Fuzzy-match repair using black-box machine translation systems: what can be expected?

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

Abstract

Computer-aided translation (CAT) tools often use a translation memory (TM) as the key resource to assist translators. A TM contains translation units (TU) which are made up of source and target language segments; translators use the target segments in the TU suggested by the CAT tool by converting them into the desired translation. Proposals from TMs could be made more useful by using techniques such as fuzzy-match repair (FMR) which modify words in the target segment corresponding to mismatches identified in the source segment. Modifications in the target segment are done by translating the mismatched source sub-segments using an external source of bilingual information (SBI) and applying the translations to the corresponding positions in the target segment. Several combinations of translated sub-segments can be applied to the target segment which can produce multiple repair candidates. We provide a formal algorithmic description of a method that is capable of using any SBI to generate all possible fuzzy-match repairs and perform an oracle evaluation on three different language pairs to ascertain the potential of the method to improve translation productivity. Using DGT-TM translation memories and the machine system Apertium as the single source to build repair operators in three different language pairs, we show that the best repaired fuzzy matches are consistently closer to reference translations than either machine-translated segments or unrepaired fuzzy matches.

1 Introduction

Computer-aided translation (CAT) tools often use a translation memory (TM), containing translation units (TU), as the key resource to assist translators. TU are made up of source and target language segments and translators use the target segments in the TU suggested by the CAT tool by converting them into the desired translation. When an exact match (100%, $s = s'$) is not available one can use a *fuzzy-match repair* method to *repair* a translation proposal t. The aim of these methods is to replace the sub-segments in t that are the translation of the sub-segments in s that do not appear in s' by the translation of the corresponding sub-segment in s'. Fuzzy-match repair is gaining more traction in modern tools such as DejaVu[1] and MemoQ[2] as a reliable method of replacing words in a proposed target-language (TL) segment t by using a source of bilingual information (SBI) such as the very same TM being used, a dictionary, or an on-line translation tool.

[1]`http://www.atril.com/content/10-deepminer-fuzzy-matches-repair-458`
[2]`https://www.memoq.com/whats-new-in-memoq-2015`

Ortega et al. (2014) describe a method that is capable of using any SBI for fuzzy-match repair. This method first aligns the words in the SL segment s of the TU being repaired (s, t) with the words in the segment to be translated s' and identifies the mismatched words in s and s', i.e. the sub-segments they do not have in common. It then uses the SBIs available to identify the sub-segments in t that are the translations of the mismatched sub-segments in s in a way similar to that used by Esplà-Gomis et al. (2011), and then to build a set of *patching operators* by translating the mismatched sub-segments in s'. Each patching operator specifies the TL sub-segment τ in t that needs to be repaired and the TL sub-segment τ' to be used for repairing. Combinations of patching operators can then be applied to obtain a set of candidate repaired TL segments from which the one to be finally used can be selected.

In this paper, we revisit Ortega et al. (2014)'s approach to fuzzy-match repair and go one step further; in particular in this paper we:

- provide an algorithmic description of the method;
- introduce a set of *principled* restrictions by establishing a set of compatibility rules between patching operators so that two patching operators are not applied on the same mismatch;
- extensively evaluate the method at the document level on DGT-TM[3] texts in three different language pairs, namely English–Spanish, Spanish–French and Spanish–Portuguese;
- provide some insight on the results by studying how often patching operators can actually be built using the SBI available.

The rest of the paper is organised as follows. The next section discusses related work on fuzzy-match repair and stresses the main differences with respect to the approach described in this paper. Section 3 then provides an algorithmic description of our fuzzy-match repair method, whereas Section 4 describes the rationale behind the principled restrictions that prevent two patching operators from working on the same mismatch. Sections 5 and 6 discuss the experimental settings and the results of an oracle evaluation we have conducted to determine the potential of the method. The paper ends with some remarks and a description of future research lines.

2 Related Work

In the literature, one can find many papers addressing the combination of machine translation and translation memories, most of which explore different ways of integrating sub-segments from the translation memory into the decoding process of a phrase-based statistical machine translation system (Biçici and Dymetman, 2008; Simard and Isabelle, 2009; Zhechev and Genabith, 2010; Koehn and Senellart, 2010; Li et al., 2016). Alternative approaches, such as those by Dandapat et al. (2011), Hewavitharana et al. (2005) and Kranias and Samiotou (2004), use instead the target segment t in a translation unit (s, t) as the *backbone* or the *basis* of the translation to be produced and describe ways to *repair* it by modifying those sub-segments in t that are the translation of the mismatched sub-segments in s. The method proposed here, which extends that of Ortega et al. (2014), belongs to this second group.

Dandapat et al. (2011)'s method first aligns, in a way similar to ours, the words in s and s' using the (word-based) edit distance (Levenshtein, 1966) and marks the mismatched sub-segments in s and s' for translation. It then aligns the mismatch sub-segments in s with their

[3]The translation memory of the Directorate General for Translation of the European Comission, `https://ec.europa.eu/jrc/en/language-technologies/dgt-translation-memory`

counterparts in t by using a sub-segmental translation memory built on the user's translation memory following the standard method to obtain phrase tables in statistical MT (Koehn, 2010). Finally the sub-segments in t aligned to mismatched sub-segments in s are replaced by the translations of the corresponding sub-segments in s' as they are found in the sub-segmental translation memory. The main differences with the approach described here are (a) that Dandapat et al. (2011) do not take into account the context words around the mismatches —which may lead to incorrect translations due to *boundary friction* problems such as incorrect agreement or incomplete word reorderings— and (b) that they rely on the user's translation memory (which may be small) rather than on an external SBI.

Hewavitharana et al. (2005) first use a modified IBM model 1 to align the mismatched words in s to sequences of one or more words ("phrases") in t and then directly map the sequence of source-side one-word edit operations (substitutions, deletions and insertions) needed to convert s into s', the segment to be translated, into an identical sequence of edit operations on the corresponding word sequences in t to generate the fuzzy-match repaired translation. An important strength of their method is that multiple alternative target-side edits are possible for each source-side insertion or substitution, and that they score them using a probabilistic model. An important limitation of their method (as compared with ours) is the lack of context around source-side one-word edits.

Kranias and Samiotou (2004) use several linguistic resources —such as bilingual dictionaries and lists of suffixes and closed-class words— to align the words in s to those in t and then uses these alignments to identify the words in t to be repaired. Finally, the words to be repaired are replaced (edited, inserted or deleted) by the translation of the corresponding mismatch in s' obtained using machine translation. This method is similar to the one we describe in this paper but differs in that it only uses context around the mismatches when the new segment s' contains words not found in s. In contrast, we always use context when available around all mismatches, which allows us to treat insertions, deletions and substitutions in the same way, and to mitigate the incomplete reordering and agreement errors that may occur because of not using context.

Finally, it is worth noting that commercial computer-aided translation software have recently begun to implement fuzzy-match repair. For example, MemoQ[4] implements a feature called MatchPatch that uses term bases and other resources for fuzzy-match repair, while Déjà Vu implements a feature called DeepMiner[5] that extracts sub-segments from the very same translation memory being used for their use for fuzzy-match repair. Unfortunately, details about how these methods work are not available.

3 Algorithm for Fuzzy-Match Repair

We describe a fuzzy-match repair algorithm that generates a set of candidate fuzzy-match- repaired segments from a translation unit (s, t) and the SL segment to be translated s' by using any SBI. First, we describe the algorithm used to build the list of patching operators to be used for fuzzy-match repair; then, we describe the algorithm that explores all possible combinations of patching operators to generate the set of candidate fuzzy-match repaired segments.

In order to build the list of patching operators (see Algorithm 1), first the alignment between the words in the SL segment to be translated s' and those in the SL segment s in the TU being repaired is obtained by a method based on the (word-level) edit-distance algorithm.[6] The string-positioned sub-segment pairs (σ, σ'), containing unaligned (unmatched) words and their corresponding positions in s and s', are then obtained by using the phrase-pair extraction algorithm used in phrase-based statistical machine translation (Koehn, 2010, section 5.2.3) to

[4]`https://www.memoq.com/whats-new-in-memoq-2015`

[5]`http://www.atril.com/software/dj-vu-x3-professional`

[6]If more than one optimal path is available to align s' and s, on of them is chosen arbitrarily.

Algorithm 1 BuildPatchOp$(s', (s,t))$ generates the set of patching operators to use.

```
Input: SL segment to be translated $s'$; TU $(s,t)$ to be repaired
Output: A list of patching operators $P$
 1: $P \leftarrow ()$  ▷ Initially $P$ is an empty list
 2: $A \leftarrow \mathrm{EditDistanceAligner}(s', s)$  ▷ Get the word alignment between $s$ and $s'$
 3: for $(\sigma, \sigma') \in \mathrm{ExtractPhrasePairs}(s', s, A)$ do
 4:     $M \leftarrow \mathrm{Translate}(\sigma)$  ▷ $M$ is a set with translations of $\sigma$
 5:     $M' \leftarrow \mathrm{Translate}(\sigma')$  ▷ $M'$ is a set with translations of $\sigma'$
 6:     for $\mu \in M$ do
 7:         for $\mu' \in M'$ do
 8:             for $\tau \in \mathrm{FindInSegment}(\mu, t)$ do
 9:                 $\tau' \leftarrow \mathrm{AttachTranslationToString}(\tau, \mu')$
10:                 append $(\sigma, \sigma', \tau, \tau')$ to $P$
11:             end for
12:         end for
13:     end for
14: end for
15: return $P$
```

obtain bilingual phrase pairs. After this, for each sub-segment pair (σ, σ') the pair of sets of translations into the TL (M, M') is obtained by using the SBI available. Finally, the translations in those sets are used to build patching operators by looking for all the occurrences in t of the target sub-segments $\mu \in M$ to get the corresponding string-positioned target sub-segments τ, and then attaching to each τ the target sub-segment μ' to get τ'.

The following example illustrates how the set of patching operators is built. Suppose the segment s' = *Bill found out about the fraud* to be translated into Spanish with the help of the TU $(s,t) =$ (*Gina found out about the news*, *Gina se enteró de las noticias*). The unmatched (unaligned) words in s' are *Bill* and *fraud*, whereas the unmatched (unaligned) words in s are *Gina*, and *news*. After word alignment these are the sub-segments pairs (σ, σ') (up to length 3) which contain at least an unmatched word together with their translations (μ, μ') into Spanish:[7]

σ	σ'	μ	μ'	μ in t?
Gina found	*found*	*Gina encontró*	*encontró*	no
Gina found	*Bill found*	*Gina encontró*	*Bill encontró*	no
Gina found out	*found out*	*Gina se enteró*	*se enteró*	yes
Gina found out	*Bill found out*	*Gina se enteró*	*Bill se enteró*	yes
found	*Bill found*	*encontró*	*Bill encontró*	no
found out	*Bill found out*	*se enteró*	*Bill se enteró*	yes
about the	*about the fraud*	*sobre el*	*de la estafa*	no
about the news	*about the*	*de noticias*	*sobre el*	yes
about the news	*about the fraud*	*de las noticias*	*de la estafa*	yes
the	*the fraud*	*el*	*la estafa*	no
the news	*the*	*las noticias*	*el*	yes
the news	*the fraud*	*las noticias*	*la estafa*	yes

Only in those cases in which μ, the translation of σ, is found in the target segment t of the TU being repaired a patching operator can be built; this is indicated by the fifth column in the table

[7]Note that the string-positioned sub-segment pairs (σ, σ') extracted from s and s' always contain an aligned word in s or s'. In this example we are assuming that the sets of translations M and M' of σ and σ' are singletons.

Algorithm 2 Patching$(P, O, n, (s,t), t^{\simeq}, D, T)$ generates all possible fuzzy-match repaired segments by backtracking.

```
Input: List of patching operators $P$; set of patching operators $O$ applied so far; position in $P$
    of the patching operator being considered, $n$; TU to be repaired $(s,t)$; fuzzy-match repaired
    segment being built $t^{\simeq}$; boolean $D$ indicating whether the $n$-th patching operator in $P$ will
    be attempted to apply (true) or not (false), list $T$ containing fuzzy-match-repaired segments
 1: if $D$ then
 2:     if $\mathrm{Compatible}(P_n, O, (s,t))$ then
 3:         $\mathrm{ApplyPatchOp}(P_n, t^{\simeq})$
 4:         $O \leftarrow O \cup \{P_n\}$   ▷ Add compatible patching operator
 5:     else
 6:         return   ▷ Prune this branch of the recursion tree
 7:     end if
 8: end if
 9: if $n = \mathrm{length}(P)$ then
10:     append $t^{\simeq}$ to $T$   ▷ Add candidate fuzzy-match repaired segment to list $T$
11:     return   ▷ All the patching operators have been considered
12: else
13:     $\mathrm{Patching}(P, O, n+1, (s,t), t^{\simeq}, \textbf{true}, T)$   ▷ Continue by applying operator $n+1$
14:     $\mathrm{Patching}(P, O, n+1, (s,t), t^{\simeq}, \textbf{false}, T)$   ▷ Continue by not applying operator $n+1$
15: end if
```

above.

Algorithm 2 generates the set of all possible fuzzy-match repaired segments by using those sets in $\mathcal{P}(P)$ (the power set of P) containing compatible patching operators. This is achieved through a backtracking algorithm that performs a recursive depth-first search and incrementally builds fuzzy-match repaired segments $t^{\simeq}$; the algorithm is initialized with two calls $\mathrm{Patching}(P, \emptyset, 1, (s,t), t, \textbf{false}, ())$ and $\mathrm{Patching}(P, \emptyset, 1, (s,t), t, \mathbf{true}, ())$, where $()$ stands for an empty list. At each level of the recursion tree a new patching operator is considered and tested for applicability ($D = \mathbf{true}$) or discarded ($D = \mathbf{false}$). For a patching operator to be applicable it needs to be compatible with the set of patching operators O applied so far to build $t^{\simeq}$ (see Section 4). If it is compatible with the rest of patching operators in O, the patching operator is added to O and applied (lines 3–4); otherwise the branch of the recursion tree is cut. When a leaf of the recursion tree is reached (i.e. $n = \mathrm{length}(P)$) the corresponding fuzzy-match repaired segment $t^{\simeq}$ is added to the list T of candidate fuzzy-match repaired segments. The algorithm $\mathrm{ApplyPatchOp}(o, t^{\simeq})$ replaces in $t^{\simeq}$ the sub-segment τ by τ'; this can be safely done if patching operator P_n is compatible with the other patching operators applied so far.

This algorithm assumes that patching operators that are compatible can be applied in any order because the repaired segment to be generated would be the same. Thanks to this assumption, the worst-case complexity of the algorithm is $O(2^n)$, with $n = \mathrm{length}(P)$, in which case 2^n repaired segments are produced. If the algorithm had to explore the application of all the patching operators in P and in all possible orders its worst-case complexity would be super-exponential.

For the example introduced above, Algorithm 2 would produce 128 repaired segments if all patching operators were compatible. However, most of them are not compatible because they edit the same words in t (see next section) and the algorithm ends up producing only 25 repaired segments. Some of these 25 repaired segments are identical but are produced by applying a different set of patching operators. For instance, the repaired segment *Bill se enteró de la estafa* is produced by applying the patching operator $(\textit{Gina found out}, \textit{Bill found out}, \textit{Gina se enteró}, \textit{Bill se enteró})$ and either the patching operator $(\textit{about the news}, \textit{about the fraud}, \textit{de las noticias}, \textit{de la estafa})$ or the patching operator $(\textit{the news}, \textit{the fraud}, \textit{las noticias}, \textit{la estafa})$.

4 Restrictions

Ortega et al. (2014) introduce three restrictions: two related to the type of sub-segments used to build the patching operators and a third one related to the words in t being edited. The first two restrictions —one restricting the length of the sub-segments and the other one requiring a certain amount of context words around mismatches— are optional and were introduced to reduce the number of patching operators to be considered. These optional restrictions throw away *legal* repairs that are however considered to be of low quality and will not be applied for the experiments reported in Section 5.

The third restriction cannot be avoided and is needed in order to prevent two patching operators from editing the same word in t. However, it may happen that two patching operators working on the same mismatch do not edit any of the words in t but introduce missing ones. In those cases, the fuzzy-match repair algorithm of Ortega et al. (2014) may end up producing candidate fuzzy-match repaired segments $t^{\simeq}$ with repeated words. The following example illustrates this situation. Suppose the segment $s' = \textit{the size does not exceed 100 cm}$ to be translated with the help of the translation unit $(s, t) = (\textit{the size does not exceed 100}, \textit{el tamaño no supera los 100})$ whose target segment can be repaired with the two patching operators $(\sigma_1, \sigma'_1, \tau_1, \tau'_1) = (\textit{exceed 100}, \textit{exceed 100 cm}, \textit{supera los 100}, \textit{supera los 100 cm})$ and $(\sigma_2, \sigma'_2, \tau_2, \tau'_2) = (\textit{100}, \textit{100 cm}, \textit{los 100}, \textit{los 100 cm})$. As both patching operators do not edit (change) any word in t they could be applied one after the other and produce the fuzzy-match repaired segment $t^{\simeq} = \textit{el tamaño no supera los 100 cm cm}$, which contains duplicated words due to the fact that the word *cm* is to be inserted by both operators.

To avoid this problem we need to identify when two patching operators work on the same mismatch, and to do so one needs to check the mismatches both in s and s' because there may be words in s not appearing in s' (the mismatch only shows up in s), or words that do not appear in s but are introduced in s' (the mismatch only shows up in s', as in the example above). Hence two patching operators $o_i = (\sigma_i, \sigma'_i, \tau_i, \tau'_i)$ and $o_j = (\sigma_j, \sigma'_j, \tau_j, \tau'_j)$ will be marked as incompatible if they edit the same word in t (as in the work by Ortega et al. (2014)) or they meet the following condition:

$$(\text{mismatch}(\sigma_i, s) \cap \text{mismatch}(\sigma_j, s) \neq \emptyset) \vee (\text{mismatch}(\sigma'_i, s') \cap \text{mismatch}(\sigma'_j, s') \neq \emptyset)$$

where $\text{mismatch}(x, y)$ returns the set of mismatch words covered by sub-segment x in segment y.

It is worth nothing that this new restriction may mark as incompatible two patching operators that, even though they work on the same mismatch, do not edit the same words in t. In those cases it is still advisable to forbid the application of the two patching operators since it is very likely that they work on the same region in t and their application interfere with one another. The following example illustrates this situation. Suppose the segment $s' = \textit{the size is around 100 cm}$ to be translated with the help of the translation unit $(s, t) = (\textit{the size is about 50 cm}, \textit{el tamaño es de unos 50 cm})$ whose target segment can be repaired with the two patching operators $o_1 = (\sigma_1, \sigma'_1, \tau_1, \tau'_1) = (\textit{is about}, \textit{is around}, \textit{es de unos}, \textit{está alrededor de})$ and $o_2 = (\sigma_2, \sigma'_2, \tau_2, \tau'_2) = (\textit{about 50}, \textit{around 100}, \textit{de unos 50}, \textit{de unos 100})$. Both operators share a mismatch (*about*) but do not edit the same words in t: o_1 edits the word *es* (which is replaced by *está*), introduces

		en–es	**es–pt**	**es–fr**
TM	# TUs	196,294	150,567	149,479
	Avg. SL segment length	9.61	27.24	27.35
Test set	# SL segments	1993	1983	1983
	# SL words	40238	45334	46350
	Avg. SL segment length	20.19	22.67	21.73

Table 1: Data about the translation memories and test sets used in the experiments.

the word *alrededor* and removes (edits) the word *unos*; o_2 edits the word *50* and replaces it by *100*. The two operators can be applied at the same time if operator o_2 is applied first —the repaired target segment being $t^{\simeq}$ = *el tamaño está alrededor de 100 cm*— but not the other way around. Recall that the algorithm described in Section 3 assumes that patching operators can be applied independently of each other and the order in which they are applied does affect the final result.

5 Experimental settings

To evaluate the potential of the fuzzy-match repair algorithm described in Section 3, we have performed an oracle evaluation (see below) on three different language pairs: English–Spanish (en–es), Spanish–Portuguese (es–pt) and Spanish–French (es–fr). These language pairs have been chosen to study how the method behaves when translating between closely-related languages (e.g. Spanish–Portuguese and Spanish–French) and when the languages involved in the translation are not so closely related (English–Spanish). In addition, of the two closely-related language pairs we have used, Spanish and Portuguese are more alike than Spanish and French: Spanish and Portuguese are both pro-drop, Ibero-Romance languages —they permit null-subject sentences— whereas French is a non-pro-drop Gallo-Romance language.

As for the corpora used for the experiments, we have used three translation memories, one per language pair, extracted from the DGT-TM 2015 multilingual translation memory;[8] each translation memory contains between 145,000 and 200,000 translation units. In addition, we have also extracted three test sets from the same source. Each test set contains around 2,000 parallel segments with source segments no longer than 100 words. The experiments consist of simulating the translation of each source segment in the test sets by using the translation memories and using the corresponding target-language segment as a reference for evaluation. Table 1 provides additional information about the translation memories and test sets used.

As a source of bilingual information we have used the free/open-source machine translation platform Apertium (Forcada et al., 2011),[9] which provides a single translation for each source segment;[10] more precisely, we have used the language-pair packages `apertium-en-es`,[11] `apertium-es-pt`[12] and `apertium-fr-es`.[13] Apertium has been used both to build patching operators by translating sub-segments σ into the target language and to translate the segments in the test set for which a fuzzy match above the given threshold has not been found. Table 2 provides the word error rate (WER) and BLEU scores attained by Apertium when translating the source-language segments in the test set; the percentage of out-of-vocabulary words (OOV) is also reported. As can be seen, the translations performed by

[8]`https://ec.europa.eu/jrc/en/language-technologies/dgt-translation-memory`
[9]`https://www.apertium.org`
[10]That is, sets M and M' in lines 4 and 5 of Algorithm 1 are singletons in this case.
[11]SVN revision 64348.
[12]SVN revision 62539.
[13]SVN revision 62696.

	en–es	**es–pt**	**es–fr**
WER	65.3%	47.4%	55.2%
BLEU	18.6%	36.4%	24.7%
OOV	2.6%	2.4%	2.4%

Table 2: Apertium's performance on the test sets and percentage of out-of-vocabulary words (OOV).

Apertium need less post-editing in the case of the two closely-related language pairs (es–pt and es–fr) than in the case of English–Spanish.

Finally, we evaluate the potential of our fuzzy-match repair method with fuzzy-match score thresholds of 60%, 70% and 80% with the aim of studying whether out method is more capable of repairing fuzzy matches above a given threshold. In this regard it is worth noting that professional translators usually set the fuzzy-match score threshold above 60% (Bowker, 2002).

5.1 Oracle Evaluation

The way to study the potential of our approach for fuzzy-match repair has been to generate, for each source segment s' in the test set, the set of all possible fuzzy-match repaired target segments T and then use t', the translation of s', to choose the best one and evaluate its quality. Obviously, in a real setting t' would not be available and the best fuzzy-match repaired segment would need to chosen using a method similar to those used for estimating the quality of machine translation output (Specia and Soricut, 2013; Avramidis, 2013).

What follows is a detailed explanation of the procedure we have followed with each source segment s' in the test set:

1. Retrieve the set of translation units U whose fuzzy-match score $\mathrm{FMS}(s', s)$ is above the desired fuzzy-match threshold θ.
2. If there is no translation unit (s, t) so that $\mathrm{FMS}(s', s) \geq \theta$, i.e. $U = \emptyset$, use machine translation to get a translation for s'. Otherwise use the TU $(s, t) \in U$ with the highest $\mathrm{FMS}(s', s)$ and produce the set T with all possible target fuzzy-match repaired segments.
3. Select the fuzzy-match repaired segment $t^{\simeq *} \in T$ with the minimum edit distance to t'.

Once all the segments in the test set have been processed the translations produced are evaluated by comparing them to the target segments in the test set and computing the error rate over the whole test set as follows:

$$\frac{\sum_{i=0}^{N} \mathrm{ED}(t_i^*, t_i')}{\sum_{i=0}^{N} \max(|t_i^*|, |t_i'|)} \quad (1)$$

where $\mathrm{ED}(x, y)$ returns the word-based edit distance between the segments x and y, N is the number of segments in the test set, and $|x|$ is the number of words of segment x. This way of computing the error rate resembles the way in which the fuzzy-match score is computed.[14]

6 Results and Discussion

Table 3 shows, for the three different language pairs on which we have evaluated our approach and for three different fuzzy-match score thresholds (FMT) —60%, 70% and 80%—, the error rate computed as described in Equation (1) when:

[14]For instance, OmegaT (`http://www.omegat.org`) computes the fuzzy-matching score between s and s' as $1 - \frac{\mathrm{ED}(s,s')}{\max(|s|,|s'|)}$.

FMT: 60%	en–es			es–pt			es–fr		
	TM	MT	FMR	TM	MT	FMR	TM	MT	FMR
Error (%)	55.0	65.3	36.5	56.5	47.4	31.3	56.4	55.2	34.7
Er. (%) on matches	20.1	65.3	17.9	22.5	47.4	17.0	20.3	55.2	16.5
# matches	1184	1993	1184	1221	1983	1221	1206	1983	1206
Avg. length	22.6	22.1	22.6	21.1	20.6	21.1	22.8	22.4	22.8

FMT: 70%	en–es			es–pt			es–fr		
	TM	MT	FMR	TM	MT	FMR	TM	MT	FMR
Error (%)	61.0	65.3	38.5	62.4	47.4	31.8	62.3	55.2	35.6
Er. (%) on matches	16.3	65.3	14.6	18.0	47.4	13.9	15.8	55.2	12.8
# matches	828	1993	828	777	1983	777	786	1983	786
Avg. length	22.4	22.1	22.5	20.8	20.6	21.1	22.6	22.4	22.6

FMT: 80%	en–es			es–pt			es–fr		
	TM	MT	FMR	TM	MT	FMR	TM	MT	FMR
Error (%)	69.7	65.3	42.6	70.1	47.4	33.8	69.5	55.2	38.2
Er. (%) on matches	13.1	65.3	11.9	15.3	47.4	11.9	12.2	55.2	9.7
# matches	660	1993	660	641	1983	641	649	1983	649
Avg. length	22.3	22.2	22.4	20.8	20.6	21.1	22.5	22.4	22.8

Table 3: For the three different language pairs considered in our evaluation and for three different means of translation —translation memory (TM), machine translation (MT) and fuzzy-match repair (FMR)— and fuzzy-match score thresholds (FMT), the table gives the error rate over the whole test set, the error rate over the segments in the test set for which a match above the given threshold is found in the translation memory, the amount of these segments (# matches) and the average length of the target segments produced.

TM: the target segment in the translation unit with the highest fuzzy-match score is used as a translation, if available; otherwise, an empty translation is used, and therefore the error reflects the need to type the words in the reference translation.

MT: the same machine translation system used as SBI (Apertium) is used to translate the source segments in the test set.

FMR: the translation to be evaluated is obtained by applying the fuzzy-match repair algorithm described in Section 3 with the translation unit with the highest fuzzy-match score, if available; otherwise, the translation is produced using machine translation.

Two error rates are reported, one computed on the whole test set and another computed only on the set of segments for which a TU with a fuzzy-match score above the given threshold is found (error on matches). The former provides and indication of the actual translation effort a translator would made to translate the source segments in the test set. The latter provides an indication of the performance of our method for fuzzy-match repair (FMR) without the interference of whole-segment machine translation, since it focuses only on those segments for which there is a translation unit to repair. This allows to directly compare FMR performance to that of using the target segment in the best TU without any repair (TM). In addition, the number of source segments for which a match is found in the translation memory and the average length of the translations produced are provided.

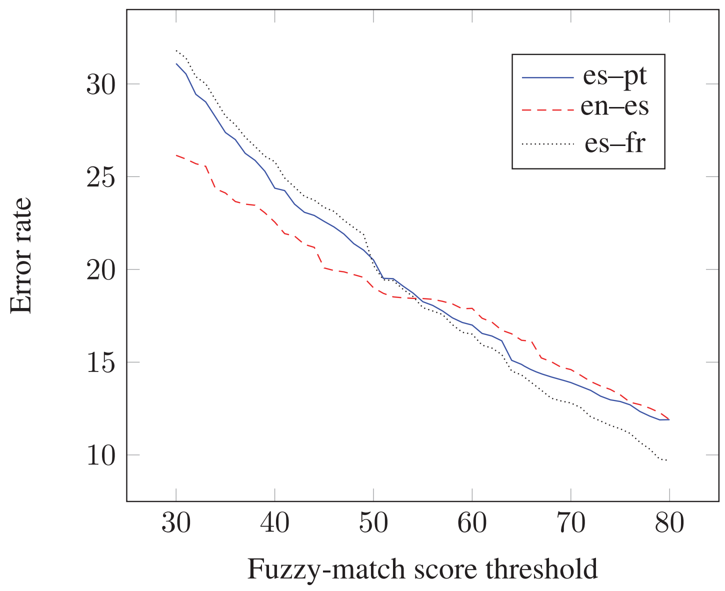
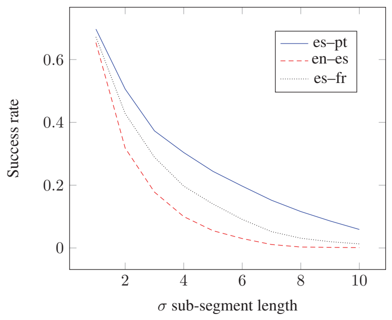


Figure 1: For the three language pairs used for evaluation, error rate over the segments in the test set for which a match above the fuzzy-match score threshold is found in the translation memory.

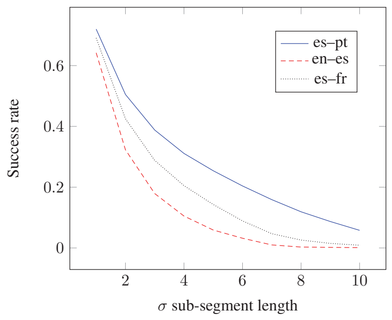
As can be seen, our method for fuzzy-match repair has the potential (recall that this is an oracle evaluation) to improve the translator's productivity for all three different language pairs: the error rate is both below that of using the target segment in the best translation unit (TM) and below that of using machine translation (MT). Furthermore, it is worthwhile to note that a good part of the difference in performance between the three language pairs can be attributed to the performance of the MT system; if we pay attention to the performance of FMR when the evaluation only focuses on those segments for which a match has been found we can see that the scores reported are quite similar for all language pairs, even though this does not happen in the case of the MT scores reported, i.e. our method for fuzzy-match repair appears to be quite robust to MT errors.

The error rate over the whole test set grows with the fuzzy-match score threshold (FMT). This happens because the greater this threshold is, the less source segments can be translated using fuzzy-match repair and, as a consequence, the amount of segments that are translated with Apertium grows. If we focus only on those segments that can be translated by means of FMR, we can see that the error rate decreases as the threshold grows; Figure 1 show how the error rate on matches behaves as a function of the fuzzy-match score threshold. This is the expected behaviour because as the threshold grows the amount of words to repair decreases.

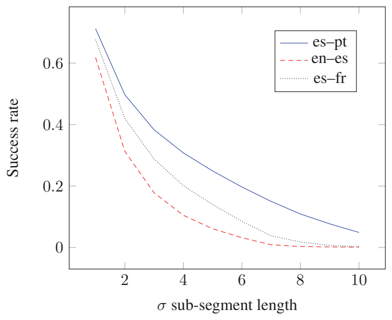
With respect to the process of building patching operators, and provided that the performance of the machine translation system differs between the language pairs, it is worth studying how successful it is our method when it comes to use Apertium to build patching operators. Figure 2 plots the success rate when building patching operators as a function of the length of the source sub-segments σ for a fuzzy-match score threshold of 60%, 70% and 80%; as can be seen, success rates for different fuzzy-match thresholds behave very similarly. A patching operator is successful when the translation of the sub-segment σ of s is found in t, that is, when the machine translation system and the proposed translation unit exactly agree on the translation



(a) σ success rate at 60% FMT



(b) σ success rate at 70% FMT



(c) σ success rate at 80% FMT

Figure 2: For the three different language pairs and for fuzzy-match score thresholds (FMT) of 60%, 70%, and 80% success rate when building patching operators as a function of the number of words in the source sub-segments σ being translated.

of a source sub-segment: this acts as a safety feature, as patching is not attempted when this agreement is absent. This is why our method is robust to machine translation errors.

As can be seen, the longer the sub-segments the harder it is that the translation obtained from the SBI is found in t. This behaviour is present in all the language pairs and is more pronounced when the translation involves non-related language pairs (en–es) than when the languages are closely related (es–pt). The average length of σ in the patching operators used to build the repaired target segment chosen by the oracle when the fuzzy-match score threshold is set to 80% is around 2.8 words for en–es, 3.7 for es–fr and 4.7 for es–pt.

7 Concluding Remarks

In this paper we have extended the approach of Ortega et al. (2014), which uses any external source of bilingual information to repair fuzzy matches coming from a translation memory, to prevent two patching operators from working on the same mismatch, and we have extensively evaluated its performance on three different language pairs and provided a more formal algorithmic description.

The oracle evaluation we have conducted reveals the potential of our approach to fuzzy-match repair. For three different language pairs we consistently improve the quality of the translations produced —both with respect to raw machine translation or unrepaired fuzzy matches— even though the SBI we have used (the machine translation system Apertium) performs below the state of the art for some language pairs. We hope that by combining different SBIs, e.g.

different machine translation systems as well as bilingual concordancers,[15] the quality of the repaired segments increase.

As a future work we plan to combine different SBI and try different methods to automatically select the best fuzzy-match repair for a given SL segment. In particular we will adapt existing techniques used for sentence-level machine translation quality estimation and devise a set of features specially designed to tackle this particular problem.

References

Avramidis, E. (2013). Sentence-level ranking with quality estimation. *Machine Translation*, 27(3-4):239–256.

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.

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. (2010). *Statistical Machine Translation*. Cambridge University Press, New York, NY, USA, 1st edition.

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.

Kranias, L. and Samiotou, A. (2004). Automatic translation memory fuzzy match post-editing: a step beyond traditional TM/MT integration. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation*, pages 331–334, Lisbon, Portugal.

Levenshtein, V. (1966). Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady.*, 10(8):707–710.

Li, L., Parra Escartín, C., and Liu, Q. (2016). Combining translation memories and syntax-based smt. *Baltic Journal of Modern Computing*, 4:165–177.

[15]Such as Reverso Context, `http://context.reverso.net/translation`, Linguee, `http://www.linguee.com/`, or TransSearch, `http://tsrali.com`

Ortega, J. E., Sánchez-Martínez, F., and Forcada, M. L. (2014). Using any machine translation source for fuzzy-match repair in a computer-aided translation setting. In *Proceedings of the 11th Biennial Conference of the Association for Machine Translation in the Americas (AMTA 2014, vol. 1: MT Rsearchers)*, pages 42–53, Vancouver, BC, Canada.

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.

Specia, L. and Soricut, R. (2013). Quality estimation for machine translation: preface. *Machine Translation*, 27(3-4):167–170.

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.