Predicting insertion positions in word-level machine translation quality estimation[☆]

Miquel Esplà-Gomis*, Felipe Sánchez-Martínez, Mikel L. Forcada

Dept. de Llenguatges i Sistemes Informàtics
Universitat d'Alacant
E-03690 St. Vicent del Raspeig (Spain)

Abstract

Word-level machine translation (MT) quality estimation (QE) is usually formulated as the task of automatically identifying which words need to be edited (either deleted or replaced) in a translation T produced by an MT system. The advantage of estimating MT quality at the word level is that this information can be used to guide post-editors since it enables the identification of the specific words in T that need to be edited in order to ease their work. However, word-level MT QE, as defined in the current literature, has an obvious limitation: it does not identify the positions in T in which missing words need to be inserted. To deal with this limitation, we propose a method which identifies both word deletions and insertion positions in T. This is, at the time of writing, the first approach allowing the identification of insertion positions in word-level MT QE. The method proposed can use any source of bilingual information —such as MT, dictionaries, or phrase-level translation memories— to extract features that are then used by a neural network to produce a prediction for both words and insertion positions (gaps between words) in the translation T. In this paper, several feature sets and neural network architectures are explored and evaluated on publicly-available datasets used in previous evaluation campaigns for word-level MT QE. The results confirm the feasibility of the proposed approach, as well as the usefulness of sharing information between the two prediction tasks in order to obtain more reliable quality estimations.

Keywords: machine translation, quality estimation, word-level quality estimation

1. Introduction

The use of machine translation (MT) systems to produce draft translations that are then corrected (post-edited) to make them adequate for a specific purpose has been shown to improve translation productivity [1, 2]. However, the quality of the translations produced by an MT system may vary from one sentence to another. Some translations may be worth post-editing, while it would be better to discard others and translate the source sentence from scratch or use other translation technologies, such as translation memories [3, 4]. Identifying the translations that are worth post-editing is, therefore, a key task as regards obtaining actual gains in productivity and it is, consequently, necessary to be able to estimate the quality of the translations produced.

MT quality estimation (QE) was first defined by Specia et al. [5] who built upon the closely-related task of MT confidence estimation [6]. MT QE is not only relevant in that it tries to reduce the need to bother professional translators or post-editors with useless translations, but also in that it may also be used to

[☆]Accepted for publication at *Applied Soft Computing* on 25th November 2018. First submitted on 19th January 2018.
*Corresponding author
Email addresses: `mespla@dlsi.ua.es` (Miquel Esplà-Gomis), `fsanchez@dlsi.ua.es` (Felipe Sánchez-Martínez), `mlf@dlsi.ua.es` (Mikel L. Forcada)

choose among several MT systems [7] for a given translation task, to estimate the post-editing effort of a given MT output, and to budget a translation job. Quality may be measured in terms of post-editing time, as the number of edit operations needed to turn the MT output into an adequate translation, or by using other related metrics, such as subjective effort metrics [8, 9]. In addition, when MT is used for assimilation, that is, for gisting, and the user of the translation has no knowledge of the source language (SL), quality labels may also be used to provide information regarding the reliability of the translation into the target language (TL).

Although most of the approaches for MT QE estimate the quality of the translations at the segment level, there are also approaches that estimate the quality of individual words [6, 10]. The former provide a quality score for the whole segment that can be used to decide whether or not it is worth post-editing it. The latter detects the words in a given translation that need to be post-edited in order to produce an adequate translation of the SL segment. This information may be used to guide post-editors in their job and help them focus on the parts of the translation that need to be fixed. In addition, word-level MT QE may facilitate the joint use of MT and translation memories since fuzzy match scores [11] —which measure the similarity between the segment to be translated and the source segment in a translation unit as the percentage of edit operations needed to transform one in the other— and percentages of words to post-edit can be easily compared. Although fuzzy-match scores are computed on the source language and the percentages of words to be post-edited are computed on the target language, they are closely linked and they are, in computer-aided translation, assumed to be so: the greater the difference between the source segments, the greater the number of edit operations it is necessary to perform on the target segment.

Most of the approaches for word-level MT QE focus on identifying those words in the machine-translated segment that need to be edited, that is, deleted or replaced [12]. This information, although useful, is not enough to estimate the extent of post-editing needed. Being capable of identifying also insertion positions would allow to predict more reliably the full sequence of edits. This is specially relevant for professional translators because it allows the creation of tools to better support the translation task. In this paper, we present a word-based MT QE approach that is capable of both predicting the words that need to be deleted and the positions into which a sequence of one or more words should be inserted. To deal with these two goals in a unified way, in this article we model substitution as a deletion followed by an insertion. To the best of our knowledge, this is the first approach which predicts insertion positions, that is, gaps into which a word or sequence of words should be inserted.

The approach presented here builds on previous work by the same authors [13, 14] in which insertion positions were not detected and a slightly different feature set was used. As in the original papers, here we use black-box bilingual resources from the Internet. In particular, we combine online MT systems and an online bilingual concordancer [15, 16] to spot sub-segment correspondences between the SL segment S and its machine translation T into the TL. This is done by dividing both S and T into all possible (overlapping) sub-segments, or n-grams, up to a certain maximum length. These sub-segments are then translated into the TL and the SL, respectively, by means of the bilingual resources mentioned. These sub-segment correspondences are then used to extract several collections of features that are fed to a neural network (NN) in order to determine the words to be deleted and the word positions into which insertions are required. One of the main advantages of this approach, when compared to the other approaches described below, is that it uses simple string-level bilingual information extracted from any available source. Obtaining this information directly from the Internet allows us to obtain quality estimations for the words in T on the fly, without having to rely on more complex sources, such as probabilistic lexicons, part-of-speech information or word nets.

We have experimented with three different NN architectures: In the first one, the words to be deleted and the positions into which words need to be inserted are predicted by two independent NNs; the second architecture uses the output of these two independent NNs for the words and insertion positions in the vicinity of the word or insertion position about which a decision is made as context; the third architecture uses a single NN to predict deletions and insertion positions and the input features used are those related to the word or insertion position on which a decision is being made, along with those in its vicinity.

The performance of this approach has been evaluated with two language pairs, English–Spanish and

English–German, using the publicly available datasets for the shared task on word-level MT QE at WMT15[1] [17] and WMT16[2] [18]. The experimental results confirm that our method, when compared to state-of-the-art methods that only detect the words to be edited (either replaced or removed) provides competitive results using considerably fewer features. In addition, our method is able to determine the insertion positions with an F_1 score of 39%, a precision of 44% and a recall of 36%.[3]

The remainder of the paper is organised as follows. The following section provides an overview of related work on word-level MT QE and stresses the main differences between these and our approach. Section 3 describes the features used, whereas Section 4 describes the three NN architectures we have evaluated. The experiments and results are then discussed in Section 5. Finally, the paper ends with some concluding remarks and two appendices, one showing the mathematical description of the NNs used, and another providing an algorithmic description of the features used.

2. Related work

Some of the early work on word-level MT QE can be found in the context of interactive MT [19, 6, 20]. While in standard MT there is no interaction between the user and the MT system during the translation process, in interactive MT the user drives the translation process by accepting or editing parts of the translation, affecting in turn the way in which other parts of the segment are translated by the MT system. In addition, interactive MT systems may provide the user with different translation suggestions for the same SL segment. Gandrabur and Foster [19] obtain confidence scores for each TL word in a given translation T of the SL segment S to help an interactive MT system to choose among the translation suggestions to be presented to the user. Similarly, Ueffing and Ney [20] obtain scores for each word in T, even though they are used to automatically decide which suggestions should be rejected. This second approach incorporates the use of probabilistic lexicons as a source of bilingual information.

Blatz et al. [6] introduced a more complex collection of features for word-level MT QE, using semantic features based on WordNet [21], translation probabilities from IBM model 1 [22], word posterior probabilities, and alignment templates [23] from statistical MT models. All the features employed are combined to train a binary classifier which is then used to determine the confidence scores.

Ueffing and Ney [10] evaluate several word-level confidence measures based on word posterior probabilities for word-level MT QE. They divide the features used by their approach into two categories: those which are independent of the MT system used for translation (black-box system-independent), as occurs with the features used in this paper, and those which are obtained from the inner workings of the statistical MT system used for translation (glass-box system-dependent). The latter features are obtained by comparing the output T of the statistical MT system to the best n translation hypotheses it produces. Several distance metrics are then used to check how often word t_j, the word at position j of T, is found in each translation hypothesis, and how far it is from position j. These features rely on the assumption that if word t_j appears in a similar position in a large number of translation hypotheses, then it is likely to be correct and does not need to be post-edited. Biçici [24] proposes a strategy by which to extend this kind of system-dependent features to what could be called a system-independent scenario. His approach consists of employing feature-decay algorithms [25] to choose parallel sentences from a parallel corpus, not necessarily the one on which the statistical MT system was trained, which are close to the segment S to be translated. Once this parallel corpus has been built, a new statistical MT system is trained and its internals are examined in order to extract these features.

Most of the recent advances in MT QE have been made by participants in the shared tasks on QE at the different editions of the Workshop on Statistical Machine Translation (WMT). Of the systems competing

[1]`http://www.statmt.org/wmt15/quality-estimation-task.html`

[2]`http://www.statmt.org/wmt16/quality-estimation-task.html`

[3]Word-level MT QE datasets are usually unbalanced, as there are more words that are adequately translated than otherwise (See Table 1 for some examples). In most evaluation scenarios, such as the shared tasks on MT QE at WMT15 and WMT16, word-level MT QE is evaluated by focusing on the less frequent class (words to delete or, in our case, positions into which insertions are required). This usually leads to relatively low scores; other metrics would surely have less pessimistic interpretations.

in WMT 2014 [9], it is worth singling out the MULTILIZER approach for sentence-level MT QE because it also uses other MT systems to translate S into the TL and T into the SL.[4] These translations are then used as a pseudo-reference and the similarity between them and the original SL and TL segments is computed and taken as an indication of quality. This approach, and that of Biçici and Yuret [25], are the most similar to the one we propose here because they also use other MT systems for QE, although they translate whole segments, whereas we translate sub-segments. Like the approach in this paper, MULTILIZER also combines several sources of bilingual information, while that of Biçici and Yuret [25] uses only one MT system. In any case, neither MULTILIZER nor the approach by Biçici and Yuret [25] work at the level of words and are able to predict insertion positions.

More recently, Blain et al. [26] proposed the use of *bilexical embeddings* [27] to model the strength of relationship between SL and TL words for their use for sentence-level and word-level MT QE. Bilexical embeddings are learned from SL and TL embeddings and word-aligned parallel corpora. The results obtained for word-level MT are below the baseline results for the WMT17 shared task [28].

With regard to the use of NNs, one of the first approaches using NNs was presented by Kreutzer et al. [29] to the word-level QE shared task at WMT15 [17]. Kreutzer et al. [29] use a deep feed-forward NN to encode SL and TL words into feature vectors using a lookup table that is tuned during training. The representation obtained from this network is then combined with the collection of baseline features provided by the shared-task organisers through linear combination. Recently, Liu et al. [30] have extended this work by building synthetic training data through the use of MT and a parallel corpus: they translate the source sentences in the parallel corpus by means of an MT system and then use the target sentences to automatically label the words in the MT output.

A more sophisticated approach was proposed by Martins et al. [31], who achieved the best results in the word-level QE shared task at WMT 2016 [18]. They combined a feed-forward NN with two recurrent NNs and used the predictions they provided as additional features for a linear sequential model [32]. This architecture has been extended [33] by adding the output of an automatic-post-editing tool to the input of the linear sequential model, resulting in a noticeable performance improvement.

At WMT17 [28], another NN approach was presented that obtained results comparable to, and in some cases even better than, those obtained by Martins et al. [33]: the Postech system [34, 35]. This system builds on a three-level stacked architecture trained in a multi-task fashion: at the first level there is a neural word prediction model trained on large-scale parallel corpora, at the second level, a word-level MT QE system, and at the third level, a sentence-level MT QE system.

Apart from the features used —Martins et al. [31] and Martins et al. [33] use lexical and syntactic features, computed on both individual words and word bi-grams, whereas Kim et al. [34, 35] do not extract any features at all— our approach differs as regards the NN architecture. We do not use any recurrent unit; instead, we define, in two of the NN architectures we have evaluated, a fixed-length context window around the word or insertion position on which a decision is being made. This architecture is easier to train (it requires less computational effort), it is easier to parallelise, and behaves similarly to a sliding-window or convolutional architecture.

3. Features based on black-box sources of bilingual information

The method described in this paper is based on previous approaches by the same authors [14, 13], which are in turn based on the work by Esplà-Gomis et al. [36], in which several online MT systems were used for word-level QE in translation-memory-based computer-aided translation tasks. The objective is for the method to be system-independent and able to use available online bilingual resources for word-level MT QE. These resources are used on-the-fly to detect relations between the original SL segment S and a given translation T in the TL as follows: first, all the (overlapping) sub-segments σ of S with lengths in $[1, L]$ are obtained and translated into the TL using the sources of bilingual information (SBI) available; the same process is carried out for all the overlapping sub-segments τ of T, which are translated into the SL. The

[4]To the best of our knowledge, there is no public description of the internal workings of MULTILIZER.

resulting sets of sub-segment translations, $M_{S\to T} = \{(\sigma,\tau)\}$ and $M_{T\to S} = \{(\sigma,\tau)\}$, are then used to spot sub-segment correspondences between T and S. Note that some SBI, such as phrase tables or bilingual concordancers, may provide additional data such as the number of occurrences (frequency of translation) or a probability; we can therefore also use the collections $M^{\text{occ}}_{S\to T} = \{(\sigma,\tau,\phi)\}$ and $M^{\text{occ}}_{T\to S} = \{(\sigma,\tau,\phi)\}$ of sub-segment pairs and their scores ϕ (number of occurrences or probabilities, depending on the resources available). In this section we describe a collection of features designed to represent these relations for their exploitation for word-level MT QE. We define two different sets of features: one whose objective is to detect the words in a translation T to be deleted (Section 3.1), and another whose objective is to detect the insertion positions in T into which a word, or sequence of words, needs to be inserted (Section 3.2). Appendix B provides pseudo-code for the different feature sets described in this section.

3.1. Features for word deletions

We define three collections of features to detect the words to be deleted: one taking advantage of the sub-segments τ that appear in T, $\text{Keep}_n(\cdot)$, another that uses the translation frequency with which a sub-segment σ in S is translated as the sub-segment τ in T, $\text{Freq}^{\text{keep}}_n(\cdot)$, and a third that uses the alignment information between T and τ and which does not require τ to appear as a contiguous sub-segment in T, $\text{Align}^{\text{keep}}_n(\cdot)$.

*3.1.1. Features for word deletions based on sub-segment pair occurrences (*Keep*)*

Given a set of sub-segment translations $M = \{(\sigma,\tau)\}$, that is, the union of $M_{S\to T}$ and $M_{T\to S}$, with $|\tau| \leq L$, obtained either when translating from SL into TL or vice versa, the first collection of features, $\text{Keep}_n(\cdot)$, is obtained by computing the amount of sub-segment translations $(\sigma,\tau) \in M$ with $|\tau| = n$ that confirm that word t_j in T should be kept in the translation of S. We consider that a sub-segment translation (σ,τ) confirms t_j if σ is a sub-segment of S, and τ is an n-word sub-segment of T that covers position j. This collection of features is defined as follows:

$$\text{Keep}_n(j,S,T,M) = \frac{|\{\tau : (\sigma,\tau) \in \text{conf}^{\text{keep}}_n(j,S,T,M)\}|}{|\{\tau : \tau \in \text{seg}_n(T) \wedge j \in \text{span}(\tau,T)\}|}$$

where $\text{seg}_n(X)$ represents the set of all possible n-word sub-segments of segment X, and function $\text{span}(\tau,T)$ returns the set of word positions spanned by the sub-segment τ in the segment T; if τ is found more than once in T, it returns all the possible positions spanned. Function $\text{conf}^{\text{keep}}_n(j,S,T,M)$ returns the collection of sub-segment pairs (σ,τ) that confirm a given word t_j, and is defined as:

$$\text{conf}^{\text{keep}}_n(j,S,T,M) = \{(\sigma,\tau) \in M \cap (\text{seg}_*(S) \times \text{seg}_n(T)) : j \in \text{span}(\tau,T)\}$$

where $\text{seg}_*(X)$ is similar to $\text{seg}_n(X)$ but without length constraints.[5]

We shall illustrate this collection of features with an example. Let us consider the Catalan segment $S =$ *ens van demanar que baixàrem el volum*, an English translation hypothesis $T =$ *they asked to make the volume go down*, and the reference translation $R =$ *they asked us to turn the volume down*. According to the reference, the words *make* and *go* in the translation hypothesis should be deleted: *go* should simply be removed, whereas *make* should be removed and the word *turn* should be inserted afterwards. In addition, the word *us* should be inserted between the words *asked* and *to*. Finally, let us suppose that the collection M of sub-segment pairs (σ,τ) is obtained by applying the available sources of bilingual information in order to translate the sub-segments in S up to length 3 into English:

M ={*(ens, us)*, *(van, did)*, *(demanar, ask)*, *(que, that)*, *(baixàrem, lower)*,
(el, the), ***(volum, volume)***, *(ens van, they going us)*,
(van demanar, they asked), *(demanar que, ask that)*, *(que baixàrem, to lower)*, *(baixàrem el, lower the)*,
(el volum, the volume),
(ens van demanar, they asked us), ***(van demanar que, they asked to)***, *(demanar que baixàrem, ask to lower)*, *(que baixàrem el, to lower the)*, *(baixàrem el volum, to turn the volume down)*}

[5]Esplà-Gomis et al. [13] conclude that constraining only the length of τ leads to better results than constraining both σ and τ.

Note that the sub-segment pairs (σ, τ) in bold type are those that fully match (thus confirming) the translation T,[6] while the rest may contradict some parts of T. The word *asked* (which corresponds to word position 2) is confirmed by two sub-segment pairs: (*van demanar*, *they asked*), with length 2, and (*van demanar que*, *they asked to*), with length 3. We therefore have that:

$$\mathrm{conf}_1^{\mathrm{keep}}(2, S, T, M) = \emptyset$$

$$\mathrm{conf}_2^{\mathrm{keep}}(2, S, T, M) = \{(\textit{van demanar}, \textit{they asked})\}$$

$$\mathrm{conf}_3^{\mathrm{keep}}(2, S, T, M) = \{(\textit{van demanar que}, \textit{they asked to})\}$$

In addition, we have that the sub-segments τ in $\mathrm{seg}_*(T)$ covering the word *asked* for lengths in $[1, 3]$ are:

$$\{\tau : \tau \in \mathrm{seg}_1(T) \wedge 2 \in \mathrm{span}(\tau, T)\} = \{\textit{asked}\}$$

$$\{\tau : \tau \in \mathrm{seg}_2(T) \wedge 2 \in \mathrm{span}(\tau, T)\} = \{\textit{they asked}, \textit{asked to}\}$$

$$\{\tau : \tau \in \mathrm{seg}_3(T) \wedge 2 \in \mathrm{span}(\tau, T)\} = \{\textit{they asked to}, \textit{asked to make}\}$$

The resulting $\mathrm{Keep}_n(\cdot)$ features for the word *volume* are, therefore:

$$\mathrm{Keep}_1(2, S, T, M) = \frac{|\{\tau : (\sigma, \tau) \in \mathrm{conf}_1^{\mathrm{keep}}(2, S, T, M)\}|}{|\{\tau : \tau \in \mathrm{seg}_1(T) \wedge 2 \in \mathrm{span}(\tau, T)\}|} = \frac{0}{1}$$

$$\mathrm{Keep}_2(2, S, T, M) = \frac{|\{\tau : (\sigma, \tau) \in \mathrm{conf}_2^{\mathrm{keep}}(2, S, T, M)\}|}{|\{\tau : \tau \in \mathrm{seg}_2(T) \wedge 2 \in \mathrm{span}(\tau, T)\}|} = \frac{1}{2}$$

$$\mathrm{Keep}_3(2, S, T, M) = \frac{|\{\tau : (\sigma, \tau) \in \mathrm{conf}_3^{\mathrm{keep}}(2, S, T, M)\}|}{|\{\tau : \tau \in \mathrm{seg}_3(T) \wedge 2 \in \mathrm{span}(\tau, T)\}|} = \frac{1}{2}$$

3.1.2. Features for word deletions based on sub-segment pair occurrences using translation frequency ($\mathrm{Freq}_n^{\mathrm{keep}}$)

The second collection of features uses the number of occurrences of the sub-segment pairs in $M^{\mathrm{occ}} = \{(\sigma, \tau, \phi)\}$. This information is not available for MT, but it is available for the bilingual concordancer we have used for the experiments (see Section 5.2). The number of occurrences of sub-segment pair (σ, τ) can be used to determine how often σ is translated as τ and, therefore, how reliable this translation is. We define $\mathrm{Freq}_n^{\mathrm{keep}}(\cdot)$ as:

$$\mathrm{Freq}_n^{\mathrm{keep}}(j, S, T, M^{\mathrm{occ}}) = \sum_{\forall (\sigma, \tau, \phi) \in \mathrm{conf}_n^{\mathrm{keep}}(j, S, T, M^{\mathrm{occ}})} \frac{\mathrm{occ}(\sigma, \tau, M^{\mathrm{occ}})}{\sum_{\forall (\sigma, \tau') \in M^{\mathrm{occ}}} \mathrm{occ}(\sigma, \tau', M^{\mathrm{occ}})}$$

where function $\mathrm{occ}(\sigma, \tau, M^{\mathrm{occ}})$ returns the number of occurrences ϕ in M^{occ} for the sub-segment pair (σ, τ). Note that each term in $\mathrm{Freq}_n^{\mathrm{keep}}(\cdot)$ is equivalent to the probability $p(\tau|\sigma)$ used in phrase-based statistical MT where M^{occ} would act as a phrase table.

To continue with the running example, and assuming that we have a sub-segmental translation memory which contains 99 occurrences of the sub-segment *van demanar* translated as *they asked*, 11 occurrences in which it is translated as *they demanded*, and 10 in which it is translated as *they inquired*, the feature using these counts for sub-segments of length 2 would be:

$$\mathrm{Freq}_2^{\mathrm{keep}}(2, S, T, M) = \frac{99}{99 + 11 + 10} = \frac{33}{40}$$

[6]These sub-segment pairs are those defined as $M \cap (\mathrm{seg}_*(S) \times \mathrm{seg}_n(T))$ in function $\mathrm{conf}_n^{\mathrm{keep}}(\cdot)$.

3.1.3. Features for word deletions based on word alignments of partial matches ($\text{Align}_n^{\text{keep}}$)

The third collection of features takes advantage of partial matches, that is, of sub-segment pairs (σ, τ) in which τ does not appear as is in T. Given this condition, only resources $M_{S\to T}$ translating from SL into TL can be used, since those translating from TL into SL would always contain any τ sub-segment appearing in T. This collection of features is defined as:

$$\text{Align}_n^{\text{keep}}(j, S, T, M, e) = \sum_{\forall\tau\in\text{segs\_edop}_n(j,S,T,M,e)} \frac{|\text{LCS}(\tau, T)|}{|\tau|} \tag{1}$$

where $\text{LCS}(X, Y)$ returns the word-based longest common sub-sequence between segments X and Y, and $\text{segs\_edop}_n(j, S, T, M, e)$ returns the set of sub-segments τ of length n from M that are a translation of a sub-segment σ from S and in which, after computing the LCS with T, the j-th word t_j is assigned the edit operation e:[7]

$$\begin{gathered}\text{segs\_edop}_n(j, S, T, M, e) = \\ \{\tau : (\sigma, \tau) \in M \wedge \sigma \in \text{seg}_*(S) \wedge |\tau| = n \wedge \text{editop}_1(t_j, T, \tau) = e\}\end{gathered} \tag{2}$$

where $\text{editop}_1(t_j, T, \tau)$ returns the edit operation assigned to t_j and e is either `delete` or `match`. If $e = \texttt{match}$ the resulting set of features provides evidence in favour of keeping the word t_j unedited, whereas when $e = \texttt{delete}$ it provides evidence in favour of removing it.

In the running example, there are three sub-segment pairs (σ, τ) for which the word *asked* has $\text{editop}_1(t_j, T, \tau) =$ `match` with $T =$*they asked to make the volume go down*: one sub-segment pair with length 2, (*van demanar*, *they asked*), and two sub-segment pairs with length 3, (*van demanar que*, *they asked to*) and (*ens van demanar*, *they asked us*). With the exception of the last one, all these sub-segments are fully matched in T; the last one has two matching words, *they* and *asked.* Consequently, the values of features $\text{Align}_n^{\text{keep}}(j, S, T, M, \texttt{match})$ for the word *asked* are:

$$\text{Align}_1^{\text{keep}}(2, S, T, M, \texttt{match}) = \frac{|\emptyset|}{|\textit{asked}|} = 0$$

$$\text{Align}_2^{\text{keep}}(2, S, T, M, \texttt{match}) = \frac{|\textit{they asked}|}{|\textit{they asked}|} = \frac{2}{2}$$

$$\text{Align}_3^{\text{keep}}(2, S, T, M, \texttt{match}) = \frac{|\textit{they asked}|}{|\textit{they asked us}|} + \frac{|\textit{they asked to}|}{|\textit{they asked to}|} = \frac{5}{3}$$

In addition, there are two sub-segment pairs (σ, τ) for which the word *asked* has $\text{editop}_1(t_j, T, \tau) =$ `delete`: *(ens van, they going us)* and *(demanar que baixàrem, ask to lower)*, both with length 3. Therefore, the value of features $\text{Align}_n^{\text{keep}}(j, S, T, M, \texttt{delete})$ for the word *asked* with $n \in [1, 2]$ is 0, while for $n = 3$ its value is:

$$\text{Align}_3^{\text{keep}}(2, S, T, M, \texttt{delete}) = \frac{|\textit{they}|}{|\textit{they going us}|} + \frac{|\textit{to}|}{|\textit{ask to lower}|} = \frac{2}{3}$$

Note that feature $\text{Align}_n^{\text{keep}}(\cdot)$ is the only one to provide evidence that a word should be deleted. For instance, in the running example, the word *go* needs to be deleted; in the case of this word, all features but $\text{Align}_n^{\text{keep}}(\cdot)$ take the value 0. For edit operation `delete` there is one sub-segment pair that provides evidence that the word *go* should be deleted: *(baixàrem el volum, to turn the volume down)*.

The three collections of features described so far, $\text{Keep}_n(\cdot)$, $\text{Freq}_n^{\text{keep}}(\cdot)$, and $\text{Align}_n^{\text{keep}}(\cdot)$, are computed for t_j for all the values of sub-segment length $n \in [1, L]$. Features $\text{Keep}_n(\cdot)$ and $\text{Freq}_n^{\text{keep}}(\cdot)$ are computed for

[7]Note that the sequence of edit operations needed to transform X in Y is obtained as a by-product of the computation of $\text{LCS}(X, Y)$; these operations are insertions, deletions or matches (when the corresponding word does not need to be edited).

the collection of sub-segment pairs M obtained by translating from SL into TL ($M_{S\to T}$), and the collection of sub-segment pairs obtained by translating in the reverse manner ($M_{T\to S}$). As a result, $2L$ features are computed for $\text{Keep}_n(\cdot)$ and $2L$ more for $\text{Freq}_n^{\text{keep}}(\cdot)$. $\text{Align}_n^{\text{keep}}(\cdot)$ uses only $M_{S\to T}$; it is, however, computed twice: once for the edit operation `match`, and once for the edit operation `delete`. This brings the total to $6L$ features per word t_j.

3.2. Features for insertion positions

In this section, we describe three collections of features, which are based on those described in Section 3.1 for word deletions, and are designed to detect insertion positions. The main difference between them is that the former apply to words, while the latter apply to gaps; we shall refer to the gap after word t_j as γ_j.[8]

*3.2.1. Features for insertion positions based on sub-segment pair occurrences (*NoInsert*)*

The first collection of features, $\text{NoInsert}_n(\cdot)$, based on the $\text{Keep}_n(\cdot)$ features defined in Section 3.1.1 for word deletions, is defined as follows:

$$\text{NoInsert}_n(j,S,T,M) = \frac{|\{\tau : (\sigma,\tau) \in \text{conf}_n^{\text{noins}}(j,S,T,M)\}|}{|\{\tau : \tau \in \text{seg}_n(T) \wedge j \in \text{span}(\tau,T) \wedge j+1 \in \text{span}(\tau,T)\}|}$$

where function $\text{conf}_n^{\text{noins}}(j,S,T,M)$ returns the collection of sub-segment pairs (σ,τ) that confirm a given insertion position γ_j, and is defined as:

$$\text{conf}_n^{\text{noins}}(j,S,T,M) = \{(\sigma,\tau) \in M \cap (\text{seg}_*(S) \times \text{seg}_n(T)) : [j,j+1] \subset \text{span}(\tau,T)\}$$

$\text{NoInsert}_n(\cdot)$ accounts for the number of times that the translation of sub-segment σ from S makes it possible to obtain a sub-segment τ that covers the gap γ_j, that is, a τ that covers both t_j and t_{j+1}. If a word is missing in position γ_j, one would expect to find fewer sub-segments τ that cover this gap, therefore obtaining low values for $\text{NoInsert}_n(\cdot)$, while if there is no word missing in this position of T, one would expect more sub-segments τ to cover the gap, therefore obtaining values of $\text{NoInsert}_n(\cdot)$ closer to 1. In order to be able to identify insertion positions before the first word or after the last word, we use imaginary sentence boundary words t_0 and $t_{|T|+1}$, which can also be matched,[9] thus allowing us to obtain evidence for gaps γ_0 and $\gamma_{|T|}$.

In the running example, gap γ_1 between the words *they* and *asked* is covered by two sub-segment pairs: *(van demanar, they asked)* and *(van demanar que, they asked to)*. The values of feature $\text{NoInsert}_n(1,S,T,M)$ for $n \in [2,3]$ are, therefore:[10]

$$\text{NoInsert}_2(1,S,T,M) = \frac{|\{\textit{they asked}\}|}{|\{\textit{they asked}\}|} = \frac{1}{1}$$

$$\text{NoInsert}_3(1,S,T,M) = \frac{|\{\textit{they asked to}\}|}{|\{\textit{they asked to}\}|} = \frac{1}{1}$$

3.2.2. Features for insertion positions based on sub-segment pair occurrences using translation frequency ($\text{Freq}_n^{\text{noins}}$*)*

The same adaptation can be carried out with the $\text{Freq}_n^{\text{keep}}(\cdot)$ feature collection defined in Section 3.1.2 to obtain the equivalent feature collection for insertion positions:

$$\text{Freq}_n^{\text{noins}}(j,S,T,M^{\text{occ}}) = \sum_{\forall(\sigma,\tau)\in\text{conf}_n^{\text{noins}}(j,S,T,M^{\text{occ}})} \frac{\text{occ}(\sigma,\tau,M^{\text{occ}})}{\sum_{\forall(\sigma,\tau')\in M^{\text{occ}}} \text{occ}(\sigma,\tau',M^{\text{occ}})}$$

[8]Note that the index of the first word in T is 1, and gap γ_0 corresponds to the space before the first word in T.
[9]These boundary words are annotated in M when this resource is built.
[10]Note that sub-segments shorter than 2 words cannot be used to identify insertion positions.

As previously described in the example for feature $\text{Freq}_n^{\text{keep}}(\cdot)$, the running example assumes a source of bilingual information which contains 99 occurrences of sub-segment *van demanar* translated as *they asked*, 11 occurrences in which it is translated as *they demanded*, and 10 in which it is translated as *they inquired*; the feature that uses these frequencies for gap γ_1 is:

$$\text{Freq}_2^{\text{noins}}(1, S, T, M) = \frac{99}{99 + 11 + 10} = \frac{33}{40}$$

3.2.3. Features for insertion positions based on word alignments of partial matches ($\text{Align}_n^{\text{noins}}$)

Finally, the collection of features $\text{Align}_n^{\text{keep}}(\cdot)$ defined in Section 3.1.3 for word deletions can be easily repurposed to detect insertion positions by setting the edit operation e in Eq. (1) to `match` and `insert` and redefining Eq. (2) as

$$\text{segs\_edop}_n(j, S, T, M, e) = \{\tau : (\sigma, \tau) \in M \wedge |\tau| = n \wedge \text{editop}_2(t_j, \tau, T) = e\}$$

where function $\text{editop}_2(t_j, \tau, T)$ is analogous to $\text{editop}_1(t_j, \tau, T)$ except for the fact that it computes the LCS between τ and T, rather than the other way round.[11] We shall refer to this last collection of features for insertion positions as $\text{Align}_n^{\text{noins}}(\cdot)$.

In the running example, the values for features $\text{Align}_n^{\text{noins}}(j, S, T, M, \texttt{match})$ are:

$$\text{Align}_2^{\text{noins}}(1, S, T, M, \texttt{match}) = \frac{|\textit{they asked}|}{|\textit{they asked}|} = \frac{2}{2}$$

$$\text{Align}_3^{\text{keep}}(1, S, T, M, \texttt{match}) = \frac{|\textit{they asked}|}{|\textit{they asked us}|} + \frac{|\textit{they asked to}|}{|\textit{they asked to}|} = \frac{5}{3}$$

In this case, there is no sub-segment τ for which $\text{editop}_2(t_1, T, \tau) = \texttt{insert}$. However, there is one sub-segment pair that indicates that the word *turn* should be added after the word *make*: (σ, τ)=(*baixàrem el volum*, *to turn the volume down*):

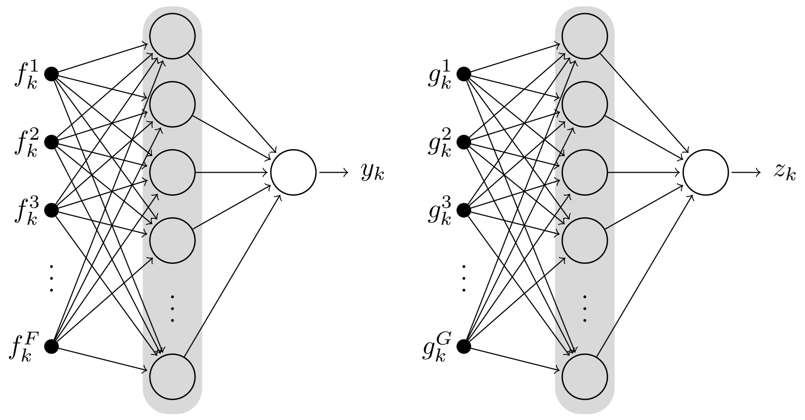
$$\text{Align}_5^{\text{noins}}(4, S, T, M, \texttt{insert}) = \frac{|\textit{to the volume down}|}{|\textit{to turn the volume down}|} = \frac{4}{5}$$

The collections of features for insertion positions, $\text{NoInsert}_n(\cdot)$, $\text{Freq}_n^{\text{noins}}(\cdot)$ and $\text{Align}_n^{\text{noins}}(\cdot)$, are computed for gap γ_j for all the values of sub-segment length $n \in [2, L]$. As in the case of the feature collections employed to detect deletions, the first two collections can be computed by using both $M_{S\to T}$ and $M_{T\to S}$, while the latter can only be computed using $M_{S\to T}$ for the edit operations `insert` and `match`. This yields $6(L-1)$ features in total per gap γ_j.

4. Neural network architecture for word-level MT QE

The features described above are used to predict the words to be deleted and the insertion positions into which insertions are required using NNs. We use NNs instead of other machine learning approaches because NNs are suitable for non-linear classification problems [37, Chapter 6] and a NN with a single hidden layer and a finite number of neurons can approximate any continuous function [38]. In addition, NNs allow us to combine information about words and insertion positions through more complex architectures, as shown in Sections 4.2 and 4.3, and train them together. In any case, we have tried with alternative machine-learning approaches; in particular, with extremely randomised trees [39] and non-linear support vector machines [40]. For extremely randomised trees we used as many trees as twice the number of features, and for non-linear

[11]It is worth noting that $\text{LCS}(X, Y) = \text{LCS}(Y, X)$, but the sequences of edit operations obtained as a by-product are different in each case.



(a) Network employed to predict whether the k-th word needs to be deleted.

(b) Network employed to predict whether insertions are required after the k-th word.

Figure 1: Two independent neural networks, one used to predict the words to be deleted and another used to predict the gaps into which insertions are required. Inputs f_k and g_k represent the features employed for word deletions and insertion positions, while y_k and z_k are the output (decision) nodes for each network.

support vector machines we used a radial basis function kernel with a kernel coefficient of 1 divided by the number of features. In both cases the results obtained where significantly lower than those obtained with NNs.

We have tried three different predictor architectures, which are explained below. All the NNs proposed have the objective of using relatively simple architectures for usability. In all cases, a special token is introduced to mark the beginning of the machine translation output, so that insertion positions are always found after a token.

4.1. Two independent neural networks

The simplest NN architecture consists of having two independent feed-forward networks, one for predicting whether the word t_k at position k needs to be deleted, and another to predict whether insertions are required at the gap γ_k after t_k. Figure 1 depicts the architectures of these two NNs in which features for word deletions (f_k) and for insertion positions (g_k) are used to feed each network.

Each network has a single hidden layer consisting of M and N units, respectively. This results in $FM + GN + 2M + 2N + 2$ parameters between the two networks, where F and G are the number of features that inform the network used to predict deletions and the number of features used as input to the network that predicts insertions, respectively. Both F and G include an additional binary input feature that will be used by the architecture described in sections 4.2 and 4.3.[12] Note that k takes values in $[1, |T|]$ for word deletions while it takes values in $[0, |T|]$ for insertion positions in order to make it possible to identify insertions before the first word of T.

4.2. Cascaded revision of prediction using context

The two networks described above do not take the predictions for neighbouring words and insertion positions into account. We therefore propose two additional feed-forward networks which revise these isolated predictions by taking into account the predictions made for the surrounding words and insertion positions. These additional networks, henceforth *cascaded-revision* NNs, take as input the outputs of the two independent networks shown in Figure 1. In particular, they use the outputs for the k-th word and insertion position, y_k and z_k respectively, along with those in the vicinity, to produce revised predictions y'_k and z'_k.

[12] In the experiments with the architecture defined in Section 4.1 the value of these additional input features is set to zero.

To handle the situation in which revising the prediction for words and insertion positions at the beginning or the end of a sentence would require as a context predictions for non-existing words and insertion positions, we introduce additional binary input features. These input features are set to zero when the associated input neuron for y_k or z_k refers to an existing word or insertion position, and to one otherwise.

Figure 2 depicts the architecture of these two additional networks using a single hidden layer with P and Q units, respectively, and C context positions on the left and on the right. Notice that the binary input features mentioned above are not shown in the figure for the sake of clarity; there is one such feature per input context neuron.

The addition of the cascaded-revision NNs signifies that the number of parameters to estimate calculated in Section 4.1 ($FM + GN + 2M + 2N + 2$) is increased by $4P + 4Q + 4CP + 4CQ + 2$.

We have tried two different ways of training the NN that results from using the output neurons of the independent NNs depicted in Figure 1 as input to the NNs shown in Figure 2: one that trains the independent and cascaded-revision NNs in two steps and another that trains them together in a single step.

Two-step training. This training strategy first trains the independent NNs described in Section 4.1 and then uses their outputs as the input to the cascaded-revision NNs to produce the revised predictions. This training process is fairly simple and only has to train four feed-forward NNs in isolation: two independent NNs and two cascaded-revision NNs.

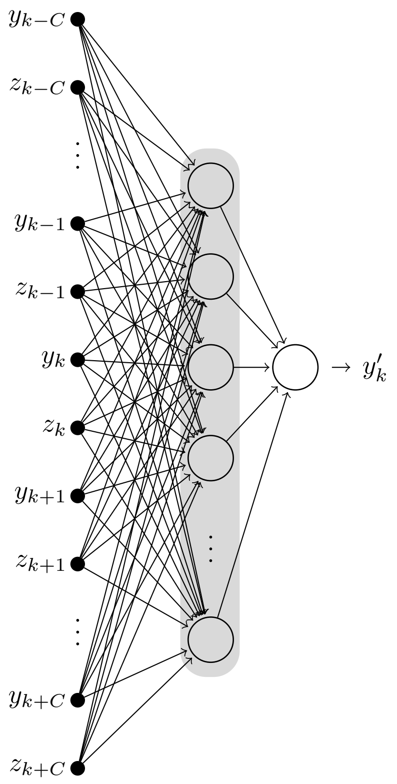
In the case of the words and insertion positions at the beginning or the end of a sentence, the right or left context may not exist and we set these predictions to 0.5.

One-step training. This second training procedure trains all the NNs simultaneously and is aimed at improving the results by allowing the independent NNs to benefit from the feedback provided by the cascaded-revision NNs. For a fair comparison, two preventive decisions were made. First, parameter tying was used between the different instances of the independent NNs in order to have the same number of parameters as in the two-step training procedure explained above. Secondly, we followed a multi-task approach, in which the independent NNs were trained to predict the actual estimate at each position in $[k - C, k + C]$ and the cascaded-revision NNs were trained to predict the actual predictions for position k; the same weight was given to each loss function. Both training strategies were, therefore, provided with exactly the same information during training. As in the case of two-step training, context may not exist for word and insertion positions at the beginning or the end of the sentence; in those cases we use feature vectors $\vec{f_i}$ and $\vec{g_i}$ with all their values set to 0.0, except for the binary input feature introduced in Section 4.1, whose value is set to one. These binary features are used to flag non-existing word or insertion positions in the range $[k - C, k + C]$; otherwise they are set to zero.

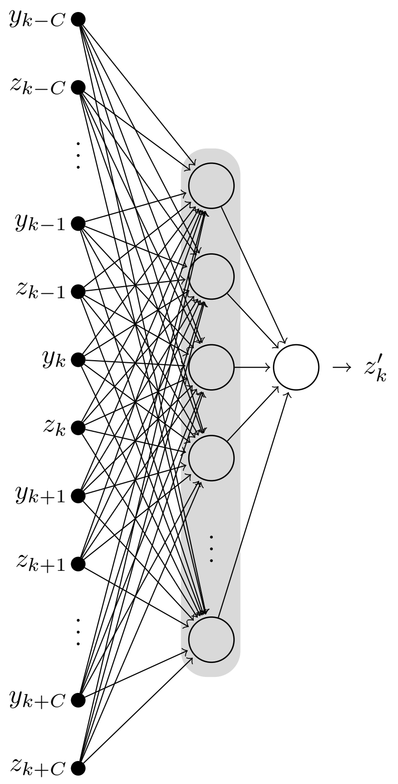
4.3. Single neural network for joint prediction of deletions and insertions

The NN proposed in Section 4.2 takes context into account by reviewing a sequence of predictions made by the independent NNs defined in Section 4.1. This is done by using these predictions as the input of an NN with a hidden layer and an output layer that retrieves the reviewed predictions. Here, we propose a slightly different NN in which, rather than obtaining predictions with the independent NNs and then reviewing them, the hidden layers of the independent NNs are directly connected to the hidden layer of the revision NN. Figure 3 depicts this architecture; as will be noted, each feature vector $\vec{f_k}$ and $\vec{g_k}$ is used as the input for the hidden layer of the corresponding independent NN in Section 4.1, and the neurons in each of these hidden layers are connected to a second hidden layer. Finally, a single output layer is added with two neurons, one that predicts deletions and another that predicts insertions. It is worth mentioning that the parameters of the hidden layers of the independent NNs are shared, thus reducing the number of parameters to be learned. As in Section 4.2, for the words and insertion positions to the left of the beginning or to the right of the end of a sentence, we use feature vectors $\vec{f_i}$ and $\vec{g_i}$ with all their values set to 0.0, except for the binary input feature introduced in Section 4.1, whose value is set to one.

The number of parameters of this NN is $2MCH + 2NCH + MH + NH + FM + GN + 3H + M + N + 2$, where F and G are, respectively, the total number of input features for each word and for each insertion position (including the additional binary input neurons), M and N are the number of hidden layers in the



(a) Network for cascaded revision of prediction made for the k-th word.



(b) Network for cascaded revision of prediction made for the word position after the k-th word.

Figure 2: Two neural networks for cascaded revision of the predictions made by the isolated NNs shown in Figure 1 by using the context of $2C$ words and word positions around the word, or word position, on which a decision is being made. In this case, input values y_k and z_k correspond to the outputs of the NN in Figure 1 and y'_k and z'_k are the cascaded-reviewed output values.

Dataset		Total number			
		segments	words	deletions	insertions
WMT'15	training	11,272	257,879	49,321 (19%)	38,246 (16%)
	development	1,000	23,098	4,455 (19%)	3,405 (16%)
	test	1,817	40,883	7,720 (19%)	6,010 (16%)
WMT'16	training	12,000	210,958	45,162 (21%)	36,217 (19%)
	development	1,000	19,487	3,809 (20%)	3,069 (17%)
	test	2,000	34,531	6,668 (19%)	6,010 (15%)

Table 1: Number of segments, number of words, number of word deletions and number of insertions in each portion of the two datasets used in the experiment.

NN that predict word deletions and insertion positions, respectively, H is the number of units in the second hidden layer and C is the amount of context to be used on each side of the word and insertion position for which quality is estimated.

5. Experiments and results

We have evaluated the method for word-level MT QE described in the previous sections using the datasets provided for the shared tasks on MT QE at the 2015 (WMT15; [17]) and 2016 (WMT16; [18]) editions of the Workshop on Statistical Machine Translation. In what follows we describe these two datasets and how they were used to identify the words to be deleted and the word positions into which insertions are required (see Section 5.1), the sources of bilingual information used (see Section 5.2), how the training of the different neural networks described in Section 4 was performed (see Section 5.5) and the results obtained (see Section 5.6).

5.1. Datasets

The WMT15 and WMT16 datasets consist of a collection of segments S in English, their corresponding machine translations T into Spanish in the case of WMT15 and into German in the case of WMT16, obtained through MT, and their post-edited versions R.

The original datasets label each word in every translation T as GOOD (`match`), when it is properly translated, or as BAD (`delete` in our experiments), when post-editing is required (the word must either be removed or replaced); however, no information is provided as regards the insertion positions. In order to evaluate our method for predicting insertion positions, we computed the sequence of edit operations required to convert T into R using the LCS algorithm [41] and subsequently used it to determine the word positions into which insertions were required.

Table 1 describes the amount of segments and words in each of the three portions of the two datasets (training, development and test), along with the amount of words to be deleted and the word positions into which insertions are required. As can be seen, the amount of insertions is slightly lower than the number of word deletions for all datasets: in general, about 19% of the words need to be deleted, and about 16% of them require an insertion after them. With regard to the number of insertion positions, Table 2 provides more detailed information by dividing them into two classes: those that are the result of a replacement (one deletion plus an insertion) and those that are independent (one or more words are inserted). The results shown in this table indicate that about 30% of the insertion positions in the datasets are independent while the rest are the result of a replacement. This accounts for the relevance of the problem tackled in this work, since these independent insertions would never have been detected by any of the approaches in the literature.

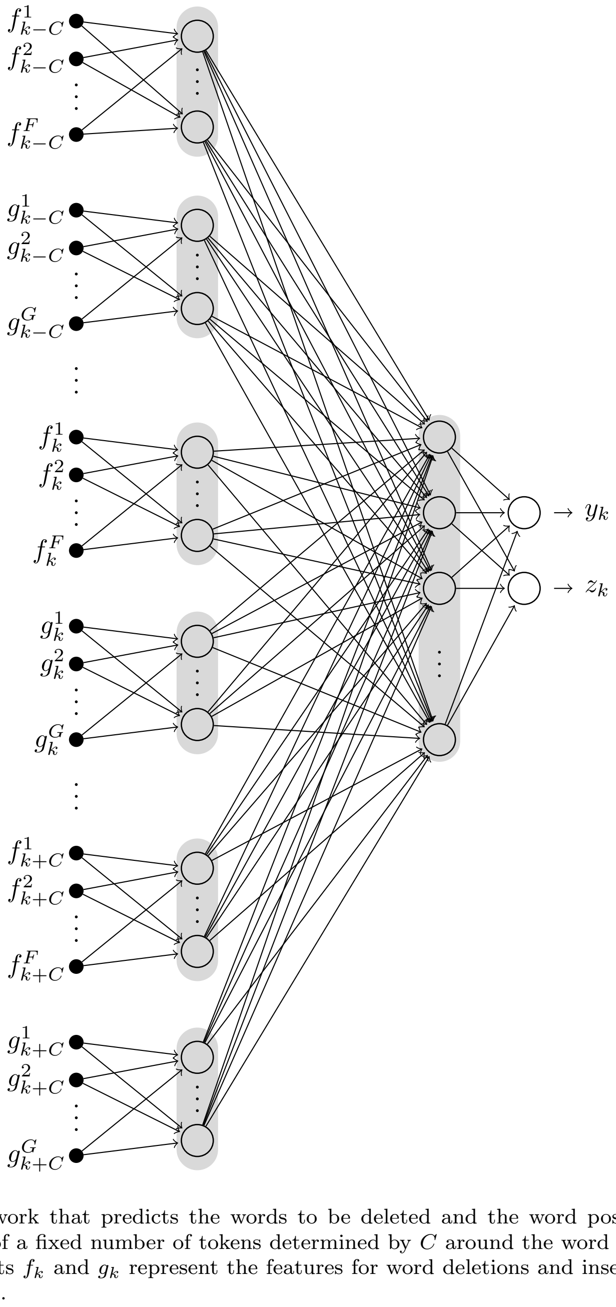


Figure 3: A single neural network that predicts the words to be deleted and the word positions into which insertions are required by using the context of a fixed number of tokens determined by C around the word and the word position for which a decision is being made. Inputs f_k and g_k represent the features for word deletions and insertion positions, while y_k and z_k are the output (decision) nodes.

Dataset		independent insertions	insertions tied to deletions
WMT'15	training	10,212 (27%)	28,034 (73%)
	development	884 (21%)	3,405 (79%)
	test	1,606 (32%)	2,521 (68%)
WMT'16	training	12,062 (33%)	24,155 (67%)
	development	1,062 (34%)	2,007 (66%)
	test	1,948 (24%)	6,010 (76%)

Table 2: Number of insertions that are independent vs. number of insertions that are the result of a replacement (a deletion plus an insertion).

Sequence of edits. The sequence of edits from which the insertion positions are derived is obtained as a by-product of the computation of the word-level LCS between T, the MT output, and its post-edited translation R. The edit operations that can be obtained with this algorithm are deletions and insertions, unlike with the edit distance algorithm [42] in which substitutions are also taken into account. The edition sequences obtained may, in some cases, be ambiguous, given that the substitution of one word for another may be modelled as an insertion followed by a deletion or as a deletion followed by an insertion (in our experiments, we chose the second option); however, as all segments in the datasets are processed in the same manner, this will have no effect on the results.

With regard to the detection of insertion positions, and given that the objective of our method is only to detect the positions in T into which words need to be inserted, and not the exact number of words to be inserted, a sequence of insertions is simplified to just one insertion. For example, for $T =$ *The European Association for the Automatic Translation is noncommercial organisation* and $R =$ *The European Association for Machine Translation is a nonprofit organisation*, the sequence of editions would be (`match`, `match`, `match`, `match`, `delete`, `delete`, `insert`, `match`, `match`, `delete`, `insert`, `match`), in which the last `insert` refers to the insertion of two words, *a* and *nonprofit*.

5.2. Sources of bilingual information

We have used two different kinds of sources of bilingual information: MT, a less informative bilingual resource (M), and a bilingual concordancer, a more informative resource that provides the number of occurrences of each sub-segment translation (M^{occ}). We used three MT systems that are freely available on the Internet: Apertium [43], Lucy,[13] and Google Translate.[14] While Google Translate was used for both datasets, Lucy was used only for WMT16 and Apertium only for WMT15. Two MT systems (of different types) were, therefore, used for each dataset.

The bilingual concordancer used is Reverso Context;[15] which provides, for a given SL sub-segment, the collection of TL translation alternatives, together with the number of occurrences of the sub-segment pair in the translation memory. The sub-segment translations obtained from this source of information are more reliable than those obtained from MT, since they are extracted from manually translated texts (although some sub-segments may be wrong owing to alignment errors). Its main weakness is, however, its lack of source coverage: although Reverso Context uses a large translation memory, no translation can be obtained for those SL sub-segments not found in it. Moreover, the sub-sentential translation memory contains only those sub-segment translations with a minimum number of occurrences. On the contrary, MT systems will always produce a translation, but it may be wrong or contain untranslated out-of-vocabulary words. We have combined these complementary sources of bilingual information to improve the performance of the approaches for word-level MT QE proposed. It is worth noting that other resources, such as phrase tables from phrase-based statistical MT systems, can be used as alternative bilingual resources.

[13]`http://www.lucysoftware.com/english/machine-translation/`
[14]`http://translate.google.com`
[15]`http://context.reverso.net/`

In our experiments, we computed the features described in Section 3 separately for both sources of information. It is worth mentioning again that the features based on translation occurrences cannot be obtained for MT. The value of the maximum sub-segment length L used was set to 5 for both languages. This value was chosen after a set of preliminary experiments in which the value of L was initialised to 1 and incremented until the performance of the independent NNs described in Section 4.1 on the WMT15 dataset converged. In fact the difference between the results with $L = 4$ and $L = 5$ were not statistically significant, even though those with $L = 5$ were slightly higher.

5.3. Evaluation

The evaluation was carried out by following the guidelines provided for each shared task to ease the comparison with the state of the art. In both WMT15 and WMT16, word-level MT QE is tackled as a binary-classification problem, signifying that standard precision, recall, and F_1-score metrics are used for evaluation. In WMT15, the main evaluation metric was the F_1 score for the least frequent class in the dataset, that is the F_1 measure for the `BAD` class (or `delete` class, as it is denominated in this paper). Conversely, in WMT16, the main evaluation metric was the product of the F_1 score of the two classes: `GOOD` and `BAD` (`match` and `delete` in our case). Although in Section 5.6 we provide all the metrics mentioned above, we used these two main metrics to tune the corresponding binary classifiers (see Section 5.5 for a description of the method used), thus enabling the results obtained for word deletions to be easily compared with those obtained by the approaches participating in these shared tasks.

To compare our approach with other state-of-the-art methods, we took the best performing systems in the word-level MT QE shared tasks in WMT15 and WMT16 as a reference; these are the works by Esplà-Gomis et al. [13] and Martins et al. [31], respectively, and focus solely on the task of identifying the words to be deleted or replaced (words tagged as `BAD`). Given the absence of previous approaches concerning the identification of insertion positions, we defined a *dumb* baseline implementing the *null hypothesis.* This is a classifier that assigns a label to each word and insertion position in a weighted-random fashion, using the a priori probability of each class in the dataset.

5.4. Baseline features

The organisers of the shared tasks on word-level MT QE at WMT15 [17] and WMT16 [18] provided the participants with a collection of baseline features obtained with the QuEst++ tool [44]. Some of these features were included in the experiments to evaluate whether any improvement could be obtained when combining them with the features described in Section 3.[16] The baseline features included in the evaluation are the following:

- Syntactic features:
  - Is the token a stop word?
  - Is the token a punctuation sign?
  - Is the token a proper noun?
  - Is the token a digit?
  - Part of speech of the current token
  - Part of speech of the SL token aligned with the current token
- Semantic features:
  - Number of alternative meanings of the current token (only available for WMT15 datasets)

[16]Some features, such as the immediate neighbour words to that for which predictions are produced, require a large amount of features to be represented, such as one-hot representations or word embedding. These features were discarded for the sake of the simplicity of the models built.

  - Number of alternative meanings of the SL token aligned with the current token (only available for WMT15 datasets)

- Language model (LM) features:
  - Longest n-gram seen by the TL LM with the current token as the last word
  - Longest n-gram seen by the TL LM with the current token as the first word
  - Backoff probability for the shortest n-gram not seen by the TL LM with the current token as the last word
  - Backoff probability for the shortest n-gram not seen by the TL LM with the current token as the first word
  - Backoff probability for the shortest n-gram not seen by the TL LM with the current token as the middle word
  - Longest n-gram seen by the SL LM with the SL token aligned to the current TL token as the last word
  - Longest n-gram seen by the SL LM with the SL token aligned to the current TL token as the first word

- Other features:
  - Number of tokens in the SL segment
  - Number of tokens in the TL segment
  - Ratio between the number of tokens in the SL and TL segments
  - Does the token appear in a given pseudo-reference? (only available for WMT15 datasets)

Note that all the features included in this list are either binary or numeric, with the exception of the part of speech of the SL and TL tokens, which are categorical. We dealt with these features by converting them into one-hot representations. The length of these one-hot representations was 50 and 57 for English–German, and 59 and 67 for English–Spanish. The difference in the number of features needed to encode part-of-speech tags together with the fact that the organisers of WMT provided some features in WMT15 that were not available for WMT16 (see the list above) lead to different amounts of baseline features for each language pair: 121 for English–German and 143 for English–Spanish.

5.5. Neural network parameters

Different configurations were tried using different numbers of neurons and hidden layers, optimisation algorithms, loss and activation functions and *dropout* values. Those producing the best results with the minimum number of parameters to be learned are described in this section.

The NNs described in Section 4 were implemented by using the Keras library [45].[17] Every NN contains in its hidden layers as many rectified linear units (ReLU; Nair and Hinton [46]) as the number of nodes in the input layer. A sigmoid activation function was chosen for the output node. The Adam [47] algorithm was used to optimise the binary cross-entropy cost function. A *dropout* of 20% was similarly set in order to minimise the risk of overfitting. The development set of each dataset was used to compute the error after each training epoch; the training process was stopped after 10 epochs without any improvement on the development set. The training was repeated 10 times for each NN with random uniform initialisations using the method defined by He et al. [48]; the model used was the one which provided the lowest error on the development set. After training, a *thresholding* strategy [49] was used to choose the threshold applied to the output node of each NN that provided the best results for the main evaluation metric: the F_1 score of the least frequent class in WMT15, and the product of the F_1 scores for both classes in WMT16. This tuning was also carried out on the development set by means of a line search.

[17]`http://www.keras.io`

Approach	**Class**	**Precision**	**Recall**	F_1**-score**	F_1**-product**
		English–Spanish			
Null hypothesis	keep	81.1%	50.0%	61.9%	14,8%
	delete	18.9%	50.1%	23.9%	
baseline	keep	88.2%	45.5%	60.0%	21.8%
	delete	24.0%	73.9%	36.2%	
SBI	keep	88.0%	70.8%	78.5%	33.2%
	delete	32.8%	59.5%	42.3%	
SBI+baseline	keep	88.1%	76.9%	82.1%	35.6%
	delete	35.7%	55.2%	**43.4%**	
WMT15 best	keep	89.1%	69.5%	78.1%	33.6%
	delete	32.6%	63.6%	43.1%	
		English–German			
Null hypothesis	keep	61.8%	80.6%	49.9%	25,0%
	delete	27.8%	19.3%	50.1%	
baseline	keep	87.2%	81.0%	84.0%	36.8%
	delete	38.8%	50.4%	43.9%	
SBI	keep	89.5%	64.2%	74.8%	31.5%
	delete	30.6%	67.8%	42.1%	
SBI+baseline	keep	87.6%	87.6%	87.6%	42.3%
	delete	48.2%	48.4%	48.3%	
WMT16 best	keep	90.1%	86.8%	88.5%	**49.6%**
	delete	52.3%	60.3%	56.0%	

Table 3: Results obtained for the task of identifying word deletions for English–Spanish and English–German. The table includes the results obtained when using an independent NN focused only on this task (see Section 4.1) and fed with the SBI features described in Section 3, the same NN using only the baseline features provided by the organisers of the MTQE shared task at WMT, the combination of both feature sets, the best performing systems at WMT15 [17] and WMT16 [18], and the null-hypothesis baseline.

5.6. Results and discussion

The following section contains the results obtained for each of the architectures proposed in Section 4 and compares the impact of taking context into account when predicting word deletions and insertions positions.

5.6.1. Predicting word deletions and insertion positions independently

Tables 3 and 4 show the results obtained when using two independent neural networks (see Section 4.1) to identify word deletions and insertion positions, respectively, both for English–Spanish and English–German. Table 3 includes the results of the approach described in Section 4.1, both when using only the baseline features described in Section 5.4 (baseline), the combination of features based on sources of bilingual information described in Section 3 (SBI), and when combining both types of features (SBI+baseline). The same NNs were used only with the baseline features in order to confirm the improvement of the combination of both feature sets. In addition to this, the results obtained with the different combinations of features are compared to both the results obtained by the best performing systems in both editions of the shared task and the null hypothesis described in Section 5.3 (the approach that uses only the a priori probabilities for each class). Note that the results in bold type are those that outperform those obtained by the rest of approaches with statistical significance of $p \leq 0.05$. Statistical significance was evaluated by using the approximate randomisation strategy described by Yeh [50].

Approach	**Class**	**Precision**	**Recall**	**F_1-score**	**F_1-product**
		English–Spanish			
Null hypothesis	no insert	86.0%	50.0%	63.2%	13.9%
	insert	14.1%	50.2%	22.0%	
SBI	no insert	90.5%	68.5%	78.0%	25.0%
	insert	22.5%	55.8%	**32.1%**	
		English–German			
Null hypothesis	no insert	75.5%	50.5%	60.2%	19.9%
	insert	24.6%	50.2%	33.1%	
SBI	no insert	79.4%	78.2%	78.8%	**29.0%**
	insert	36.0%	37.6%	36.8%	

Table 4: Results obtained for the task of identifying insertion positions for English–Spanish and English–German datasets. The table includes the results obtained when using an independent NN focused only on this task (see Section 4.1) and the null-hypothesis baseline.

As will be noted, our approach outperforms the null hypothesis in the case of both datasets and, in both cases, the approach that combines the SBI and the baseline features is better than that which uses only the SBI and the baseline features separately (in the case of word deletions, for which baseline features are available). In general, the results obtained by the SBI feature set is quite similar for English–Spanish and English–German. However, the baseline features lead to much better results in the case of English–German; this explains why the SBI+baseline combination for English–German leads to slightly better results than the same combination for English–Spanish. Focusing on the latter language combination, the results obtained for English–Spanish are comparable to those obtained by the best performing system in WMT15, a result which is reasonable given that both systems use a very similar approach, even though the approach presented in this paper uses less features (see Table 8) because it does not use the *negative features* originally proposed by Esplà-Gomis et al. [14]. In the case of English–German, the approach presented in this paper does not attain the performance of the best system in WMT16 and would rank third among the fourteen systems submitted to the shared task.

Table 4 shows the results of the approach with which to identify insertion positions described in Section 4.1. In the absence of other insertion prediction systems, it was only possible to compare it to the null hypothesis. For the same reason, no baseline features are available for this approach, and only the SBI-based features described in Section 3.2 could be used. In the case of both English–Spanish and English–German the proposed approach clearly outperforms the null hypothesis with a statistical significance of $p \leq 0.05$. It is worth noting that the results obtained when identifying insertion positions are worse than those obtained when identifying word deletions. This may indicate that the former problem is more difficult than the latter. We additionally evaluated the performance of this approach as regards both insertions that are related to a word deletion, that is, those that are the result of a replacement, and independent insertions; the recall obtained for both types of editions is almost the same, signifying that both tasks have a similar degree of difficulty.

5.6.2. Predicting word deletions and insertion positions taking context into account

Tables 5 and 6 show the results obtained following the cascaded-revision (see Section 4.2) and single-NN (see Section 4.3) strategies that use context. With regard to the cascaded-revision method, the two training approaches described in Section 4.2 were evaluated: the two-step training that first trains the independent networks and then builds on their predictions to train the cascaded-revision NNs, and the one which trains all the NNs simultaneously. For both approaches, the features used for word deletions were the same as in the SBI+baseline approach, while for insertion positions, the features employed by the SBI approach were used.

Approach	Class	Precision	Recall	F_1-score	F_1-product
		English–Spanish			
WMT15 best	keep	89.1%	69.5%	78.1%	33.6%
	delete	32.6%	63.6%	43.1%	
SBI+baseline	keep	88.1%	76.9%	82.1%	35.6%
	delete	35.7%	55.2%	**43.4%**	
Cascaded rev. 2-step training	keep	89.8%	73.0%	80.5%	35.2%
	delete	33.7%	62.3%	43.7%	
Cascaded rev. 1-step training	keep	89.2%	71.8%	79.5%	35.1%
	delete	34.0%	62.6%	44.1%	
Single NN	keep	90.1%	69.4%	78.5%	35.3%
	delete	33.8%	67.2%	**45.0%**	
		English–German			
SBI+baseline	keep	87.6%	87.6%	87.6%	42.3%
	delete	48.2%	48.4%	48.3%	
Cascaded rev. 2-step train	keep	89.7%	84.3%	86.9%	43.4%
	delete	44.7%	56.7%	50.0%	
Cascaded rev. 1-step train	keep	88.9%	84.5%	86.6%	43.8%
	delete	46.3%	52.9%	50.6%	
Single NN	keep	89.4%	84.7%	87.0%	45.5%
	delete	47.6%	58.2%	52.4%	
WMT16 best	keep	90.1%	86.8%	88.5%	**49.6%**
	delete	52.3%	60.3%	56.0%	

Table 5: Results obtained for the task of identifying word deletions for English–Spanish and English–German. The table includes the results obtained when using the cascaded-revision approach described in Section 4.2, both when training the networks in two steps and when doing so in a single step, and the single-NN approach described in Section 4.3. The shaded rows contain the results obtained by the best performing systems in Table 3 and have the objective of easing the comparison between the new results and the previous ones.

All the methods were evaluated using different values of context C and the experiments showed that values of C greater than 1 did not lead to better results. In general, for different values of C, the F_1-score and F_1-product metrics vary by about 0.5 percent and their differences are not statistically significant. This may be interpreted as an indication that only the immediately preceding and following edit operations are relevant to predict the current edit operation; operations that are more distant do not sufficiently influence such decisions. All the results in Tables 5 and 6 use, therefore, this level of context in order to reduce the complexity of the networks.

As can be seen, the methods using context outperform those focusing on a single word or insertion position. With regard to the results in Table 5, all the results provided outperform those obtained by the SBI+baseline approach with a statistical significance of $p \leq 0.05$. Namely, the approach that performed best was the one that used a single NN to predict both word deletions and insertion positions, which, for both datasets, obtained better results than a cascaded-revision with a statistically significant difference ($p \leq 0.05$). It is worth noting that in the case of the WMT16 dataset, none of the approaches suggested outperforms the best performing system in the shared task. However, the results obtained by the single NN approach do not show statistically significant differences with the method ranking second in the task (UNBABEL/linear), which attained an F_1-product of 46.3 [18, Table 21].

In the case of Table 6, which contains the results obtained for insertion positions, the conclusions are

Context size	**Class**	**Precision**	**Recall**	**F_1-score**	**F_1-product**
		English–Spanish			
SBI	no insert	90.5%	68.5%	78.0%	25.0%
	insert	22.5%	55.8%	32.1%	
Cascaded rev. 2-step training	no insert	91.8%	72.1%	80.7%	29.5%
	insert	26.2%	60.6%	36.6%	
Cascaded rev. 1-step training	no insert	91.9%	71.4%	80.4%	29.5%
	insert	26.1%	61.6%	36.6%	
Single NN	no insert	91.3%	78.2%	84.3%	31.9%
	insert	29.1%	54.5%	**37.9%**	
		English–German			
SBI	no insert	79.4%	78.2%	78.8%	29.0%
	insert	36.0%	37.6%	36.8%	
Cascaded rev. 2-step training	no insert	89.8%	82.9%	86.2%	34.2%
	insert	33.8%	48.1%	39.7%	
Cascaded rev. 1-step training	no insert	90.2%	80.9%	85.3%	34.3%
	insert	32.9%	51.5%	40.2%	
Single NN	no insert	91.0%	83.3%	87.0%	**38.5%**
	insert	37.3%	54.5%	44.3%	

Table 6: Results obtained for the task of identifying insertion positions for English–Spanish and English–German. The table includes the results obtained when using the cascaded-revision approach described in Section 4.2, both when training the networks in two steps and when doing so in a single step, and the single-NN approach described in Section 4.3. The shaded rows contain the results obtained by the best performing system (SBI) in Table 4 and have the objective of easing the comparison between the new results and the previous ones.

mostly the same. In this case, the results obtained with the two training strategies for the cascaded-revision approach are even closer, but are both outperformed by the single NN approach with a statistically significant difference of $p \leq 0.05$.

In general, it would appear obvious that using NNs that take into account the previous and following words and insertion positions to those being evaluated lead to substantially better results. In our evaluation we also considered the possibility of using a context containing only word deletions and only insertion positions. However, providing this context independently led to significantly worse results than those shown in tables 5 and 6. It therefore seems obvious that the combination of the information obtained for both types of editions helps to mutually improve their results.

The results obtained also indicate that the use of context is especially useful in the task of identifying insertion positions. This would appear to be reasonable, given that, according to the results obtained, the models trained to identify word deletions are more reliable than those trained to identify insertion positions and, therefore, the former help the latter more than in the opposite case.

Table 7 compares our best performing system for predicting word deletions (single NN) to the best approaches in the literature that, in spite of not having participated in the WMT15 an WMT16 shared tasks, have used these datasets for word-level MT QE. The results in this table confirm that the single NN approach clearly outperform the most basic approaches using deep NNs, such as those by Kreutzer et al. [29] and Liu et al. [30]. When compared to approaches based on much more complex neural architectures, such as the one by Martins et al. [33], which is based on the best performing system at WMT16, or the one by Kim et al. [35], the winner of WMT17, results are not that clear. On the WMT15 dataset, our approach outperforms the system by Kim et al. [35] and obtains results close to those by Martins et al. [33]. However, the distance to these two approaches becomes larger when we compare the results on the

Approach	WMT15 (F_1-score$_{\text{delete}}$)	WMT16 (F_1-product)
Kreutzer et al. [29]	43.1	—
Liu et al. [30]	38.0	—
Single NN (SBI)	45.0	45.5
Kim et al. [35]	42.7	50.1
Martins et al. [33]	**47.1**	**57.5**

Table 7: Results obtained for word deletion with our best-performing system (single NN architecture) and the best performing approaches in the literature evaluated on the WMT15 and WMT16 datasets.

Approach	# feat. del.	# feat. ins.	# parameters
Null hypothesis	0	0	1
SBI	51	41	4,468
SBI+baseline	172/194	–	31,693/39,789
WMT15 best	213	–	45,796
WMT16 best		not available	
Cascaded rev.	518/584	125	31,935/40,031
Single NN	516/582	123	441,931/492,201

Table 8: Number of features and parameters to be learned for each of the approaches discussed in Section 5.6. Note that two values are provided for the number of features and parameters for the SBI+baseline, Cascaded and Single NN approaches, because the number of baseline features available for the WMT15 and WMT16 is different.

WMT16 dataset. The fact that these two approaches lead to better results (at least for some datasets) is quite reasonable if one takes into account the extremely complex neural architectures described by their authors. It is worth noting that the approaches proposed in this paper require much less computational resources (see Section 5.6.3) and, still, they lead to results that are competitive when compared to the state of the art and even better than some much more complex and costly neural approaches.

5.6.3. Discussion regarding the performance of the approaches evaluated

For a more detailed analysis of the approaches compared in this section, Table 8 shows the total number of features used and the number of parameters to be learned by each of them. This allows us to discuss the complexity and computational cost of each approach compared to the results obtained. Please recall that the SBI+baseline approach can be computed only for word deletions, given that the baseline features are only available for this task. The same occurs with the best performing systems at WMT15 and WMT16, which were designed only to predict word deletions. In the case of the null hypothesis, no features are used and only the a priori probability of each class in the training data is computed.

It is worth noting that the number of baseline features provided for the English–Spanish (WMT15) dataset is slightly higher than that provided for the English–German (WMT16) dataset; as a result, two values are provided for the features and parameters of those approaches that use them, that is the SBI+baseline and the cascaded-revision and single NN approaches. Sixty baseline features are available for English–German, while this amount increases to 91 for Spanish–English, as defined in Section 5.4.

According to the data provided, it would appear that the cascaded-revision strategy (using the SBI and SBI+baseline collections of features) provides the best compromise between computational cost and performance. This is particularly noticeable when comparing the results obtained by this approach to the best performing systems at WMT15 and WMT16. In the first case, it outperforms the WMT15 system using less parameters, even when this approach is also learning to identify insertion positions, something that could not be done by the best-performing WMT15 system. In the case of the second, the details regarding the implementation of the best performing approach at WMT16 are not available. However, the description by Martins et al. [31] specifies that a combination of five instances of: (a) a convolutional recurrent network,

	training time on CPU		training time on GPU	
Approach	**per epoch**	**total**	**per epoch**	**total**
Independent NNs	9±1 s	6.5 min	4±1 s	4.0 min
Cascaded rev. 2-step	18±2 s	11.5 min	7±2 s	8.5 min
Cascaded rev. 1-step	45±2 s	17.0 min	23±1 s	10.5 min
Single NN	64±2 s	34.5 min	5±1 s	4.0 min

Table 9: Time per epoch and total time needed to train each NN architecture described in Section 4 on a CPU and on a GPU. All times are computed for the WMT15 dataset. For *Independent NNs*, we only include the time for word deletions, since it is the network that takes more time to train as it has much more features (194 vs. 41) and both networks can be trained in parallel.

(b) a bilingual recurrent language model, and (c) a feed-forward network, are used, summing 15 NNs in a voting scheme, which leads us to believe that this architecture requires tens, if not hundreds of millions, of parameters to be learned. On the other hand, the single NN approach proved to be the best performing of all the methods proposed in this paper, with the only exception of the winner of the WMT16 shared task. However, even though it must learn hundreds of thousands of parameters, the complexity of the NN proposed is still sufficiently simple for it to be trained on a standard CPU in a reasonable amount of time (see Table 9), something that would not be possible with any of the deep-learning approaches at WMT15 and WMT16.

Table 9 provides the actual time (per epoch and total amount) required to train the different NN architectures described in Section 4, both on a CPU[18] and on a GPU.[19] These results were obtained for the WMT15 dataset, the one using most features: 194 to identify word deletions and 41 to identify insertion positions. The training time shown for the independent NNs was only computed for word deletions, which is the most time-consuming network to train as it has almost five times more features than the independent NN used to identify insertion positions.

As expected, training on a GPU is appreciably faster (time is at least halved). When training on the CPU, time grows with the complexity of the networks. However, when training on a GPU, results vary slightly. In this last case, the cascaded-revision NNs training is the most time-consuming process. In the case of the two-step training, this is due to the fact that the independent NNs and the cascaded-revision NN have to be trained separately, which prevents the process to benefit from the high computational parallelisation provided by GPU. In the case of the one-step training process, the injection of error signals at two different levels of the neural network may be rendering the backpropagation calculation harder to parallelise. Finally, in the case of the single NN, it may be the opposite: grouping of tensor calculations in blocks seems to bring about a sharp speed-up.

In general, the results in Table 9 demonstrate that the approaches described in this work not only lead to competitive results, but are also feasible even with non-specialised computational resources.

6. Concluding remarks

In this work, we have presented a new method for word-level MT QE that partially builds on the approach by Esplà-Gomis et al. [13]. The results obtained confirm that this method makes it possible not only to identify the words in the output of an MT system that need to be deleted or replaced, as most word-level MT QE approaches do, but also to detect the positions into which one or more words need to be inserted. The latter is particularly relevant, given that, at the time of writing, this is the first work in the literature to tackle this problem.

This paper proposes a collection of features that builds on those defined by Esplà-Gomis et al. [13] and can be obtained from any source of bilingual information: in our experiments, online MT systems and an online

[18]An AMD Opteron(tm) Processor 6128, with 16 cores and 64 GB of RAM.
[19]A Geforce GTX 1080 Ti card with 11GB DDR5X.

bilingual concordancer were used. The results obtained on the datasets published for the word-level MT QE shared tasks at WMT15 and WMT16 confirm the good performance of the approach proposed, which is able to reproduce or even improve on the results obtained by Esplà-Gomis et al. [13] and Esplà-Gomis et al. [51]. The features used have, however, been redesigned to reduce their number, which has led to methods that require a lower computational cost. In addition to the features proposed, several NN architectures are explored for word-level MT QE: one that uses two independent NNs to predict word deletions and insertion positions, one that revises each prediction by taking into account the predictions made for the words and insertion positions surrounding it, and another that uses a single NN to predict both word deletions and insertion positions simultaneously. The experiments carried out confirm the relevance of the latter two approaches, that is, those using context. These results have led us to the conclusion that the simultaneous identification of both word deletions and insertion positions may lead to better results than those in the state of the art, in which only word deletions are identified.

The experiments carried out confirm the feasibility of the method proposed to identify insertion positions in T. These results are especially relevant, given that being capable of identifying both word deletions and insertion positions will allow the prediction of the full edit sequence required to post-edit a translation T, something that is not currently possible. This research paves the way towards the creation of systems that may support the task of professional translators by, for example, helping them to obtain reliable budgets based on the predicted technical effort[20] required for a given translation task. It would also be possible to provide metrics similar to fuzzy-match scores [11], a very popular and easy-to-interpret metric used by professional translators to measure the effort required to post-edit a suggestion from a translation memory in a computer-aided translation environment. It would even be possible to go one step further and build systems that could guide post-editors by indicating which parts of T require an action, as is done by Esplà-Gomis et al. [52] for translation memories.

With regard to improving the results obtained, one natural step would be to study other features used by other systems in the state of the art of word-level MT QE and attempt to adapt them in order to identify insertion positions.

Apart from this, one of the most obvious and promising next steps would be to adapt the techniques described in this work to the problem of sentence-level MT QE; that is, the task of predicting the total post-editing effort required for a sentence. In most shared tasks [9, 17, 18] this effort is measured using the human-targeted translation error rate (HTER) metric [53], which consists of identifying the number of deletions, insertions, substitutions and movements of sub-sequences of words (block shifts). Given that three[21] of these operations can be identified by our approach, it would be natural to attempt to apply it to this new task. It would even be possible to design new architectures and features that would make it possible to predict the fourth operation type used in HTER, that is, movements of sub-sequences of words.

Acknowledgements

Work funded by the Spanish Government through the EFFORTUNE project [project number TIN2015-69632-R]. We would particularly like to thank Reverso-Softissimo and Prompsit Language Engineering for providing access to the Reverso Context concordancer through their API, the University Research Program for Google Translate that granted us access to the Google Translate service, and Anna Civil from Lucy Software for producing sub-segment translations with the Lucy LT MT system. Felipe Sánchez-Martínez gratefully acknowledges the support of NVIDIA Corporation with the donation of the Titan X Pascal GPU used for this research.

[20]Technical effort may be predicted as the number of edit operations required to produce a post-edited translation. Other effort metrics could be explored, such as keystroke ratio or even post-editing time, although they would not be as straightforward to predict from edit operations.

[21]Actually two, but replacements can be straightforwardly obtained if we consider them as deletions followed by replacements or vice-versa.

Appendix A. Mathematical description of the neural networks used

This appendix contains the equations that describe the NNs used proposed in Section 4. Equations A.1 and A.2 describe how to obtain predictions y_k and z_k for the word t_k and the gap γ_k, respectively, using the NNs defined in Section 4.1.

$$y_k = \text{sigmoid}\left(\sum_{i=1}^{M} w_i^{yp}(p_k)_i + w_0^{yp}\right) \tag{A.1}$$

$$z_k = \text{sigmoid}\left(\sum_{i=1}^{N} w_i^{zq}(q_k)_i + w_0^{zq}\right) \tag{A.2}$$

where sigmoid$(\cdot)$ is the activation function defined as:

$$\text{sigmoid}(x) = \frac{\exp(x)}{\exp(x)+1}. \tag{A.3}$$

M and N are the total number of neurons in the hidden layer, $\vec{w}^{yp}$ and $\vec{w}^{zq}$ are the collection of weights learned in the output neuron for each of them, and $(p_k)_i$ and $(q_k)_i$ are defined as:

$$(p_k)_i = \text{ReLU}\left(\sum_{j=1}^{F} w_{ij}^{pf}(f_k)_j + w_{i0}^{pf}\right),\ i \in [1, M] \tag{A.4}$$

$$(q_k)_i = \text{ReLU}\left(\sum_{j=1}^{G} w_{ij}^{qg}(g_k)_j + w_{i0}^{qg}\right),\ i \in [1, N] \tag{A.5}$$

where ReLU$(\cdot)$ is an activation function [46] defined as:

$$\text{ReLU}(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases} \tag{A.6}$$

F and G are the total number of features for word deletions (f_k) and insertion positions (g_k), respectively, and $\vec{w}_i^{pf}$ and $\vec{w}_i^{qg}$ are the weights learned by the i-th neuron in the hidden layer of each of the NNs defined.

Similarly, the predictions produced by the cascaded-revision architecture NNs (see Section 4.2) would be defined as follows:

$$y'_k = \text{sigmoid}\left(\sum_{i=1}^{P} w_i^{y's}(s_k)_i + w_0^{y's}\right) \tag{A.7}$$

$$z'_k = \text{sigmoid}\left(\sum_{i=1}^{Q} w_i^{z'u}(u_k)_i + w_0^{z'u}\right) \tag{A.8}$$

where y'_k and z'_k are the reviewed predictions for the word t_k and the gap γ_k, respectively, P and Q are the number of neurons in each of the hidden layers, $\vec{w}^{y's}$ and $\vec{w}^{z'u}$ are the collection of weights learned in the output neuron for each of them, and

$$(s_k)_i = \text{ReLU}\left(\sum_{l=-C}^{l=+C} w_{il}^{sy} y_{k+l} + \sum_{l=-C}^{l=+C} w_{il}^{sz} z_{k+l} + w_{i0}^{s(yz)}\right),\ i \in [1, P] \tag{A.9}$$

$$(u_k)_i = \text{ReLU}\left(\sum_{l=-C}^{l=+C} w_{il}^{uy} y_{k+l} + \sum_{l=-C}^{l=+C} w_{il}^{uz} z_{k+l} + w_{i0}^{u(yz)}\right),\ i \in [1, Q] \tag{A.10}$$

In this case, $(s_k)_i$ and $(u_k)_i$ take the outputs of Equations A.1 and A.2 as inputs. Here, C is the size of the context used. It is worth noting the dependency on k as a result of parameter sharing.

Finally, Equations A.7 and A.8 could be adapted for the single NN defined in 4.3 as follows:

$$y'_k = \text{Sigmoid}\left(\sum_{i=1}^{H} w_i^{y'v}(v_k)_i + w_0^{y'v}\right) \tag{A.11}$$

$$z'_k = \text{Sigmoid}\left(\sum_{i=1}^{H} w_i^{z'v}(v_k)_i + w_0^{z'v}\right) \tag{A.12}$$

where H is the size of the common second-level hidden layer, and,

$$(v_k)_i = \text{ReLU}\left(\sum_{l=-C}^{l=+C}\sum_{j=1}^{M} w_{ijl}^{vp}(p_{k+l})_j + \sum_{l=-C}^{l=+C}\sum_{j=1}^{N} w_{ijl}^{vq}(q_{k+l})_j + w_{i0}^{v(pq)}\right), \; i \in [1, H] \tag{A.13}$$

where $(p_{k+l})_j$ and $(q_{k+l})_j$ are defined in Equations A.4 and A.5.

Appendix B. Pseudo-code for feature extraction

This appendix provides an algorithmic description of the six feature sets defined in Section 3. It is worth noting that, for the sake of clarity, an independent algorithm is provided for each feature set. The actual implementation is more efficient and does not compute each feature independently to avoid iterating over the same sets more than once. The following auxiliary functions, which were previously defined in Section 3, are used inside these algorithms:

- $\text{seg}_n(X)$: returns the set of all possible n-word sub-segments of segment X;
- $\text{seg}_*(X)$: returns the set of all possible sub-segments of segment X, regardless of length;
- $\text{span}(\tau, T)$: returns the set of word positions spanned by sub-segment τ in segment T;
- $\text{LCS}(X, Y)$: returns the word-based longest common sub-sequence between segments X and Y;
- $\text{occ}(\sigma, \tau, M^{\text{occ}})$: returns the number of occurrences of sub-segment pair (σ, τ) in M^{occ};
- $\text{editop}_1(X_j, X, Y)$: returns the edit operation assigned to the word X_j, obtained as a by-product of the computation of the longest-common subsequence of segments X and Y; and
- $\text{editop}_2(Y_j, X, Y)$: returns the edit operation assigned to the word Y_j, obtained as a by-product of the computation of the longest-common subsequence of segments X and Y.

References

[1] M. Plitt, F. Masselot, A Productivity Test of Statistical Machine Translation Post-Editing in a Typical Localisation Context, The Prague Bulletin of Mathematical Linguistics 93 (2010) 7–16.

[2] A. Guerberof Arenas, Productivity and quality in the post-editing of outputs from translation memories and machine translation, The International Journal of Localisation 7 (1) (2009) 11–21.

[3] L. Bowker, Computer-aided translation technology: a practical introduction, chap. Translation-memory systems, University of Ottawa Press, 92–127, 2002.

[4] H. Somers, Computers and translation: a translator's guide, chap. Translation memory systems, John Benjamins Publishing, Amsterdam, Netherlands, 31–48, 2003.

[5] L. Specia, M. Turchi, N. Cancedda, M. Dymetman, N. Cristianini, Estimating the Sentence-Level Quality of Machine Translation Systems, in: 13th Annual Conference of the European Association for Machine Translation, Barcelona, Spain, 28–37, URL `http://www.mt-archive.info/EAMT-2009-Specia.pdf`, 2009.

[6] J. Blatz, E. Fitzgerald, G. Foster, S. Gandrabur, C. Goutte, A. Kulesza, A. Sanchis, N. Ueffing, Confidence Estimation for Machine Translation, in: Proceedings of the 20th International Conference on Computational Linguistics, COLING '04, Geneva, Switzerland, 315–321, 2004.

Algorithm 1 Algorithm for the Keep$_n$ feature set (Section 3.1.1)

```
1: procedure KEEP($j$,$S$,$T$,$M$,$n$)
2:     Input:
3:         $S$: segment in SL;
4:         $T$: segment in TL;
5:         $j$: position of a word in $T$;
6:         $M$: collection of sub-segment pairs $(\sigma, \tau)$;
7:         $n$: length of $\tau$ in words
8:     Output:
9:         value of Keep$_n(j, S, T, M)$
10:    confirm_segs $\leftarrow 0$
11:    segs$_s$ $\leftarrow$ seg$_*(\sigma)$
12:    segs$_t$ $\leftarrow$ seg$_n(T)$
13:    for $(\sigma, \tau) \in M$ do
14:        if $\sigma \in$ segs$_s$ $\wedge$ $\tau \in$ segs$_t$ $\wedge$ $j \in$ span$(\tau, T)$ then
15:            confirm_segs $\leftarrow$ confirm_segs + 1
16:    total_segs $\leftarrow 0$
17:    for $\tau \in$ segs$_t$ do
18:        if $j \in$ span$(\tau, T)$ then
19:            total_segs $\leftarrow$ total_segs + 1
20:    return confirm_segs/total_segs
```

Algorithm 2 Algorithm for the Freq$_n^{\text{keep}}$ feature set (Section 3.1.2)

```
1: procedure FREQ$^{\text{keep}}$($j$,$S$,$T$,$M^{\text{occ}}$,$n$)
2:     Input:
3:         $S$: segment in SL;
4:         $T$: segment in TL;
5:         $j$: position of a word in $T$;
6:         $M^{\text{occ}}$: collection of sub-segment pairs
7:             and their number of occurrences $(\sigma, \tau, \phi)$;
8:         $n$: length of target sub-segment in words
9:     Output:
10:        value of Freq$_n^{\text{keep}}(j, S, T, M^{\text{occ}})$
11:    total_occs $\leftarrow 0$
12:    segs$_s$ $\leftarrow$ seg$_*(S)$
13:    segs$_t$ $\leftarrow$ seg$_n(T)$
14:    for $(\sigma, \tau) \in M^{\text{occ}}$ do
15:        if $\sigma \in$ segs$_s$ $\wedge$ $\tau \in$ segs$_t$ $\wedge$ $j \in$ span$(\tau, T)$ then
16:            confirm_occs $\leftarrow$ occ$(\sigma, \tau, M^{\text{occ}})$
17:            all_occs $\leftarrow 0$
18:            for $\tau' \in$ segs$_t$ do
19:                all_occs $\leftarrow$ all_occs + occ$(\sigma, \tau', M^{\text{occ}})$
20:            total_occs $\leftarrow$ total_occs + confirm_occs/all_occs
21:    return total_occs
```

Algorithm 3 Algorithm for the $\text{Align}_n^{\text{keep}}$ feature set (Section 3.1.3)

```
1: procedure ALIGN^keep($j$,$S$,$T$,$M$,$e$,$n$)
2:     Input:
3:         $S$: segment in SL;
4:         $T$: segment in TL;
5:         $j$: position of a word in $T$;
6:         $M$: collection of sub-segment pairs $(\sigma, \tau)$;
7:         $n$: length of target sub-segment in words;
8:         $e$: edit operation (either delete or match)
9:     Output:
10:        value of $\text{Align}_n^{\text{keep}}(j, S, T, M, e)$
11:    total_algs ← 0
12:    $\text{segs}_s \leftarrow \text{seg}_*(S)$
13:    for $(\sigma, \tau) \in M$ do
14:        if $\sigma \in \text{segs}_s \land |\tau| = n \land \text{editop}(t_j, T, \tau) = e$ then
15:            total_algs ← total_algs + $|\text{LCS}(\tau, T)|/|\tau|$
16:    return total_algs
```

Algorithm 4 Algorithm for the NoInsert_n feature set (Section 3.2.1)

```
1: procedure NOINSERT($j$,$S$,$T$,$M$,$n$)
2:     Input:
3:         $S$: segment in SL;
4:         $T$: segment in TL;
5:         $j$: position of a word in $T$;
6:         $M$: collection of sub-segment pairs $(\sigma, \tau)$;
7:         $n$: length of target sub-segment in words
8:     Output:
9:         value of $\text{NoInsert}_n(j, S, T, M)$
10:    confirm_segs ← 0
11:    $\text{segs}_s \leftarrow \text{seg}_*(S)$
12:    $\text{segs}_t \leftarrow \text{seg}_n(T)$
13:    for $(\sigma, \tau) \in M$ do
14:        if $\sigma \in \text{segs}_s \land \tau \in \text{segs}_t \land j \in \text{span}(t, T) \land j + 1 \in \text{span}(t, T)$ then
15:            confirm_segs ← confirm_segs + 1
16:    total_segs ← 0
17:    for $\tau \in \text{segs}_t$ do
18:        if $j \in \text{span}(\tau, T) \land j + 1 \in \text{span}(\tau, T)$ then
19:            total_segs ← total_segs + 1
20:    return confirm_segs/total_segs
```

Algorithm 5 Algorithm for the $\text{Freq}_n^{\text{noins}}$ feature set (Section 3.2.2)

```
1: procedure $\textsc{Freq}^{\text{noins}}(j,S,T,M^{\text{occ}},n)$
2:     Input:
3:         $S$: segment in SL;
4:         $T$: segment in TL;
5:         $j$: position of a word in $T$;
6:         $M^{\text{occ}}$: collection of sub-segment pairs
7:             and their number of occurrences $(\sigma, \tau, \phi)$;
8:         $n$: length of target sub-segment in words
9:     Output:
10:        value of $\text{Freq}_n^{\text{noins}}(j, S, T, M^{\text{occ}})$
11:    $\text{total\_occs} \leftarrow 0$
12:    $\text{segs}_s \leftarrow \text{seg}_*(S)$
13:    $\text{segs}_t \leftarrow \text{seg}_n(T)$
14:    for $(\sigma, \tau) \in M^{\text{occ}}$ do
15:        if $\sigma \in \text{segs}_s \wedge \tau \in \text{segs}_t \wedge j \in \text{span}(\tau, T) \wedge j + 1 \in \text{span}(\tau, T)$ then
16:            $\text{confirm\_occs} \leftarrow \text{occ}(\sigma, \tau, M^{\text{occ}})$
17:            $\text{all\_occs} \leftarrow 0$
18:            for $\tau' \in \text{segs}_t$ do
19:                $\text{all\_occs} \leftarrow \text{all\_occs} + \text{occ}(\sigma, \tau', M^{\text{occ}})$
20:            $\text{total\_occs} \leftarrow \text{total\_occs} + \text{confirm\_occs}/\text{all\_occs}$
21:    return total_occs
```

Algorithm 6 Algorithm for the $\text{Align}_n^{\text{noins}}$ feature set (Section 3.2.3)

```
1: procedure $\textsc{align}^{\text{noins}}(j,S,T,M,e,n)$
2:     Input:
3:         $S$: segment in SL;
4:         $T$: segment in TL;
5:         $j$: position of a word in $T$;
6:         $M$: collection of sub-segment pairs $(\sigma, \tau)$;
7:         $n$: length of target sub-segment in words;
8:         $e$: edit operation (either insert or match)
9:     Output:
10:        value of $\text{Align}_n^{\text{noins}}(j, S, T, M, e)$
11:    $\text{total\_algs} \leftarrow 0$
12:    $\text{segs}_s \leftarrow \text{seg}_*(S)$
13:    for $(\sigma, \tau) \in M$ do
14:        if $\sigma \in \text{segs}_s \wedge |\tau| = n \wedge \text{editop}_2(t_j, \tau, T) = e$ then
15:            $\text{total\_algs} \leftarrow \text{total\_algs} + |\text{LCS}(\tau, T)|/|\tau|$
16:    return total_algs
```

[7] M. L. Forcada, F. Sánchez-Martínez, A general framework for minimizing translation effort: towards a principled combination of translation technologies in computer-aided translation, in: Proceedings of the 18th Annual Conference of the European Association for Machine Translation, Antalya, Turkey, 27–34, 2015.

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

[9] 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, A. Tamchyna, Findings of the 2014 Workshop on Statistical Machine Translation, in: Proceedings of the Ninth Workshop on Statistical Machine Translation, Baltimore, MD, USA, 12–58, 2014.

[10] N. Ueffing, H. Ney, Word-Level Confidence Estimation for Machine Translation, Computational Linguistics 33 (1) (2007) 9–40, ISSN 0891-2017, URL `http://dx.doi.org/10.1162/coli.2007.33.1.9`.

[11] R. Sikes, Fuzzy Matching in Theory and Practice, Multilingual 18 (6) (2007) 39–43.

[12] O. Bojar, C. Buck, C. Callison-Burch, C. Federmann, B. Haddow, P. Koehn, C. Monz, M. Post, R. Soricut, L. Specia, Findings of the 2013 Workshop on Statistical Machine Translation, in: Proceedings of the Eighth Workshop on Statistical Machine Translation, Sofia, Bulgaria, 1–44, URL `http://www.aclweb.org/anthology/W13-2201`, 2013.

[13] M. Esplà-Gomis, F. Sánchez-Martínez, M. Forcada, UAlacant word-level machine translation quality estimation system at WMT 2015, in: Proceedings of the Tenth Workshop on Statistical Machine Translation, Lisbon, Portugal, 309–315, URL `http://aclweb.org/anthology/W15-3036`, 2015.

[14] M. Esplà-Gomis, F. Sánchez-Martínez, M. L. Forcada, Using on-line available sources of bilingual information for word-level machine translation quality estimation, in: Proceedings of the 18th Annual Conference of the European Association for Machine Translation, Antalya, Turkey, 19–26, 2015.

[15] M. Barlow, Parallel concordancing and translation, in: Proceedings of ASLIB Translating and the Computer 26, London, UK, 2004.

[16] L. Bowker, M. Barlow, Bilingual concordancers and translation memories: a comparative evaluation, in: Proceedings of the Second International Workshop on Language Resources for Translation Work, Research and Training at Coling 2004, Geneva, Switzerland, 70–79, 2004.

[17] O. Bojar, R. Chatterjee, C. Federmann, B. Haddow, M. Huck, C. Hokamp, P. Koehn, V. Logacheva, C. Monz, M. Negri, M. Post, C. Scarton, L. Specia, M. Turchi, Findings of the 2015 Workshop on Statistical Machine Translation, in: Proceedings of the Tenth Workshop on Statistical Machine Translation, Lisbon, Portugal, 1–46, 2015.

[18] O. Bojar, R. Chatterjee, C. Federmann, Y. Graham, B. Haddow, M. Huck, A. Jimeno Yepes, P. Koehn, V. Logacheva, C. Monz, M. Negri, A. Neveol, M. Neves, M. Popel, M. Post, R. Rubino, C. Scarton, L. Specia, M. Turchi, K. Verspoor, M. Zampieri, Findings of the 2016 Conference on Machine Translation, in: Proceedings of the First Conference on Machine Translation, Berlin, Germany, 131–198, URL `http://www.aclweb.org/anthology/W/W16/W16-2301`, 2016.

[19] S. Gandrabur, G. Foster, Confidence Estimation for Translation Prediction, in: Proceedings of the 7th Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4, CONLL '03, Edmonton, Canada, 95–102, 2003.

[20] N. Ueffing, H. Ney, Application of word-level confidence measures in interactive statistical machine translation, in: Proceedings of the 10th European Association for Machine Translation Conference "Practical applications of machine translation", Budapest, Hungary, 262–270, 2005.

[21] G. A. Miller, WordNet: A Lexical Database for English, Communications of the ACM 38 (11) (1995) 39–41.

[22] P. F. Brown, V. J. D. Pietra, S. A. D. Pietra, R. L. Mercer, The Mathematics of Statistical Machine Translation: Parameter Estimation, Computational Linguistics 19 (2) (1993) 263–311.

[23] F. J. Och, H. Ney, The Alignment Template Approach to Statistical Machine Translation, Computational Linguistics 30 (4) (2004) 417–449.

[24] E. Biçici, Referential Translation Machines for Quality Estimation, in: Proceedings of the 8th Workshop on Statistical Machine Translation, Sofia, Bulgaria, 343–351, 2013.

[25] E. Biçici, D. Yuret, Instance selection for machine translation using feature decay algorithms, in: Proceedings of the 6th Workshop on Statistical Machine Translation, Edinburgh, UK, 272–283, 2011.

[26] F. Blain, C. Scarton, L. Specia, Bilexical embeddings for quality estimation, in: Proceedings of the Second Conference on Machine Translation, 545–550, 2017.

[27] P. S. Madhyastha, X. Carreras Pérez, A. Quattoni, Learning task-specific bilexical embeddings, in: Proceedings of the 25th International Conference on Computational Linguistics, 161–171, 2014.

[28] O. Bojar, R. Chatterjee, C. Federmann, Y. Graham, B. Haddow, S. Huang, M. Huck, P. Koehn, Q. Liu, V. Logacheva, C. Monz, M. Negri, M. Post, R. Rubino, L. Specia, M. Turchi, Findings of the 2017 Conference on Machine Translation (WMT17), in: Proceedings of the Second Conference on Machine Translation, Volume 2: Shared Task Papers, Association for Computational Linguistics, Copenhagen, Denmark, 169–214, URL `http://www.aclweb.org/anthology/W17-4717`, 2017.

[29] J. Kreutzer, S. Schamoni, S. Riezler, QUality Estimation from ScraTCH (QUETCH): Deep Learning for Word-level Translation Quality Estimation, in: Proceedings of the Tenth Workshop on Statistical Machine Translation, Association for Computational Linguistics, Lisbon, Portugal, 316–322, URL `http://aclweb.org/anthology/W15-3037`, 2015.

[30] L. Liu, A. Fujita, M. Utiyama, A. Finch, E. Sumita, L. Liu, A. Fujita, M. Utiyama, A. Finch, E. Sumita, Translation quality estimation using only bilingual corpora, IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP) 25 (9) (2017) 1762–1772.

[31] A. F. T. Martins, R. Astudillo, C. Hokamp, F. Kepler, Unbabel's Participation in the WMT16 Word-Level Translation Quality Estimation Shared Task, in: Proceedings of the First Conference on Machine Translation, Berlin, Germany, 806–811, URL `http://www.aclweb.org/anthology/W16-2387`, 2016.

[32] D. Wang, N. S. Chaudhari, Binary neural network training algorithms based on linear sequential learning, International

journal of neural systems 13 (5) (2003) 333–351.

[33] A. F. Martins, M. Junczys-Dowmunt, F. N. Kepler, R. Astudillo, C. Hokamp, R. Grundkiewicz, Pushing the limits of translation quality estimation, Transactions of the Association for Computational Linguistics 5 (2017) 205–218.

[34] H. Kim, H.-Y. Jung, H. Kwon, J.-H. Lee, S.-H. Na, Predictor-Estimator: Neural Quality Estimation Based on Target Word Prediction for Machine Translation, ACM Transactions on Asian and Low-Resource Language Information Processing (TALLIP) 17 (1) (2017) 3.

[35] H. Kim, J.-H. Lee, S.-H. Na, Predictor-estimator using multilevel task learning with stack propagation for neural quality estimation, in: Proceedings of the Second Conference on Machine Translation, 562–568, 2017.

[36] M. Esplà-Gomis, F. Sánchez-Martínez, M. L. Forcada, Using machine translation in computer-aided translation to suggest the target-side words to change, in: Proceedings of the Machine Translation Summit XIII, Xiamen, China, 172–179, 2011.

[37] R. O. Duda, P. E. Hart, D. G. Stork, Pattern Classification, John Wiley and Sons Inc., second edn., 2000.

[38] K. Hornik, M. Stinchcombe, H. White, Multilayer Feedforward Networks Are Universal Approximators, Neural Networks 2 (5) (1989) 359–366.

[39] P. Geurts, D. Ernst, L. Wehenkel, Extremely randomized trees, Machine Learning 63 (1) (2006) 3–42, ISSN 1573-0565, doi:\bibinfo{doi}{10.1007/s10994-006-6226-1}, URL `https://doi.org/10.1007/s10994-006-6226-1`.

[40] B. Scholkopf, K.-K. Sung, C. J. Burges, F. Girosi, P. Niyogi, T. Poggio, V. Vapnik, Comparing support vector machines with Gaussian kernels to radial basis function classifiers, IEEE transactions on Signal Processing 45 (11) (1997) 2758–2765.

[41] V. Chvátal, D. Sankoff, Longest common subsequences of two random sequences, Journal of Applied Probability 12 (2) (1975) 306–315.

[42] R. Wagner, M. Fischer, The String-to-String Correction Problem, Journal of the ACM 21 (1) (1974) 168–173.

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

[44] L. Specia, G. Paetzold, C. Scarton, Multi-level Translation Quality Prediction with QuEst++, in: Proceedings of ACL-IJCNLP 2015 System Demonstrations, Beijing, China, 115–120, URL `http://www.aclweb.org/anthology/P15-4020`, 2015.

[45] F. Chollet, et al., Keras, `https://github.com/fchollet/keras`, 2015.

[46] V. Nair, G. E. Hinton, Rectified linear units improve restricted boltzmann machines, in: Proceedings of the 27th international conference on machine learning, Haifa, Israel, 807–814, 2010.

[47] D. P. Kingma, J. Ba, Adam: A Method for Stochastic Optimization, in: Proceedings of the 3rd International Conference for Learning Representations, San Diego, 807–814, 2015.

[48] K. He, X. Zhang, S. Ren, J. Sun, Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification, in: Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV), ICCV '15, Washington, DC, USA, ISBN 978-1-4673-8391-2, 1026–1034, URL `http://dx.doi.org/10.1109/ICCV.2015.123`, 2015.

[49] Z. C. Lipton, C. Elkan, B. Naryanaswamy, Optimal Thresholding of Classifiers to Maximize F1 Measure, Springer Berlin Heidelberg, Berlin, Heidelberg, 225–239, 2014.

[50] A. Yeh, More Accurate Tests for the Statistical Significance of Result Differences, in: Proceedings of the 18th Conference on Computational Linguistics - Volume 2, COLING '00, Stroudsburg, PA, USA, 947–953, URL `https://doi.org/10.3115/992730.992783`, 2000.

[51] M. Esplà-Gomis, M. Forcada, S. Ortiz Rojas, J. Ferrández-Tordera, Bitextor's participation in WMT'16: shared task on document alignment, in: Proceedings of the First Conference on Machine Translation, Berlin, Germany, 685–691, URL `http://www.aclweb.org/anthology/W/W16/W16-2367`, 2016.

[52] M. Esplà-Gomis, F. Sánchez-Martínez, M. L. Forcada, Target-Language Edit Hints in CAT Tools Based on TM by Means of MT, Journal of Artificial Intelligence Research 53 (2015) 169–222.

[53] M. Snover, B. Dorr, R. Schwartz, L. Micciulla, J. Makhoul, A study of translation edit rate with targeted human annotation, in: Proceedings of Association for Machine Translation in the Americas, vol. 200, Cambridge, MA, USA, 2006.