforehand (Allen & Hogan, 2000; Knight & Chander, 1994). However, to this day, there exists no solution to fully automate the post-editing process.

In order to fully automate the post-editing process, one could try to learn post-editing rules automatically from concrete corrections. This has the main advantage of being system independent. With this approach, however, one cannot generalize to correct the same structural error with a different surface realization, namely a different word. Furthermore, several thousands of sentences might need to be corrected for the same error.

Additionally, it seems reasonable to want to update one's underlying MT system to adapt to the evolution of language and knowledge processing.

This paper describes an alternative to Automated Post-Editing (APE) that attacks the root of the problem by fixing the source of the error in the MT system, as opposed to automatically fixing MT output. Our approach uses post-editing information, collected through an online tool, to automati-

cally detect incorrect lexical and grammar rules responsible for the errors and propose concrete fixes to such rules. Once the grammar and lexicon have been refined, the MT system not only produces the correct translation result of the post-editing process, but is also able to generalize and correctly translate unseen sentences. With the goal of maximizing the degree of reliable generalization, our research focuses on an intermediate step of relatively safe generalization.

This approach naturally requires access to translation rules, which can be manually written or automatically created (Font-Llitjós *et al*., 2004), and thus is mostly relevant to transfer-based systems, even though we believe it could also be extended to improve Statistical systems. Such extension would require a different approach that would have to take into account a very large number of rules and their interactions as well as tuning of their weights.[1]

2 Related Work

Post-editing has often been defined as the correction of MT output by human linguists or editors. However, many MT errors can be corrected with high accuracy by bilingual speakers who are not linguists, editors or translators, as shown in an initial set of user studies (Font Llitjós & Carbonell, 2004).

Our approach was originally motivated by the problem of improving inaccurate or partial MT systems in resource-scarce scenarios, where experts who can improve MT systems or even MT output are often not available. For this reason, our approach does not require experts to provide post-editing feedback nor post-editing rules to be applied automatically.

Therefore, the intended users of the online post-editing tool, known as the Translation Correction Tool (TCTool), are non-expert bilingual speakers, 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.

Besides being useful for resource-scarce contexts, having a method to accurately elicit post-editing information from non-trained bilingual speakers is a revolutionary idea. One that can lead to the collection of large amounts of MT output feedback, which can be used to improve all kinds of MT systems.

For languages with more presence on the Internet, we envision the use of TCTool as an online game, which would allow bilingual speakers to correct MT output and get rewards for making good corrections, and compare their scores and speed with other users (Font Llitjós 2006). The use of online games with a purpose has already been implemented and has been shown to work for labeling images, a difficult problem that current computer vision algorithms can still not accurately solve (Von Ahn et al. 2006).

Back in 1988, Nishida and colleagues described a Post-Editing Correction information Feedback system (PECOF) in its early stages, which attempts to improve a Transfer-based MT system. There are many differences between their approach and the one described here, but the main ones are: 1) the use of expert post-editors, whose work is not only to correct MT output but also to formulate correcting procedures corresponding to unseen error patterns, which are then executed by the PECOF system, and 2) the use of two MT systems in order to detect discrepancies between intermediate representations of the source language and the target language side, namely an *original* MT system (Japanese to English) and a *reverse* MT system (English to Japanese) which is applied to the post-edited English translation.

Our transfer-based MT system is particular in the sense that its rules integrate information from the three components of a typical transfer system, namely syntactic analysis (parsing), transfer and generation. And thus, in comparison with the PECOF system, blame assignment and correction become highly simplified.

More recently, some researchers have looked at other ways of including user feedback in the MT loop. Phaholphinyo and colleagues (2005) proposed adding post-editing rules to their English-Thai MT system with the use of a post-editing tool.

[1] Statistical MT systems need exponentially more data to improve by very small amounts, and for most language pairs such a brute force approach will simply not work. System-relevant post-editing information provides a great way to obtain annotated data, for which new smarter training algorithms can be developed.

However, they use context sensitive pattern-matching rules, which make it impossible to fix errors involving missing words. Unlike our approach, in their system, the rules are created by experienced linguists and their approach requires a large corpus. They mention an experiment with 6,000 bilingual sentences but report no results due to data sparseness.

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.

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 and colleagues 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.

3 Eliciting Post-Editing Information

The first step of the Automatic Rule Refinement approach is the correction of MT errors with an online tool. The main challenge of the error elicitation process is how to elicit minimal post-editing information from non-expert bilingual speakers.

If an MT-produced translation is incorrect, the assumption is that a bilingual speaker can diagnose the presence of an error reliably using the online Translation Correction Tool. An example of an English-Spanish sentence pair generated by our MT system is "*Gaudí was a great artist - Gaudí era un artista grande*". Using the online tool, bilingual speakers had not trouble modifying the incorrect translation to obtain a correct one: "*Gaudí era un gran artista*".

Bilingual speakers, however, cannot be expected to diagnose which complex translation rules produced the error, and even less, determine how to improve those rules. And so the Translation Correction Tool effectively abstracts away from any complexity of the underlying representation, leaving the inference of which correction actions to perform to the Automatic Rule Refiner. In this case, the Rule Refiner will add the special case rule for Spanish pre-nominal adjectives to the grammar.

3.1. Translation Correction Tool

The Translation Correction Tool (TCTool) is a user-friendly online tool that allows both error detection and error remediation, as well as an implicit error categorization with four top-level categories, which correspond to the different correction actions: add, delete and modify words (and alignments), as well as change word order.

The error typology in Figure 1 shows the four classes into which users implicitly classify the errors, simply by correcting the MT output with the TCTool, followed by finer grained classes that are inferred by the Automatic Rule Refiner in order to fix the rules accordingly.

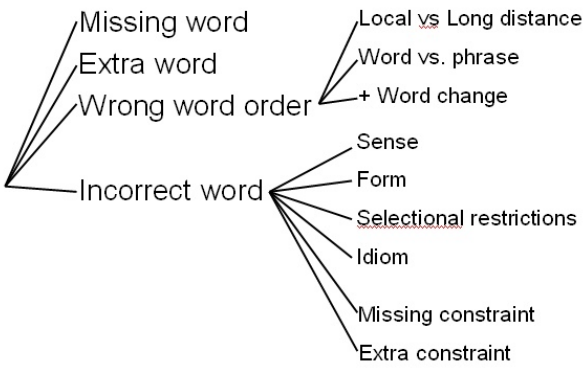

Figure 1: Error Typology given by user post-editing actions (on the left), and inferred by the Automatic Rule Refiner to fix the incorrect rules (on the right).

A set of user studies was conducted to discover the right amount of error information that non-

expert bilingual speakers can detect reliably when using the TCTool. These studies showed that a simple variation of traditional post-editing (error classes on the left of Figure 1) can be elicited much more reliably (F1 0.89) than error type information (error classes on the right of Figure 1; F1 0.72) (Font Llitjós and Carbonell, 2004). Most importantly, it became apparent that the list of correction actions with information about error and correction words and their alignments is sufficient for Automatic Rule Refinement purposes.

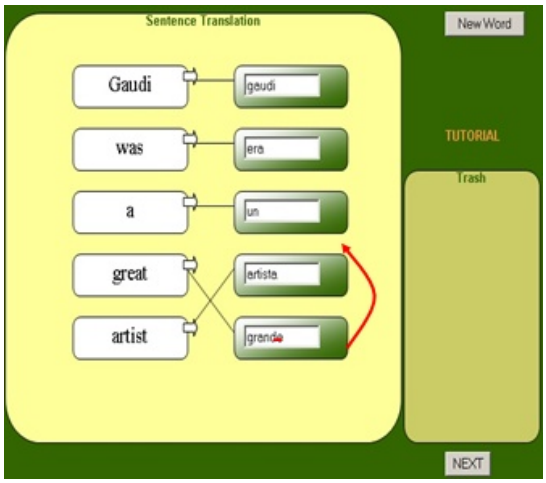

Figure 2. TCTool snapshot with initial translation pair

Building on the example introduced above, Figure 2 shows the initial state of the TCTool, once the user has decided that the translation produced by the MT system is not correct.

In this case, the bilingual speaker changed 'grande' to 'gran' and dragged 'gran(de)' in front of 'artista', effectively flipping the order of these two words.

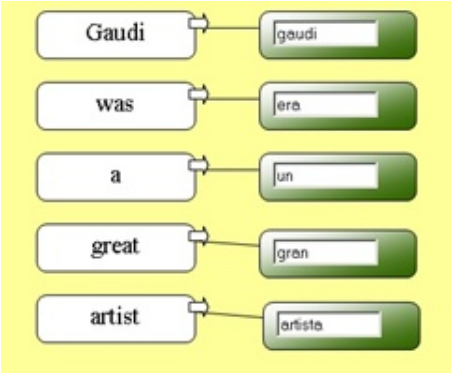

Figure 3. TCTool snapshot after user has corrected the translation.

Figure 3 shows the state of the TCTool after the user corrections. Given user post-editing[2], the real challenge is how to perform blame assignments on lexical entries and transfer rules, and then to automatically edit these in the most general feasible way to avoid repetitions of the same error.

3.2. Extracting Error Information

User correction actions are registered into a log file. The Automatic Rule Refiner (RR) extracts all the relevant information from the TCTool log files and stores it into a Correction Instance (Figure 4).

The RR module processes one action at a time. So in this approach, the order in which users correct a sentence does have an impact on the order in which refinements apply.

SL: *Gaudí was a great artist*
TL: *Gaudí era un artista grande*
AL: ((1,1),(2,2),(3,3),(4,5),(5,4))

Action 1: edit (*grande→ gran*)
Temp CTL: *Gaudi era un artista gran*

Action 2: change word order
(*gran artista*)

CTL: *Gaudí era un gran artista*
AL: ((1,1),(2,2),(3,3),(4,4),(5,5))

Figure 4. A Correction Instance stores the source language sentence (SL), the target language sentence (TL) and the initial alignments (AL), as well as all the correction actions done by the user. It also provides the corrected translation (CTL) and final alignments.

4 Automatic Rule Refinement

After having extracted and stored the location of the error in the translated sentence (*error-locus)* and *error-type* information from non-expert bilingual speakers, the Automatic Rule Refiner can trace the errors back to incorrect lexical and grammar rules responsible for the errors and propose concrete fixes to such rules.

More specifically, the RR can automatically add missing lexical entries and detect incomplete or incorrect rules (both manually written and automatically learned) that applied during the generation of MT output.

[2] This could also be the result of automated post-editing if properly instrumented to capture each edit step.

4.1. REFINE vs. BIFURCATE

There are two main refinement operations that can be applied to both grammar rules and lexical entries, with some minor differences: REFINE and BIFURCATE.

The REFINE operation consists of modifying an existing rule (R0), effectively replacing it with a more specific, correct rule (R1). Sometimes a rule is mostly correct, but is missing an agreement constraint, for an example see Figure 5.

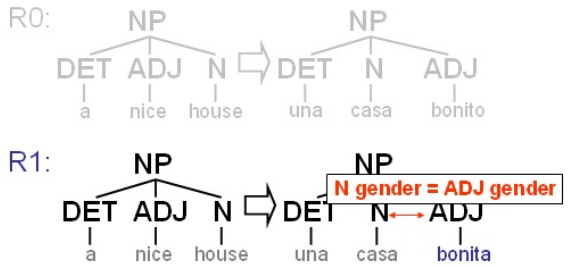

Figure 5: The resulting rule from the REFINE operation (R1) is like the original rule (R0) but with an additional agreement constraint. Even though the rules in this figure appear lexicalized, this is just for illustration purposes and the rules in our grammar are at the POS level.

In the REFINE case, the new refined rule needs to translate the same sentences as before plus the corrected sentence.

The BIFURCATE operation makes a copy of the original rule (R0) and refines the copy (R1) so that it covers an exception to the general rule. In the BIFURCATE case both the original rule and the refined rule coexist in the grammar (Figure 6).

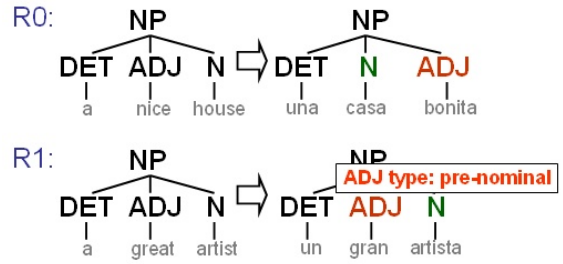

Figure 6: R1 is the result of BIFURCATing the general rule for nouns and adjectives in Spanish (R0) to cover the pre-nominal order.

This is appropriate for cases when the general rule has correctly applied before, for example in translating *a nice house – una casa bonita,* the grammar already contains the general rule to deal with nouns and adjectives in Spanish , and we want the grammar to also account for an exception to the general rule, namely pre-nominal adjectives.

4.2. Formalizing Error Information

The RR represents TL sentences 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 (CTL) as follows:

$$CTL_m = (W_1,...,W_i^{'},...,W_{clue},...,W_{n'})$$

where W_i represents the error, namely the word or multiword phrase that needs to be modified, deleted or dragged into a different position by the user to correct the sentence; 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.

The clue word (W_{clue}) represents a word that provides a clue with respect to what triggered the correction: 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_{clue} should be instantiated to *coche*, as it is the word that gives us the clue about what the gender agreement feature value of W_i should be. In this case, masculine (*rojo*).

W_{clue} 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_{clue} is not always present and it can be before or after W_i. They can be contiguous or separated by one or more words.

For more information about the theoretical framework of the Rule Refiner, refer to Font-Llitjós *et al.* (2005).

5 From Post-Editing Actions To Rule Refinements

One of the biggest challenges of this work is how to map simple post-editing actions onto complex refinements to the appropriate rules.

The core component of the rule refinement process is the one that decides what rule refinement operations need to apply to address a specific error correction. This is also the component that is most sensitive to the set of correction actions currently allowed by the TCTool. The reason for this is that the rule refinement operations that are applied by

the system crucially depend on what types of correction actions were chosen by users.

Given the correction action type (add, edit, delete and change_word_order) and the error and correction words, the RR applies the appropriate refinement algorithm. In general, the Rule Refiner addresses lexical refinements first and then moves on to refinements of the grammar rules, if it is still necessary.

The following subsections describe a simplified version of algorithm underlying the Rule Refiner illustrated with a few examples.

5.1 Add Word

When users add a word (by clicking on the [New Word] button on the TCTool interface and then by writing the word in the newly created box), there is no error word *per se*, however the RR can reliably identify a correction word (W'), namely the newly added word (in this case 'se'). See Figure 7.

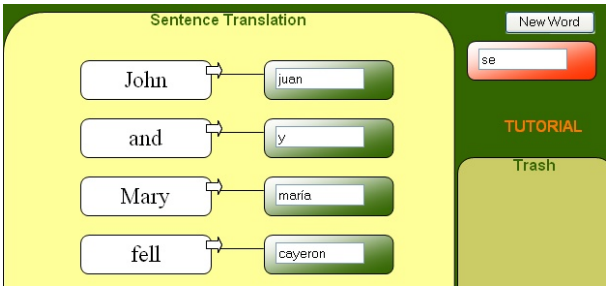

Figure 7. TCTool snapshot after having created a new word (*se*).

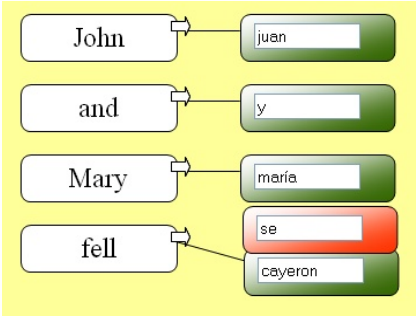

Figure 8. TCTool snapshot after having added the newly created word into the right position (Action 1).

After having added the new word, users drag it to the right position in the translation. This operation allows the system to instantiate the new word with a specific position in the Corrected Target Language (CTL) vector, W_i' (Figure 8), the next step is to check if the user added any alignments from any word(s) in the SL sentence to the corrected word W_i', and if so, to extract them (Figure 9).

Alignment information is required to retrieve the relevant lexical entries and determine the necessary refinements.

In our example, [fell→se] and [fell→se cayeron] are not in the lexicon, however [fell→cayeron] is there.

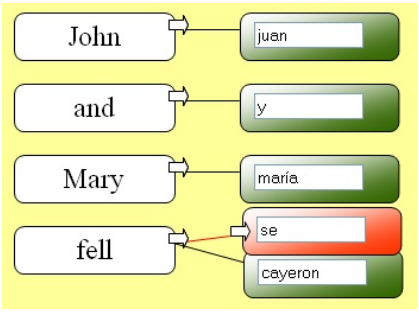

Figure 9. TCTool snapshot showing Action 2: Adding manual alignment.

At this point, the RR BIFURCATES the lexical entry [fell→cayeron] creating a copy of it and REFINES it by replacing the TL side with W_i' plus the other word aligned to the SL word: [fell→se cayeron]. The resulting refined entry is displayed below:

V::V |: [fell] -> [**"se cayeron"**]
((x0 form) = fall)
((x0 actform) = fell)
((x0 tense) = past)
((y0 agr pers) = 3)
((y0 agr num) = pl)

The new lexical entry is added to the Lexicon and the Refined Lexicon is loaded to the MT system to assess the effect of the rule refinement.

The translation alternatives output by the system are now checked against the CTL sentence as corrected by the user. If the RR finds that CTL sentence is being generated by the MT system, it stops, otherwise, it proceeds to grammar refinements. For this example, the algorithm described above successfully refined the lexicon and the output of the refined MT system already contains the CTL sentence.

If the word added (W_i') is not aligned to any word in the SL sentence, then there is nothing to be done at the lexical level and the algorithm proceeds to grammar refinements.

The first step is to search the grammar for rules with a TL side that contain the new Word/POS

sequence. If the sequence suggested by the user's refinement is not in the grammar, the RR adds W_i' in the position indicated by the user to the appropriate incorrect rule.

For example, given the translation pair *you saw the woman − viste la mujer* and the user correction of adding the word "a" in front of *mujer*, the RR will detect that 'a' is not aligned to any words in the SL sentence and will proceed to look for the following sequences ["a" NP] and ["a" DET N] in the TL grammar. Since such a sequence does not exist, the refiner has three candidate rules for refinement, namely V, NP and VP:

```
(S    (VP   (VP (V  'viste'))
      → "a"
      (NP   (DET  'la')
            (N  'mujer') ) ) )
```

Adding an 'a' in the right position to either of these three rules (VP[V "a"], VP[VP "a" NP]) and NP["a" DET N] would result into the desired final output, and even though from a linguistics perspective the second and third are better options for this example, there will be cases when adding a specific preposition to the preceding verb is the linguistically motivated thing to do (eg. "preocupado por").

In general, the RR algorithm will choose to refine the rule that better generalizes over a regression test set, as estimated by automatic evaluation metrics (Section 7).

5.2 Edit Word

When users modify a word (W_i) into a related form or sense (W_i'), there are two possible scenarios. The one most favorable to generalization is that the lexicon already discriminates between these two forms, usually by giving them a different value for the same feature attribute (example: [*red-roja* gender: fem] and [*red-rojo* gender: masc]). The one with less immediate impact is that the two words are identically defined in the lexicon, namely they have the same POS and the same feature attributes and values (e.g. [women-mujer] and [guitar-guitarra] are both singular feminine nouns in Spanish).

If the lexicon already discriminates between the two lexical entries, the RR extracts the grammar rule for the immediate common parent of W_i and W_{clue} (as identified by the user or guessed by the system) and adds an agreement constraint with the triggering feature[3] between the constituents corresponding to W_i and W_{clue}.

SL: I see the red car
TL: veo el auto roja
Alignments: ((2,1),(3,2),(4,4),(5,3))

Action 1: edit (W_i=roja → W_i'=rojo; W_{clue}=auto)

CTL: veo el auto rojo
Alignments: ((2,1),(3,2),(4,4),(5,3))

Figure 10. Correction Instance for edit action.

For the correction instance represented in Figure 10 (*I see the red car)*, the user edits *roja* into *rojo* (by clicking on the word and changing 'a' into 'o'), and the system finds that the difference (delta set) between the lexical entry for roja and rojo is [agr gen].

At this point, the RR moves on to the Grammar Refinement.

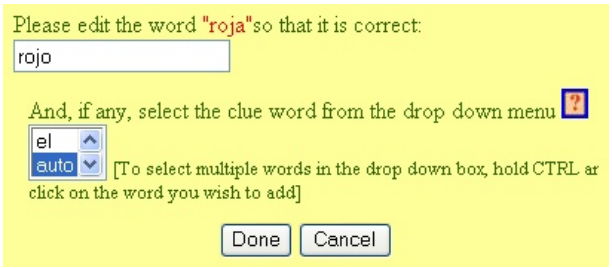

Figure 11. Edit Word window eliciting for clue word information.

Since the user identified 'auto' as being the clue word as shown in Figure 11, the RR algorithm can now instantiate what variables do W_i and W_{clue} correspond to in the relevant rule (**NP,8:** ADJ N → **N ADJ**), namely the system internal variables that represent the TL adjective and noun.[4]

Next, the Rule Refiner adds an [agr gen] constraint to rule NP,8 between the noun and the adjective: **NP,8:** ADJ N → **N ADJ**

[(N agr gen) = (ADJ agr gen)]

However, if the lexicon does not already discriminate between the two lexical entries (W_i and W_i'), the RR postulates a new feature attribute and adds a binary value constraint to each lexical entry,

[3] The triggering feature is the attribute name for which the two lexical entries have a different value.
[4] The relevant rule is extracted from the translation tree output by the MT system, making blame assignment straightforward.

in order to allow the grammar to distinguish between the two forms/senses of the same SL word automatically.

For example, given the sentence *Mary plays guitar* and its translation as produced by our MT system, **María juega guitarra*, the user will edit *juega* into *toca*, and since this new sense is not listed in the lexicon, the RR will BIFURCATE the original lexical entry [play→juega] and REFINE it by replacing the TL side. Naturally, [play→toca] is otherwise an exact copy of [play→juega] (with the same POS and features), and so the system postulates a new feature (*feat_0*) to distinguish between the two and adds the following constraints to the lexical entries:

[play→toca$_{((feat_0) = +)}$] [play→juega$_{((feat_0) = -)}$]

Note that in the absence of a semantically annotated lexicon, our approach will only be able to solve such errors on a case by case basis.

5.3 Delete Word

If a user deletes a word, first the RR algorithm needs to make sure this is not followed by a word being added in the same position, which is the equivalent to editing a word.

After making sure it is really a delete case, the RR algorithm checks if there were any alignments from the deleted word (W_i) to one or more SL words, and if so, it looks ahead to see if there was any other word in the TL sentence that was aligned to the SL word(s) at a later point in the session. If there is a TL word aligned to any of the relevant SL words, then the RR algorithm checks if it's already in the lexicon, and if it isn't, it adds it.

If no alignment is added to the relevant SL word(s), the RR algorithm adds a new lexical entry for the SL word with an empty TL side ([SL word → ""]), which results into the MT system not translating the SL word.

5.4 Word Order Change

To change the order of the TL words, users can drag and drop words into a different position in the TL sentence using the TCTool.

The Rule Refiner detects which word(s) were moved to a different position and extracts what were their initial (*i*) and final (*i'*) positions. The Rule Refiner can only reliably execute refinement operations iff, given a word that has moved (W_i), both the initial and final positions fall inside the scope of one of the rules in the grammar. If a word undergoes a long-distance move and thus is placed at the beginning or the end of the sentence far from its original position, automatic refinements become less reliable.

If the initial and final positions are contained within a rule in the grammar, then the RR algorithm can extract the rule that immediately subsumes the constituents in both positions and BIFURCATE it in order to change the constituents on the target language side of the rule copy.

For example, if the grammar already has a general NP rule that reverses the order of the adjectives and nouns in Spanish, but is lacking a specific rule for pre-nominal adjectives, like in the *Gaudi was a great artist* example introduced in section 3, given relevant correction feedback, the RR can extract the general NP rule and flip the order of N and ADJ on the TL side of the rule (Figure 12).

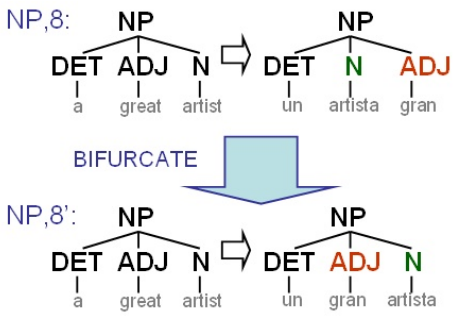

Figure 12: The RR applies the BIFURCATE operation to rule NP,8, by which the order of the noun and adjective constituents is flipped.

The next step is to further constrain the newly created rule so that it only applies in the right context, namely to 'gran' but not to 'grande'. Again this can be done in a general way if the lexicon already distinguishes between the lexical entries that are affected by this change and the general cases. A constraint with the appropriate feature attribute is added to the specific rule and a blocking constraint is added to the general rule.

If there is no current feature attribute to distinguish between the special case and the general case, the RR postulates a new binary feature and REFINES both the grammar rules (Figure 13) as well as the appropriate lexical entries by adding a value constraint (Figure 14).

In the Gaudi example, [great→gran] and [great→grande] are identical at the feature level,

and so the RR module postulates a new binary feature, say *feat1*[5], which serves the purpose of distinguishing between two words that are otherwise identical according to our lexicon.

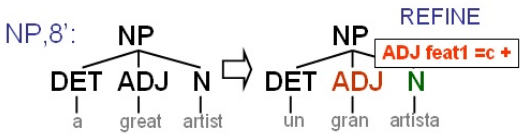

Figure 13: The bifurcated rule NP,8' is further REFINEd by the RR by adding a value constraint for the adjective indicating that it will only apply to adjectives with (*feat1* = +).

```
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) = +))
```

Figure 14: Lexical entries for *grande* and *gran* REFINED with a value constraint for *feat1*, so that the RR can discriminate them.

These two refinements result in the MT system generating the desired translation, namely "*Gaudí era un gran artista*" and not the previous incorrect translation.

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

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

6 Generalization power

The difference between this approach and mere post-editing is that the resulting refinements affect not only to the translation instance corrected by the user, but also to other similar sentences where the same error would manifest. After the above refinements have been applied to the grammar for the Gaudi example sentence, sentences like "*Irina is a great friend*" and "the young professor is a great person" will now correctly translate as "*Irina es una gran amiga*" and "*el profesor joven es una gran persona*", instead of "**Irina es una amiga grande*" and "**el professor joven es una persona grande*". As shown by the last example, generalization goes beyond just lexical variation, and applies to constituent generalization. Moreover, the generalization power of this approach is greater when the refinements involve information that is already encoded in the lexicon and the grammar. In our lexicon, this means mostly errors of lexical and morpho-syntactic nature.

7 Automatic Evaluation Methods

In order to fully close the feedback loop and make sure that automatic refinements lead to real improvements of the MT system and therefore its output, the Rule Refiner uses different methods to evaluate the refined MT output and guide its decisions. The first one is a simplification of recall which tells us whether the corrected translation, as provided by the user post-editing actions, is currently being generated by the system as one of the alternatives or not. Since users implicitly also give information about which translations are not correct, we also want to measure if refinements managed to eliminate such incorrect translations (precision at rank k). At the same time, we are interested in knowing whether the number of alternatives produced by the system has decreased. This indicates that the refinements effectively reduced the ambiguity of the grammar (reduction ratio).

On the other hand, we want to evaluate the effect of refinements on test data, to make sure refinements generalize well. For this, we can also rely on standard metrics to evaluate MT output before and after refinement on a test set. Initial experiments have shown that both modified BLEU and METEOR [Lavie et al., 2004] can automatically distinguish between raw MT output and cor-

rected MT output, even for a small set of sentences. Such metrics can automatically calculate the ngram overlap between the output generated by the refined MT system and the user-corrected translation, namely the reference translation. Unlike most evaluation settings that rely on independent reference translations, this approach has the great advantage that reference translations are guaranteed to be relevant to the system.

8 Conclusions and Future Work

We have described an alternative approach to traditional Automated Post-Editing that recycles post-editing efforts back into the MT system, improving the system itself as new sentences are manually corrected by bilingual speakers.

Guided by post-editing information, the RR module can decide to add a lexical entry, modify a current lexical entry, bifurcate a rule and modify the copy, usually making it into a more specific rule, or refine a rule that is too general, by adding a missing agreement constraint, for example.

The Rule Refinement process is not invariable. It depends on the order in which refinement operations are applied. In batch mode, the RR module can rank correction instances in such a way as to maximize translation accuracy.

In some cases, in order to determine the right level of granularity of the refinements proposed by the Automatic Rule Refiner, an implementation of the system with interactive mode as well as Active Learning techniques would be required.

Finally, the TCTool can also be seen under a slightly different light, that of benefiting second language learners. For this purpose, we envision the TCTool as an interactive game that would bring together a native speaker of English who is learning Spanish, say, with a native speaker of Spanish who is learning English, and would allow them to interact until they agree on what is a minimal correction of any given translation, given the original sentence.

9 Acknowledgements

We thank Alon Lavie for theoretical discussion of this work, William Ridmann for his implementation work and Stephan Vogel for implementation discussion and for his support. This research was funded in part by NSF grant number IIS-0121-631.

References

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

Allen, J. & Hogan C. (2000). *Toward the Development of a Post editing Module for Raw Machine Translation Output: A Controlled Language Perspective*. CLAW.

Font Llitjós, A. (2006). *Can the Internet help Improve Machine Translation*. Doctoral Consortium at HLT-NAACL.

Font Llitjós, A.; J. Carbonell and A. Lavie (2005). *A Framework for Interactive and Automatic Refinement of Transfer-based Machine Translation*. EAMT.

Font Llitjós, A. & J. Carbonell (2004). *The Translation Correction Tool: English-Spanish user studies.* LREC.

Font Llitjós, A.; K. Probst and J. Carbonell (2004). *Error Analysis of Two Types of Grammar for the Purpose of Automatic Rule Refinement*. AMTA.

Imamura, K., E. Sumita and Y. Matsumoto (2003). Feedback cleaning of Machine Translation Rules Using Automatic Evaluation. ACL.

Knight Kevin & Ishwar Chander (1994). Automated postediting of documents. In Proceedings of the twelfth national conference on Artificial intelligence.

Lavie, A; K. Sagae and S. Jayaraman. 2004. *The Significance of Recall in Automatic Metrics for MT Evaluation*. AMTA.

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

Nishida, F.; S. Takamatsu, T. Tani and T. Doi (1988). *Feedback of Correcting Information in Posteding to a Machine Translation System*. COLING.

Phaholphinyo, S.; T. Modhiran, N. Kritsuthikul and T. Supnithi (2005). *A Practical of Memory-based Approach for Improving Accuracy of MT*. MT Summit.

Su, K.; J. Chang and Y. Hsu, (1995). A corpus-based statistics-oriented two-way design for parameterized MT systems. TMI.

Von Ahn, Luis; S. Ginosar, M. Kedia, R. Liu and M. Blum (2006). *Improving Accessibility of the Web with a Computer Game.* In Notes at the International conference for human-computer interaction (CHI).