# Using *any* machine translation source for fuzzy-match repair in a computer-aided translation setting

**John E. Ortega** jeo10@alu.ua.es
**Felipe Sánchez-Martínez** fsanchez@dlsi.ua.es
**Mikel L. Forcada** mlf@dlsi.ua.es
Dept. de Llenguatges i Sistemes Informàtics, Universitat d'Alacant, E-03071, Alacant, Spain

**Abstract**

When a computer-assisted translation (CAT) tool does not find an exact match for the source segment to translate in its translation memory (TM), translators must use *fuzzy matches* that come from translation units in the translation memory that do not completely match the source segment. We explore the use of a *fuzzy-match repair* technique called *patching* to repair translation proposals from a TM in a CAT environment using any available machine translation system, or any external bilingual source, regardless of its internals. Patching attempts to aid CAT tool users by repairing fuzzy matches and proposing improved translations. Our results show that patching improves the quality of translation proposals and reduces the amount of edit operations to perform, especially when a specific set of restrictions is applied.

## 1 Introduction

Computer-aided translation (CAT) tools based on translation memories (TM) are one of the most popular technologies among professional translators (Bowker, 2002; Somers, 2003). CAT tools exploit existing, segment-aligned translations to help the translator translate a new document by recycling as much target-language (TL) text as possible. To do so, CAT tools first split the source-language (SL) document to translate into segments, and for each SL segment $s'$ they look up the translation memory for segment pairs $(s,t)$ (called *translation units*) where $t$ is the translation of $s$ and $s$ is similar to $s'$. These translation units are then shown to the translator in decreasing order of similarity, together with an indication of the words in $s$ that do not match those in $s'$. Finally, the translator decides which translation unit to use and which parts of its TL segment $t$ have to be edited to produce $t'$, the desired translation of $s'$, and performs such editions.

The similarity between $s$ and $s'$ is computed by means of a *fuzzy-match score* (FMS) function whose output is between 0% (no match at all) and 100% (a perfect match, $s = s'$). Commercial CAT systems implement proprietary versions of FMS, but a reasonable approximation is given by:

$$\text{FMS}(s,s') = \left(1 - \frac{\text{ED}(s,s')}{\max(|s|,|s'|)}\right) \cdot 100\% \tag{1}$$

where $\text{ED}(s,s')$ is the (word-based) *edit distance* (Wagner and Fischer, 1974) between $s$ and $s'$ —the minimum number of one-word deletions, insertions and substitutions needed to transform $s$ into $s'$— and $|x|$ stands for the number of words in segment $x$. Many times translation tools use a *fuzzy-match score threshold* (FMT), for instance 80%, to reduce the number of translation proposals.

When a perfect match is not found in the translation memory, and before making any changes to the TL segment $t$ in the proposed translation unit, the translator has to identify the sub-segments of $t$ that correspond to the sub-segments of $s$ that are not common to $s'$. To help in finding the sub-segments of $t$ that need to be edited, but without actually editing them, Esplà-Gomis et al. (2011) use machine translation (MT) to find sub-segment alignments between $s$ and $t$, and train a classifier to classify the words in $t$ as words to be kept unedited or words to be changed to transform $t$ into the desired translation $t'$.

Other researchers have gone one step further and have explored different ways to combine the TL segment of the proposed translation unit and the output of a *statistical machine translation* (SMT) system to produce a translation closer to $t'$. Biçici and Dymetman (2008), for example, use a phrase-based SMT system trained on a bilingual corpus in the same domain as the TM and combine it with the TM's fuzzy match by extracting a phrase table that is dynamically added to the usual set of bi-phrases used for decoding the source. Their implementation augments the internal translation table in the SMT system with bilingual discontiguous sub-segments (phrases) that have source sub-segments in common with $s'$. Alignments in the system created by Biçici and Dymetman (2008) are detected using word alignments directly obtained from the SMT system training process and are used to find the parts of $t$ that need to be edited (mismatches).

Similarly, Simard and Isabelle (2009) use a phrase-based SMT system by adding phrase pairs (sub-segment pairs) of any length (obtained using a statistical aligner on the TM) to the SMT system's phrase table and introduce a feature to indicate that the phrase-pairs came from their TM. After that, they optimize the weighting of the TM-based phrase table in a regular SMT decoder. By means of optimization and phrase table inclusion they are able to make their SMT system produce a translation close to the desired translation $t'$.

Additional work done by Zhechev and Genabith (2010) makes use of a phrase-based SMT system along with an alignment method that, like Simard and Isabelle (2009), connects sub-segments from the target translation $t$ with those in $s$. The alignment method Zhechev and Genabith (2010) use takes advantage of a tree-based structural alignment created from a bilingual dictionary after training their SMT system with phrase pairs. After aligning the words in $s$ with those in $t$, Zhechev and Genabith (2010) are able to identify words that should appear in the final translation $t'$.

Koehn and Senellart (2010) take a similar approach to Biçici and Dymetman (2008). They first align words in $s'$ and $s$ to find mismatches. Then, they align the words in $s$ and $t$ to identify target matches and remove the words in $t$ that are aligned to the mismatched words in $s$. Target mismatches are sent to the SMT decoder for translation. Mismatched words in Koehn and Senellart (2010)'s system are treated separately; that is, context around a mismatch, while indirectly taken into account by the language model, is not directly taken into account of when applying phrase pairs.

Ma et al. (2011), on the other hand, decided to research the shortcomings of using a fuzzy-match score as a threshold for determining translation unit matches that serve as translations for other segments. Ma et al. (2011)'s approach uses discriminative learning and support vector machines to salvage translations of matched words to select a translation unit that would have been otherwise thrown away due to the fuzzy-match score being used as a threshold. Their work, unlike Koehn and Senellart (2010), takes matched parts in $s$ and replaces them with their counterparts in $t$. The main drawback of the approaches from Ma et al. (2011), Koehn and Senellart (2010), Zhechev and Genabith (2010), and Biçici and Dymetman (2008) is that they are all based on SMT and either have access to the internals of an SMT system trained on the user's or related data or modify its behavior in some way.

Other research work (Hewavitharana et al., 2005; Dandapat et al., 2011) focuses on the identification of the sub-segments in the TL segment $t$ of the translation unit $(s, t)$ needed to produce $t'$ and then produces a translation by applying a set of edit operations over $t$. In particular, Hewavitharana et al. (2005) first align the mismatches in $s$ to their TL translations in $t$ by means of a modified IBM model 1 and then apply the same edit operations —substitutions, deletions and insertions— that are needed to convert $s$ into $s'$ to the TL segment $t$. Their resulting translation may contain agreement and reordering errors because their method assumes that edit operations applied on the SL are exactly the ones needed in the TL and do not take word context into account.

Dandapat et al. (2011) were also able to successfully translate texts in the TL by marking mismatched words for translation. Dandapat et al. (2011)'s example-based machine translation (EBMT) and SMT work marks sub-segments for translation from $s'$ and $s$ in a manner similar to that of this paper. Their work involves creating sub-segment pairs to form a sub-segment TM, marking mismatched words, aligning matched words, and finally substituting words marked for translation in what they call a *recombination* step. Their recombination step substitutes words using a sub-segment TM, that is, a mismatched phrase table obtained from the user's TM in an SMT training job. Sub-segments are translated and "plugged" (i.e. inserted or replaced) into $t$ according to how they are found in the source text without taking into account other context around the mismatched sub-segments. Plugging and similar approaches, like the one from Hewavitharana et al. (2005), have some shortcomings due to the lack of contextual information around a mismatched sub-segment and differ from our approach in this respect.

We investigate fuzzy-match repair using a technique called *patching* that uses *any* external bilingual source to translate mismatched sub-segments; patching could use a glossary, a terminological database (Bowker, 2003), or another translation memory containing smaller segment pairs. Our approach, while related to the research described above, exhibits three main novelties: ($i$) it removes the dependency on knowledge of the internal workings of the MT system used, ($ii$) it removes the need to modify an MT system's behavior in some way and ($iii$) avoids having to pre-process a user's TM. We repair the mismatched sub-segments in a translation unit using a simple, yet novel, method that, unlike those by Hewavitharana et al. (2005) and Dandapat et al. (2011), takes context around mismatched words into account. Patching uses overlapping sub-segments as powerful anchors much like Brown et al. (2003)'s *maximal left-overlap compositional (example-based) MT* system where the use of overlapping sub-segments reduces "boundary friction" problems and increases the likelihood of producing a correct translation. Since patching treats sources of external bilingual translation information as black boxes, we generate translations on the fly without training SMT models on the user's TM. We are not aware of any research work that: (a) uses any source of bilingual information for translation or (b) uses the context around mismatched words for repair.

In the following sections, we show how the fuzzy-match repair method mentioned can be applied in a CAT setting. The rest of the paper is organized as follows. The next section describes in detail our approach and illustrates how it works with an example. Section 3 describes the experiments we have conducted and the results achieved. The paper closes with some concluding remarks and potential future research.

## 2 Methodology

We begin with a foundation similar to Esplà-Gomis et al. (2011): an engine-agnostic approach which, unlike other work done so far, only requires that the CAT tool is able to invoke the external translation system, in order to translate short source-side sub-segments $\sigma$ of $s$ to obtain the corresponding short target-side sub-segments $\tau$ of $t$. As Esplà-Gomis et al. (2011), the method described here can use online MT or any MT system “out of the box” — indeed, as described earlier, any source of such sub-segmental translation units $(\sigma, \tau)$ may in principle be queried.

Our methodology, unlike Esplà-Gomis et al. (2011), does not only mark words from $t$ for editing, it goes one step further and edits them using a patching method that can be described in 5 steps:

1. Align the words in the SL segment $s'$ to be translated to those in the SL segment $s$ of the translation unit and find mismatched words.
2. Translate the sub-segments $\sigma$ of $s$ and $\sigma'$ of $s'$ containing at least one mismatched word, up to a given sub-segment length, by querying the sources of bilingual information available.
3. Match each translated sub-segment $\sigma$ of the SL segment $s$ to those sub-segments $\tau$ in the TL segment $t$ of the translation unit (as in Esplà-Gomis et al. (2011)).
4. Pair the translation $\tau$ of the (mismatched) sub-segments $\sigma$ in $s$ for which a match has been found in $t$ to the translation $\tau'$ obtained for sub-segments $\sigma'$ in $s'$. The pairs $(\tau, \tau')$ are the *patching operators* which replace mismatched sub-segments in $t$ with the translation of the corresponding mismatched sub-segments in $s'$ to generate an improved translation candidate $t^{\simeq}$.
5. Apply the patching operators to build all possible translation hypotheses by selecting valid sets of patching operators that can be applied to form a final proposal. Restrictions can be applied to limit the amount of patching operators considered valid.

As patching can, in general, yield more than one solution, translation hypotheses could then be ranked according to their estimated quality so that the best one is shown to the translator for validation or post-editing. Here, in the absence of a *quality estimation* (QE) method that we plan as future work, we experiment with restrictions to discard less reliable patching operators and reduce the number of translation hypotheses to generate. With restrictions in place, we then evaluate the average quality of the resulting repaired sentences on a test set as well as the quality of the best proposal.

In the following sub-sections, we illustrate the patching process (steps 1 through 5 above) in detail by building patching operators for an English segment and then applying them to produce a translation in Spanish.

## 2.1 Step 1: align and find mismatches

We first find mismatched sub-segments from the source side $(s', s)$ segments of the document and translation memory using fuzzy matching. Imagine we are translating from English to Spanish and the new segment to be translated is:

$s' =$ “The blue dog barks loud when it rains at night”

The system shows a fuzzy-match $(s, t)$ from the translation memory and marks mismatched words (in bold below):

$s' =$ “The **blue** dog barks loud when it rains at night”

$s =$ “The **red** dog barks loud **sometimes** when it rains at night”

$t =$ “El perro rojo ladra fuerte a veces cuando llueve por la noche”

According to Eq. (1) the fuzzy-match score is $\text{FMS}(s', s) = 81.8\%$, and as a side result of the computation of the edit distance, the alignment between the words in $s'$ and those in $s$ is produced:

| $s'$ | the | **blue** | dog | barks | loud | | when | it | rains | at | night |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $s$ | the | **red** | dog | barks | loud | **sometimes** | when | it | rains | at | night |

It is clear that there are two words in $s$ that do not match $s'$: **red** and **sometimes**. Using the edit distance (Wagner and Fischer, 1974), the edit operations to convert $s$ into $s'$ would consist of:

- one substitution - replace the word **red** in $s$ for the word **blue** from $s'$ and
- one deletion - delete the word **sometimes** from $s$ between the words *loud* and *when*.

## 2.2 Step 2: translate source-side sub-segments covering mismatches

Each one of the sub-segments $\sigma$ from $s$ covering a mismatch is sent to an external machine translation system or bilingual source of information to find their corresponding TL translations $\tau$. The hope is to find a translation $\tau$ that exists in $t$ so that the corresponding sub-segment can be later modified to produce the desired translation $t'$. When $\tau$ is a substring of $t$, it is marked as applicable for patching.

If we choose Apertium (Forcada et al., 2011) as the external source of bilingual information to translate the mismatched $(\sigma, \tau)$ pairs and the maximum length of the sub-segments to translate is set to 3, we generate the following translated pairs:

- (the red dog, *el perro rojo*)
- (the red, *el rojo*)
- (red dog, *perro rojo*)
- (red, *rojo*)
- (loud sometimes when, *fuerte a veces cuando*)
- (loud sometimes, *fuerte a veces*)
- (sometimes when, *a veces cuando*)
- (sometimes, *a veces*)

## 2.3 Step 3: match the source-side translations to $t$

In step 3, we identify the $(\sigma, \tau)$ pairs that have a matching sub-segment in $t$ and can, therefore, be used to build the patching operators. We keep the following $(\sigma, \tau)$ pairs whose $\tau$ appears in $t$:

| $\sigma$ | position in $s$ | $\tau$ | position in $t$ | outcome |
|---|---|---|---|---|
| the red dog | 1–3 | *el perro rojo* | 1–3 | kept |
| the red | 1–2 | *el rojo* | *none* | discarded |
| red dog | 2–3 | *perro rojo* | 2–3 | kept |
| red | 2–2 | *rojo* | 3–3 | kept |
| loud sometimes when | 5–7 | *fuerte a veces cuando* | 5–8 | kept |
| loud sometimes | 5–6 | *fuerte a veces* | 5–7 | kept |
| sometimes when | 6–7 | *a veces cuando* | 6–8 | kept |
| sometimes | 6–6 | *a veces* | 6–7 | kept |

Sub-segments that do not have a match in $t$ (e.g. the second one in the example above) are discarded and not used further in the patching process. Word positions in $t$ not covered by any translation $\tau$ of any segment $\sigma$ in $s$ contain words for which there is no evidence to modify them. In the absence of information, they will not be changed.

### 2.4 Step 4: pair translations of $\tau$ and $\tau'$ to form patching operators

After the initial matching occurs from the translations of $s$ to form $(\sigma, \tau)$ pairs, $(\sigma', \tau')$ pairs are created by translating mismatched sub-segments from $s'$. The alignment found between words in $s$ and words in $s'$ during fuzzy matching are used by an algorithm, analogous to that by Och and Ney (2000), to extract phrase pairs that project mismatched $\sigma$ sub-segments in $s$ into the corresponding sub-segments $\sigma'$ in $s'$. The $\sigma'$ sub-segments are sent to the MT system or other source of bilingual information to obtain their translations $\tau'$. The final result is a set of patching operators that contain translations that match the previously mismatched words in $s'$ and $s$.

In steps 1 through 3, we have already created the $(\sigma, \tau)$ pairs; now we translate $s'$ sub-segments to form $(\sigma', \tau')$ pairs. In our example, the $(\sigma', \tau')$ pairs translated by Apertium (Forcada et al., 2011) are:

| $\sigma'$ | $\sigma$ | positions | $\tau$ | $\tau'$ |
|---|---|---|---|---|
| the **blue** dog | the **red** dog | $\sigma'$=1–3, $\sigma$=1–3 | *el perro **rojo*** | *el perro **azul*** |
| **blue** dog | **red** dog | $\sigma'$=2–3, $\sigma$=2–3 | *perro **rojo*** | *perro **azul*** |
| **blue** | **red** | $\sigma'$=2–2, $\sigma$=2–2 | ***rojo*** | ***azul*** |
| loud when | loud **sometimes** when | $\sigma'$=5–6, $\sigma$=5–7 | *fuerte **a veces** cuando* | *fuerte cuando* |

In the example above, most of the $\tau'$ are aligned word by word to their corresponding $\tau$ in part because their source sub-segments ($\sigma'$ and $\sigma$) are also aligned word by word. Notice however that the last $\tau'$ (*fuerte cuando*) does not align word by word to its corresponding $\tau$ (*fuerte a veces cuando*) because it is a deletion case where the words *a veces* should be deleted.

A patching operator consists of a ($\tau$,$\tau'$) pair and its positions in $t$. To obtain safer patching operators, we keep only those patching operators where there is overlap between $\tau$ and $\tau'$. On top of that, deletions are required to have at least two words (one on each side of the mismatched sub-segment) of overlapping. The fraction of words in $\tau'$ overlapping $\tau$ (and therefore $t$) may be a good indicator of the quality of the patching operator.

The resulting operators from our example with overlap underlined are:

|  | $\tau$ | pos in $t$ | $\tau'$ | result |
|---|---|---|---|---|
| #1 | $\underline{\textit{el perro}}$ *rojo* | 1–3 | $\underline{\textit{el perro}}$ *azul* | **safe**, overlap |
| #2 | $\underline{\textit{perro}}$ *rojo* | 2–3 | $\underline{\textit{perro}}$ *azul* | **safe**, overlap |
| #3 | *rojo* | 3–3 | *azul* | **unsafe**, no overlap |
| #4 | $\underline{\textit{fuerte}}$ *a veces* $\underline{\textit{cuando}}$ | 5–8 | $\underline{\textit{fuerte cuando}}$ | **safe**, deletion with context on both sides |
| #5 | $\underline{\textit{fuerte}}$ *a veces* | 5–7 | $\underline{\textit{fuerte}}$ | **unsafe**, deletion with context on one side only |
| #6 | *a veces* $\underline{\textit{cuando}}$ | 6–8 | $\underline{\textit{cuando}}$ | **unsafe**, deletion with context on one side only |
| #7 | *a veces* | 6–7 | $\epsilon$ | **unsafe**, deletion with no context on either side |

## 2.5 Step 5: applying the patching operators

Once the applicable patching operators have been determined, the final step is to apply them to sub-segments in $t$ to create the final translation that is presented to the translator. It is entirely possible to have multiple combinations of patching operators that form multiple repaired segments $t^{\simeq}$. Here are some possible results of applying patching operators to $t$ from our patching example comparing them to the reference translation $t'$ – *el perro azul ladra fuerte cuando llueve por la noche*:

- $t_1^{\simeq}$ = *el perro azul ladra fuerte cuando llueve por la noche* - (**correct**, produced by #1 and #4 above)
- $t_2^{\simeq}$ = *el perro azul ladra fuerte **a veces** cuando llueve por la noche* - (**incorrect**, produced by #2 above)
- $t_3^{\simeq}$ = *el perro **rojo** ladra fuerte **a veces** cuando llueve por la noche* - (**incorrect**, produced by #4 above)
- $t_4^{\simeq}$ = *el perro azul ladra fuerte cuando llueve por la noche* - (**correct**, produced by #2 and #4 above)

Patching operators can deal with all three types of edit operations (substitution, insertion, and deletion). The patching examples shown above depict a substitution and a deletion example. Nonetheless, insertions can be handled using patching also. Insertions occur when $\tau'$ contains new words not in $\tau$ (most likely because its corresponding $\sigma'$ contained words that were not in the corresponding $\sigma$).

Note that deletions are always produced with some overlap, that is, in context. Context words around a word to be deleted along with their positions are necessary to determine if an operation is valid. The example above, for instance, creates a patching operator (#4) that deletes the word *sometimes*. We are able to use the context of the surrounding words *loud* and *when* to determine deletion. Context used for patching in this manner is different from other research: Hewavitharana et al. (2005), for example, apply all possible context-free deletions according to statistical word alignment, and later score the resulting segments to determine the best translation.

It is worthwhile to note that patching operators, as seen above, may not always be applicable because positions in $t$ that are modified by a patching operator may not be available for another patching operator when $\tau$ does not match the partially-patched sentence. That would make the two patching operators incompatible. Our method generates all possible $t^{\simeq}$ obtained by applying all possible sets of mutually compatible patching operators. In our experiments, $t^{\simeq}$s are determined to be correct (or not) by comparing the $t^{\simeq}$ translation to a previously-translated "gold" segment in a test set. Below, in the Experiments section, all of the patching operators applied are compared against their "gold" $t'$ counterparts. During real-world deployment of fuzzy-match repair, and in the absence of a reference translation, patching operators should be assessed before applying them in order to present only the best quality translations to CAT tool users.

### 2.6 Selecting the best $t^{\simeq}$

The patching process can produce a large amount of patched segments $t^{\simeq}$. Sometimes, various patches exist for the same $(\sigma, \sigma')$ mismatch. In our patching example, three different $(\tau, \tau')$ patching operators are produced that cover the same (*blue*, *red*) mismatch:

1. (*el perro rojo*, *el perro azul*)
2. (*perro rojo*, *perro azul*)
3. (*rojo*, *azul*)

In order to present the optimum translations to a CAT tool user, it would be beneficial to apply various restrictions to discard redundant or low quality patching operators like those above. The restrictions that we have tested in our experiments are:

- *Restriction 1* ($R_1$) = Establish a minimum source-side $\sigma$ length in order to disallow patches that are too short.
- *Restriction 2* ($R_2$) = Disallow patches that do not have context on both sides of the mismatched word to be translated.
- *Restriction 3* ($R_3$) = Set $|\tau'| = |\tau| = 3$ to handle one-word substitutions with a minimum amount of context.

By applying the restrictions above to the corpus used in experimentation, we show that by using the patching method we are able to increase the translation accuracy of proposals from a translation memory via fuzzy-match repair. The amount of patching operators is limited by our restrictions; experiments display the effects of restriction on performance.

## 3 Experiments

### 3.1 Experimental Settings

#### 3.1.1 Corpus

Our experiments were performed using an English–Spanish parallel corpus; note, however, that the fuzzy-match repair method would work with any language pair. We have used a data set[1] extracted from the DGT translation memories,[2] which was also used in a project[3] on editing TL hints in a CAT system. The corpus provides an ideal number of segmented sentences both for input and testing as well as a translation memory.

The two main components that we used for experimentation are:

**Test Set** - The input (in English) and output (in Spanish) containing 1500 source sentences to translate and their corresponding "gold" target sentences.

[1]http://transducens.dlsi.ua.es/~mespla/resources/mtacat/02.40.10.40/
[2]https://open-data.europa.eu/en/data/dataset/dgt-translation-memory
[3]http://www.dlsi.ua.es/~mespla/edithints.html

| FMS Threshold (%) | # Matched SL Seg's | # Patched TL Seg's | # TL Words Edited |
|---|---|---|---|
| [80, 85) | 76 | 686 | 750 |
| [85, 90) | 123 | 239 | 285 |
| [90, 95) | 82 | 115 | 109 |
| [95, 100) | 1 | 1 | 1 |
| Total: | 282 | 1041 | 1145 |

**Table 1:** Segments and target words at different FMS thresholds

**TM** - The translation memory with 6000 translation units in English and Spanish.

Since our implementation is a proof of concept, and to avoid onerous computations, we chose to limit the amount of words in the source segments. Segments ($s$ and $s'$) with more than 25 words or a FMS below 80 percent were discarded because they are harder to deal with in reasonable time due to the combinatorial explosion of patching operator sets.

The amount of mismatched sub-segments varies according to the FMS threshold used. Table 1 shows, for different FMS threshold intervals, the amount of matching (source language) segments, the amount of patched (target) segments and the amount of words in target segments that have been edited.

### 3.1.2 System Setup

Translations for the patching algorithm were performed using an MT system as a black box. In this case, segments were translated directly using the Apertium shallow-transfer MT system (Forcada et al., 2011). The word error rate (WER: see below) on the test set for this rule-based MT system is 56%; this value was computed by translating the source segments in the test set and using their target "gold" translations as references. It is important to note that MT errors are less likely to affect the quality of the patching operators because the $\tau$'s obtained are always required to match a sub-segment in $t$.

## 3.2 Evaluation

To evaluate the performance of patching and the different types of restrictions ($R_1$ – $R_3$) that can be applied to patching operators, we computed the following metrics:

- The average WER between the "gold" translation ($t'$) and the patch translation for all repaired hypotheses ($t^{\simeq}$).
- The average WER between the "gold" translation and the best repaired hypothesis (this metric provides an indication of the best results we can achieve by patching)
- The average number of repaired hypotheses per segment

WER in our experiments is defined as the complement of the FMS defined in eq. (1):

$$\mathrm{WER}(t, t^{\simeq}) = \left( \frac{\mathrm{ED}(t, t^{\simeq})}{\max(|t|, |t^{\simeq}|)} \right) \cdot 100\% \qquad (2)$$

WER tells us how patching performs and by computing the average WER for hypotheses with and without patching along with each individual restriction, we are able to gain better knowledge of restrictions that could be beneficial; that is, those reducing the number of hypotheses while retaining the best hypotheses as often as possible.

| Restriction(s) | Avg Amount of Hypotheses | Avg WER | Best WER |
|---|---|---|---|
| No Patching | 0 | 21.37% | 21.37% |
| No Restrictions | 68.96 | 20.07% | 17.42% |
| Restriction #1 | 14.20 | 20.74% | 18.42% |
| Restriction #2 | 21.99 | 20.07% | 17.43% |
| Restriction #3 | 23.14 | 19.78% | 17.44% |
| Restrictions #1 - #3 combined | 1.67 | 21.03% | 19.62% |

**Table 2:** Evaluation of Restriction 1 through Restriction 3 with 6000 source segments

### 3.3 Patching Results

Patching restrictions that help improve system quality are those described in section 2.6: $R_1$, restricting the $\sigma$ length (restricted here to a minimum of 2 and a maximum of 5); $R_2$, disallowing non-anchored patches, and $R_3$, which sets $|\tau|' = |\tau| = 3$ to handle one-word repairs with minimal context.

Table 2 displays the results achieved for the restrictions and computations described in the Evaluation section. A separate execution is done during evaluation for the following cases: 1) evaluation without applying patching at all, 2) evaluation applying patching without any restrictions, 3) evaluation using each particular restriction, and 4) evaluation using all possible restrictions. A calculation is done that shows the average amount of hypotheses per segment for each setting along with the average WER and the best WER.

The patching technique seems to work better when applied to $\tau$ with less words as seen from the results above for $R_2$. In order to achieve quality patches, we use $(\sigma, \sigma')$ pairs that contain anchored words with overlap on both sides. For example, if the source input file contains a segment with the words “power input of 5 watts” and the translation memory source file contains words in a parallel sentence “power input of about 5 watts”, the system is more likely to correctly replace the “of about 5” by “of 5” due to the fact that there is overlap on both sides of the word “about”.

Table 2 shows a clear distinction between patching with restrictions and without them. Patching improves WER in all cases since the average WER for any restriction $R_i$ is lower than the average WER without patching. But, with respect to restrictions $R_1$ – $R_3$, only $R_3$ is able to improve the average quality of the hypotheses, thus reducing the average WER. In addition, $R_3$’s average amount of hypotheses is considerably lower (23.14). Restriction $R_2$, while not scoring as well as $R_3$, was able to maintain the same quality (average WER of 20.07%) as patching without restrictions with an average amount of hypotheses (21.99) even lower than $R_3$. Restriction $R_1$ is of less quality when compared to patching without restrictions. $R_1$’s best WER (18.42%) is worse, in absolute terms, than $R_2$ and $R_3$’s best WER.

When combining the restrictions ($R_1$ – $R_3$), we notice that both the average WER and Best WER perform considerably worse (a quality loss of near 1% or above). On the other hand, $R_2$ and $R_3$, retain the best WER much like their averages above with a very slight degradation of between .01% and .02%. That means that restrictions $R_2$ and $R_3$ should be considered the better restrictions for future use and; due to its lower performance, $R_1$ would probably not be used in the future.

It is worthwhile to note that another useful way to reduce redundant patches would be by deleting patching operator $(\tau_1, \tau_1')$ when there is another operator $(\tau_2, \tau_2')$ such that $\tau_1$ is a substring of $\tau_2$ and $\tau'_1$ is a substring of $\tau'_2$ or vice-versa. It could be used as a way of reducing patches; but, it does not change the quality of patching as it only reduces patches that cover the same sub-segments. This type of filtering is left for future work and not covered in this paper.

## 4 Conclusion

We have presented a novel approach to fuzzy-match repair using any MT system or bilingual information source called *patching*. Patching focuses on a problem in the CAT environment: presenting accurate translation proposals to CAT tool users. In order to help CAT tool users translate faster, patching may be applied to repair and improve fuzzy-match proposals from a translation memory alone. By using any external source for gathering sub-segment translations, translation systems would be able to take advantage of patching by simply adding the patching library to an existing CAT tool and setting the external source of bilingual information.

In this work, we have applied a preliminary set of restrictions which have been shown to effectively reduce the number of repaired segments while keeping the best ones. We have shown that by applying patching to fuzzy matches from a translation memory, we can achieve better WERs as opposed to raw fuzzy matches. Future research will explore further restrictions and use the results to inspire the design of features relevant to learn quality estimators capable of ranking repaired fuzzy-matches so that only the most useful ones are shown to the professional translator using the CAT environment.

**Acknowledgements:** The authors thank the Spanish Ministry of Economy and Competitiveness for support through grant TIN2012-32615 and Rafael C. Carrasco for useful comments.

## References

Biçici, E. and Dymetman, M. (2008). Dynamic translation memory: Using statistical machine translation to improve translation memory fuzzy matches. *Computational Linguistics and Intelligent Text Processing*, pages 454–465.

Bowker, L. (2002). *Computer-aided translation technology: a practical introduction*. University of Ottawa Press.

Bowker, L. (2003). Terminology tools for translators. In Somers, H., editor, *Computers and Translation: a Translator's Guide*, pages 49–65. John Benjamins.

Brown, R. D., Hutchinson, R., Bennett, P. N., Carbonell, J. G., and Jansen, P. (2003). Reducing Boundary Friction Using Translation-Fragment Overlap. In *Proceedings of the Ninth Machine Translation Summit*, pages 24–31.

Dandapat, S., Morrissey, S., Way, A., and Forcada, M. L. (2011). Using example-based MT to support statistical MT when translating homogeneous data in a resource-poor setting. In *Proceedings of the 15th conference of the European Association for Machine Translation*, pages 201–208. Leuven, Belgium.

Esplà-Gomis, M., Sánchez-Martínez, F., and Forcada, M. L. (2011). Using machine translation in computer-aided translation to suggest the target-side words to change. In *Proceedings of the 13th Machine Translation Summit*, pages 172–179, Xiamen, China.

Forcada, M. L., Ginestí-Rosell, M., Nordfalk, J., O'Regan, J., Ortiz-Rojas, S., Pérez-Ortiz, J. A., Felipe Sánchez-Martínez, G. R.-S., and Tyers, F. M. (2011). Apertium: a free/open-source platform for rule-based machine translation. *Machine Translation*, 25(2):127–144.

Hewavitharana, S., Vogel, S., and Waibel, A. (2005). Augmenting a statistical translation system with a translation memory. In *Proceedings of the 10th conference of the EAMT on 'Practical applications of machine translation'*, pages 126–132, Carnegie Mellon University, Pittsburgh, USA.

Koehn, P. and Senellart, J. (2010). Convergence of translation memory and statistical machine translation. In *Proceedings of AMTA Workshop on MT Research and the Translation Industry*, pages 21–31, Edinburgh, United Kingdom and Paris, France.

Ma, Y., He, Y., Way, A., and van Genabith, J. (2011). Consistent translation using discriminative learning - a translation memory-inspired approach. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1239–1248, Portland, Oregon, USA. Association for Computational Linguistics.

Och, F. J. and Ney, H. (2000). Improved statistical alignment models. pages 440–447, Hongkong, China.

Simard, M. and Isabelle, P. (2009). Phrase-based machine translation in a computer-assisted translation environment. *Proceeding of the Twelfth Machine Translation Summit (MT Summit XII)*, pages 120–127.

Somers, H. (2003). Translation memory systems. In Somers, H., editor, *Computers and Translation: a Translator's Guide*, pages 31–47. John Benjamins.

Wagner, R. A. and Fischer, M. J. (1974). The string-to-string correction problem. *J. ACM*, 21(1):168–173.

Zhechev, V. and Genabith, J. V. (2010). Seeding statistical machine translation with translation memory output through tree-based structural alignment. In *Proceedings of SSST-4 - 4th Workshop on Syntax and Structure in Statistical Translation*, pages 43–49, Dublin,Ireland.