A general framework for minimizing translation effort: towards a principled combination of translation technologies in computer-aided translation

Mikel L. Forcada **Felipe Sánchez-Martínez**
Dept. de Llenguatges i Sistemes Informàtics
Universitat d'Alacant
E-03071 Alacant (Spain)
{mlf,fsanchez}@dlsi.ua.es

Abstract

This paper motivates the need for an homogeneous way of measuring and estimating translation effort (quality) in computer-aided translation. It then defines a general framework for the measurement and estimation of translation effort so that translation technologies can be both optimized and combined in a principled manner. In this way, professional translators will benefit from the seamless integration of all the technologies at their disposal when working on a translation job.

1 Introduction

Imagine that you are a professional translator and you are given a translation job. The text to be translated comes divided in N segments, $s_1, s_2, \ldots s_N$: your job is therefore the ordered set $\{s_i\}_{i=1}^N$. You are supposed to turn this into a translation, that is, an ordered set $\{t_i\}_{i=1}^N$ with the translations of each segment, and get paid for that. Yes, this is a simplified view of your work: the translation of each sentence is treated as an independent event, which is not always the case. In any case, even in this simplified form, the job is already quite challenging.

Help coming your way? Of course, you could translate each segment s_i by hand, i.e. from scratch, into an suitable segment t_i in the target language. You are a translator and you know this is usually harder than most people think. However, there are *translation technologies* out there that are supposed to help you by reducing your translation effort; they usually come packaged as *computer-aided translation* (CAT).

Machine translation: You could, for instance, use *machine translation* (MT) to get a draft of the translation of each segment, $\mathrm{MT}(s_i)$; vendors and experts tell you that you will save effort by *post-editing* $\mathrm{MT}(s_i)$ into your desired translation t_i.[1] Machine translation output $\mathrm{MT}(s_i)$ may just be text, but it could come with annotations to help you make the most of it; for instance, words could be color-coded according to how confident the system is about them (Ueffing and Ney, 2007; Ueffing and Ney, 2005; Blatz et al., 2004), or unknown words that come out untranslated may be marked so that you spot them clearly. Machine-translated segments could even be accompanied by indicators of their estimated quality (Specia and Soricut, 2013; Specia et al., 2010; Blatz et al., 2004) which may be used to ascertain whether the output of the MT system is worth being post-edited or not. If someone measured your post-editing effort (in time, in number of keystrokes, in number of words changed, in money you would have to pay another translator to do it, etc.), when turning $\mathrm{MT}(s_i)$ into t_i, they could call that effort e_i^{MT}.

Translation memory: You could also use a *translation memory* (TM; (Somers, 2003)), where previously translated segments s are stored together with their translations t in pairs called translation units (s, t). The software searches the TM for the source segment $s_i^\star$ that best matches each one of your segments s_i, and delivers the corresponding

[1] We leave aside the debate about whether post-editors should be considered different from translators, as in the end of the day, you will be producing a translation which should be adequate for the purpose at hand, and you will be as responsible of it as if you had produced it from scratch; we will therefore call everything *translation* for the purpose of this paper.

target segment $t_i^\star$ as a proposal, but not alone: it also gives you information about how good the match was between your new segment s_i and the best *fuzzy match* $s_i^\star$ —usually as a percentage called *fuzzy match score* that accounts for the amount of text that is common to both segments[2]— and even marks for you the words in $s_i^\star$ that do not match those in s_i. Let's call all this information $\mathrm{TM}(s_i)$: your job is to use it to turn $t_i^\star$ into the final translation t_i. If the *fuzzy match* is good, you will spend less effort than if you started from scratch. Let us call e_i^{TM} the effort to turn the $t_i^\star$ provided by $\mathrm{TM}(s_i)$ into the desired translation t_i.[3]

Mixing them up: You could even have available another technology, *fuzzy-match repair* (FMR; (Ortega et al., 2014; Dandapat et al., 2011; Hewavitharana et al., 2005; Kranias and Samiotou, 2004)), that integrates the two technologies just mentioned: after a suitable fuzzy match is found, machine translation (or another source of bilingual information) is used to *repair*, i.e. edit some parts of $t_i^\star$, to take into account what changes from $s_i^\star$ to s_i to try to save even more effort; it tells you all that $\mathrm{TM}(s_i)$ tells you, but also marks the parts that have been repaired. Fuzzy-match repair is one of the technologies that TAUS, the Translation Automation User Society, calls *advanced leveraging*;[4] commercial examples of these are DeepMiner in Atril's Déjà Vu,[5] and ALTM in MultiCorpora's MultiTrans.[6] It takes an effort e_i^{FMR} to turn the output of fuzzy-match repair, $\mathrm{FMR}(s_i)$, into the desired t_i.

And many more: To summarize, each technology X you can use —where X may be MT, TM, FMR, etc.— takes each segment s_i and produces an output $X(s_i)$ that takes an effort e_i^{X} to turn into t_i. For a more general discussion, we will also consider a technology the case in which no technology is used, i.e. when the translation is performed from scratch: technologies may not be helpful at all sometimes. We will call $\mathcal{X}$ the set of all technologies X available in the CAT environment. Note that there is another simplification here: the effort e_i^{X} is assumed to depend only on s_i, but translators may vary over time, either during a job, or between jobs; they may become tired, or the effectiveness of each technology may vary.

Isn't this getting too complicated to be considered help? At this point, you are probably wondering how can you decide which technology to use for each segment s_i if the information available for each technology, such as quality indicators in the case of machine translation and fuzzy matching scores in the case of translation memories, are not directly comparable. Or even better, couldn't the decision of selecting the best technology $X_i^\star$, that is, the one that minimizes your effort for each segment s_i, be made automatically?

It is therefore clear that a framework that allows to seamlessly integrate all the translation technologies available in the CAT system is very much needed to make the most of all of them and minimize translation effort as much as possible.

Previous work on technology selection: The specific case of automatically choosing between machine translation output and translation memory fuzzy matches has received attention in the last years. Simard and Isabelle (2009) proposed a simple approach called β-combination, which simply selects machine translation when there is no translation memory proposal with a fuzzy match score above a given threshold β, which can be tuned. He et al. (2010a) and He et al. (2010b) approach this problem, which they call *translation recommendation*, by training a classifier which selects which of the two, $\mathrm{TM}(s_i)$ or $\mathrm{MT}(s_i)$, gets the lowest value for an approximate indicator of effort, called *translation error rate* (TER, (Snover et al., 2006)). Their training compares outputs to preexisting reference translations; their ideas are generalized in the approach proposed in this paper.

The next section explains two ways to minimize the effort needed to perform a translation job in a CAT environment integrating different technologies. Section 3 then describes our proposal for a general framework for training the whole CAT environment. Finally, we discuss the implications of having such a framework.

[2]The fuzzy match score is usually based on a text similarity measure like the word-level edit distance or Levenshtein distance (Levenshtein, 1966). Commercial CAT systems use trade-secret, proprietary versions of it aimed at estimating better than the edit distance the remaining effort.

[3]Interestingly enough, in contrast with the case of machine translation, even if you are actually post-editing a fuzzy-matched proposal, there does not seem to be much debate as to whether you are a translator or a *fuzzy match post-editor*.

[4]http://www.taus.net/reports/advanced-leveraging

[5]http://tinyurl.com/x3dejavu

[6]http://multicorpora.com/resources/advanced-leveraging/

2 Minimizing translation effort

Translation technology designers understand that translators want to minimize their total effort. To compute this total effort, the actual measurements of effort e_i^X need to be *extensive* magnitudes,[7] that is, magnitudes that grow with the length of the segment and make sense when added for all the segments of the job. Examples of extensive measurements have already been given: amount of words or characters changed, total amount of keystrokes, time spent in editing and total price.[8] These are magnitudes that can be compared regardless of technology and allow the total cost of a new translation job to be simply calculated as

$$E = \sum_{i=1}^{N} e_i^{X_i^\star}$$

where $e_i^{X_i^\star}$ is the effort expended in translating segment s_i using the best technology $X_i^\star$ for that segment, that is, the one that minimizes that effort.

To minimize the translation effort on a specific task, designers have to work in two main areas:

Improving each technology: One is to *improve the output of each technology* X, ideally focusing on those cases when X is going to be selected. Some such technologies have tunable parameters; for instance, *feature weights* in statistical MT (Koehn, 2010, p. 255); for other technologies, this is not usually reported, but it is not impossible to think, for instance, of fuzzy-match scores that give different weights to different kinds of edit operations. Let us call $\vec{\lambda}^X$ the vector of tunable parameters for technology X; as the output of technology X varies with these parameters, we can write its output like this: $X(s_i; \vec{\lambda}^X)$.

Learning to select the best technology: The other one is that the CAT environment needs a way to select the best technology $X_i^\star$ for each segment s_i, obviously without measuring the actual effort. To do this, CAT designers need to come up with a set of estimators $\tilde{e}^X$, one for each technology. These estimators should be trained to give the best possible estimate of the actual measured effort $e^X(X(s_i; \vec{\lambda}^X))$. If we call $\vec{\theta}^X$ the set of tunable parameters of the estimator for technology X, the output of the estimator for X can be written as $\tilde{e}^X(X(s_i; \vec{\lambda}^X); \vec{\theta}^X)$. These estimators can be used to estimate the total cost of a new translation job as

$$\tilde{E} = \sum_{i=1}^{N} \tilde{e}^{X_i^\star}(X_i^\star(s_i; \vec{\lambda}^{X_i^\star}); \vec{\theta}^{X_i^\star}),$$

where

$$X_i^\star = \operatorname*{argmin}_{X \in \mathcal{X}} \tilde{e}^X(X(s_i; \vec{\lambda}^X); \vec{\theta}^X).$$

In the case of MT, the estimators of effort (Specia and Soricut, 2013; Specia et al., 2010; Blatz et al., 2004) are based on a number of features obtained from s_i and $\mathrm{MT}(s_i; \vec{\lambda}^{\mathrm{MT}})$. The vector of parameters $\vec{\theta}^{\mathrm{MT}}$ is tuned on a development set made of bilingual segments and translation effort measurements $e^{\mathrm{MT}}(\mathrm{MT}(s_i; \vec{\lambda}^{\mathrm{MT}}))$.[9]

Getting a good estimate of effort is hard: One problem for technologists is that actual measurements of effort are expensive to collect, and they are not likely to be available for all technologies and for all segments. Therefore it is in principle not easy to determine the parameters $\vec{\theta}^X$ to get good estimates $\tilde{e}^X(X(s_i; \vec{\lambda}^X); \vec{\theta}^X)$. We will see a way to do this below.

Tuning technologies is also hard: Technologies may have tunable parameters $\vec{\lambda}^X$ which determine the output they produce. Obviously, one cannot just repetitively measure the actual effort spent by translators in editing their output for a wide variety of values of $\vec{\lambda}^X$, as this is clearly impracticable; therefore, an alternative is needed. When $X = \mathrm{MT}$, this is usually done by means of an algorithm that optimizes (Och, 2003; Chiang, 2012) *automatic evaluation measures*, such as BLEU (Papineni et al., 2002), using *reference translations* in a *development set*. Most of these automatic

[7] This concept is borrowed from physics: see, e.g., `http://en.wikipedia.org/wiki/Intensive_and_extensive_properties`.

[8] Examples of measurements that are not extensive, that is, *intensive*, could be the percentage of words or characters changed, the ratio of the total amount of keystrokes to the sentence length, the time spent per word, or the price per word. These are intensive properties as they are the ratio of two extensive properties.

[9] The measurements of effort that one can find in literature vary from simple scores for "perceived" post-editing effort (usually scores taking 3 or 4 values) to actual post-editing time (see, for instance, the quality estimation task in WMT 2014 (Bojar et al., 2014))

measures are measures of similarity (or dissimilarity) between raw and reference translations. Researchers hope that their use during tuning will lead to a reduction in translation effort, although this is not currently guaranteed —for instance, Denkowski and Lavie (2012) found that BLEU could not distinguish between raw and post-edited machine translation. Generally, an automatic evaluation measure for technology X may have the form $\hat{e}^X(X(s_i;\vec{\lambda}^X),\{t_{ij}\}_{j=1}^{n_i};\vec{\mu}^X)$, where $\{t_{ij}\}_{j=1}^{n_i}$ is the set of reference translations for segment s_i in the development set and $\vec{\mu}^X$ is a set of tunable parameters. Ideally, $\hat{e}^X(X(s_i;\vec{\lambda}^X),\{t_{ij}\}_{j=1}^{n_i};\vec{\mu}^X)$ should approximate $e^X(X(s_i;\vec{\lambda}^X))$, but tuning of $\vec{\mu}^X$ is surprisingly absent from current MT practice (with some exceptions, see Denkowski and Lavie (2010)). In fact, $\hat{e}^X(X(s_i;\vec{\lambda}^X),\{t_{ij}\}_{j=1}^{n_i};\vec{\mu}^X)$ can be seen as a special estimator of effort, much like $\tilde{e}^X(X(s_i;\vec{\lambda}^X);\vec{\theta}^X)$, but informed with reference translations $\{t_{ij}\}_{j=1}^{n_i}$ when they are available. This is similar to the use of pseudo-reference translations in machine translation quality estimation (Shah et al., 2013; Soricut and Narsale, 2012; Soricut et al., 2012; Soricut and Echihabi, 2010), but with actual references.

Table 1 summarizes the main concepts and the notation used along the paper.

3 A general framework for training the whole CAT environment

We describe a possible workflow to tune simultaneously the different technologies that may be used in a CAT environment and the estimators used to select them on a segment basis:

1. Design automatic evaluation measures $\hat{e}^X(X(s_i,\vec{\lambda}^X),\{t_{ij}\};\vec{\mu}^X)$ and estimators $\tilde{e}^X(X(s_i;\vec{\lambda}^X);\vec{\theta}^X)$ for each technology $X\in\mathcal{X}$, based on a series of relevant features that can easily be extracted from s_i and $X(s_i)$, and which will depend on parameters $\vec{\mu}^X$ and $\vec{\theta}^X$, respectively.

2. Give reasonable starting values to the parameters of $\tilde{e}^X(X(s_i;\vec{\lambda}^X);\vec{\theta}^X)$ for each technology X in $\mathcal{X}$, so that they can be used to preliminarily select technologies. These initial estimators $\tilde{e}_0^X(X(s_i;\vec{\lambda}^X);\vec{\theta}^X)$ could, for instance, be the ones that worked well for a related task.

3. Put together a development set $D=\{s_k\}_{k=1}^{n_D}$ having n_D segments, representative of the task at hand, and which provides reference translations $\{t_{kl}\}_{l=1}^{n_k}$ for each segment. This development set should be large enough to be used to tune the technologies in step 7; it could also be used to pre-tune the technologies (for instance, statistical machine translation could be pre-tuned using customary evaluation measures such as BLEU). Development sets of thousands of segments are common, for instance, in statistical machine translation, but the actual number may depend on the number of parameters in each $\vec{\lambda}^X$.

4. Have translators work on a representative subset $M=\{s_k\}_{k=1}^{n_M}$ of the development set D, and measure their effort when translating each segment using the best technology, selected according to the available version of estimators $\tilde{e}^X(X(s_k;\vec{\lambda}^X);\vec{\theta}^X)$. Of course, M is a small *subset* of D because translating thousands of sentences, as in a typical development set, is clearly out of the question, as this would be more like the size of a whole translation job. Note that translator work can add additional references for those segments in D which are also in M. Note also that we might be combining here measurements for a team of more than one translator. Therefore, the resulting combination of technologies may not be expected to be optimal for each individual translator, but rather *on average* for the team.

   A richer set of measurements could be obtained by having translators translate segments in M using also other technologies not selected by the estimators. This would be costly but could help mitigate the bias introduced by the initial set of estimators.

   To get evaluation measures $\hat{e}_i^X$ and estimators $\tilde{e}_i^X$ which are useful in the scenario of a new translation job, translation memories are not allowed to grow or change when translators translate the set M, and the resulting reference set is fixed after this step.

5. Use these measurements to fit all the automatic evaluation measures $\hat{e}^X(X(s_i,\vec{\lambda}^X),\{t_{ij}\};\vec{\mu}^X)$ together by varying their vectors $\vec{\mu}^X$ by means of eq. (1):

Notation	Definition
$\{s_i\}_{i=1}^{N}$	Translation task made up of N source segments s_i.
X	Translation technology; e.g. $X = \mathrm{MT}$, machine translation; $X = \mathrm{TM}$, translation memory; $X = \mathrm{FMR}$, fuzzy-match repair; etc.
$\mathcal{X}$	The set of all translation technologies available.
$X_i^{\star}$	Technology selected for the translation of the source segment s_i.
$X(s_i; \vec{\lambda}^X)$, also $X(s_i)$	Output produced by translation technology X for input segment s_i. Many provide additional information in addition to its output.
$\vec{\lambda}^X$	Optional vector of tunable parameters used by translation technology X to produce the best possible translation proposal.
$e^X(X(s_i; \vec{\lambda}^X))$, also e_i^X	Actual measured effort to produce an adequate translation starting from the proposal provided by technology X.
$\tilde{e}^X(X(s_i; \vec{\lambda}^X); \vec{\theta}^X)$, also $\tilde{e}_i^X$	Estimated effort to produce an adequate translation starting from the proposal provided by technology X.
$\vec{\theta}^X$	Optional vector of tunable parameters used in the estimator of effort $\tilde{e}^X(\cdot)$. The parameters may be tuned so that the estimated effort is as close as possible to the measured effort.
$\hat{e}^X(X(s_i; \vec{\lambda}^X), \{t_{ij}\}_{j=1}^{n_i}; \vec{\mu}^X)$, also $\hat{e}_i^X$	Estimated effort to produce an adequate translation starting from the proposal of technology X, specifically informed by a set of reference translations (sometimes called *automatic evaluation measure*).
$\{t_{ij}\}_{j=1}^{n_i}$, also $\{t_{ij}\}$	Set of *reference translations* for segment s_i.
$\vec{\mu}^X$	Optional vector of tunable parameters used in estimator $\hat{e}^X(\cdot)$. These parameters may be tuned so that the estimated effort is as close as possible to the measured effort (parameters are seldom tuned in *automatic evaluation measures*).

Table 1: A summary of the main concepts defined in the paper and the notation used.

$$\min_{\{\vec{\mu}^X\}_{X\in\mathcal{X}}} \sum_{k\in[1,n_M]} \mathcal{L}\left(\hat{e}^{X_k^\star}(X_k^\star(s_k,\vec{\lambda}^{X_k^\star}),\{t_{kl}\}_{l=1}^{n_k};\vec{\mu}^{X_k^\star}), e^{X_k^\star}(X_k^\star(s_k;\vec{\lambda}^{X_k^\star}))\right) \quad (1)$$

Here, $\mathcal{L}(x)$ is ideally a differentiable loss function (for instance $\mathcal{L}(x)=\frac{1}{2}x^2$), and $X_k^\star$ is the technology selected for s_k by using the initial estimator (if measurements had been made in step 4 for more than one technology per segment, they could be used here to get a better approximation). The result is a set of functions $\{\hat{e}^X\}$ which estimate, using reference translations, the effort needed to deal with the output of each technology X, without having to actually measure that effort.

6. **Training the technology selectors:** Use the available measurements of effort e_k^X where they are available, and the automatic evaluation measures $\hat{e}^X(X(s_k),\{t_{kl}\}_{l=1}^{n_k};\vec{\mu}^X)$ where they are not, to obtain a set of better estimators $\tilde{e}^X(X(s_k;\vec{\lambda}^X);\vec{\theta}^X)$ by varying their vectors $\vec{\theta}^X$ using the whole development set D and eq. (2):

$$\min_{\{\vec{\theta}^X\}_{X\in\mathcal{X}}} \sum_{k\in[1,n_D]} \mathcal{L}\left(\tilde{e}^{X_k^\star}(X_k^\star(s_k,\vec{\lambda}^{X_k^\star});\vec{\theta}^{X_k^\star}), \bar{e}^{X_k^\star}(X_k^\star(s_k,\vec{\lambda}^{X_k^\star}),\{t_{kl}\}_{l=1}^{n_k};\vec{\mu}^{X_k^\star})\right) \quad (2)$$

Here, $\bar{e}^X(X(s_k,\vec{\lambda}^X),\{t_{kl}\}_{l=1}^{n_k};\vec{\mu}^X)$ is the actual effort measured $e^X(X(s_k;\vec{\lambda}^X))$ if it is available, and $\hat{e}^X(X(s_k,\vec{\lambda}^{X_k^\star}),\{t_{kl}\}_{l=1}^{n_k};\vec{\mu}^X)$ otherwise, and $X_k^\star$ is the actual selection for those segments where measurements were taken, or the technology with the best $\hat{e}^X(X(s_k,\vec{\lambda}^X),\{t_{kl}\}_{l=1}^{n_k};\vec{\mu}^X)$ where they were not. The result is a set of functions $\{\tilde{e}^X\}$ which estimate, in the absence of reference translations, the effort needed to deal with the output of each technology X, without having to actually measure that effort. These functions will be used in the translator's CAT tool to automatically select the best technology for each segment.

7. **Training the technologies themselves:** The same development set, and the functions $\{\hat{e}^X\}$ may be used to train —or, as usually said in statistical machine translation, to *tune* — the technologies themselves by searching for those values of $\vec{\lambda}^X$ leading to the minimum effort for translators:

$$\min_{\{\vec{\lambda}^X\}_{X\in\mathcal{X}}} \sum_{k\in[1,n_D]} \hat{e}^{X_k^\star}(X_k^\star(s_k,\vec{\lambda}^{X_k^\star}),\{t_{kl}\}_{l=1}^{n_k};\vec{\mu}^{X_k^\star}), \quad (3)$$

where $X_k^\star$ is the translation technology with the best $\hat{e}^X(X(s_k,\vec{\lambda}^X),\{t_{kl}\}_{l=1}^{n_k};\vec{\mu}^X)$.

This training would be analogous to minimum-error-rate training (MERT) (Och, 2003), MIRA (Hasler et al., 2011), or similar iterative methods used in statistical machine translation to vary the parameters of a system and optimize the output with respect to a certain quality indicator such as BLEU (Papineni et al., 2002), but it would be applied to all sources $X\in\mathcal{X}$ and use the estimators $\hat{e}_i^X$ of extensive effort measurements tuned in step 5. Eq. (3) tunes each technology using only those segments for which it was selected. Alternatively, one may prefer to optimize all technologies on all segments as follows:

$$\min_{\{\vec{\lambda}^X\}_{X\in\mathcal{X}}} \sum_{X\in\mathcal{X}} \sum_{k\in[1,n_D]} \hat{e}^X(X(s_k,\vec{\lambda}^X),\{t_{kl}\}_{l=1}^{n_k};\vec{\mu}^X), \quad (4)$$

and run the risk of spreading optimization too thin to significantly optimize those technologies likely to be actually selected in a real translation job.

The result is a set of technologies $\mathcal{X}$ that are (approximately) optimized to reduce effort. These technologies will be used in the CAT tool to provide the best possible assistance to translators.

Note that the new estimators obtained in step 6 could have led to different technology choices, and therefore different measurement sets, if these choices had been made in step 4. This coupling between parameter sets should in principle be taken into account in an improved setting by feeding this all the way back to step 4 in some way to achieve self-consistency while keeping the need for additional effort measurements, that is, additional manual translations of segments in M, to a minimum; feasible or approximate ways of doing this should definitely be explored.

Note also that the workflow above is a *batch* workflow. *Online* workflows which improve the technologies X as translators work, would certainly be more complex, but could be devised following the rationale behind the batch workflow just described, once it is proven useful.

4 Discussion

We have introduced a unified, general framework for effort (quality) measurement, evaluation and estimation. This framework allows to simultaneously tune all the components of a computer-aided translation environment. To that end, we propose the use of estimators of remaining effort that are comparable across translation-assistance technologies.

On the one hand, tuning all the translation technologies together (which is not common in current practice), and in a way that takes into account when they are actually selected to produce a proposal for the translator, will lead to an improvement of the translation proposals these technologies produce when this improvement is relevant; that is, when the technology is actually used to propose a translation to the translator.

On the other hand, having estimators of effort whose output is comparable across technologies will allow for seamless integration of translation technologies: the CAT tool will be able to automatically select the best technology on a segment basis. In addition, if the estimations of effort are measured in time spent in editing, or in money, they could be used to accurately budget new translation jobs.

The framework proposed in this paper provides a principled way to adapt the mix of technologies to reduce total effort in a specific computer-aided translation job.

We hope that this unified framework will ease the integration of existing research being performed to actually reduce translators' effort and improve their productivity. We also hope that it will inspire new approaches and encourage best practice in computer-aided translation research and development.

Acknowledgements: We acknowledge support from the Spanish Ministry of Industry and Competitiveness through project Ayutra (TIC2012-32615) and from the European Commission through project Abu-Matran (FP7-PEOPLE-2012-IAPP, ref. 324414) and thank all three anonymous referees for very useful comments on the paper.

References

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

Bojar, O., C. Buck, C. Federmann, B. Haddow, P. Koehn, J. Leveling, C. Monz, P. Pecina, M. Post, H. Saint-Amand, R. Soricut, L. Specia, and A. Tamchyna. 2014. Findings of the 2014 workshop on statistical machine translation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 12–58, Baltimore, Maryland, USA.

Chiang, D. 2012. Hope and fear for discriminative training of statistical translation models. *Journal of Machine Learning Research*, 13:1159–1187.

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

Denkowski, M. and A. Lavie. 2010. METEOR-NEXT and the METEOR paraphrase tables: Improved evaluation support for five target languages. In *Proceedings of the Joint 5th Workshop on Statistical Machine Translation and MetricsMATR (Uppsala, Sweden)*, page 339–342.

Denkowski, M. and A. Lavie. 2012. Challenges in predicting machine translation utility for human post-editors. In *Proceedings of The Tenth Biennial Conference of the Association for Machine Translation in the Americas*, San Diego, CA, USA.

Hasler, Eva, Barry Haddow, and Philipp Koehn. 2011. Margin infused relaxed algorithm for moses. *The Prague Bulletin of Mathematical Linguistics*, 96:69–78.

He, Y., Y. Ma, J. van Genabith, and A. Way. 2010a. Bridging SMT and TM with translation recommendation. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 622–630, Uppsala, Sweden.

He, Yifan, Yanjun Ma, Johann Roturier, Andy Way, and Josef van Genabith. 2010b. Improving the post-editing experience using translation recommendation: A user study. In *Proceedings of the Ninth Conference of the Association for Machine Translation in the Americas*.

Hewavitharana, S., S. Vogel, and A. Waibel. 2005. Augmenting a statistical translation system with a translation memory. In *Proceedings of the 10th conference of the European Association for Machine Translation*, pages 126–132, Budapest, Hungary.

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

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

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

Och, F.J. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*, pages 160–167, Sapporo, Japan.

Ortega, J. E., M. L. Forcada, and F. Sánchez-Martínez. 2014. Using any machine translation source for fuzzy-match repair in a computer-aided translation setting. In al Onaizan, Yaser and Michel Simard, editors, *Proceedings of the 11th Conference of the Association for Machine Translation in the Americas*, volume 1: MT Researchers, pages 42–53, Vancouver, BC, Canada.

Papineni, K., S. Roukos, T. Ward, and W. J. Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, PA, USA.

Shah, K., T. Cohn, and L. Specia. 2013. An investigation on the effectiveness of features for translation quality estimation. In *Proceedings of the XIV Machine Translation Summit*, pages 167–174, Nice, France.

Simard, M. and P. Isabelle. 2009. Phrase-based machine translation in a computer-assisted translation environment. In *Proceedings of the 12th Machine Translation Summit*, pages 120–127, Ottawa, Canada.

Snover, M., B. Dorr, R. Schwartz, L. Micciulla, and J. Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proceedings of Association for Machine Translation in the Americas Conference*, pages 223–231, August.

Somers, H., 2003. *Computers and translation: a translator's guide*, chapter Translation memory systems, pages 31–48. John Benjamins Publishing, Amsterdam, Netherlands.

Soricut, R. and A. Echihabi. 2010. Trustrank: Inducing trust in automatic translations via ranking. In *In Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 612–621, Uppsala, Sweden.

Soricut, R. and S. Narsale. 2012. Combining quality prediction and system selection for improved automatic translation output. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 163–170, Montreal, Canada.

Soricut, R., N. Bach, and Z. Wang. 2012. The SDL Language Weaver systems in the WMT12 quality estimation shared task. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 145–151, Montreal, Canada.

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

Specia, L., D. Raj, and M. Turchi. 2010. Machine translation evaluation versus quality estimation. *Machine Translation*, 24(1):39–50.

Ueffing, Nicola and Hermann Ney. 2005. Word-level confidence estimation for machine translation using phrase-based translation models. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, pages 763–770.

Ueffing, Nicola and Hermann Ney. 2007. Word-level confidence estimation for machine translation. *Computational Linguistics*, 33(1):9–40, March.