# PEPr: Post-Edit Propagation Using Phrase-based Statistical Machine Translation

**Michel Simard** **George Foster**
National Research Council Canada
1200 Montreal Road
Ottawa, Ontario K1A 0R6 Canada
firstname.lastname@nrc.gc.ca

## Abstract

Translators who work by post-editing machine translation output often find themselves repeatedly correcting the same errors. We propose a method for *Post-edit Propagation* (PEPr), which learns post-editor corrections and applies them on-the-fly to further MT output. Our proposal is based on a phrase-based SMT system, used in an automatic post-editing (APE) setting with online learning. Simulated experiments on a variety of data sets show that for documents with high levels of internal repetition, the proposed mechanism could substantially reduce the post-editing effort.

## 1 Introduction

While post-editing of machine translation is an increasingly widespread practice, very few technological solutions exist that are aimed specifically at facilitating the work of the post-editor. This paper addresses this gap, by proposing a mechanism that automatically propagates post-editor corrections to further machine-translated sentences within a document. We call this process *Post-edit Propagation*, or PEPr for short.

One recurrent complain from post-editors is that they often have to fix the same error repeatedly. Repeated errors can happen for a number of reasons: if the MT system's training data was too small or heterogeneous, or if it was from a different domain as the document under consideration, then it is not uncommon for a given word or phrase to be systematically mistranslated. Carpuat & Simard (2012) have shown that SMT systems tend to be highly consistent, meaning that multiple occurrences of any given source-language word or phrase will tend to be translated by the same target-language phrase. If that translation happens not to be appropriate in the current context, i.e. if the system is consistently wrong, then the post-editor will need to fix that translation several times. As pointed out by Lagoudaki (2008), post-editing systems should have the ability to "learn from the decisions/choices made by users (e.g. which potential translations are preferred, which were rejected and why), so that errors in future translation assemblies are reduced."

Developers of commercial translation memory (TM) systems have already taken note of this requirement. The idea behind TM technology is that translators should not have to translate the same material more than once. To achieve this, all translations are archived, new texts are systematically compared to the contents of the archive, and when segments are found that have been translated previously, their translation is retrieved and proposed to the translator for reuse.

Of course, this idea is based on the assumption that text segments do indeed repeat. There are some specialized domains for which this often appears to be true, e.g. contractual documents, user manuals, etc. But in general, the degree of repetitivity of text varies greatly over the myriad of possible text domains. However, it is sometimes remarked that if a segment is to repeat at all, this has the greatest chance of happening within the same document where the segment initially appeared (Church and Gale, 1995). This phenomenon, which we call *document-internal repetition*, is the motivation for developping TM systems with real-time update capabilities (an example of such a system is *Fusion One*[1]). Within such systems, each text segment is archived as soon as it is processed by the translator, so that if it reappears within the same document, its most recent translation is immediately available for re-use. Viewed as

[1] http://www.jivefusiontech.com

Sima'an, K., Forcada, M.L., Grasmick, D., Depraetere, H., Way, A. (eds.)
*Proceedings of the XIV Machine Translation Summit (Nice, September 2–6, 2013), p. 191–198.*

an active process, this is something we might call *translation propagation*.

A similar idea can be applied in the context of MT post-editing: here, translation proposals are produced by an MT system for every input text segment, and these proposals are processed one by one (either accepted, corrected or re-written) by a (human) post-editor. If the MT system under consideration has learning abilities (such as SMT systems), then the analog of TM's real-time archiving of processed translations is something that is sometimes refered to as "incremental training" or "online learning": the MT system integrates the post-edited translation, as they are produced, in the hope of improving the future performance of the system.

Whether this sort of mechanism makes it possible to effectively propagate post-edits depends on a number of factors. First, let us clarify what we understand by PEPr: given this ability, an MT system would take user corrections into consideration when translating new material: given some text segment $s$, which the MT originally translated as $t$, and which the post-editor corrected as $r$, the system would learn to translate $s$ as $r$ in later translations, when appropriate. We add "when appropriate", because it might not be adequate to translate $s$ as $r$ in all contexts: some corrections might be specific to the way in which $s$ is used within a given document, or even be specific to the more local context: for example, it may not be relevant to propagate corrections pertaining to agreement or other similar grammatical phenomena.

Rather than modify the behavior of the MT system directly, it is simpler and might be just as effective to learn how to reproduce the corrections. Thus, it would truly be the post-edits that are propagated, not the post-edited translations. Under this model, the PEPr mechanism learns by looking at post-editor corrections (how $t$ is transformed into $r$), and selectively re-applies these corrections to further system translation proposals within the same document. Note that this is done without any knowledge of the source-language $s$ of which $t$ is the proposed translation. It also makes it possible to view the MT system that produced $t$ as a black box: in effect, it could be an online MT service, or any other kind of system on whose behavior the user has no or little control; it doesn't even have to be machine translation system: it could be a TM system, a combined MT/TM system, etc. Because only local post-edits are considered (i.e. those performed within the current document or user-session), such a system manipulates very little data, making the whole process lightweight. Finally, this strategy allows simpler control over the relative weight of prior (background) and newly acquired (local) knowledge.

We argue that the PEPr task can be effectively carried out by a phrase-based statistical MT system. The system is then effectively an *automatic post-editing* (APE) system with online learning capabilities. We detail how this idea can be realized (Section 2), then validate our design with experiments that simulate PE sessions (Section 3). Related work is reviewed in Section 4.

## 2 PEPr using Phrase-based SMT

An APE system is one that performs transformations on MT output. If this APE system is based on a SMT system, as proposed in Simard et al. (2007), then it can be seen as a system that attempts to translate target-language (TL) MT output into proper (human quality) TL text. Typically, it will be trained on a corpus of post-edited MT: pairs of machine-translated sentences along with their post-edited counterparts.

To perform post-edit propagation, we propose to use an SMT system in an APE setting: the system will be trained incrementally, using pairs of machine-translated and post-edited segments as they are produced. The whole process is assumed to take place within the context of a single document $D$. The general set-up is illustrated in Figure 1:

- The source text is a document $D$, consisting of a sequence of source-language sentences $S = s_1...s_n$;
- A machine-translation system $M$ provides target language versions $t_k$ of each source sentence $s_k$;
- Each MT output $t_k$ is automatically post-edited by a sentence-specific APE system $A_k$, producing a second target-language version $p_k$;
- Working from source sentences $s_1...s_n$ and the automatically post-edited target-language

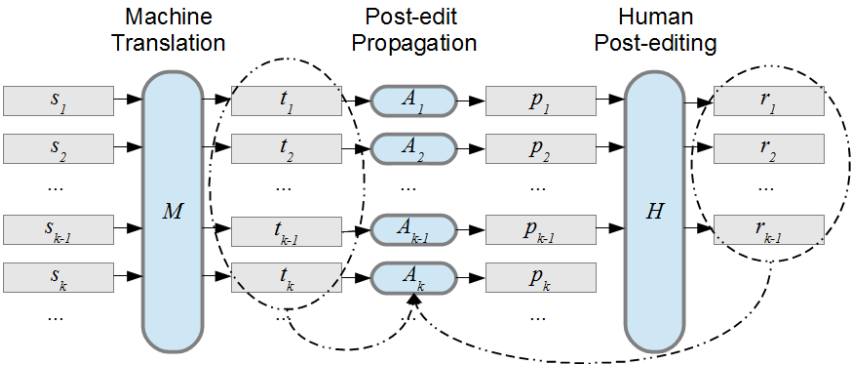


Figure 1: Incremental learning set-up for PEPr. Dotted lines indicate the sources of information used to build system $A_k$ that propagate post-edits to translation $t_k$.

sentences $p_1 ... p_n$, a human post-editor $H$ produces final (reference) target-language versions $r_1 ... r_n$.

Each APE system $A_k$ is implemented as a phrase-based SMT system, created specifically to post-edit sentence $t_k$. While in practice this process can be viewed (and implemented in large part) as incremental training, it is formally simpler to view each system as a new one. Each system $A_k$ relies on a phrase table and language model built from the previously post-edited segments: $t_1 ... t_{k-1}$ and $r_1 ... r_{k-1}$.

Because everything takes place within the context of document $D$, we further make the assumption that system $A_1$ is actually an "empty" system, i.e. one that copies its input onto its output. It can therefore be assumed that $p_1 = t_1$. Alternatively, $A_1$ could be "primed" using data from similar, previously post-edited documents; we do not make this assumption here. Also, to simplify matters, we assume that each sentence in $D$ is reviewed exactly once by the post-editor: no sentence is left unreviewed, and no sentence is reviewed more than once. The indices $k = 1...n$ represent the order in which the post-editor $H$ reviews the sentences of $D$; this is not necessarily the order in which the sentences appear in $D$.

**Phrase Table and Translation Model** In an SMT framework, the post-editing "rules" that implement PEPr are embodied within phrase pairs in the system's phrase table. The phrase table for system $A_k$ contains all phrases extracted from the pairs of previously post-edited segments: $\langle t_1, r_1 \rangle ... \langle t_{k-1}, r_{k-1} \rangle$. These phrase pairs implicitly contain all previously observed post-edits (within the limits of phrase-based SMT's ability to capture these edits).

Because learned post-edits should only be applied when appropriate, we must take care that the APE system always has the option of not editing its input. We call this the *do-nothing option*. The standard mechanism for handling out-of-vocabulary words, which passes through unknown words, cannot be relied upon for the *do-nothing option*, because it is inhibited as soon as a word appears in the phrase table: if on its first occurrence a word $w$ is post-edited to $v$, then that correction will automatically be applied the second time $w$ is encountered. Instead, we explicitly include in the phrase table unedited versions of all input phrases. This can be achieved by adding the pair $\langle t_k, t_k \rangle$ to $A_k$'s training set; this way, the phrase table contains phrase pairs that explicitly sanction leaving a word or phrase unedited in the APE output.

While phrase pair extraction is performed using standard methods (Koehn et al., 2007), a word-level alignment between machine translations $t_i$ and reference translations $r_i$ is required, which, in standard SMT (or APE) approaches, is computed using IBM-style translation models. In a PEPr scenario, there is typically too little data to train such a model: in many cases, the training corpus will be ridiculously small. But since the "source" and "target" are the same language, a straightforward alternative is to obtain the alignment from the minimal sequence of edits $e = e_1 ... e_m$ corresponding to the word-level Levenshtein distance between $t_i$ and $r_i$. To achieve more precise alignments, we rely on Damerau-Levenshtein distance which, in addition to insertions, deletions and substitu-

tions, also considers local inversions ("x y" → "y x"); furthermore, all edits have a unit cost, except substitutions, whose cost is the length-normalized character-level edit distance between the substituted words. The alignment is produced by drawing links between words of $t_i$ and $r_i$ that are substituted, swapped or copied (not edited); inserted and deleted words are left unaligned.

**Language Models** Analog to the phrase table, the APE system's language model (LM) component is trained using previously post-edited target-language sentences $r_1...r_{k-1}$. To enable the *do-nothing option*, it is necessary to also include the unedited versions of the TL sentences $t_1...t_{k-1}$ into the language model's training set. Rather than combine all of this information into a single training set, we build two distinct language models: a *reference LM* is trained on the manually post-edited (reference) translations $r_1...r_{k-1}$; a separate *MT LM* is trained on all unedited MT sentences of the document, $t_1...t_n$. At decoding time, the parameters of the two models are combined linearly, following the method proposed in Foster el al. (2007).

The training sets for LM's are typically very small; this results in weak models, which may lead to APE systems that apply post-edits with little regard for the fluency of the result. One workaround to this problem is to replace or complement the *MT LM* with a model trained on larger amounts of data. In-domain or related target language material can be used if available; otherwise, even out-of-domain data can be useful. In our experiments we used a *generic LM*, trained on a very large corpus harvested from the Web (see Section 3.2).

**Reordering Model** In phrase-based SMT, the reordering (or distortion) model controls the relative order of SL and TL words. Local word reorderings are typically captured within phrase pairs in the phrase table, and need not be handled by these models. Phrase-level reordering is already a complicated matter, and it is doubtful that we can reliably model it in this extreme sparse-data environment. We also conjecture that phrase-level reorderings are a rare event in post-editing, and therefore completely inhibit them for PEPr.

**Parameter Optimization** Parameters of the APE systems' log-linear model are optimized using batch-MIRA (Cherry and Foster, 2012). This procedure normally assumes a development set, which is repeatedly translated with a single translation system. In a PEPr setting, the system dynamically changes after each post-edited sentence. We nevertheless make the assumption that it is possible to find a set of parameters that is globally optimal under these varying conditions. We modify the decoding step in the optimization loop so that each development set sentence is translated by a different system, as described above.

In our current setting, the parameters controlling the LM mixture are optimized manually on the development set. This approach and possible alternatives are discussed in Section 3.4.

## 3 Experiments

We evaluate the potential of the proposed PEPr approach by simulation: given a set of test documents for which we have both a source language version $S$ and a reference translation $R$, we produce a machine translation $T$; for each sentence, we then create an APE system as described in Section 2, and use it to produce automatically post-edited versions $P$. We take reference translations of $R$ as post-edited versions of $P$, and use them both to feed the PEPr process and to evaluate the performance of the system.

### 3.1 MT and APE Systems

For the purpose of generating machine translations $T$, we used a "generic" MT system, i.e. a system not intended for a particular text domain or genre. We used Portage, a typical phrase-based SMT system which has achieved competitive results in recent WMT (Larkin et al., 2010) and NIST evaluations, and trained it using a very large corpus of English-French Canadian government data harvested from the Web (domain `gc.ca`), containing over 500M words in each language. We used the following feature functions in the log-linear model: 5-gram language model with Kneser-Ney smoothing (1 feature); relative-frequency and lexical translation model probabilities in both directions (4 features); lexicalized distortion (6 features); and word count (1 feature). The parameters of the log-linear model were tuned by optimizing BLEU on the development set using the batch variant of MIRA (Cherry and Foster, 2012). Phrase extraction was done by aligning the cor-

pus at the word level using both HMM and IBM2 models, using the union of phrases extracted from these separate alignments for the phrase table, with a maximum phrase length of 7 tokens. Phrase pairs were filtered so that the top 30 translations for each source phrase were retained.

We also used the Portage system for the APE system implementing PEPr. The components of the APE system were set up as described in Section 2, and the log-linear model combines the following feature functions: linear mixture language model (1 feature); relative-frequency translation model probabilities in both directions (2 features); and word count (1 feature). Phrases were limited to 7 tokens. The *Reference LM* and *MT LM* used in the LM mixture are trigram models with Witten-Bell smoothing; the *generic LM's* are those used by the MT system above. All components are trained on true case data: the intention is to capture case-related corrections.

### 3.2 Data

Our experimental data consists of documents extracted from the ECB and EMEA corpora of the OPUS corpus (Tiedemann, 2009), and a collection of scientific abstracts from Canadian publications; we focused on French and English versions of these datasets, and performed experiments in both translation directions. The choice of the test data was motivated by the need for real document-like discourse units, but also by the technical and specialized nature of the texts, which makes them particularly difficult for a generic MT system. Furthermore, some of these texts (ECB and EMEA in particular) feature high levels of document-internal and domain-specific repetition, as suggested by their high token/type ratios (see Table 1).

We have limited document size to 100 sentences, so as to avoid larger documents biasing the results; longer documents were truncated[2]. The collection of scientific abstracts is also highly technical, but most documents are very short, even though we excluded abstracts shorter than 5 sentences. Therefore, each document contains little internal repetition. To better understand the effect of document length, we examined the effect of PEPr on this corpus under two different conditions: with abstracts considered as individual documents, and grouping multiple abstracts from the same journal and year into a single "digest".

| | ECB | EMEA | Science Abstracts | Science Digests |
|---|---|---|---|---|
| **Development Sets** | | | | |
| documents | 6 | 7 | 75 | 23 |
| sentences | 600 | 538 | 578 | 551 |
| tokens (EN) | 17 142 | 8448 | 14 580 | 13 930 |
| token/type | 3.54 | 3.65 | 1.75 | 2.04 |
| **Test Sets** | | | | |
| documents | 13 | 11 | 312 | 77 |
| sentences | 1313 | 795 | 2426 | 2453 |
| tokens (EN) | 44 312 | 13 387 | 65 973 | 19 631 |
| token/type | 3.87 | 2.67 | 1.78 | 2.14 |

Table 1: Experimental Data

The development sets used to optimize the parameters of the APE systems were intentionally made relatively small, on the order of 10-15K words. Intuitively, this is intended to correspond to about a week's worth of human post-editing. In a real-life setting, this data could be collected during a "warm-up" period. Alternatively, the system could be initially deployed with random parameters, and its parameters periodically re-optimized.

### 3.3 Results

We tested our approach on all datasets, under two different conditions: first by mixing the *reference LM* of the PEPr system with an *MT LM* (trained on the MT output, as described in Section 2); second by mixing the *reference LM* with a background model, trained on large amounts of "general language" data (*generic LM*). In our experiments, this happens to be the same LM that is used by the first stage MT system. The weight of the *reference LM* was manually set to 0.9 in the linear mixture with the *MT LM* and to 0.5 when combining with the *generic LM*. We further discuss the effect of varying this parameter in Section 3.4.

Table 2 presents the results of these experiments. The impact of PEPr is measured in terms of WER and BLEU gain (for convenience, we report WER scores as 100-WER, so that larger values denote better translations, and negative "gains" can be interpreted as "losses") [3]. For each corpus and language, we first report the scores obtained by the

[2]Intuitively, 100 sentences approximately corresponds to the daily production of a professional translator. For longer documents, we expect our systems to behave more like standard, batch-trained APE systems.

[3]TER would arguably have been a better metric to evaluate the potential of our approach in a post-editing setting; in practice, however, WER is known to behave very similarly to TER (Cer et al., 2010).

| Corpus | System | 100-WER | BLEU |
|---|---|---|---|
| **ECB** | | | |
| en→fr | MT | 32.24 | 25.87 |
| | +PEPr-MTLM | +5.71 | +6.16 |
| | +PEPr-GLM | +6.53 | +7.08 |
| fr→en | MT | 32.65 | 22.56 |
| | +PEPr-MTLM | +3.45 | +7.42 |
| | +PEPr-GLM | +5.38 | +9.27 |
| **EMEA** | | | |
| en→fr | MT | 32.75 | 25.05 |
| | +PEPr-MTLM | +3.83 | +5.37 |
| | +PEPr-GLM | +4.76 | +6.56 |
| fr→en | MT | 30.05 | 25.13 |
| | +PEPr-MTLM | +3.27 | +5.35 |
| | +PEPr-GLM | +4.56 | +7.44 |
| **Science Abstracts** | | | |
| en→fr | MT | 37.00 | 24.00 |
| | +PEPr-MTLM | -0.94 | +0.73 |
| | +PEPr-GLM | -0.10 | -0.16 |
| fr→en | MT | 39.64 | 24.82 |
| | +PEPr-MTLM | +0.84 | +0.74 |
| | +PEPr-GLM | -0.50 | -0.04 |
| **Science Digests** | | | |
| en→fr | MT | 36.96 | 23.93 |
| | +PEPr-MTLM | +0.37 | +0.37 |
| | +PEPr-GLM | -1.88 | +0.09 |
| fr→en | MT | 39.68 | 24.81 |
| | +PEPr-MTLM | +0.56 | +0.57 |
| | +PEPr-GLM | -0.64 | -0.16 |

Table 2: Experimental results

raw machine translation, prior to performing PEPr (**MT**), then the effect of PEPr mixing the *reference LM* with the *MT LM* (**+PEPr-MTLM**), and last the effect of PEPr mixing the *reference LM* with the *generic LM* (**+PEPr-GLM**).

For the ECB and EMEA corpora, PEPr has a clear positive impact: WER is reduced by 3.27 to 6.53, while BLEU increases by 5.35 to 9.27. Mixing the *reference LM* with a generic background LM (+PEPr-GLM) appears to work better than with a locally-trained *MT LM* (+PEPr-MTLM). This is not entirely surprising: While the *MT LM* knows little more than how to do nothing, the *generic LM* is a rich source of additional knowledge that the APE system can exploit to produce more fluent translations.

The Science corpora illustrate situations where PEPr is unlikely to bring significant improvements to the initial MT. In fact, in many of these conditions, PEPr slightly degrades translation quality, as measured with WER and BLEU. In practice, the Science abstracts are simply too short to contain document-internal repetition that PEPr can exploit advantageously (average length of documents is 7.7 sentences). When combined into yearly digests, the documents become substantially larger (31.9 sentences per document), but they are too heterogeneous to contain any exploitable repeated corrections.

| | 100-WER | | BLEU | |
|---|---|---|---|---|
| Corpus | Actual | Oracle | Actual | Oracle |
| **ECB** | | | | |
| en→fr | 38.83 | 39.76 | 33.00 | 33.80 |
| fr→en | 38.07 | 39.53 | 31.97 | 33.04 |
| **EMEA** | | | | |
| en→fr | 37.68 | 38.45 | 31.67 | 32.77 |
| fr→en | 44.64 | 45.75 | 32.56 | 33.86 |
| **Science Abstracts** | | | | |
| en→fr | 36.77 | 37.09 | 23.26 | 23.38 |
| fr→en | 38.83 | 39.72 | 24.14 | 24.40 |
| **Science Digests** | | | | |
| en→fr | 34.99 | 36.89 | 23.96 | 24.14 |
| fr→en | 39.01 | 40.02 | 24.57 | 24.92 |

Table 3: Oracle scores for PEPr-GLM

Since PEPr relies only on information in the MT output and revisions, an interesting question is whether it could be improved by using the source text as well, as proposed e.g. in Béchara et al. (2011). To get an approximate bound on the gains that might be obtained this way, we generated 1000-best lists of PEPr output and computed oracle scores using the references. As shown in Table 3, the gains are quite modest, in the order of 1-2 WER. In all cases PEPr achieves over 85% of this estimated upper bound, which would be difficult to match in practice.[4]

### 3.4 Mixture LM Parameter Optimization

It is instructive to examine the behavior of the PEPr layer as we vary the relative weight of the *reference LM* in the LM mixture. This is shown for the ECB fr→en development set in Figure 2. The black "○" curve denotes the amount of edits performed by PEPr, measured in terms of WER (on a scale of 0 - 100). The plot on the left-hand side ("PEPr + MTLM") illustrates the situation for mixtures with the *MT LM*, which is intended to implement the *do-nothing option*. When all the weight is assigned to the *MT LM*, the PEPr layer performs virtually no changes to its MT input; conversely, assigning all the weight to the *reference LM* results in more than 20% of the words being edited.

[4]This is not strictly speaking an upper bound for adapting the base MT system, since it might be able to draw on other resources to infer domain-specific translations. However, such techniques would be considerably more complex than what we propose here.

Between these two extremes, the amount of PEPr edits grows monotonically. WER and BLEU gains (red "□" and blue "△" curves, respectively) appear to follow the same kind of progression. This suggests that, while assigning more weight to the *MT LM* does make the system less aggressive, it does not make it more discriminant: PEPr corrections are inhibited regardless of their potential value for the post-editor.

This contrasts with the plot on the right-hand side ("PEPr + GLM"), which corresponds to mixtures with a rich background LM. Here again, the amount of PEPr corrections increases dramatically as more weight is assigned to the *reference LM*. Here, however, WER and BLEU gains follow a different pattern, displaying optimal values somewhere between the two extreme settings. (Interestingly, in this case, the outcome will be substantially different whether we optimize relative to WER or BLEU; this behavior is not generalized, however.) The *generic LM* provides additional information, which the PEPr system can exploit to make better decisions. This suggests that, when such a background LM is available, it makes sense to automatically optimize its relative weight on development data.

However, the effect of the mixture parameter on the global behavior of PEPr is striking, and it could also be interesting to leave its setting to the user, as a way of controlling how aggressive or conservative the system is.

## 4 Related Work

Automatic post-editing using SMT was proposed in Simard et al. (2007). The idea of dynamically updating an APE system after each sentence revised by a translator was the subject of an early proposal by Knight and Chander (1994). As far as we are aware, it has not been investigated experimentally in previous work, although as noted above certain commercial TM systems allow dynamic updates.

Similar ideas have been explored for SMT, beginning with Nepveu et al. (2004), who used a cache-based approach to incorporate recent word-for-word translations and ngrams into an early interactive SMT system. Hardt and Elming (2010) applied a similar strategy to a modern phrase-based SMT system, using heuristic IBM4-based word alignment techniques to augment a local phrase table with material from successive post-edited sentences. Two related themes in SMT research are incremental training (Levenberg et al., 2010) and context-based adaptation without user feedback (Tiedemann, 2010; Gong et al., 2011). These techniques have not yet been applied to automatic post-editing, outside the work of Hardt and Elming (2010).

## 5 Conclusion

We have proposed a method to perform automatic post-edit propagation (PEPr), using a phrase-based SMT system in an APE setting, with incremental training. Experiments simulating post-editing sessions suggest that our method is particularly effective when translating technical documents with high levels of internal repetition. Because the method is designed to work with extremely small amounts of training data, we believe that it can be implemented into an efficient, lightweight process.

There are potentially many ways in which our method could be improved. Using Levenshtein-Damerau distance to capture post-edits into a phrase table appears to be effective, but there are likely many situations where this approach runs into trouble, especially for heavily post-edited sentences. Alternative or complementary alignment techniques should be investigated, e.g. using IBM-style translation models trained on existing post-edited data. Phrase table filtering techniques could also be considered, to discard dubious phrase pairs.

In Section 3.4, we proposed to leave the setting of the LM mixture parameters to the user, as a way of controlling the aggressivity of the PEPr layer. We believe this sort of control is valuable, but it may be equally or even more effectively achieved by modifying the translation model probabilities. The LM mixture parameter could then be optimized on development data, so as to minimize post-editing effort.

Finally, all experiments were performed by simulation, using data that was not produced by post-editing. Independently-produced translations are possibly more distant from MT outputs than real post-edits; however, it is not clear how this affects our evaluation of the approach. In the end, to properly evaluate the value of our proposal will require implemententing a prototype version and submitting it to a real user evaluation.

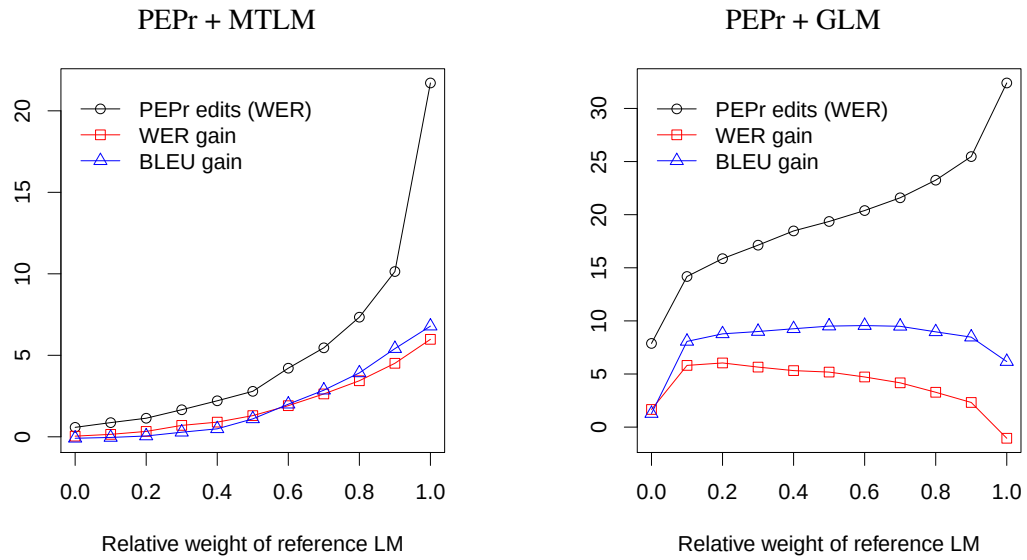


Figure 2: Amount of PEPr edits on the **ECB fr→en Development Set** (measured as WER between raw MT and PEPr output – in black "○"), WER gain (red "□") and BLEU gain (blue "△") as a function of the relative weight of the *reference LM* in LM mixtures with *MT LM* (left) and *generic LM* (right).

## References


Béchara, H., Y. Ma, and J. van Genabith. 2011. Statistical Post-editing for a Statistical MT System. In *MT Summit XIII*, pages 308–315.

Carpuat, M. and M. Simard. 2012. The trouble with SMT consistency. In *WMT*, pages 442–449.

Cer, D., C. D. Manning, and D. Jurafsky. 2010. The Best Lexical Metric for Phrase-Based Statistical MT System Optimization. In *HLT-NAACL*, pages 555–563.

Cherry, C. and G. Foster. 2012. Batch tuning strategies for statistical machine translation. In *NAACL*, volume 12, pages 34–35.

Church, K.W. and W.A. Gale. 1995. Poisson mixtures. *Natural Language Engineering*, 1(2):163–190.

Foster, G. and R. Kuhn. 2007. Mixture-Model Adaptation for SMT. In *WMT*.

Gong, Z., M. Zhang, and G. Zhou. 2011. Cache-based document-level statistical machine translation. In *EMNLP*.

Hardt, D. and J. Elming. 2010. Incremental Retraining for Post-Editing SMT. In *AMTA*.

Knight, K. and I. Chander. 1994. Automated postediting of documents. In *National Conference on Artificial Intelligence*, pages 779–779. JOHN WILEY & SONS LTD.

Koehn, P., H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, et al. 2007. Moses: Open Source Toolkit for Statistical Machine Translation. In *ACL*, pages 177–180.

Lagoudaki, E. 2008. The value of machine translation for the professional translator. In *AMTA*, pages 262–269.

Larkin, S., B. Chen, G. Foster, U. Germann, E. Joanis, H. Johnson, and R. Kuhn. 2010. Lessons from NRC's Portage system at WMT 2010. In *the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR*, pages 127–132.

Levenberg, A., C. Callison-Burch, and M. Osborne. 2010. Stream-based Translation Models for Statistical Machine Translation. In *NAACL*.

Nepveu, L., G. Lapalme, P. Langlais, and G. Foster. 2004. Adaptive Language and Translation Models for Interactive Machine Translation. In *EMNLP*.

Simard, M., C. Goutte, and P. Isabelle. 2007. Statistical Phrase-based Post-editing. In *Proceedings of NAACL HLT*, pages 508–515.

Tiedemann, J. 2009. News from OPUS-A collection of multilingual parallel corpora with tools and interfaces. In *Recent Advances in Natural Language Processing*, volume 5, pages 237–248.

Tiedemann, J. 2010. Context adaptation in statistical machine translation using models with exponentially decaying cache. In *DANLP*.