Boosting Neural Machine Translation with Similar Translations

Jitao Xu, Josep-Maria Crego, Jean Senellart

Boosting Neural Machine Translation with Similar Translations

Jitao Xu[†‡], **Josep Crego**[†], **Jean Senellart**[†]

[†]SYSTRAN, 5 rue Feydeau, 75002 Paris, France
firstname.lastname@systrangroup.com
[‡]Université Paris-Saclay, CNRS, LIMSI, 91400, Orsay, France
jitao.xu@limsi.fr

Abstract

This paper explores data augmentation methods for training Neural Machine Translation to make use of similar translations, in a comparable way a human translator employs fuzzy matches. In particular, we show how we can simply feed the neural model with information on both source and target sides of the fuzzy matches, we also extend the similarity to include semantically related translations retrieved using distributed sentence representations. We show that translations based on fuzzy matching provide the model with "*copy*" information while translations based on embedding similarities tend to extend the translation "*context*". Results indicate that the effect from both similar sentences are adding up to further boost accuracy, are combining naturally with model fine-tuning and are providing dynamic adaptation for unseen translation pairs. Tests on multiple data sets and domains show consistent accuracy improvements. To foster research around these techniques, we also release an Open-Source toolkit with efficient and flexible fuzzy-match implementation.

1 Introduction

For decades, the localization industry has been proposing Fuzzy Matching technology in CAT tools allowing the human translator to visualize one or several fuzzy matches from translation memory when translating a sentence leading to higher productivity and consistency (Yamada, 2011). Hence, even though the concept of fuzzy match scores is not standardized and differs between CAT tools (Bloodgood and Strauss, 2014), translators generally accept discounted translation rate for sentences with "high" fuzzy matches[1]. With improving machine translation technology and training of models on translation memories, machine translated output has been progressively introduced as a substitute for fuzzy matches when no sufficiently "good" fuzzy match is found and proved to also increase translator productivity given appropriate post-editing environment (Plitt and Masselot, 2010).

These two technologies are entirely different in their finality - indeed, for a given source sentence, fuzzy matching is just a database retrieval and scoring technique always returning a pair of source and target segments, while machine translation is actually building an original translation. However, with Statistical Machine Translation, the two technologies are sharing the same simple idea about managing and retrieving optimal combination of longest translated n-grams and this property led to the development of several techniques like use of fuzzy matches in SMT decoding (Koehn and Senellart, 2010; Wang et al., 2013), adaptive machine translation (Zaretskaya et al., 2015) or "fuzzy match repairing" (Ortega et al., 2016).

With Neural Machine Translation (NMT), the integration of Fuzzy Matching is less obvious since NMT does not keep nor build a database of aligned sequences and does not explicitly use n-gram language models for decoding. The only obvious and important use of translation memory is to use them to train an NMT model from scratch or to adapt a generic translation model to a specific domain (fine-tuning) (Chu and Wang, 2018). While some works propose architecture changes (Zhang et al., 2018) or decoding constraints (Gu et al., 2018); a recent work (Bulté and Tezcan, 2019; Bulté et al., 2018) has proposed a simple and elegant framework where, like for human translation, translation of fuzzy matches are presented simultaneously with source sentence and the network learns to use this additional information. Even though this method has showed huge gains in quality, it also opens

[1] https://signsandsymptomsoftranslation.com/2015/03/06/fuzzy-matches/.

Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 1580–1590

many questions.

In this work, we are pushing the concept further a) by proposing and evaluating new integration methods, b) by extending the notion of similarity and showing that fuzzy matches can be extended to embedding-based similarities, c) by analyzing how online fuzzy matching compares and combines with offline fine-tuning. Finally, our results also show that introducing similar sentence translation is helping NMT by providing sequences to copy (*copy effect*), but also providing additional context for the translation (*context effect*).

2 Translation Memories and NMT

A translation memory (TM) is a database that stores translated segments composed of a source and its corresponding translations. It is mostly used to match up previous translations to new content that is similar to content translated in the past.

Assuming that we translated the following English sentence into French: *[How long does the flight last?]* ⇝ *[Combien de temps dure le vol?]*. Both the English sentence and the corresponding French translation are saved to the TM. This way, if the same sentence appears in a future document (an *exact match*) the TM will suggest to reuse the translation that has just been saved. In addition to exact matches, TMs are also useful with *fuzzy matches*. These are useful when a new sentence is similar to a previously translated sentence, but not identical. For example, when translating the input sentence: *[How long does a cold last?]*, the TM may also suggest to reuse the previous translation since only two replacements (*a cold* by *the flight*) are needed to achieve a correct translation. TMs are used to reduce translation effort and to increase consistency over time.

2.1 Retrieving Similar Translations

More formally, we consider a TM as a set of K sentence pairs $\{(s_k, t_k) : k = 1, \ldots, K\}$ where s_k and t_k are mutual translations. A TM must be conveniently stored so as to allow fast access to the pair (s_k, t_k) that shows the highest similarity between s_k and any given new sentence. Many methods to compute sentence similarity have been explored, mainly falling into two broad categories: *lexical matches* (*i.e.* fuzzy match) and *distributional semantics*. The former relies on the number of overlaps between the sentences taken into account. The latter counts on the generalisation power of neural networks when building vector representations. Next, we describe the similarity measures employed in this work.

Fuzzy Matching Fuzzy matching is a lexicalised matching method aimed to identify non-exact matches of a given sentence. We define the fuzzy matching score $FM(s_i, s_j)$ between two sentences s_i and s_j as:

$$FM(s_i, s_j) = 1 - \frac{ED(s_i, s_j)}{max(|s_i|, |s_j|)}$$

where $ED(s_i, s_j)$ is the Edit Distance between s_i and s_j, and $|s|$ is the length of s. Many variants have been proposed to compute the edit distance, generally performed on normalized sentences (ignoring for instance case, number, punctuation, space or inline tags differences that are typically handled at a later stage). Also, IDF and stemming techniques are used to give more weight on significant words or less weight on morphological variants (Vanallemeersch and Vandeghinste, 2015; Bloodgood and Strauss, 2014).

Since we did not find an efficient TM fuzzy match library, we implemented an efficient and parameterizable algorithm in C++ based on suffix-array (Manber and Myers, 1993) that we open-sourced[2]. Fuzzy matching offers a great performance under large overlapping conditions. However, in some cases, sentences with large overlaps may receive low FM scores. Consider for instance the input: *[How long does the flight arriving in Paris from Barcelona last?]* and the TM entry of our previous example: *[How long does the flight last?]* ⇝ *[Combien de temps dure le vol?]*. Even though the TM entry may be of great help when translating the input sentence, it receives a low score ($1 - \frac{5}{12} = 0.583$) because of the multiple insertion/deletion operations needed. We thus introduce a second lexicalised similarity measure that focuses on finding the longest of n-gram overlap between sentences.

[2] https://github.com/systran/FuzzyMatch

N-gram Matching[3] We define the N-gram matching score $NM(s_i, s_j)$ between s_i and s_j:

$$NM(s_i, s_j) = \left| max\Big(\{\mathcal{S}(s_i) \cap \mathcal{S}(s_j)\}\Big) \right|$$

where $\mathcal{S}(s)$ denotes the set of n-grams in sentence s, $max(q)$ returns the longest n-gram in the set q and $|r|$ is the length of the n-gram r. For N-gram matching retrieval we also use our in-house open-sourced toolkit.

Distributed Representations The current research on sentence similarity measures has made tremendous advances thanks to distributed word representations computed by neural nets. In this work, we use `sent2vec`[4] (Pagliardini et al., 2018) to generate sentence embeddings. The network implements a simple but efficient unsupervised objective to train distributed representations of sentences. The authors claim that the algorithm performs state-of-the-art sentence representations on multiple benchmark tasks in particular for unsupervised similarity evaluation.

We define the similarity score $EM(s_i, s_j)$ between sentences s_i and s_j via cosine similarity of their distributed representations h_i and h_j:

$$EM(s_i, s_j) = \frac{h_i \cdot h_j}{||h_i|| \times ||h_j||}$$

where $||h||$ denotes the magnitude of vector h.

To implement fast retrieval between the input vector representation and the corresponding vector of sentences in the TM we use the `faiss`[5] toolkit (Johnson et al., 2019).

2.2 Related Words in TM Matches

Given an input sentence s, retrieving TM matches consists of identifying the TM entry (s_k, t_k) for which s_k shows the highest matching score. However, with the exception of perfect matches, not all words in s_k or s are present in the match. Considering the example in Section 2, the words *the flight* and *a cold* are not related to each other, from that follows that the TM target words *le vol* are irrelevant for the task at hand. In this section we discuss an algorithm capable of identifying the set of target words $\mathcal{T} \in t_k$ that are related to words of the input sentence s. Thus, we define the set $\mathcal{T}$ as:

$$\mathcal{T} = \left\{ \begin{array}{ll} t \in t_k : & \\ \quad \exists s \in \mathcal{S} \mid (s,t) \in \mathcal{A} & \\ \wedge \quad \forall s \notin \mathcal{S} \mid (s,t) \notin \mathcal{A} & \end{array} \right\}$$

where $\mathcal{A}$ is the set of word alignments between words in s_k and t_k and $\mathcal{S}$ is the LCS (Longest Common Subsequence) set of words in s_k and s. The LCS is computed as a by-product of the edit distance (Paterson and Dančík, 1994).

$\mathcal{S}$ is found as a sub-product of computing fuzzy or n-gram matches. Word alignments are performed by `fast_align`[6](Dyer et al., 2013). Figure 1 illustrates the alignments and LCS words between input sentences and their corresponding fuzzy (top) and N-gram (bottom) matches.

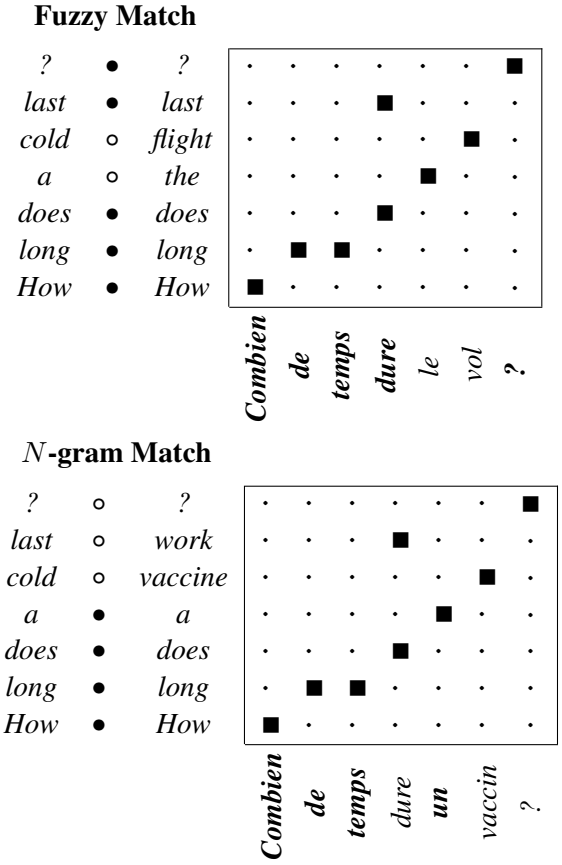

Figure 1: English-French TM entries with corresponding word alignments (right) and LCS of words with the input sentence (left). Matches are found following Fuzzy (top) and N-gram (bottom) techniques.

The TM source sentence s_k of the fuzzy matching example has a LCS set of 5 words $\mathcal{S}$ =

[3] Note that this practice is also called "subsequence" or "chunk" matching in CAT tools and is usually combined with source-target alignment in order to help human translators easily find translation fragments.

[4] https://github.com/epfml/sent2vec

[5] https://github.com/facebookresearch/faiss

[6] https://github.com/clab/fast_align

$\{How, long, does, last, ?\}$. The set of related target words $\mathcal{T}$ is also composed of 5 words $\{Combien, de, temps, dure, ?\}$, all aligned to at least one word in $\mathcal{S}$ and to no other word. The N-gram match example has a LCS set of 4 words $\mathcal{S}$ = $\{How, long, does, a\}$, while related target words consists of $\mathcal{T}$ = $\{Combien, de, temps, un\}$. The target word *dure* is not part of $\mathcal{T}$ as it is aligned to *work* and *work* $\notin \mathcal{S}$. Notice that sets $\mathcal{S}$ and $\mathcal{T}$ consist of collections of indices (word positions in their corresponding sentences) while word strings are used in the previous examples to facilitate reading.

2.3 Integrating TM into NMT

We retrieve fuzzy, n-gram and sentence embedding matches as detailed in the previous section. We explore various ways to integrate matches in the NMT workflow. We follow the work by (Bulté and Tezcan, 2019) where the input sentence is augmented with the translation retrieved from the TM showing the highest matching score (FM, NM or EM). One special integration of fuzzy matching, denoted FM_T, is rescoring fuzzy matches based on the *target edit distance*. This special integration, that is only performed on training data, is discussed in the **Target Fuzzy matches** section.

Figure 2 illustrates the main integration techniques considered in this work and detailed below. The input English sentence *[How long does the flight last?]* is differently augmented. For each alternative we show: the TM (English) sentence producing the match; the augmented input sentence with the corresponding TM (French) translation. Note that LCS words are displayed in boldface.

FM$^{\#}$ We implement the same format as detailed in (Bulté and Tezcan, 2019). The input English sentence is concatenated with the French translation with the (highest-scored) fuzzy match as computed by $FM(s_i, s_j)$. The token || is used to mark the boundary between both sentences.[7]

FM* We modify the previous format by masking the French words that are not related to the input sentence. Thus, sequences of unrelated tokens are replaced by the || token. The mechanism to identify relevant words is detailed in Section 2.2.

FM^{+} As a variant of FM*, we now mark target words which are not related to the input sentence in an attempt to help the network identify those target words that need to be copied in the hypothesis. However, we use an additional input stream (also called *factors*) to let the network access to the entire target sentence. Tokens used by this additional stream are: **S** for source words; **R** for unrelated target words and **T** for related target words.

NM^{+} In addition to fuzzy matches, we also consider arbitrary large n-gram matching. Thus, we use the same format as for FM^{+} but considering the highest scored n-gram match as computed by $NM(s_i, s_j)$.

EM^{+} Finally, we also retrieve the most similar TM sentences as computed by $EM(s_i, s_j)$. In this case, marking the words that are not related to the input sentence is not necessary since similar sentences retrieved following EM score do not necessarily present any lexical overlap. Note from the example in Table 2 that similar sentences retrieved with distributed representations may contain many word reorderings or synonyms (*i.e.*: *duration* − *last* or *flu* − *cold*) that makes it difficult to align both sentences. Hence, the same format employed for FM can be used here. However, since we plan to combine different kind of matches in a single model we adopt the format employed by NM^{+} and FM^{+} with a new factor label **E**.

FM$^{\#}$	*How long does the flight last ?*
How long does *a cold* ***last ?*** || *Combien de temps dure le vol ?*	
FM*	*How long does the flight last ?*
How long does *a cold* ***last ?*** || *Combien de temps dure* || *?*	
FM^{+}	*How long does the flight last ?*
How long does *a cold* ***last ?*** || *Combien de temps dure le vol ?*	
S S S S S S S R T T T T R R T	
NM^{+}	*How long does a vaccine work ?*
How long does a *cold last ?* || *Combien de temps dure un vaccin ?*	
S S S S S S S R T T T R T R R	
EM^{+}	*What is the duration of flu symptoms ?*
How long does a cold last ? || *Quelle est la durée de la grippe ?*	
S S S S S S S E E E E E E E E E	

Figure 2: Input sentence augmented with different TM matches: FM$^{\#}$ (Bulté and Tezcan, 2019), FM*, FM^{+} and EM^{+}.

[7] The original paper uses '@@@' as break token. We made sure that || was not part of the vocabulary.

3 Experimental Framework

3.1 Corpora and Evaluation

We used the following corpora in this work[8] (Tiedemann, 2012): Proceedings of the European Parliament (EPPS); News Commentaries (NEWS); TED talk subtitles (TED); Parallel sentences extracted from Wikipedia (Wiki); Documentation from the European Central Bank (ECB); Documents from the European Medicines Agency (EMEA); Legislative texts of the European Union (JRC); Localisation files (GNOME, KDE4 and Ubuntu) and Manual texts (PHP). Detailed statistics about these are provided in Appendix A. We randomly split the corpora by keeping 500 sentences for validation, 1,000 sentences for testing and the rest for training. All data is preprocessed using the OpenNMT tokenizer[9] (conservative mode). We train a 32K joint byte-pair encoding (BPE) (Sennrich et al., 2016b) and use a joint vocabulary for both source and target.

Our NMT model follows the state-of-the-art Transformer base architecture (Vaswani et al., 2017) implemented in the `OpenNMT-tf`[10] toolkit (Klein et al., 2017). Further configuration details are given in Appendix B.

3.2 TM Retrieval

We perform fuzzy matching, ignoring exact matches, and keep the single best match if $FM(s_i, s_j) \geq 0.6$ with no approximation. Similarly, the largest N-gram match is used for each test sentence with a threshold $NM(s_i, s_j) \geq 5$. A similarity threshold $EM(s_i, s_j) \geq 0.8$ is also employed when retrieving similar sentences using distributed representations. The `EM` model is trained on the source training data with default *fasttext* params on 200 dimension, and 20 epochs.

Algorithm	Indexing (s)	Retrieval (word/s)
FM	546	607
NM	546	40,888
EM	181+342	4,142

Table 1: Indexing and retrieval time for the different matching algorithm run on single thread Intel Core i7, 2.8GHz. EM index time is the sum of embedding building for the 2M sentences and *faiss* index building.

[8] Freely available from `http://opus.nlpl.eu`

[9] `https://github.com/OpenNMT/Tokenizer`

[10] `https://github.com/OpenNMT/OpenNMT-tf`

The *faiss* search toolkit is used through python API with exact *FlatIP* index. Building and retrieval times for each algorithm on a 2M sentences translation memory (Europarl corpus) are provided in Table 1. Note that all retrieval algorithms are significantly faster than NMT Transformer decoding, thus, implying a very limited decoding overhead.

4 Results

We compare our baseline model, without augmenting input sentences, to different augmentation formats and retrieval methods. Our `base` model is built using the concatenation of all the original corpora. All other models extend the original corpora with sentences retrieved following various retrieval methods. It is worth to notice that extended bitexts share the target side with the original data.

Individual comparison of Matching algorithms and Augmentation methods In this experiment, all corpora are used to build the models while matches of a given domain are retrieved from the training data of this domain. Models are built using the original source and target training data (`base`), and after augmenting the source sentence as detailed in Section 2.3: $\texttt{FM}^{\#}$, $\texttt{FM}^{\#}_{T}$, $\texttt{FM}^{*}$, $\texttt{FM}^{+}$, $\texttt{NM}^{+}$ and $\texttt{EM}^{+}$. Test sentences are augmented following the same technique as for training sentences[11]. Table 2 summarises the results that are divided in three blocks, showing results for the three types of matching studied in this work (`FM`, `NM` and `EM`).

Best scores are obtained by models using augmented inputs except for corpora not suited for translation memory usage: News, TED for which we observe no gains correlated to low matching rates. For the other corpora, large gains are achieved when evaluating test sentences with matches (up to +19 BLEU on GNOME corpus), while a very limited decrease in performance is observed for sentences that do not contain matches. This slight decrease is likely to come from the fact that we kept the corpus size and number of iterations identical while giving harder training tasks. Results are totally on par with the findings of (Bulté and Tezcan, 2019).

All types of matching indicate their suitability showing accuracy gains. In particular for fuzzy matching, which seems to be the best for our task. Among the different techniques used to insert fuzzy matching, $\texttt{FM}^{+}$ obtains the best results, validating

[11] Except for $\texttt{FM}^{\#}_{T}$ for which we use $\texttt{FM}^{\#}$ test set

Model	News	TED	ECB	EMEA	JRC	GNOME	KDE4	PHP	Ubuntu	Avg
%FM	*3.1%*	*10.3%*	*49.8%*	*69.8%*	*50.1%*	*59.7%*	*47.3%*	*41.0%*	*23.3%*	–
base	**37.16**	43.23	49.19	50.14	59.19	51.14	50.16	30.24	45.52	47.94
			57.69 - **41.95**	54.88 - 44.10	66.34 - 52.84	55.80 - **47.92**	53.05 - **48.77**	42.19 - 25.25	56.05 - 42.27	
FM#	36.68	42.93	55.15	61.16	66.35	61.82	54.37	33.10	48.26	54.32
			69.79 - 41.54	70.87 - 43.53	80.46 - 53.55	73.61 - 45.83	65.57 - 47.85	47.04 - 26.08	66.72 - 42.08	
FM#T	36.79	43.14	55.41	60.32	66.41	62.01	53.65	33.22	**49.75**	54.40
			70.46 - 41.41	68.63 - **44.90**	80.57 - 53.57	74.05 - 45.58	64.77 - 47.20	46.31 - 26.30	**69.16** - **43.32**	
FM*	36.44	**43.27**	54.52	59.49	65.24	59.54	53.30	32.77	48.74	53.37
			68.43 - 41.68	67.64 - 44.85	77.59 - **54.10**	70.16 - 45.19	62.63 - 48.00	44.50 - **26.31**	68.34 - 42.20	
FM+	37.12	42.62	**56.18**	**61.97**	**66.91**	**62.68**	**54.59**	**33.81**	48.62	**54.97**
			72.26 - 41.25	**71.52** - 44.72	**81.58** - 53.62	**74.99** - 45.83	**65.95** - 48.01	**47.74** - 26.27	67.49 - 42.37	
%NM	*45.5%*	*36.9%*	*69.9%*	*60.4%*	*69.6%*	*31.1%*	*22.9%*	*33.7%*	*14.1%*	–
base	**37.16**	**43.23**	49.19	50.14	59.19	51.14	50.16	30.24	45.52	47.94
			49.97 - **46.44**	50.94 - **47.43**	60.32 - **55.70**	53.86 - **46.59**	54.16 - **45.89**	34.64 - **26.88**	58.29 - **40.68**	
NM+	36.74	43.07	**55.40**	**59.17**	**65.60**	**58.46**	**51.54**	**31.87**	**46.16**	**52.60**
			58.65 - 44.06	**62.69** - 46.60	**69.24** - 54.32	**70.05** - 42.21	**59.87** - 42.11	**39.35** - 26.10	**63.22** - 39.59	
base	**37.16**	**43.23**	49.19	50.14	59.19	51.14	50.16	30.24	45.52	47.94
			52.09 - 40.74	52.07 - 40.08	62.60 - 48.16	54.20 - **45.88**	51.62 - **48.60**	42.22 - **21.42**	52.20 - 41.82	
EM+	36.50	42.89	**54.02**	**56.41**	**66.04**	**58.07**	**53.70**	**32.37**	**49.88**	**52.93**
			58.52 - **40.86**	**59.47** - **40.16**	**71.45** - **48.33**	**66.09** - 44.06	**59.43** - 47.43	**46.91** - 20.96	**62.04** - **43.20**	

Table 2: The first row in each block indicates the percentage of test sentences for which a match was found. Cells below contain the BLEU score over the entire test set (top number) and over the subset of test sentences augmented with matches (bottom left) and without matches (bottom right). Best scores of each column are outlined with bold fonts. Last column is the average of all corpus but News and TED.
For instance on KDE4: the base model obtains a BLEU score of 50.16 while FM+ obtains the highest score 54.59. Most of the gains are obtained over the test sentences having a fuzzy match (**65.95** vs. 53.05) while for sentences without fuzzy match the best score is obtained with the base system (**48.77** compared to 48.01).

Model	ECB	EMEA	JRC	GNOME	KDE4	PHP	Ubuntu	Avg
FM+	56.18	61.97	66.91	62.68	54.59	33.81	48.62	54.97
⊖(FM+, NM+)	56.83	60.60	67.52	61.97	54.67	32.38	47.13	54.44
⊖(FM+, EM+)	56.71	61.61	67.64	62.71	54.82	**33.60**	**49.98**	55.30
⊖(FM+, NM+, EM+)	56.20	61.30	67.43	62.14	55.05	32.33	48.96	54.77
⊕(FM+, EM+)	**57.08**	**62.27**	**68.06**	**63.30**	**55.48**	33.39	49.50	**55.58**
FT(base)	52.65	54.06	61.58	56.16	54.20	33.54	50.14	51.76
FT(⊖(FM+, EM+))	57.07	63.11	69.44	**65.97**	**59.30**	**36.26**	**52.77**	**57.70**
FT(⊕(FM+, EM+))	**57.44**	**63.41**	**69.82**	65.72	58.71	35.49	52.40	57.57

Table 3: BLEU scores of models combining several types of matches (2nd block) and over Fine-Tuned models (3rd block). We include again results of the FM+ model (1st block) to facilitate reading.

our hypothesis that marking related words is beneficial for the model. Masking sequences of unrelated words, FM* under-performs showing that the neural network is more challenged when dealing with incomplete sentences than with sentences containing unrelated content.

Target fuzzy matches To evaluate if the fuzzy match quality is really the primary criterion for the observed improvements, we consider $\texttt{FM}^{\#}_{T}$ where the fuzzy matches are rescored (on the training set only) with the edit distance between the reference translation and the target side of the fuzzy match. By doing so, we reduce the fuzzy match average FM source score by about 2%, but increase target edit distance from 61% to 69%.

The effect can be seen in Table 2 in the line $\texttt{FM}^{\#}_{T}$ vs. $\texttt{FM}^{\#}$. In average, this technique is performing better with large individual gains of +1.5 BLEU on the Ubuntu corpus. This shows that in this configuration where we do not differentiate related and unrelated words, the model mainly learns to copy fuzzy target words.

Unseen matches Note that in the previous experiments, matches were built over domain corpora that are already used to train the model. This is a common use case: the same translation memory used to train the system will be used in run time, but now we evaluate the ability of our model in a different context where a test set is to be translated for which we have a new TM that has never been seen when learning the original model. This use case corresponds to typical translation task where new entries will be added continuously to the TM and shall be used instantly for translation of following sentences. Hence, we only use EPPS, News, TED and Wiki data to build two models: the first employs only the original source and target sentences (`base`) the second learns to use fuzzy matches ($\texttt{FM}^{+}$). Table 4 shows results for this use case.

Model	ECB	EMEA	JRC	GNOME	KDE4	PHP	Ubuntu	Avg
$\%FM$	49.8	69.8	50.1	59.7	47.3	41.0	23.3	–
`base`	36.48	26.31	45.03	27.90	23.62	19.50	25.85	29.24
$\texttt{FM}^{+}$	**43.28**	**36.09**	**53.52**	**38.40**	**30.91**	**23.10**	**30.53**	**36.55**

Table 4: BLEU scores when models are only trained over EPPS, News, TED and Wiki datasets.

As it can be seen, the model using fuzzy matches shows clear accuracy gains. This confirms that gains obtained by $\texttt{FM}^{+}$ are not limited to remember an example previously "seen" during training. The model using fuzzy matches acquired the ability to actually copy or recycle words from the provided fuzzy matches and therefore is suitable for adaptive translation workflows. Note that all scores are lower than those showed in Table 2 as a result of discarding all in-domain data when training the models showing also that online use of translation memory is not a substitute for in-domain model fine-tuning as we will further investigate in **Fine Tuning**.

Combining matching algorithms Next, we evaluate the ability of our NMT models to combine different matching algorithms. First, we use $\ominus(M_1, M_2, ...)$ to denote the augmentation of an input sentence that considers first the match specified by M_1, if no match applies for the input sentence then it considers using the match specified by M_2, and so on. Note that at most one match is used. Sentences for which no match is found are kept without augmentation. Similar to Table 2, models are learned using all the available training data. Table 3 (2nd block) illustrates the results of this experiment. The first 3 lines show BLEU scores of models combining $\texttt{FM}^{+}$, $\texttt{NM}^{+}$ and $\texttt{EM}^{+}$. The last row illustrates the results of a model that learns to use two different matching algorithms. We use the best combination of matches obtained so far ($\texttt{FM}^{+}$ and $\texttt{EM}^{+}$) and augment input sentences with both matches. Figure 3 illustrates an example of an input sentence augmented with both a fuzzy match and an embedding match ($\texttt{FM}^{+}$ and $\texttt{EM}^{+}$). Notice that the model is able to distinguish between both types of augmented sequences by looking at the token used in the additional stream (*factor*). As it can be seen in Table 3 (2nd block), the best combination of matches is achieved by $\oplus(\texttt{FM}^{+}, \texttt{EM}^{+})$ further boosting the performance of previous configurations. It is only surpassed by $\ominus(\texttt{FM}^{+}, \texttt{EM}^{+})$ in two test sets by a slight margin.

Fine Tuning Results so far evaluate the ability of NMT models to integrate similar sentences. However, we have run our comparisons over a "generic" model built from a heterogeneous training data set while it is well known that these models do not achieve best performance on homogeneous test sets. Thus, we now assess the capability of our augmentation methods to enhance fine-tuned (Luong and Manning, 2015) models, a well known technique that is commonly used in domain adaptation scenarios obtaining state-of-the-art results. Table 3 illustrates the results of the model configurations previously described after fine-tuning the models towards each test set domain. Thus, building 7 fine-tuned models for each configuration. Note that similar sentences (matches) are retrieved from the same in-domain data sets used for fine tuning. As

$\oplus$(FM$^+$, EM$^+$)																						
How long does a cold last ? || *Combien de temps dure le vol ?* || *Combien de temps dure un vaccin ?*																						
S	**S**	**S**	**S**	**S**	**S**	**S**	**R**	**T**		**T**	**T**	**T**	**R**	**R**	**T**	**E**	**E**		**E**	**E**	**E**	**E**	**E**	**E**

Figure 3: Input sentence augmented with a fuzzy match FM$^+$ and an embedding match EM$^+$.

Token	base	FM$^\#$	FM$^+$	base	NM$^+$	base	EM$^+$	base	FT($\ominus$(FM$^+$, EM$^+$))
T	66.3%	79.9%	80.3%	68.9%	83.3%	–	–	66.3%	79.3%
R	31.3%	54.6%	49.3%	27.0%	34.4%	–	–	31.3%	46.2%
E	–	–	–	–	–	45.7%	58.6%	33.0%	37.7%

Table 5: Percentage of Tokens **T**, **R** and **E** effectively appearing in the translation.

shown in Table 3 (3rd block), models with FM/EM also increase performance of fine-tuned models gaining in average +6 BLEU on fine-tuned model baselines, and +2.5 compared to FM/EM on generic translation. This add-up effect is interesting since both approaches make use of the same data.

Copy Vs. Context We observe that models allowing for augmented input sentences effectively learn to output the target words used as augmented translations. Table 5 illustrates the rates of usage. We compute for each word added in the input sentence as **T** (part of a lexicalised match), **R** (not in the match) and **E** (from an embedding match), how often they appear in the translated sentence. Results show that **T** words increase their usage rate by more than 10% compared to the corresponding base models. Considering **R** words, models incorporating fuzzy matches increase their usage rate compared to base models, albeit with lower rates than for **T** words. Furthermore, the number of **R** words output by FM$^+$ is clearly lower than those output by FM$^\#$, demonstrating the effect of marking unrelated matching words. Thus, we can confirm the copy behaviour of the networks with lexicalised matches. Words marked as **E** (embedding matches) increase their usage rates when compared to base models but are far from the rates of **T** words. We hypothesize that these sentences are not copied by the translation model, rather they are used to further contextualise translations.

5 Related Work

Our work stems on the technique proposed by (Bulté and Tezcan, 2019) to train an NMT model to leverage fuzzy matches inserted in the source sentence. We extend the concept by experimenting with more general notions of similar sentences and techniques to inject fuzzy matches.

The use of similar sentences to improve translation models has been explored at scale in (Schwenk et al., 2019), where the authors use multilingual sentence embeddings to retrieve pairs of similar sentences and train models uniquely with such sentences. In (Niehues et al., 2016), input sentences are augmented with pre-translations performed by a phrase-based MT system. In our approach, similar sentence translations are provided dynamically to guide translation of a given sentence.

Similar to our work, (Farajian et al., 2017; Li et al., 2018) retrieve similar sentences from the training data to dynamically adapt individual input sentences. To compute similarity, the first work uses n-gram matches, the second includes dense vector representations. In (Xu et al., 2019) the same approach is followed but authors consider for adaptation a bunch of semantically related input sentences to reduce adaptation time.

Our approach combines source and target words within a same sentence - the same type of approach has also been proposed by (Dinu et al., 2019) for introduction of terminology translation.

Last, we can also compare the extra-tokens appended in augmented sentences as "side constraints" activating different translation paths on the same spirit than the work done by (Sennrich et al., 2016a; Kobus et al., 2017) for controlling translation.

6 Conclusions and Further Work

This paper explores augmentation methods for boosting Neural Machine Translation performance by using similar translations.

Based on "neural fuzzy repair" technique, we introduce tighter integration of fuzzy matches informing neural network of source and target and propose extension to similar translations retrieved

from their distributed representations. We show that the different types of similar translations and model fine-tuning provide complementary information to the neural model outperforming consistently and significantly previous work. We perform data augmentation at inference time with negligible speed overhead and release an Open-Source toolkit with an efficient and flexible fuzzy-match implementation.

In our future work, we plan to optimise the thresholds used with the retrieval algorithms in order to more intelligently select those translations providing richest information to the NMT model and generalize the use of edit distance on the target side.

We would also like to explore better techniques to inject information of small-size n-grams with possible convergence with terminology injection techniques, unifying framework where target clues are mixed with source sentence during translation. As regards distributed representations, we plan to study alternative networks to more accurately model the identification and incorporation of additional context.

Acknowledgments

We would like to thank Professor François Yvon for his insightful comments as well as the anonymous reviewers for the useful suggestions.

References

Michael Bloodgood and Benjamin Strauss. 2014. Translation memory retrieval methods. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 202–210.

Bram Bulté and Arda Tezcan. 2019. Neural fuzzy repair: Integrating fuzzy matches into neural machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1800–1809, Florence, Italy. Association for Computational Linguistics.

Bram Bulté, Tom Vanallemeersch, and Vincent Vandeghinste. 2018. M3tra: integrating tm and mt for professional translators. pages 69–78. EAMT.

Chenhui Chu and Rui Wang. 2018. A survey of domain adaptation for neural machine translation. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1304–1319, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Georgiana Dinu, Prashant Mathur, Marcello Federico, and Yaser Al-Onaizan. 2019. Training neural machine translation to apply terminology constraints. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3063–3068, Florence, Italy. Association for Computational Linguistics.

Chris Dyer, Victor Chahuneau, and Noah A. Smith. 2013. A simple, fast, and effective reparameterization of IBM model 2. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 644–648, Atlanta, Georgia. Association for Computational Linguistics.

M. Amin Farajian, Marco Turchi, Matteo Negri, and Marcello Federico. 2017. Multi-domain neural machine translation through unsupervised adaptation. In *Proceedings of the Second Conference on Machine Translation*, pages 127–137, Copenhagen, Denmark. Association for Computational Linguistics.

Jiatao Gu, Yong Wang, Kyunghyun Cho, and Victor OK Li. 2018. Search engine guided neural machine translation. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander Rush. 2017. OpenNMT: Open-source toolkit for neural machine translation. In *Proceedings of ACL 2017, System Demonstrations*, pages 67–72, Vancouver, Canada. Association for Computational Linguistics.

Catherine Kobus, Josep Crego, and Jean Senellart. 2017. Domain control for neural machine translation. In *Proceedings of the International Conference Recent Advances in Natural Language Processing, RANLP 2017*, pages 372–378, Varna, Bulgaria. INCOMA Ltd.

Philipp Koehn and Jean Senellart. 2010. Convergence of Translation Memory and Statistical Machine Translation. In *Proceedings of AMTA Workshop on MT Research and the Translation Industry*, pages 21–31, Denver.

Xiaoqing Li, Jiajun Zhang, and Chengqing Zong. 2018. One sentence one model for neural machine translation. In *Proceedings of the 11th Language Resources and Evaluation Conference*, Miyazaki, Japan. European Language Resource Association.

Minh-Thang Luong and Christopher D. Manning. 2015. Stanford neural machine translation systems for spoken language domain. In *International Workshop on Spoken Language Translation*, Da Nang, Vietnam.

Udi Manber and Gene Myers. 1993. Suffix arrays: a new method for on-line string searches. *siam Journal on Computing*, 22(5):935–948.

Jan Niehues, Eunah Cho, Thanh-Le Ha, and Alex Waibel. 2016. Pre-translation for neural machine translation. *CoRR*, abs/1610.05243.

John E Ortega, Felipe Sánchez-Martınez, and Mikel L Forcada. 2016. Fuzzy-match repair using black-box machine translation systems: what can be expected. In *Proceedings of AMTA*, volume 1, pages 27–39.

Matteo Pagliardini, Prakhar Gupta, and Martin Jaggi. 2018. Unsupervised learning of sentence embeddings using compositional n-gram features. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 528–540, New Orleans, Louisiana. Association for Computational Linguistics.

Mike Paterson and Vlado Dančík. 1994. Longest common subsequences. In *International Symposium on Mathematical Foundations of Computer Science*, pages 127–142. Springer.

Mirko Plitt and François Masselot. 2010. A productivity test of statistical machine translation post-editing in a typical localisation context. *The Prague bulletin of mathematical linguistics*, 93:7–16.

Holger Schwenk, Guillaume Wenzek, Sergey Edunov, Edouard Grave, and Armand Joulin. 2019. Ccmatrix: Mining billions of high-quality parallel sentences on the web. *arXiv preprint arXiv:1911.04944*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016a. Controlling politeness in neural machine translation via side constraints. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 35–40, San Diego, California. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016b. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Jörg Tiedemann. 2012. Parallel data, tools and interfaces in OPUS. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 2214–2218, Istanbul, Turkey. European Language Resources Association (ELRA).

Tom Vanallemeersch and Vincent Vandeghinste. 2015. Assessing linguistically aware fuzzy matching in translation memories. In *Proceedings of the 18th Annual Conference of the European Association for Machine Translation*, pages 153–160, Antalya, Turkey.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Kun Wang, Chengqing Zong, and Keh-Yih Su. 2013. Integrating translation memory into phrase-based machine translation during decoding. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11–21, Sofia, Bulgaria. Association for Computational Linguistics.

Jitao Xu, Josep Crego, and Jean Senellart. 2019. Lexical micro-adaptation for neural machine translation. In *International Workshop on Spoken Language Translation*, Honk Kong, China.

Masaru Yamada. 2011. The effect of translation memory databases on productivity. *Translation research projects*, 3:63–73.

Anna Zaretskaya, Gloria Corpas Pastor, and Miriam Seghiri. 2015. Integration of machine translation in cat tools: State of the art, evaluation and user attitudes. *Skase Journal of Translation and Interpretation*, 8(1):76–89.

Jingyi Zhang, Masao Utiyama, Eiichro Sumita, Graham Neubig, and Satoshi Nakamura. 2018. Guiding neural machine translation with retrieved translation pieces. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1325–1335, New Orleans, Louisiana. Association for Computational Linguistics.

A Corpora Statistics

Corpus	#Sents (K)	L_{mean}		Vocab (K)	
		English	French	English	French
EPPS	1,992.8	27.7	32.0	129.5	149.2
News	315.3	25.3	31.7	90.5	96.7
TED	156.1	20.1	22.1	58.7	71.4
Wiki	749.0	25.9	23.5	527.5	506.6
ECB	174.1	28.6	33.8	45.3	53.5
EMEA	336.8	16.8	20.3	62.8	68.9
JRC	475.2	30.1	34.5	81.0	83.5
GNOME	51.9	9.6	11.6	19.0	21.6
KDE4	163.9	9.1	12.4	48.7	64.7
PHP	15.1	16.7	18.0	13.3	15.5
Ubuntu	7.1	6.7	8.3	7.5	7.9

Table 6: Corpora statistics. Note that K stands for thousands and L_{mean} is the average length in words.

B NMT Network Configuration

We use the next set of hyper-parameters: size of word embedding: 512; size of hidden layers: 512; size of inner feed forward layer: $2,048$; number of heads: 8; number of layers: 6; batch size: $4,096$ tokens. Note that when using factors ($\texttt{FM}^{+}$, $\texttt{NM}^{+}$ and $\texttt{EM}^{+}$) the final word embedding is built after concatenation of the word embedding (508 cells) and the additional factor embedding (4 cells).
We use the lazy Adam optimiser. We set warmup steps to $4,000$ and update learning rate for every 8 iterations. Models are optimised during $300K$ iterations. Fine-tuning is performed continuing Adam with the same learning rate decay schedule until convergence on the validation set. All models are trained using a single NVIDIA P100 GPU.
We limit the target sentence length to 100 tokens.The source sentence is limited to 100, 200 and 300 tokens depending on the number of sentences used to augment the input sentence. We use a joint vocabulary of 32K for both source and target sides. In inference we use a beam size of 5. For evaluation, we report BLEU scores computed by `multi-bleu.perl`.

C Example of Embedding Matching

The table below gives examples of retrieved `EM` with matching distance ≥ 0.8 and with Fuzzy Match distance lower than threshold 0.6.

Distance	Source Sentence	Matched Sentence
0.86	(i) supply of gas to power producers (CCGTs [10]);	(a) Gas supply to power producers (CCGTs)
0.87	The Commission shall provide the chairman and the secretariat for these working parties.	The Commission shall provide secretariat services for the Forum, the Bureau and the working parties.
0.93	Admission to a course of training as a pharmacist shall be contingent upon possession of a diploma or certificate giving access, in a Member State, to the studies in question, at universities or higher institutes of a level recognised as equivalent.	Admission to basic dental training presupposes possession of a diploma or certificate giving access, for the studies in question, to universities or higher institutes of a level recognised as equivalent, in a Member State.