Fuzzy-match repair guided by quality estimation

John E. Ortega, Mikel L. Forcada, Felipe Sánchez-Martínez

Abstract—Computer-aided translation tools based on translation memories are widely used to assist professional translators. A translation memory (TM) consists of a set of translation units (TU) made up of source- and target-language segment pairs. For the translation of a new source segment s', these tools search the TM and retrieve the TUs (s, t) whose source segments are more similar to s'. The translator then chooses a TU and edit the target segment t to turn it into an adequate translation of s'. *Fuzzy-match repair* (FMR) techniques can be used to automatically modify the parts of t that need to be edited. We describe a language-independent FMR method that first uses machine translation to generate, given s' and (s, t), a set of candidate fuzzy-match repaired segments, and then chooses the best one by estimating their quality. An evaluation on three different language pairs shows that the selected candidate is a good approximation to the best (oracle) candidate produced and is closer to reference translations than machine-translated segments and unrepaired fuzzy matches (t). In addition, a single quality estimation model trained on a mix of data from all the languages performs well on any of the languages used.

Index Terms—fuzzy-match repair, computer-aided translation, translation memories, quality estimation

1 Introduction

Computer-aided translation (CAT) tools [1] based on translation memories are one of the most used translation technologies among professional translators [2]. They exploit a translation memory (TM), that is, a collection of translation units (TU), to improve translation productivity by providing the translators with target segments similar to those they have to produce. In particular, a TU (s, t) is made up of a source language segment s and a target language segment t; TM-based CAT tools exploit them by looking for TUs whose source segment are similar to the segment s' to be translated. When, instead of an exact match (100%, $s = s'$), a *fuzzy match* is available, the translator has to manually edit the translation proposal t to make the changes necessary to produce t', an adequate translation of s'. Alternatively, they can use a *fuzzy-match repair* (FMR) method [3], [4], [5], [6], [7] to *repair* the translation proposal t. The aim of FMR 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'. Therefore, FMR aims at translating only the mismatched sub-segments and keeping the parts of t that can be reused because they have been professionally translated.

In this paper, we describe a method for FMR that first generates a set of candidate fuzzy-match repaired segments from a TU (s, t) and the source segment to be translated s', and then estimates the quality of these candidate fuzzy-match repaired segments to select the best one.

The algorithm for generating the set of candidate fuzzy-match repaired segments, introduced in a conference paper by the same authors [6], can use any available source of bilingual information (SBI), such as an on-line MT system, for the translation of the mismatched sub-segments. It first aligns the words in the source segment s of the TU being repaired (s, t) with the words in the source segment to be translated s' and identifies the mismatched words in s and s', that is, the sub-segments they do not have in common.[1] 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 of [8], and builds a set of *repair operators* by translating the mismatched sub-segments in s'. Each repair 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 compatible repair operators are then applied to obtain a set of candidate fuzzy-match repaired segments from which the one to be finally used can be selected.

The method for estimating the quality of the candidate fuzzy-match repaired segments produced, inspired by the work on sentence-level MT quality estimation (QE) [9], [10], uses regressors trained on a combination of black-box, system-independent features and glass-box, system-dependent features. In our experiments we use *extremely randomised trees* [11] as regressors and evaluate our approach on three different language pairs for varying fuzzy-match score thresholds. The results show that the set of features we propose are informative enough to obtain regressors that allow us to select candidate fuzzy-match repaired segments close to the best (oracle) one among those produced by our FMR algorithm. The selected candidates are consistently closer to the reference translations in our test sets than both the non-repaired target segments t and the translations obtained by translating whole source segments s' using the MT system used as SBI. Moreover, the best regressors are most of the times obtained when they are trained on all language pairs together, which signifies that the values of the features proposed and the regressors learned are language-independent.

1. This is usually obtained as a by-product of fuzzy matching.

- *J.E. Ortega is with the Dep. de Llenguatges i Sistemes Informàtics, Universitat d'Alacant, Spain. E-mail: jeo10@alu.ua.es*
- *M.L. Forcada and F. Sánchez-Martínez are with the Dep. de Llenguatges i Sistemes Informàtics, Universitat d'Alacant, Spain. E-mail: {mlf,fsanchez}@dlsi.ua.es*

To appear in IEEE Transactions on Pattern Analysis and Machine Intelligence. Accepted version. DOI: 10.1109/TPAMI.2020.3021361

The rest of the paper is organised as follows. The next section discusses related work on FMR and stresses the main differences with respect to the approach described in this paper. Section 3 then presents the algorithm used to generate all possible candidate fuzzy-match repaired segments, whereas Section 4 describes the QE approach we propose to automatically select the best one. Sections 5 and 6 discuss, respectively, the experimental settings and the results obtained when evaluating the success of our approach. Section 7 ends the paper with concluding remarks.

2 Related Work

Most of the approaches combining MT and TM do so by integrating sub-segments from the TM into the decoding process of a phrase-based statistical MT system [12], [13], [14], [15], [16], [17] or a neural MT system [18], [19]. There are also approaches that incorporate target segments retrieved from the TM along with the source segment to be translated [20] so that the NMT system can use them to produced translation closer to the TM matches. Other approaches [3], [4], [5], [7], however, use the target segment t in a TU (s, t) as the *basis* of the translation to be produced instead of as an additional source of information: they *repair* t by modifying those sub-segments that are the translation of the sub-segments not common to s and s', the segment to be translated. The method described in this paper belongs to this last group.[2]

Kranias and Samiotou [3] 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 use these alignments to identify the words in t to be repaired. The words to be repaired are then replaced (edited, inserted or deleted) by the translation of the corresponding mismatch in s' obtained using MT. This method is similar to the one we describe in this paper, but differs in that it uses context around the mismatches only when the new segment s' contains words not found in s that need to be inserted. In contrast, we use context around all mismatches, when available, and this allows us to treat insertions, deletions and substitutions in the same way. It also allow us to mitigate the incomplete reordering and agreement errors that may occur because of not using context. In addition, in [3] only a single fuzzy-match repaired segment is produced, whereas we produce as many fuzzy-match repaired segments as possible and then select the best one using QE techniques.

In [4] the authors first use a modified IBM model 1 to align the mismatched words in s to sequences of one or more words 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' into an identical sequence of edit operations on the corresponding word sequences in t to generate the repaired translation. An important strength of the method in [4] is that multiple alternative target-side edits are possible for each source-side insertion or substitution, and that they are scored using a probabilistic model. An important limitation, as compared with ours, is the lack of source context around source-side one-word edits.

The method described in [5] first aligns, in a way similar to ours, the words in s and s' using the (word-based) edit distance [21] and marks the mismatched sub-segments in s and s' for translation. The mismatched sub-segments in s are then aligned with their counterparts in t by using a sub-segmental TM built on the user's TM by means of the method used to obtain phrase tables in statistical MT [22]. 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 TM. There are three main differences with the approach described in this paper. First, context around mismatches is not taken into account, which may lead to incorrect translations due to *boundary friction* problems [23, p. 341], such as incorrect agreement or incomplete word reorderings. Second, the methods relies on the user's TM (which may be small) rather than on an external SBI. And third, a single fuzzy-match repaired segment is produced, even when the sub-segmental TM contains several translation alternatives per sub-segment, whereas we generate as many fuzzy-match repaired segments as possible and then use QE to select the best one.

A method for FMR to be applied on close matches is described in [7]. It first performs a simple punctuation repair and then applies a set of edit operations —deletions, insertions and substitutions— to the target segment t of the TU (s, t) being repaired. To detect the target sub-segments to be repaired it uses statistical word alignment models. For deletions, it just removes from t the translation of the mismatched sub-segment in s, and the word to the left of the sub-segment to be removed if it is not aligned to any word in s. For insertions, it inserts in t the new sub-segment between the two target words aligned with the matched words in s surrounding the sub-segment in s' not appearing in s. For substitutions, it translates *anchored* mismatched sub-segments in s', that is, mismatched sub-segments surrounded by words common to s, one word at each end. The mismatched sub-segments are translated in-context by translating the whole segment s' using statistical MT and constraining the output of the MT system to use the sub-segments of t aligned to sub-segments of s common to s' in a way similar to that used in [13], [14], [15]. The main differences between this FMR approach and ours is that it uses statistical word alignments and treats in a different way substitutions, deletions and insertions, whereas our approach does not explicitly compute the alignment between the words in s and t and simply performs string substitutions in t. In addition, this approach is not able to repair a target segment t if the translation of the mismatched source sub-segment does not consist of contiguous words in t; our approach does not have this limitation because it does not impose any constraint on the amount of anchor words surrounding a mismatch sub-segment and their location. Lastly, it can only use MT systems, such as Moses [24], that allow part of the translation to be fixed beforehand; in contrast, our approach can benefit from any available SBI.

Finally, it is worth noting that some commercial CAT tools implement FMR methods. For example, MemoQ[3] implements a feature called MatchPatch that uses term bases

2. This section updates the related work section of a conference paper by the same authors [6] and contains paragraphs taken from that paper.

3. https://www.memoq.com/whats-new-in-memoq-2015

and other resources for FMR, while Déjà Vu implements a feature called DeepMiner[4] that extracts sub-segments from the very same TM being used for their use for FMR. Unfortunately, details about how these methods work are not available.

3 Generation of candidate fuzzy-match repaired segments

This section describes the algorithm to generate a set of candidate fuzzy-match repaired segments from a TU (s, t) and the source segment to be translated s' using any SBI. [5]

In order to generate as many candidate fuzzy-match repaired segments as possible, first a list of *repair operators* is built (Section 3.1), and then all possible combinations of repair operators are explored to generate the set of candidate fuzzy-match repaired segments (Section 3.2).

A repair operator is a 4-tuple $(\sigma, \tau, \sigma', \tau')$, where

- σ is a *string-positioned* sub-segment of s —that is, a sub-segment of s together with an indication of word positions spanned—,
- σ' is a string-positioned sub-segment of s' aligned with σ,
- τ is a string-positioned sub-segment of t which is also one of the possible translations of σ according to the SBI, and
- τ' is one of the translations of σ' according to the SBI, indeed the sub-segment to be used for repairing t by replacing τ.

Repair operators work locally and fully-repaired segments can be built by combining them.

3.1 Generation of repair operators

Algorithm 1 describes the procedure for building the list of repair operators to be applied for the generation of candidate fuzzy-match repaired segments.

To obtain the set of repair operators to be used, first, the alignment between the words in s' and those in s is obtained as a by-product of the computation of the (word-level) edit-distance [21] between s' and s, a component of the fuzzy-match score. 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 MT to obtain bilingual phrase pairs [22, section 5.2.3]. These sub-segment pairs are then translated into the target language to obtain the sets M and M' of sub-segment translations μ and μ' respectively using the SBI available. Finally, these translations are used to build repair operators by looking for all the occurrences in t of each target sub-segment μ to get the corresponding string-positioned target sub-segments τ, and then associating to each τ the target sub-segment μ' to get τ', the sub-segment to be used for repairing. If μ is not found in t, no repair operator can be built. This acts as a quality check that prevents the algorithm from building low-quality repair operators.

4. https://atril.com/key-features/

5. This section is a copy of a similar section in a conference paper by the same authors [6] and is included here for the shake of completeness.

Algorithm 1 BuildRepairOp$(s', (s, t))$ generates the list of repair operators to use.

Input: SL segment to be translated s'; TU (s, t) to be repaired
Output: A list of repair operators P

```
P ← ()   ▷ Initially P is an empty list
A ← EditDistanceAligner(s', s)
for (σ, σ') ∈ ExtractPhrasePairs(s', s, A) do
    M ← Translate(σ)   ▷ Set with translations of σ
    M' ← Translate(σ')   ▷ Set with translations of σ'
    for μ ∈ M do
        for μ' ∈ M' do
            for τ ∈ FindInSegment(μ, t) do
                τ' ← AttachTranslationToString(τ, μ')
                append (σ, σ', τ, τ') to P
            end for
        end for
    end for
end for
return P
```

The example in Figure 1 illustrates how the list of repair operators is built. It is worth noting that only in those cases in which μ, the translation of σ, is found as a contiguous segment of words in the target segment t of the TU being repaired a repair operator can be built; this is indicated by the fifth column in the table. This acts as a quality check to avoid creating repair operators for sub-segments with insufficient context that lead to translations that do not appear in t.

3.2 Generation of fuzzy-match repaired segments

Candidate fuzzy-match repaired segments are built from the list of repair operators P by combining them in all possible ways. This is done through a backtracking depth-first exhaustive search, depicted in Algorithm 2, that incrementally builds fuzzy-match repaired segments $t^{\simeq}$.

The algorithm is initialized with two calls, $\mathrm{Repair}(P, \emptyset, 1, (s, t), t, \mathbf{false}, ())$ and $\mathrm{Repair}(P, \emptyset, 1, (s, t), t, \mathbf{true}, ())$, where $()$ stands for an empty list. At each level of the recursion tree a new repair operator is considered and tested for applicability ($D = \mathbf{true}$) or discarded ($D = \mathbf{false}$). For a repair operator to be applicable it needs to be compatible with the set of repair operators O applied so far to build $t^{\simeq}$; two repair operators are incompatible if they edit the same word in t or if they work on the same source-side mismatch (see Section 3.2.1). If it is compatible with the rest of repair operators in O, the repair operator is applied and added to O; otherwise the branch of the recursion tree is cut. When a leaf of the search 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 $\mathrm{ApplyRepairOp}(o, t^{\simeq})$ replaces in $t^{\simeq}$ the sub-segment τ by τ'; this can be safely done if repair operator P_n is compatible with the other repair operators applied so far.

The algorithm takes advantage of the fact that repair operators that are compatible can be applied in any order

σ	σ'	μ	μ'	μ in t?
Gina found [1-2]	*found* [2]	*Gina encontró*	*encontró*	no
Gina found [1-2]	*Bill found* [1-2]	*Gina encontró*	*Bill encontró*	no
Gina found out [1-3]	*found out* [2-3]	*Gina se enteró*	*se enteró*	yes
Gina found out [1-3]	*Bill found out* [1-3]	*Gina se enteró*	*Bill se enteró*	yes
found [2]	*Bill found* [1-2]	*encontró*	*Bill encontró*	no
found out [2-3]	*Bill found out* [1-3]	*se enteró*	*Bill se enteró*	yes
about the [4-5]	*about the fraud* [4-6]	*sobre el*	*de la estafa*	no
about the news [4-6]	*about the* [4-5]	*de noticias*	*sobre el*	no
about the news [4-6]	*about the fraud* [4-6]	*de las noticias*	*de la estafa*	yes
the [5]	*the fraud* [5-6]	*el*	*la estafa*	no
the news [5-6]	*the* [5]	*las noticias*	*el*	yes
the news [5-6]	*the fraud* [5-6]	*las noticias*	*la estafa*	yes

Figure 1: Example illustrating how the list of repair operators is built. The segment s' = *Bill found out about the fraud* is to be translated into Spanish with the help of the TU (s, t) = (*Gina found out about the news*, *se enteró de las noticias*). Unmatched (unaligned) words in s' are *Bill* and *fraud*; unmatched (unaligned) words in s are *Gina* and *news*. The string-positioned sub-segment pairs (σ, σ') shown are those up to length 3 that contain at least an unmatched word. Their translations (μ, μ') into Spanish are also provided. In this example, we assume that every σ and σ' has a single translation, that is, that M and M' are singletons.

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

Input: List of repair operators P; set of repair operators O applied so far; position in P of the repair 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 repair operator in P will be attempted to apply (true) or not (false); list T containing fuzzy-match repaired segments

```
if D then
    if Compatible(P_n, O, (s, t)) then
        ApplyRepairOp(P_n, t^≃)
        O ← O ∪ {P_n}
    else
        return ▷ Prune this branch of the recursion tree
    end if
end if
if n = length(P) then
    append t^≃ to T
    return ▷ All the operators have been considered
else
    Repair(P, O, n + 1, (s, t), t^≃, true, T)
    Repair(P, O, n + 1, (s, t), t^≃, false, T)
end if
```

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 $n = \text{length}(P)$, in which case 2^n fuzzy-match repaired segments would be produced.[6] However, as many repair operators are incompatible, the actual complexity of the algorithm is well below the worst case (see Section 6.4).

For the example in Figure 1, our method would produce $2^6 = 64$ fuzzy-match repaired segments if all 6 repair operators were compatible. However, most of them are not compatible because they edit the same words in t; as a result, the algorithm ends up producing only 25 repaired segments. Some of these 25 fuzzy-match repaired segments are identical even if they are produced by applying a different set of repair operators. For instance, the fuzzy-match repaired segment *Bill se enteró de la estafa* is produced by applying the repair operator (*Gina found out*, *Bill found out*, *Gina se enteró*, *Bill se enteró*) followed by either the repair operator (*about the news*, *about the fraud*, *de las noticias*, *de la estafa*) or the repair operator (*the news*, *the fraud*, *las noticias*, *la estafa*).

3.2.1 Compatibility of repair operators

Two repair operators are deemed incompatible, and therefore cannot be applied to build the same candidate fuzzy-match repaired segment, if they edit the same word in t or if they work on the same source-side mismatch, that is, if they take care of the same change in s. Note that there may be repair operators that do not edit any word in t but introduce missing ones. In those cases, if they were applied to build the same candidate fuzzy-match repaired segment, they could end up producing candidate fuzzy-match repaired segments $t^{\simeq}$ with repeated words. The following example illustrates this situation. Suppose the segment s' = *the size does not exceed 100 cm* to be translated with the help of the translation unit (s, t) = (*the size does not exceed 100*, *el tamaño no supera los 100*) whose target segment can be repaired with the two repair operators $(\sigma_1, \sigma'_1, \tau_1, \tau'_1)$ = (*exceed 100*, *exceed 100 cm*, *supera los 100*, *supera los 100 cm*) and $(\sigma_2, \sigma'_2, \tau_2, \tau'_2)$ = (*100*, *100 cm*, *los 100*, *los 100 cm*). Both repair operators do not edit (change) any word in t but if they are applied one after the other the result would be the fuzzy-match repaired segment $t^{\simeq}$ = *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 repair operators work on the same source-side mismatch, and to do so, one needs to check the mismatches both in

6. If the algorithm had to explore the application of all the repair operators in P and in all possible orders, its worst-case complexity would be super-exponential.

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 repair 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 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 mismatched words covered by sub-segment x in segment y.

It is worth nothing that this last restriction may mark as incompatible two repair 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 repair 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' = *the size is around 100 cm* to be translated with the help of the translation unit (s, t) = (*the size is about 50 cm*, *el tamaño es de unos 50 cm*) whose target segment can be repaired with the two repair operators $o_1 = (\sigma_1, \sigma'_1, \tau_1, \tau'_1)$ = (*is about*, *is around*, *es de unos*, *está alrededor de*) and $o_2 = (\sigma_2, \sigma'_2, \tau_2, \tau'_2)$ = (*about 50*, *around 100*, *de unos 50*, *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 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 3.2 assumes that repair operators can be applied independently of each other and the order in which they are applied does not affect the final result.

4 Quality estimation of candidate fuzzy-match repaired segments

In the context of MT, sentence-level quality estimation (QE) methods [9], [10] have been developed during the last two decades to avoid bothering professional translators with low-quality translations, to choose among a set of different translations produced by different MT systems for a given source segment, or to estimate the effort to post-edit a given MT output. Quality is usually measured in terms of post-editing time, in terms of the amount of edit operations needed to turn the translation into an adequate translation, or using other related metrics [25], [26].

MT QE techniques can easily be adapted for estimating the quality of the different candidate fuzzy-match repaired segments produced for the source segment to be translated and pick the best one. There are mainly two different approaches to achieve this: the use of a binary classifiers and the use of a regressor. The former can be used to select the best translation on a pairwise comparison basis [27]; the latter can be used to rank the set of candidate fuzzy-match repaired segments. In this paper we follow this last approach; in particular, we use, after preliminary experiments with linear and support-vector regressors [28], extremely randomised trees [11] for regression.

In the following, we describe the features used by the regressor. In particular, we describe two sets of features: one using information readily available to CAT tools (black-box features, Section 4.1), and a second one that exploits information from the inner workings or the repair algorithm used to generate the set of candidate fuzzy-match repaired segments (glass-box features, Section 4.2).

4.1 Black-box (system-independent) features

The following features, some of which are borrowed from the set of 17 baseline features used in the shared tasks on MT QE at the WMT MT contests and implemented in QuEst++ [29],[7] use only information already present in the source segment to be translated s', in the candidate fuzzy-match repaired segment $t^{\simeq}$ or in the translation unit being repaired (s, t):

- BB1 Number of tokens in the source segment s'.
- BB2 Number of tokens in the candidate fuzzy-match repaired segment $t^{\simeq}$.
- BB3 Ratio of the number of tokens in $t^{\simeq}$ to the number of tokens in s'.
- BB4 Number of punctuation marks in s'.
- BB5 Number of punctuation marks in $t^{\simeq}$.
- BB6 Ratio of the number of punctuation marks in $t^{\simeq}$ to the number of punctuation marks in s'.
- BB7 Number of digits in s'.
- BB8 Number of digits in $t^{\simeq}$.
- BB9 Ratio of the number of digits in $t^{\simeq}$ to the number of digits in s'.
- BB10 Source fuzzy-match score: $\text{FMS}(s, s')$.
- BB11 Target fuzzy-match score: $\text{FMS}(t, t^{\simeq})$.
- BB12 Ratio of the source fuzzy-match score to the target fuzzy-match score:
$$\frac{\text{FMS}(s, s')}{\text{FMS}(t, t^{\simeq})}.$$
- BB13 Source sub-segment-level alignment mismatch score: $\text{MMS}_{\text{seg}}(s, s')$. It is computed by using the matching and mismatching sub-segments as the building blocks when computing the edit distance; this implies a monotonic segmentation of (s, s') performed on the basis of the word alignments obtained as a by-product of the edit distance.[8] It measures the ratio of mismatched sub-segments to the total number of sub-segments.
- BB14 Target sub-segment-level fuzzy-match score: $\text{MMS}_{\text{seg}}(t, t^{\simeq})$.
- BB15 Ratio of the source sub-segment-level fuzzy-match score to the target sub-segment-level fuzzy-match score:
$$\frac{\text{MMS}_{\text{seg}}(s, s')}{\text{MMS}_{\text{seg}}(t, t^{\simeq})}.$$

It is worth noting that features BB1, BB4, BB7, BB10 and BB13 do not pay attention to the candidate fuzzy-match

7. https://github.com/ghpaetzold/questplusplus

8. This is similar to the n-gram tuples used in n-gram based statistical MT [30, Sec. 2.1]

repaired segment but to the source segment to be translated (s') and its relation to the source segment (s) in the TU being repaired. Nevertheless, the learning algorithm can rely on them to specialise the trees to be used for regression, that is, to use different sub-trees for regression depending on their value.

4.2 Glass-box (system-dependent) features

The features described next make use of information about the inner workings of the repair algorithm used to generate candidate fuzzy-match repaired segments; more precisely, they capture information about the repair operators used and their form:

GB1 Ratio of word positions in $t^{\simeq}$ covered by at least one repair operator to the number of words in $t^{\simeq}$:

$$\frac{|\{j : \exists \tau' \wedge t^{\simeq}_j \text{ is part of } \tau'\}|}{|t^{\simeq}|},$$

where $t^{\simeq}_j$ is the j-th word of $t^{\simeq}$.

GB2 Sum of the length of the repair operators used to build $t^{\simeq}$ divided by the length of $t^{\simeq}$:

$$\frac{\sum_{i=1}^{N} |\tau'_i|}{|t^{\simeq}|},$$

where N is the number of repair operators used to get $t^{\simeq}$. Notice that a word in $t^{\simeq}$ may be covered by more than one repair operator and is counted as many times as it is covered.

GB3 Ratio of word positions common to t and $t^{\simeq}$ that are covered by at least one repair operator to the number of words positions common to t and $t^{\simeq}$; i.e. average overlap:

$$\frac{|\{j : \exists \tau \wedge t_j \text{ is part of } \tau \wedge j \in \text{match}(t, t^{\simeq})\}|}{|\text{match}(t, t^{\simeq})|},$$

where function $\text{match}(t, t^{\simeq})$ returns a set with the word positions common to t and $t^{\simeq}$.

GB4 Sum of the length of the overlapping sub-segments of the repair operators used to build $t^{\simeq}$ divided by the length of $t^{\simeq}$:

$$\frac{\sum_{i=1}^{N} |\text{match}(t, \tau'_i)|}{|t^{\simeq}|}.$$

The overlapping sub-segments of a repair operator are those containing words common to t and $t^{\simeq}$. As above, a word common to t and $t^{\simeq}$ may be covered by more than one repair operator and is counted as many times as it is part of an overlapping sub-segment.

GB5 Ratio of words in s' covered by at least one σ' used to build a repair operator to the number of words in s':

$$\frac{|\{j : \exists \sigma' \wedge s'_j \text{ is part of } \sigma'\}|}{|s'|},$$

where s'_j is the j-th word of s'.

GB6 Sum of the length of the σ's used to build a repair operator divided by the length of s':

$$\frac{\sum_{i=1}^{N} |\sigma'_i|}{|s'|}.$$

Similarly to GB2, a word in s' may be covered by more than one σ' and is counted as many times as it is covered by a σ'.

GB7 Ratio of matched source word positions that are covered by at least one σ used to build a repair operator to the number of matched source words positions; i.e. average overlap in the source language:

$$\frac{|\{j : \exists \sigma \wedge s_j \text{ is part of } \sigma \wedge j \in \text{match}(s, s')\}|}{|\text{match}(s, s')|}.$$

GB8 Sum of the length of the overlapping sub-segments of the (σ, σ') used to build the repair operators used to get $t^{\simeq}$ divided by the length of s':

$$\frac{\sum_{i=1}^{N} |\text{match}(s, \sigma'_i)|}{|s'|}.$$

The overlapping sub-segments here are those containing words common to s and s'. As above, a word that is part of the matching between s and s' may be covered by more than one (σ, σ') and is counted as many times as it is part of an overlapping sub-segment.

GB9 Mean target context per repair operator:

$$\frac{\sum_{i=1}^{N} |\text{LCS}(\tau_i, \tau'_i)|}{\sum_{i=1}^{N} \min(|\tau_i|, |\tau'_i|)},$$

where $\text{LCS}(x, y)$ is the longest common subsequence to x and y.

GB10 Mean source context per repair operator:

$$\frac{\sum_{i=1}^{N} |\text{LCS}(\sigma_i, \sigma'_i)|}{\sum_{i=1}^{N} \min(|\sigma_i|, |\sigma'_i|)}.$$

GB11 Measure of how evenly distributed are the mismatched target-language sub-segments in the repair operators used to build the fuzzy-match repaired segment $t^{\simeq}$:

$$\frac{\sum_{k=1}^{N} |\tau_k| \prod_{j=1}^{n_k} \frac{|m_j|-1}{|\tau_k|-2n_k+1}}{\sum_{k=1}^{N} |\tau_k|},$$

where n_k is the number of matching sub-segments in τ_k and m_i is the i-th matching substring. When computing n_k it must be taken into account that every matched sub-segment must consist of contiguous words both in τ and τ'. The closer the value of the feature is to 1, the more evenly distributed the mismatched sub-segments are.

GB12 Measure of how evenly distributed are the mismatched source-language sub-segments in the repair operators used to build the fuzzy-match repaired segment $t^{\simeq}$:

$$\frac{\sum_{k=1}^{N} |\sigma_k| \prod_{j=1}^{n_k} \frac{|m_j|-1}{|\sigma_k|-2n_k+1}}{\sum_{k=1}^{N} |\sigma_k|},$$

where n_k is the number of matching sub-segments in σ_k and m_i is the i-th matching substring. As above, when computing n_k it must

be taken into account that every matched sub-segment must consists of contiguous words both in σ and σ', as in GB11. The closer the value of the features is to 1, the more evenly distributed the mismatched sub-segments are.

GB13 Number of repair operators used to get $t^{\simeq}$ divided by the number of mismatched words in s.

GB14 Number of repair operators used to get $t^{\simeq}$ divided by the number of mismatched sub-segments in s (sequences of contiguous word positions).

GB15 Number of repair operators used to get $t^{\simeq}$.

GB16 Ratio of *grounded* repair operators to the number of repair operators used to get $t^{\simeq}$. A repair operator is considered to be grounded if at least one word at each end of σ is matched.

GB17 Binary feature to flag whether all the repair operators used to get $t^{\simeq}$ are grounded or not.

5 Experimental settings

We evaluate our FMR approach on three different languages pairs, namely English–Spanish (`en-es`), Spanish–Portuguese (`es-pt`) and Spanish–French (`es-fr`), using texts from the DGT translation memory [31] and an MT system as SBI. We use these language pairs to study how our FMR approach behaves when translating between closely-related languages (`es-pt` and `es-fr`) and when the languages involved in the translation are not so closely related (`en-es`). Of the two closely-related languages we use, 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. English is non-pro-drop.

This section describes the resources used, the way the training samples used to train the QE component of the method were generated, the regressor used and how it is trained, and the metrics used to evaluate the performance of our FMR approach.

5.1 Resources

We use the DGT-TM 2015 parallel corpus[9] for the three language pairs (`en-es`, `es-pt` and `es-fr`). For each fuzzy-match-score threshold (FMT) ϕ used in our experiments, we split the DGT TM 2015 corpus into three sets: one for training the regressor used for QE (training), another to select the configuration of the regressor to be finally used (see next section; development) and a third one for performance evaluation (testing). This splitting can be formally described as follows. Let P be the parallel corpus, J_ϕ a randomly drawn subset of P, a *translation job*, for the FMT ϕ, and $M_\phi = P - J_\phi$ the translation memory to be used. J_ϕ is built by randomly selecting parallel segments (s', t') from P so that the final set meets the following condition:

$$\forall (s', t') \in J_\phi, \exists (s, t) \in M_\phi : \mathrm{FMS}(s, s') \geq \phi.$$

9. https://ec.europa.eu/jrc/en/language-technologies/dgt-translation-memory

FMT (ϕ)	Language pair	Training		Development		Test	
		$N_{s'}$	$N_{t^{\simeq}}$	$N_{s'}$	$N_{t^{\simeq}}$	$N_{s'}$	$N_{t^{\simeq}}$
60%	`en-es`	3,249	17.6	238	14.8	460	32.2
	`es-pt`	3,194	14.9	220	15.5	1,084	11.3
	`es-fr`	2,930	14.0	120	14.6	1,070	16.4
70%	`en-es`	2,289	9.8	157	16.6	355	25.5
	`es-pt`	1,981	10.8	134	18.5	678	11.7
	`es-fr`	2,376	12.4	94	17.1	716	17.2
80%	`en-es`	1,250	6.1	50	8.3	194	5.5
	`es-pt`	1,267	7.6	83	9.0	554	9.5
	`es-fr`	1,812	5.7	122	10.1	588	13.2
90%	`en-es`	239	6.9	18	8.3	56	3.3
	`es-pt`	483	7.2	33	13.1	229	6.8
	`es-fr`	762	5.4	50	4.2	236	6.6

Table 1: For each fuzzy-match score threshold (FMT; ϕ) and language pair, number of segments to be translated ($N_{s'}$) and average number of candidate fuzzy-match repaired segments per segment to be translated ($N_{t^{\simeq}}$) in the training, development and test sets.

After building J_ϕ, we remove those segments for which, with the MT system used as SBI, it is not possible to build at least one repair operator. The resulting set is then divided into three disjoint subsets, $J_{\phi,\mathrm{train}}$, $J_{\phi,\mathrm{dev}}$ and $J_{\phi,\mathrm{test}}$, such that $J_\phi = J_{\phi,\mathrm{train}} \cup J_{\phi,\mathrm{dev}} \cup J_{\phi,\mathrm{test}}$. Table 1 shows, for each FMT used in the experiments and for each language pair, the amount of segments to be translated and the average number of candidate repaired segments $t^{\simeq}$ per segment to be translated s'. These sets were obtained as explained above from a parallel corpus P with 196,294 segments for `en-es`, 150,567 for `es-pt` and 149,479 for `es-fr`. For convenience, the TM to be used is the same for all FMTs, $M_{60\%}$, that is the one obtained for $\phi = 60\%$.

The figures in Table 1 show that, as expected, the number of segments to be translated decreases as the FMT increases: the chances of finding a match with a 60% FMT is higher than that of a 90% FMT. It is also worth noting that the average number of candidate fuzzy-match repaired segments per segment to be translated also goes down as the FMT grows because the percentage of words to be repaired, and therefore the number of repair operators, is reduced.

For each $(s', t') \in J_{\phi,\mathrm{train}}$ a set of training samples is generated as follows. First the repair algorithm described in Section 3 is run to generate the set of candidate fuzzy-match repaired segments $\{t^{\simeq}\}$ for the translation of s' using the TU with the highest FMS, that is, the TU $(s, t) \in M_{60\%} : (\nexists (s'', t'') \in M_{60\%} : \mathrm{FMS}(s', s) < \mathrm{FMS}(s', s''))$.[10] Then, from each candidate fuzzy-match repaired segment $t^{\simeq}$ a training sample is generated by computing the features described in Section 4 and the error rate to be predicted. This error rate is defined as:

$$\xi(t^{\simeq}, t') = \frac{\mathrm{ED}(t^{\simeq}, t')}{\max(|t^{\simeq}|, |t'|)} \quad (1)$$

where $\mathrm{ED}(x, y)$ returns the word-based edit distance [21] between the segments x and y. This way of computing the error rate resembles the way in which the fuzzy-match score is computed.[11] Table 2 shows, for the different fuzzy-match

10. If there is more than one TU meeting this condition, one is selected at random.

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

Language pair	FMT (ϕ)			
	60%	70%	80%	90%
en-es	57,816	22,333	7,653	1,661
es-pt	47,675	21,305	9,589	3,492
es-fr	41,124	29,432	10,351	4,140

Table 2: For the training set, number of samples for the different fuzzy-match-score thresholds (FMT) used in the experiments.

score thresholds used, the amount of training samples used to train the regressor used for QE.

The corpora used to build the training, development and test sets may contain parallel segments that are *free* translations of one another, and this may be introducing noise affecting the regressor's performance. This problem was already detected in the DGT-TM 2015 parallel corpus and a simple filtering method to discard this noise was proposed [32]. This method removes the candidate repaired segments obtained from a segment to be translated s', reference translation t' and TU to be repaired (s, t) for which $|\mathrm{FMS}(s, s') - \mathrm{FMS}(t, t')| > 0.05$. This filtering is based on the assumption that the number of words that differ in both pairs of segments should be similar for both languages. In Section 6, we report results when using both filtered and non-filtered corpora to study the effect of this noise on the regressor's performance. By applying this filtering the amount of segments that are discarded is around 22%.

Regarding the MT system to be used for FMR, we have used the Apertium MT platform [33].[12] More precisely we have used the following language pair packages: `apertium-en-es`, `apertium-es-pt` and `apertium-fr-es`.[13]

5.2 Regressor

In Section 6, we report the results obtained when using *extremely randomised trees* (ERT) [11] because it is the regressor that performed best when evaluated on the development corpus on a set of preliminary experiments in which we also tried with linear regression and support vector regression (SVR) [28]. In the case of SVR, we performed a 3-fold cross-validation grid search to find the optimum hyper-parameter values of the Gaussian radial-basis function kernel we used and tried with different feature selection methods, such as recursive feature elimination [34], chi-square, and the Gini importance computed over extremely randomised trees [35].[14]

Extremely randomised trees (ERT) is a tree-based ensemble method for classification and regression [11]. At each internal tree node the best feature is determined from a subset of features selected at random from the whole set of features whereas the cut-off point is selected fully at random. In our implementation the best feature is selected according to the *Gini importance*, also know as *mean decrease in impurity* [36, ch. 4] (see Section 6.3 for more information). When ERTs are used for regression, the prediction is computed as the average of the output of all the trees in the ensemble.

The main hyper-parameters controlling the learning process of ERT are the size of the subset of features randomly selected (N) and the amount of trees in the ensemble (M). If $N = 1$, the feature to use in each internal node is selected fully at random and the method builds completely randomised trees. If N equals the total amount of features (F), randomisation only happens in the selection of the cut-off point. In our experiments we tried with $N = \sqrt{F}$, $N = \log_2(F)$ and $N = F$; the best results on the development set were obtain with $N = F$, as reported in [37]. As regards the amount of trees in the ensemble we tried with 10, 100, 200, 500, 800 and $1,000$ and, overall, the best results on the development set were obtained for $M = 100$. The difference between using 100 or more trees is negligible. With respect to the rest of parameters, we used the default ones in the ERT implementation of scikit-learn.[15]

Finally, due to the randomisation of trees, different training executions give as a results regressors with small differences in the output they provide. The results we report in the following section are those obtained with the ERT performing best on the development set out of ten different training executions.

5.3 Evaluation

We measure the performance of our FMR guided by QE approach with the following metrics:

- Error rate over the whole test set, which accounts for the amount of edit operations needed to carry out a *translation job*:

$$\frac{\sum_{i=1}^{N} \mathrm{ED}(t_i^{\square}, t'_i)}{\sum_{i=1}^{N} \max(|t_i^{\square}|, |t'_i|)},$$

  where t'_i is the gold standard translation in the test set for the source segment s'_i and $t_i^{\square}$ is the translation being evaluated; $t_i^{\square}$ may be the (unrepaired) target segment t_i, a candidate fuzzy-match repaired segment $t_i^{\simeq}$ produced by our repair algorithm or the translation of s_i produced by the MT system.
- *Mean absolute error* (MAE) of the error rate predicted by the ERT regressor on each candidate fuzzy-match repaired segment in the test set. MAE evaluates the ability of the regressor to predict error rates at the segment level. It is computed as

$$\frac{\sum_{i=1}^{N} \sum_{j=1}^{M_i} |\xi(t_{ij}^{\simeq}, t'_i) - \hat{\xi}(t_{ij}^{\simeq}, t'_i)|}{\sum_{i=1}^{N} M_i},$$

  where $\xi(\cdot)$ and $\hat{\xi}(\cdot)$ are the predicted error rate (see Equation (1)) and the error rate to be predicted, respectively, and M_i is the number of candidate fuzzy-match repaired segments produced for source segment s_i.
- *Success rate* (SR) of the ERT regressor used to select the best candidate fuzzy-match repaired segment. It is computed by comparing the amount of edit operations that are saved when editing the candidate fuzzy-match repaired segment $t_i^{\simeq}$ with the lowest

12. https://www.apertium.org

13. SVN revisions 64348, 62539 and 62696,respectively.

14. SVR implementation used: version 0.19 of scikit-learn, http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVR.html.

15. Version 0.19, http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.ExtraTreesRegressor.html.

predicted error rate, instead of the (unrepaired) target segment t_i of the TU being repaired, to the amount of edit operations saved when editing the best possible (oracle) fuzzy-match repaired segment $t_i^\star$:

$$\frac{\sum_{i=1}^{N} \frac{|\mathrm{ED}(t_i,t'_i)-\mathrm{ED}(t_i^{\simeq},t'_i)|}{|\mathrm{ED}(t_i,t'_i)-\mathrm{ED}(t_i^{\star},t'_i)|}}{N}$$

where t'_i is the gold standard translation in the test set for the source segment s'_i. This metric shows how good is $t_i^{\simeq}$ as compared to the best possible repaired segment (oracle) produced by the repair algorithm described in Section 3: the numerator is the actual change in the edit distance when replacing t_i with the repaired version $t_i^{\simeq}$ and the denominator is the change in edit distance that would be produced if t_i was replaced with the *oracle* (best possible) repair t_i^*.

6 Results and Discussion

We first evaluate the ability of our FMR approach to produce, for each source segment in the test sets, a candidate fuzzy-match repaired segment as close as possible to the desired translation (Section 6.1). We then evaluate how well our QE approach performs when selecting the best candidate fuzzy-match repaired segment among the whole set of candidates produced (Section 6.2). Finally we discuss on the informative of the features used for QE (Section 6.3) and the actual complexity of the Repair algorithm (Section 6.4).

6.1 Oracle evaluation

Table 3 shows the error rate, computed as described above, when the target segment to be evaluated is the target segment t in the TU being repaired (TM), when it is produced by translating the whole source segment s' using the MT system being used for FMR, and when it is the best possible (*Oracle*) candidate fuzzy-march repaired segment produced by our algorithm. We provide error rates computed on non-filtered corpora as well as on the corpora filtered following the method [32] described in Section 5.1.[16]

As can be seen for the three language pairs and for all FMT, our repair algorithm is capable of producing a fuzzy-match repaired segment with an error rate below that of the target segment in the TU being repaired and that of the translation produced by the MT system used for repairing. It is worthwhile to recall that the test sets used are made up of segments for which there is at least a match above the given FMT; this explains why the error rate of the TM proposals gets lower as the FMT grows. As regards the corpora, the results show lower error rates for all three approaches across the table when evaluated on the filtered corpora, which is consistent with the way in which the corpora are filtered.

The difference between the best candidate fuzzy-match repaired produced (Oracle) by our algorithm and the MT-produced translations reveals that our FMR approach is quite robust to MT errors. This is because for a repair operator to be successfully built, τ, the translation of the sub-segment σ of s, must appear as a contiguous text sub-segment in t, the translation proposal being repaired. This acts as a quality filter that makes our FMR approach not to use for repairing sub-segments for which the SBI does not match the TM. Obviously, this quality check cannot be performed on τ', the translation of the sub-segment σ' in s' aligned with σ. However, it seems that having a quality check on τ helps to ensure that most of the repair operators built are of good quality.

FMT	Non-filtered corpora			Filtered corpora		
	TM	MT	Oracle	TM	MT	Oracle
English–Spanish						
60%	27.16	57.61	23.24	22.13	56.73	17.91
70%	22.83	56.21	19.32	18.53	56.43	14.73
80%	17.31	61.60	14.31	13.27	56.00	9.71
90%	11.63	66.84	5.85	11.21	58.91	4.78
Spanish–Portuguese						
60%	22.08	40.56	15.86	17.29	36.86	10.07
70%	18.20	40.00	13.41	13.63	36.30	8.03
80%	15.29	39.67	11.13	10.62	35.55	5.52
90%	10.42	44.87	7.69	7.24	40.23	3.97
Spanish–French						
60%	19.78	49.66	15.34	15.27	48.59	10.81
70%	15.97	49.75	12.52	12.17	48.54	8.71
80%	12.48	49.56	9.38	9.84	48.51	6.56
90%	7.71	51.42	5.36	6.34	50.72	3.75

Table 3: For the non-filtered and filtered corpora, error rate (%) for the target segment t in the TU (s,t) being repaired (TM), for the translation produced by the MT system for the whole source segment s' (MT) and for the best possible fuzzy-match repaired segment $t^\star$ (Oracle).

6.2 Quality estimation evaluation

In order to determine the best configuration to train the regressor to be used to estimate the quality of fuzzy-match repaired segments, we have tried with the following setups: (1) training different ERT regressors for different FMTs and language pairs, (2) training one regressor per language pair regardless of the FMT, and (3) training a single regressor for all language pairs and FMTs. As above, we provide results when the ERT is trained and evaluated both on filtered and non-filtered corpora.

Table 4 shows the error rates obtained for the target segment in the TU being repaired (TM) —the one with the highest fuzzy-match above the FMT—, for the fuzzy-match repaired segment with the lowest predicted error rate (ERT) and for the best possible fuzzy-match repaired segment produced by the repair algorithm (Oracle). The table also provides the success rate (SR) obtained by comparing the error rate of the oracle and the error rate of the fuzzy-match repaired segment with the lowest predicted error rate, and the mean absolute error (MAE) of the ERT regressor. For each FMT and language pair a different ERT regressor was trained.

The results show that the use of the ERT regressor to select, for a given source segment and TU, the candidate fuzzy-match repaired segment with the lowest predicted error rate performs better on the filtered corpora, where

16. It is worthwhile noting that our FMR approach is independent of the MT system used as SBI. English–Spanish experiments using a Transformer neural MT (NMT) system trained on the JRC-Acquis corpus provided analogous results to those in Table 3; that is, using the NMT system both to repair and as a baseline, the Oracle is better than both the unrepaired target segment t and the raw NMT output.

FMT	Non-filtered corpora					Filtered corpora				
	TM	ERT	Oracle	SR	MAE	TM	ERT	Oracle	SR	MAE
English–Spanish										
60%	27.16	25.71	23.24	0.37	0.06	22.13	19.38	17.91	0.65	0.04
70%	22.83	21.58	19.32	0.36	0.06	18.53	15.71	14.73	0.74	0.04
80%	17.31	16.14	14.31	0.39	0.07	13.27	10.91	9.71	0.66	0.05
90%	11.63	7.55	5.85	0.71	0.07	11.21	5.77	4.78	0.85	0.02
Spanish–Portuguese										
60%	22.08	17.74	15.86	0.70	0.11	17.29	11.84	10.07	0.76	0.06
70%	18.20	15.35	13.41	0.59	0.09	13.63	9.77	8.03	0.69	0.04
80%	15.29	13.54	11.13	0.42	0.08	10.62	6.79	5.52	0.75	0.03
90%	10.42	8.73	7.69	0.62	0.08	7.24	4.56	3.97	0.82	0.03
Spanish–French										
60%	19.78	17.31	15.34	0.56	0.08	15.27	12.45	10.81	0.63	0.05
70%	15.97	14.44	12.52	0.44	0.07	12.17	10.37	8.71	0.52	0.05
80%	12.48	10.99	9.38	0.48	0.05	9.84	7.44	6.56	0.73	0.03
90%	7.71	6.50	5.36	0.52	0.06	6.34	4.57	3.75	0.68	0.03

Table 4: For both the non-filtered and filtered corpora, error rate (%) for the target segment in the TU being repaired (TM), for the fuzzy-match repaired segment with the lowest predicted error rate (ERT) and for the best possible one (Oracle). Success rates (SR) and mean absolute errors (MAE) are also provided. A different ERT regressor was trained for each FMT and language pair.

noisy translation units have been removed from the training, development and test sets, than on the non-filtered corpora. The difference in performance is noteworthy for FMT below 90%, especially in the case of English–Spanish. Additionally, the success rate, that is the proportion of edit operations saved when editing the selected candidate fuzzy-match repaired segment over the number of edit operations saved when editing the best possible fuzzy-match repaired segment, increases as the FMT grows; for 60% FMT the success rates on the filtered corpora are around 0.68, whereas for 90% FMT they scale up to 0.78, surpassing 0.80 for English–Spanish and Spanish–Portuguese. Our QE method is clearly better at ranking fuzzy-match repaired segments when the amount of mismatched sub-segments is small; which is the typical scenario where TM-based CAT is used.[17]

If we pay attention to the performance of the ERT regressor when evaluated as such, we can see that the MAEs reported are below 0.10 in all cases but one (`es-pt`, 60% FMT, non-filtered corpora), and that, for the filtered corpora they are around 0.05. This accounts for the high informativeness of the features defined in Section 4. We do not report the root mean square error because it shows a similar trend. It is worth noting that MAE is computed at the sample-level and, as a result, all candidate fuzzy-match repaired segments are equally judged, regardless of length.

The results in Table 4 were obtained when using a different ERT regressor per FMT and language pair. In order to study how dependent is the ERT regressor on the FMT used for training, we repeated the experiments reported in Table 4 but using a single regressor per language pair. In particular, we used the regressor trained on samples obtained from TUs for which the fuzzy-match is above the 60% FMT; the results are reported in Table 5.

From the comparison of the results in tables 4 and 5 we can conclude that using a single ERT regressor per language pair is the best option. The results obtained with a single regressor are better than those obtained with a regressor per FMT, especially for the 80% and 90% FMTs. This is probably due to the fact that the amount of training samples for 80% FMT and 90% FMT is an order of magnitude lower as compared to 60% FMT (see Table 2), and as a result the ERT regressor was not learning adequately.

None of the features used and listed in Section 4 is language-dependent, although the distribution of their values and how informative they are may differ from one language to another. To ascertain whether or not the ERTs informed by these features are also language-independent, we repeated the experiments reported in Table 5 but training an ERT regressor on the samples obtained from TUs for which the fuzzy-match is above the 60% FMT for *all language pairs together*. Table 6 reports the results obtained; it is worth noting that the same ERT regressor is used regardless of the FMT and language pair, and that the amount of training samples is 146,615.

The results in Table 6 are quite similar to those in Table 5, where a different ERT is used per language pair. For some language pairs and FMTs results improve slightly, while for others they worsen slightly. This allow us to conclude that, given the small difference in the success rates reported in both tables it is advisable to use a single regressor trained on 60% FMT for all languages together, at least for languages which are related in some sense (such as the four Western Indo-European languages in these experiments).

Next, we study how fast the learning process converges to see if the amount of training samples used for training is enough to learn the best possible ERT models. Figure 2 plots the learning curve computed when training ERT models for regression for the English–Spanish language pair and for all languages together using a 60% FMT, both on filtered and non-filtered corpora. The learning curves show cross-validation (dashed line) and training (solid lines) scores of the ERT models as a function of the number of training samples. For cross validation we performed 10 iterations of *shuffle* split: in each iteration 20% of the training set was randomly selected as a validation set. These scores correspond to the coefficient of determination [38, ch. 4]: the best possible score is 1.0 and it can be negative if the

17. Fuzzy-matches are seldom used for FMS < 70%.

FMT	Non-filtered corpora					Filtered corpora				
	TM	ERT	Oracle	SR	MAE	TM	ERT	Oracle	SR	MAE
English–Spanish										
60%	27.16	25.71	23.24	0.37	0.06	22.13	19.38	17.91	0.65	0.04
70%	22.83	21.30	19.32	0.44	0.06	18.53	15.68	14.73	0.75	0.05
80%	17.31	16.07	14.31	0.41	0.07	13.27	10.54	9.71	0.77	0.04
90%	11.63	6.29	5.85	0.92	0.07	11.21	5.00	4.78	0.96	0.02
Spanish–Portuguese										
60%	22.08	17.74	15.86	0.70	0.11	17.29	11.84	10.07	0.76	0.06
70%	18.20	15.14	13.41	0.64	0.09	13.63	9.54	8.03	0.73	0.04
80%	15.29	12.75	11.13	0.61	0.09	10.62	6.61	5.52	0.79	0.04
90%	10.42	8.66	7.69	0.64	0.09	7.24	4.49	3.97	0.84	0.03
Spanish–French										
60%	19.78	17.31	15.34	0.56	0.08	15.27	12.45	10.81	0.63	0.05
70%	15.97	14.19	12.52	0.52	0.07	12.17	9.90	8.71	0.66	0.05
80%	12.48	10.82	9.38	0.53	0.06	9.84	7.71	6.56	0.65	0.03
90%	7.71	6.43	5.36	0.55	0.06	6.34	4.46	3.75	0.73	0.02

Table 5: Error rate for the target segment in the TU being repaired (MT), for the fuzzy-match repaired segment with the lowest predicted error rate (ERT) and for the best possible one (Oracle). Success rates (SR) and mean absolute errors (MAE) are also reported. A single ERT regressor was trained on 60% FMT for each language pair.

FMT	Non-filtered corpora					Filtered corpora				
	TM	ERT	Oracle	SR	MAE	TM	ERT	Oracle	SR	MAE
English–Spanish										
60%	27.16	25.64	23.24	0.39	0.06	22.13	19.72	17.91	0.57	0.04
70%	22.83	21.20	19.32	0.47	0.06	18.53	15.77	14.73	0.73	0.04
80%	17.31	15.97	14.31	0.45	0.08	13.27	10.40	9.71	0.81	0.05
90%	11.63	6.36	5.85	0.91	0.10	11.21	5.00	4.78	0.96	0.02
Spanish–Portuguese										
60%	22.08	17.69	15.86	0.71	0.11	17.29	11.42	10.07	0.81	0.05
70%	18.20	15.23	13.41	0.62	0.09	13.63	9.11	8.03	0.81	0.04
80%	15.29	12.70	11.13	0.62	0.07	10.62	6.34	5.52	0.84	0.04
90%	10.42	8.83	7.69	0.58	0.07	7.24	4.55	3.97	0.82	0.04
Spanish–French										
60%	19.78	17.18	15.34	0.59	0.08	15.27	12.02	10.81	0.73	0.05
70%	15.97	14.34	12.52	0.47	0.07	12.17	9.84	8.71	0.67	0.04
80%	12.48	11.17	9.38	0.42	0.05	9.84	7.60	6.56	0.68	0.03
90%	7.71	6.27	5.36	0.61	0.06	6.34	4.47	3.75	0.72	0.02

Table 6: Error rate for the target segment in the TU being repaired (MT), for the fuzzy-match repaired segment with the lowest predicted error rate (ERT) and for the best possible one (Oracle). Success rates (SR) and mean absolute errors (MAE) are also shown. The same ERT regressor trained on 60% FMT is used for all language pairs.

model gets arbitrarily worse. Our ERT models for 60% FMT converge to values of the coefficient of determination above 85%, some of them even above 95%, which indicates that our ERT models are learning and that the amount of samples used for training is adequate for the task.

6.3 Discussion on the informativeness of features

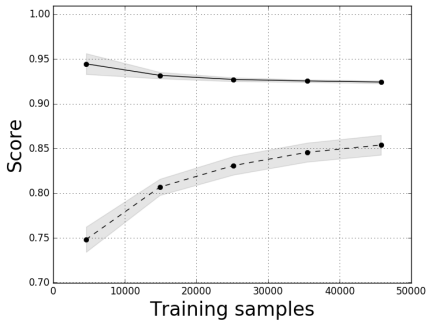
To better understand how our ERT models work, we present an analysis of the informativeness of the features used to create them. As mentioned above, the ERT implementation we have used uses the *Gini importance*, also know as *mean decrease in impurity* [36, ch. 4], to decide the splitting of the nodes while building the trees.[18] The mean decrease in impurity is directly related to the *purity* of a tree node. A node in a tree is considered to be pure if its probability for the training samples reaching it is 1; that is, if all samples reaching the node are in the same class (in the case of classification) or are close to a single target value (in the case of regression).

Our analysis is based on the Gini importance computed as the total decrease in node impurity, weighted by the probability of reaching that node (which is approximated by the proportion of samples reaching that node) and averaged over all trees in the ensemble.[19] We used Gini importance instead of other methods, such as *permutation importance*, which has been shown to have "better statistical properties" [39], because it is less computationally expensive [40] and is the measure used by the scikit-learn implementation of ERT we have used.

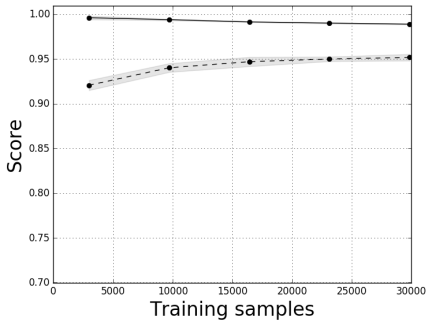
Since the learning curve of the ERT models that use all language pairs (see Figure 2) is similar and in some cases better than the ones computed for individual language pairs, we analyse the informativeness of the features obtained from the ERT models for all language pairs trained on filtered and non-filtered corpora with a 60% FMT. Figure 3

18. This way of deciding on the splitting of nodes differs from that of the original paper describing ERT [11].

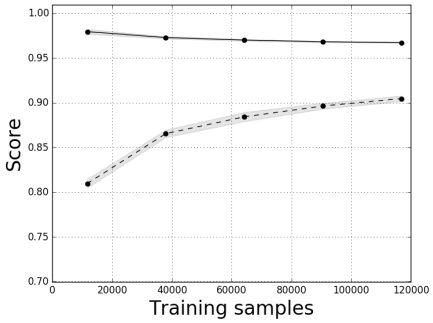
19. For more information about the computation of the Gini importance, see [35].



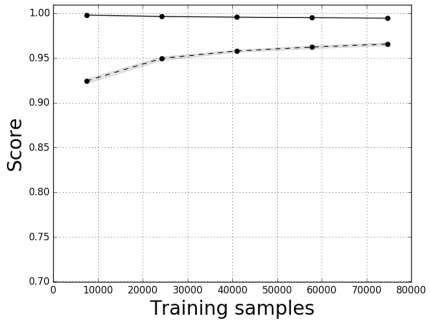
(a) English–Spanish, non-filtered corpora



(b) English–Spanish, filtered corpora



(c) All language pairs, non-filtered corpora



(d) All language pairs, filtered corpora

Figure 2: Learning curve for training ERT models for English–Spanish (((a), (b)) and for all language pairs together ((c), (d)) when using non-filtered ((a), (c)) and filtered ((b), (d)) corpora. The solid line is the learning curve computed over the training corpus, whereas the dashed line corresponds to the cross-validation learning curve.

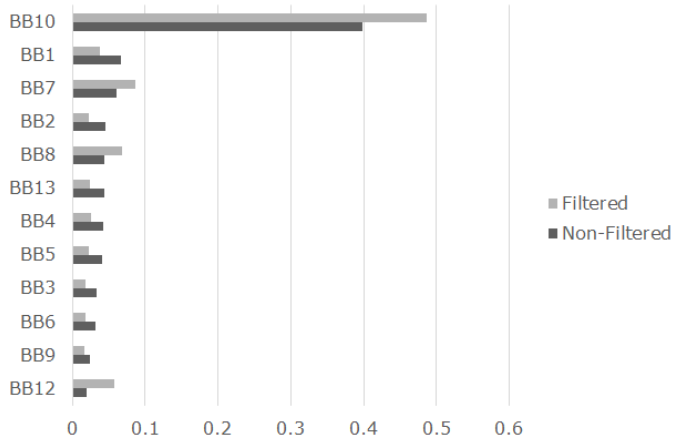


Figure 3: Gini importance for the top 12 features computed over ERT regressors trained for all language pairs with a 60% FMT and on filtered and non-filtered corpora (ordered according to the Gini importance on non-filtered corpora).

shows the Gini importance of the top 12 features for the ERT models aforementioned. The rest of features, although less relevant, still have a Gini importance above 0.0, which means that they help the ERT regressor; in fact, removing any of them degrades the regressor performance.

As Figure 3 shows, the top 12 features are all black-box features that do not take advantage of the information about the inner workings of the repair algorithm. Of these top 12 features, BB10 scores unusually high while BB7 and BB1 are the best performing features which are closer to the median of the Gini importance. The value of BB10, that is, the fuzzy match between s and s', may be working as a means for specialising the trees to be used for regression so that the ones used for small values of BB10 are different from those used for larger ones. In this regard, it is worth noting that the value of BB10 is the same for all training samples obtained from the segment to be translated s' and the TU (s, t) to be repaired. Other high-scoring features like BB1 and BB7 are in a similar situation because they count tokens and digits from the source segment s' to be translated, and provide a nice check of the validity of the repair operators used (the number of digits should be invariant).

6.4 Actual complexity of the Repair algorithm

The worst-case complexity of Algorithm 2 is $O(2^n)$, where n is the number of repair operators available for a TU (s, t) and segment to be translated s'. Usually, not all repair operators are compatible, and cannot therefore be applied together to produced a fuzzy-match repaired segment (see Section 3.2.1); as a result, the actual complexity of the algorithm is drastically reduced. Figure 4 shows, for `en-es`, the ratio of fuzzy-match repair segments produced to those that would be produced in the worst case (2^n) as a function of n. As can be seen, this ratio decreases as n increases. This accounts for the practicality of the approach.

7 Concluding Remarks

In this paper we have described a fuzzy-match repair (FMR) approach that, for a source segment to be translated s'

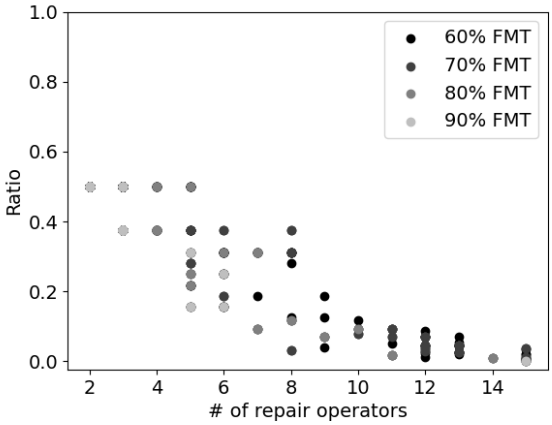


Figure 4: For en-es, ratio of fuzzy-match repair segments produced to those that would be produced in the worst case.

and translation unit (s, t) to be repaired, first generates, using any source of bilingual information, a set of candidate fuzzy-match repaired segments in the target language, and then estimates the quality of the repaired segments produced to select the one to be finally used for translation. For selecting the best fuzzy-match repaired segment, we propose a set of features and train a tree-based regressor to predict the amount of edit operations needed to convert each candidate fuzzy-match repaired segment into an adequate translation of s'. The candidate fuzzy-match repaired segment with the lowest predicted amount of edit operations needed is the one selected as best.

We have extensively evaluated the performance of this approach on three different language pairs, namely English–Spanish, Spanish–Portuguese and Spanish–French, with different fuzzy-match score thresholds (FMT), and using raw (non-filtered) corpora and (filtered) corpora from which *noisy* translation units have been removed. The best results are obtained on the filtered corpora and with regressors trained on training samples obtained using a 60% FMT. We have also evaluated the performance when the regressor is trained on a mix of training samples from all language pairs and then tested on each different language pair. This last evaluation showed that not only the features we propose are language-independent, but also the regressor based on those features is.

The performance of the quality estimation approach used to select the best candidate fuzzy-match repaired segment depends on the similarity between the segment to be translated s' and the source segment s in the TU (s, t) being repaired. The more similar they are (the greater the FMT), the more successful it is. For a 90% FMT, the QE method selects the best possible candidate fuzzy-match repair segment generated by the repair algorithm most of the times, resulting, on average, in a success rate on filtered corpora above 0.83. This means that the use of the selected candidate fuzzy-match repaired segment allows to save, on average, 83% of the edit operations that would have been saved if the best possible (oracle) candidate segment would have been chosen. If the candidate repaired segments were selected at random the saving of edit operations would be, on average, 44%.

As regards the features used by the regressors, we have proposed a set of features made up of black-box (system-independent) and glass-box (system-dependent) features; black-box features are easier to compute as they do not exploit any information about the repair operators used to generate the candidate fuzzy-match repaired segments. Black-box features are found to be more informative that glass-box ones, although all of them are useful to some extent. This result suggests that the QE method used to select the best fuzzy-match repaired segment could be used to rank candidate fuzzy-match repaired segments produced by other FMR approaches.

Acknowledgments

This work was supported by the Spanish Government through the EFFORTUNE project [TIN-2015-69632-R].

References

[1] L. Bowker, *Computer-aided translation technology: a practical introduction*. University of Ottawa Press, 2002.

[2] A. G. Schjoldager and T. P. Christensen, "Translation-memory (TM) research: what do we know and how do we know it?" *Hermes: Journal of Language and Communication in Business*, vol. 44, pp. 89–101, 2010.

[3] L. Kranias and A. Samiotou, "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*, Lisbon, Portugal, 2004, pp. 331–334.

[4] S. Hewavitharana, S. Vogel, and A. Waibel, "Augmenting a statistical translation system with a translation memory," in *Proceedings of the 10th Annual Conference of the European Association for Machine Translation*, Budapest, Hungary, 2005, pp. 126–132.

[5] S. Dandapat, S. Morrissey, A. Way, and M. L. Forcada, "Using example-based MT to support statistical MT when translating homogeneous data in a resource-poor setting," in *Proceedings of the 15th Annual Conference of the European Association for Machine Translation*, Leuven, Belgium, 2011, pp. 201–208.

[6] J. E. Ortega, F. Sánchez-Martínez, and M. L. Forcada, "Fuzzy-match repair using black-box machine translation systems: what can be expected?" in *Proceedings of the 12th Biennial Conference of the Association for Machine Translation in the Americas (AMTA 2016, vol. 1: MT Researchers' Track)*, Austin, USA, October 2016, pp. 27–39.

[7] B. Bulté, T. Vanallemeersch, and V. Vandeghinste, "M3TRA: integrating TM and MT for professional translators," in *Proceedings of the 21st Annual Conference of the European Association for Machine Translation*, Alacant, Spain, May 2018, pp. 69–78.

[8] M. Esplà-Gomis, F. Sánchez-Martínez, and M. L. Forcada, "Using machine translation in computer-aided translation to suggest the target-side words to change," in *Proceedings of the 13th Machine Translation Summit*, Xiamen, China, September 2011, pp. 172–179.

[9] J. Blatz, E. Fitzgerald, G. Foster, S. Gandrabur, C. Goutte, A. Kulesza, A. Sanchis, and N. Ueffing, "Confidence estimation for machine translation," in *Proceedings of the 20th International Conference on Computational Linguistics*, Geneva, Switzerland, 2004, pp. 315–321.

[10] L. Specia, M. Turchi, N. Cancedda, M. Dymetman, and N. Cristianini, "Estimating the sentence-level quality of machine translation systems," in *Proceedings of the 13th Annual Conference of the European Association for Machine Translation*, Barcelona, Spain, 2009, pp. 28–37.

[11] P. Geurts, D. Ernst, and L. Wehenkel, "Extremely randomized trees," *Machine Learning*, vol. 63, no. 1, pp. 3–42, 2006.

[12] E. Biçici and M. Dymetman, "Dynamic translation memory: Using statistical machine translation to improve translation memory fuzzy matches," *Computational Linguistics and Intelligent Text Processing*, pp. 454–465, 2008.

[13] M. Simard and P. Isabelle, "Phrase-based machine translation in a computer-assisted translation environment," in *Proceeding of the 12th Machine Translation Summit (MT Summit XII)*, Quebec, Canada, 2009, pp. 120–127.

[14] V. Zhechev and J. V. Genabith, "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*, Dublin, Ireland, 2010, pp. 43–49.

[15] P. Koehn and J. Senellart, "Convergence of translation memory and statistical machine translation," in *Proceedings of AMTA Workshop on MT Research and the Translation Industry*, Denver, USA, 2010, pp. 21–31.

[16] L. Li, C. Parra Escartín, and Q. Liu, "Combining translation memories and syntax-based SMT," *Baltic Journal of Modern Computing*, vol. 4, pp. 165–177, 2016.

[17] Y. Liu, K. Wang, C. Zong, and K.-Y. Su, "A unified framework and models for integrating translation memory into phrase-based statistical machine translation," *Computer Speech & Language*, vol. 54, pp. 176–206, March 2019.

[18] C. Hokamp and Q. Liu, "Lexically constrained decoding for sequence generation using grid beam search," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Vancouver, Canada, July 2017, pp. 1535–1546.

[19] J. Gu, Y. Wang, K. Cho, and V. O. Li, "Search engine guided neural machine translation," in *Proceedings of the 32 AAAI Conference on Artificial Intelligence*, New Orleans, USA, February 2018, pp. 5133–5140.

[20] B. Bulte and A. Tezcan, "Neural fuzzy repair: Integrating fuzzy matches into neural machine translation," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, Florence, Italy, Jul. 2019, pp. 1800–1809.

[21] R. A. Wagner and M. J. Fischer, "The string-to-string correction problem," *Journal of the Association for Computing Machinery*, vol. 21, no. 1, pp. 168–173, Jan. 1974. [Online]. Available: http://doi.acm.org/10.1145/321796.321811

[22] P. Koehn, *Statistical Machine Translation*. Cambridge University Press, 2010.

[23] M. Carl and A. Way, Eds., *Recent Advances in Example-Based Machine Translation*. Springer, 2003.

[24] P. Koehn, H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, C. Shen, W.and Moran, R. Zens, C. Dyer, O. Bojar, A. Constantin, and E. Herbst, "Moses: Open source toolkit for statistical machine translation," in *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume, Proceedings of the Demo and Poster Sessions*, Prague, Czech Republic, 2007, pp. 177–180.

[25] L. Specia, "Exploiting objective annotations for measuring translation post-editing effort," in *Proceedings of the 15th Annual Conference of the European Association for Machine Translation*, Leuven, Belgium, 2011, pp. 73–80.

[26] O. Bojar, C. Buck, C. Federmann, B. Haddow, P. Koehn, J. Leveling, C. Monz, P. Pecina, M. Post, H. Saint-Amand, R. Soricut, L. Specia, and A. Tamchyna, "Findings of the 2014 Workshop on Statistical Machine Translation," in *Proceedings of the Ninth Workshop on Statistical Machine Translation*, Baltimore, USA, 2014, pp. 12–58.

[27] E. Avramidis, "Sentence-level ranking with quality estimation," *Machine Translation*, vol. 27, pp. 239–256, 2013.

[28] D. Basak, S. Pal, and D. C. Patranabis, "Support vector regression," *Neural Information Processing – Letters and Reviews*, vol. 11, no. 10, pp. 203–224, 2007.

[29] L. Specia, G. Paetzold, and C. Scarton, "Multi-level translation quality prediction with QuEst++," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference of the Asian Federation of Natural Language Processing - System Demonstrations*, Beijing, China, 2015, pp. 115–120.

[30] J. B. Marino, R. E. Banchs, J. M. Crego, A. de Gispert, P. Lambert, J. A. Fonollosa, and M. R. Costa-Jussà, "N-gram-based machine translation," *Computational Linguistics*, vol. 32, no. 4, pp. 527–549, 2006.

[31] R. Steinberger, A. Eisele, S. Klocek, S. Pilos, and P. Schlüter, "DGT-TM: A freely available translation memory in 22 languages," in *Proceedings of the 8th International Conference on Language Resources and Evaluation*, Istambul, Turkey, 2012, pp. 454–459.

[32] M. Esplà-Gomis, F. Sánchez-Martínez, and M. L. Forcada, "Using machine translation to provide target-language edit hints in computer aided translation based on translation memories," *Journal of Artificial Intelligence Research*, vol. 53, no. 1, pp. 169–222, 2015.

[33] M. L. Forcada, M. Ginestí-Rosell, J. Nordfalk, J. O'Regan, S. Ortiz-Rojas, J. A. Pérez-Ortiz, F. Sánchez-Martínez, G. Ramírez-Sánchez, and F. M. Tyers, "Apertium: a free/open-source platform for rule-based machine translation," *Machine Translation*, vol. 25, no. 2, pp. 127–144, 2011.

[34] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik, "Gene selection for cancer classification using support vector machines," *Machine Learning*, vol. 46, no. 1-3, pp. 389–422, 2002.

[35] G. Louppe, L. Wehenkel, A. Sutera, and P. Geurts, "Understanding variable importances in forests of randomized trees," in *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 1*, Lake Tahoe, USA, 2013, pp. 431–439.

[36] L. Breiman, J. Friedman, C. J. Stone, and R. Olshen, *Classification and Regression Trees*. Taylor & Francis, 1984.

[37] P. Geurts and G. Louppe, "Learning to rank with extremely randomized trees," in *Proceedings of Machine Learning Research, Volume 14: Proceedings of the Learning to Rank Challenge*, Haifa, Israel, June 2011, pp. 49–61.

[38] C. R. Rao, *Linear statistical inference and its applications*. John Wiley & Sons, Inc., 1973.

[39] C. Strobl, A.-L. Boulesteix, T. Kneib, T. Augustin, and A. Zeileis, "Conditional variable importance for random forests," *BMC bioinformatics*, vol. 9, no. 1, p. 307, 2008.

[40] L. Breiman and A. Cutler, "randomforest: Breiman and cutler's random forests for classification and regression," https://www.stat.berkeley.edu/~breiman/RandomForests/, last accessed: 12th November 2019, 2019.