Statistical Phrase-based Post-editing

Michel Simard **Cyril Goutte** **Pierre Isabelle**
Interactive Language Technologies
National Research Council of Canada
Gatineau, Canada, K1A 0R6
FirstName.LastName@nrc.gc.ca

Abstract

We propose to use a statistical phrase-based machine translation system in a *post-editing* task: the system takes as input raw machine translation output (from a commercial rule-based MT system), and produces post-edited target-language text. We report on experiments that were performed on data collected in precisely such a setting: pairs of raw MT output and their manually post-edited versions. In our evaluation, the output of our *automatic post-editing* (APE) system is not only better quality than the rule-based MT (both in terms of the BLEU and TER metrics), it is also better than the output of a state-of-the-art phrase-based MT system used in standalone translation mode. These results indicate that automatic post-editing constitutes a simple and efficient way of combining rule-based and statistical MT technologies.

1 Introduction

The quality of machine translation (MT) is generally considered insufficient for use in the field without a significant amount of human correction. In the translation world, the term *post-editing* is often used to refer to the process of manually correcting MT output. While the conventional wisdom is that post-editing MT is usually not cost-efficient compared to full human translation, there appear to be situations where it is appropriate and even profitable. Unfortunately, there are few reports in the literature about such experiences (but see Allen (2004) for examples).

One of the characteristics of the post-editing task, as opposed to the revision of human translation for example, is its partly repetitive nature. Most MT systems invariably produce the same output when confronted with the same input; in particular, this means that they tend to make the same mistakes over and over again, which the post-editors must correct repeatedly. Batch corrections are sometimes possible when multiple occurrences of the same mistake appear in the same document, but when it is repeated over several documents, or equivalently, when the output of the same machine translation system is handled by multiple post-editors, then the opportunities for factoring corrections become much more complex. MT users typically try to reduce the post-editing load by customizing their MT systems. However, in Rule-based Machine Translation (RBMT), which still constitutes the bulk of the current commercial offering, customization is usually restricted to the development of "user dictionaries". Not only is this time-consuming and expensive, it can only fix a subset of the MT system's problems.

The advent of Statistical Machine Translation, and most recently phrase-based approaches (PBMT, see Marcu and Wong (2002), Koehn et al. (2003)) into the commercial arena seems to hold the promise of a solution to this problem: because the MT system learns directly from existing translations, it can be automatically customized to new domains and tasks. However, the success of this operation cru-

Proceedings of NAACL HLT 2007, pages 508–515,
Rochester, NY, April 2007.

cially depends on the amount of training data available. Moreover, the current state of the technology is still insufficient for consistently producing human readable translations.

This state of affairs has prompted some to examine the possibility of automating the post-editing process itself, at least as far as "repetitive errors" are concerned. Allen and Hogan (2000) sketch the outline of such an automated post-editing (APE) system, which would automatically learn post-editing rules from a *tri-parallel* corpus of source, raw MT and post-edited text. Elming (2006) suggests using tranformation-based learning to automatically acquire error-correcting rules from such data; however, the proposed method only applies to lexical choice errors. Knight and Chander (1994) also argue in favor of using a separate APE module, which is then portable across multiple MT systems and language pairs, and suggest that the post-editing task could be performed using statistical machine translation techniques. To the best of our knowledge, however, this idea has never been implemented.

In this paper, we explore the idea of using a PBMT system as an automated post-editor. The underlying intuition is simple: if we collect a parallel corpus of raw machine-translation output, along with its human-post-edited counterpart, we can train the system to translate from the former into the latter. In section 2, we present the case study that motivates our work and the associated data. In section 3, we describe the phrase-based post-editing model that we use for improving the output of the automatic translation system. In section 4, we illustrate this on a dataset of moderate size containing job ads and their translation. With less than 500k words of training material, the phrase-based MT system already outperforms the rule-based MT baseline. However, a phrase-based post-editing model trained on the output of that baseline outperforms both by a fairly consistent margin. The resulting BLEU score increases by up to 50% (relative) and the TER is cut by one third.

2 Background

2.1 Context

The Canadian government's department of Human Resources and Social Development (HRSDC) maintains a web site called *Job Bank*,[1] where potential employers can post ads for open positions in Canada. Over one million ads are posted on *Job Bank* every year, totalling more than 180 million words. By virtue of Canada's Official Language Act, HRSDC is under legal obligation to post all ads in both French and English. In practice, this means that ads submitted in English must be translated into French, and vice-versa.

To address this task, the department has put together a complex setup, involving text databases, translation memories, machine translation and human post-editing. Employers submit ads to the *Job Bank* website by means of HTML forms containing "free text" data fields. Some employers do periodical postings of identical ads; the department therefore maintains a database of previously posted ads, along with their translations, and new ads are systematically checked against this database. The translation of one third of all ads posted on the *Job Bank* is actually recuperated this way. Also, employers will often post ads which, while not entirely identical, still contain identical sentences. The department therefore also maintains a translation memory of individual sentence pairs from previously posted ads; another third of all text is typically found *verbatim* in this way.

The remaining text is submitted to machine translation, and the output is post-edited by human experts. Overall, only a third of all submitted text requires human intervention. This is nevertheless very labour-intensive, as the department tries to ensure that ads are posted at most 24 hours after submission. The *Job Bank* currently employs as many as 20 post-editors working full-time, most of whom are junior translators.

2.2 The Data

HRSDC kindly provided us with a sample of data from the *Job Bank*. This corpus consists in a collection of parallel "blocks" of textual data. Each block contains three parts: the source language text, as submitted by the employer, its machine-translation, produced by a commercial rule-based MT system, and its final post-edited version, as posted on the website.

[1] `http://www.jobbank.gc.ca`

The entire corpus contains less than one million words in each language. This corresponds to the data processed in less than a week by the *Job Bank*. Basic statistics are given in Table 1 (see Section 4.1). Most blocks contain only one sentence, but some blocks may contain many sentences. The longest block contains 401 tokens over several sentences. Overall, blocks are quite short: the median number of tokens per source block is only 9 for French-to-English and 7 for English-to-French. As a consequence, no effort was made to segment the blocks further for processing.

We evaluated the quality of the Machine Translation contained in the corpus using the Translation Edit Rate (TER, cf. Snover et al. (2006)). The TER counts the number of edit operations, including phrasal shifts, needed to change a hypothesis translation into an adequate and fluent sentence, and normalised by the length of the final sentence. Note that this closely corresponds to the post-editing operation performed on the *Job Bank* application. This motivates the choice of TER as the main metric in our case, although we also report BLEU scores in our experiments. Note that the emphasis of our work is on reducing the *post-edition effort*, which is well estimated by TER. It is not directly on *quality* so the question of which metric better estimates translation quality is not so relevant here.

The global TER (over all blocks) are 58.77% for French-to-English and 53.33% for English-to-French. This means that more than half the words have to be post-edited in some way (delete / substitute / insert / shift). This apparently harsh result is somewhat mitigated by two factors.

First, the distribution of the block-based TER[2] shows a large disparity in performance, cf. Figure 1. About 12% of blocks have a TER higher than 100%: this is because the TER normalises on the length of the references, and if the raw MT output is longer than its post-edited counterpart, then the number of edit operations may be larger than that length.[3] At the other end of the spectrum, it is also clear that many blocks have low TER. In fact more than 10% have a TER of 0. The global score therefore hides a large range of performance.

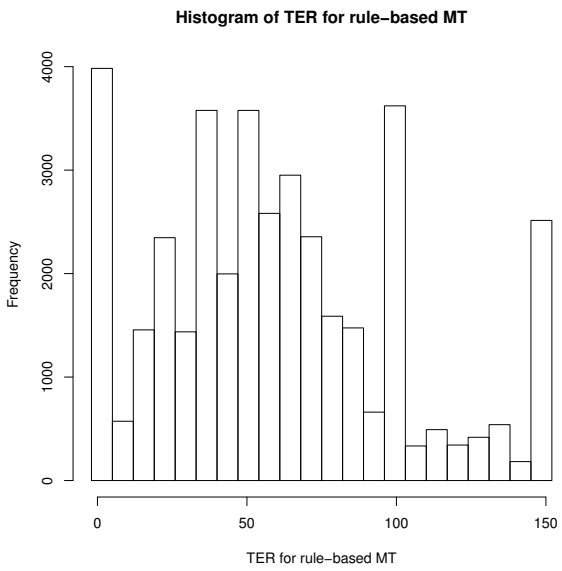

Figure 1: Distribution of TER on 39005 blocks from the French-English corpus (thresholded at 150%).

The second factor is that the TER measures the distance to an adequate *and fluent* result. A high TER does not mean that the raw MT output is not understandable. However, many edit operations may be needed to make it fluent.

3 Phrase-based Post-editing

Translation post-editing can be viewed as a simple transformation process, which takes as input raw target-language text coming from a MT system, and produces as output target-language text in which "errors" have been corrected. While the automation of this process can be envisaged in many different ways, the task is not conceptually very different from the translation task itself. Therefore, there doesn't seem to be any good reason why a machine translation system could not handle the post-editing task. In particular, given such data as described in Section 2.2, the idea of using a statistical MT system for post-editing is appealing. *Portage* is precisely such a system, which we describe here.

Portage is a phrase-based, statistical machine translation system, developed at the National Research Council of Canada (NRC) (Sadat et al.,

[2]Contrary to BLEU or NIST, the TER naturally decomposes into block-based scores.

[3]A side effect of the normalisation is that larger TER are measured on small sentences, e.g. 3 errors for 2 reference words.

2005). A version of the Portage system is made available by the NRC to Canadian universities for research and education purposes. Like other SMT systems, it learns to translate from existing parallel corpora.

The system translates text in three main phases: preprocessing of raw data into tokens; decoding to produce one or more translation hypotheses; and error-driven rescoring to choose the best final hypothesis. For languages such as French and English, the first of these phases (tokenization) is mostly a straightforward process; we do not describe it any further here.

Decoding is the central phase in SMT, involving a search for the hypotheses t that have highest probabilities of being translations of the current source sentence s according to a model for $P(t|s)$. Portage implements a dynamic programming beam search decoding algorithm similar to that of Koehn (2004), in which translation hypotheses are constructed by combining in various ways the target-language part of phrase pairs whose source-language part matches the input. These phrase pairs come from large *phrase tables* constructed by collecting matching pairs of contiguous text segments from word-aligned bilingual corpora.

Portage's model for $P(t|s)$ is a log-linear combination of four main components: one or more n-gram target-language models, one or more phrase translation models, a distortion (word-reordering) model, and a sentence-length feature. The phrase-based translation model is similar to that of Koehn, with the exception that phrase probability estimates $P(\tilde{s}|\tilde{t})$ are smoothed using the Good-Turing technique (Foster et al., 2006). The distortion model is also very similar to Koehn's, with the exception of a final cost to account for sentence endings.

Feature function weights in the loglinear model are set using Och's minium error rate algorithm (Och, 2003). This is essentially an iterative two-step process: for a given set of source sentences, generate n-best translation hypotheses, that are representative of the entire decoding search space; then, apply a variant of Powell's algorithm to find weights that optimize the BLEU score over these hypotheses, compared to reference translations. This process is repeated until the set of translations stabilizes, i.e. no new translations are produced at the decoding step.

To improve raw output from decoding, Portage relies on a rescoring strategy: given a list of n-best translations from the decoder, the system reorders this list, this time using a more elaborate loglinear model, incorporating more feature functions, in addition to those of the decoding model: these typically include IBM-1 and IBM-2 model probabilities (Brown et al., 1993) and an IBM-1-based feature function designed to detect whether any word in one language appears to have been left without satisfactory translation in the other language; all of these feature functions can be used in both language directions, i.e. source-to-target and target-to-source.

In the experiments reported in the next section, the Portage system is used both as a translation and as an APE system. While we can think of a number of modifications to such a system to better adapt it to the post-editing task (some of which are discussed later on), we have done no such modifications to the system. In fact, whether the system is used for translation or post-editing, we have used exactly the same translation model configuration and training procedure.

4 Evaluation

4.1 Data and experimental setting

The corpus described in section 2.2 is available for two language pairs: English-to-French and French-to-English.[4] In each direction, each block is available in three versions (or *slices*): the original text (or *source*), the output of the commercial rule-based MT system (or *baseline*) and the final, post-edited version (or *reference*).

In each direction (French-to-English and English-to-French), we held out two subsets of approximately 1000 randomly picked blocks. The *validation* set is used for testing the impact of various high-level choices such as pre-processing, or for obtaining preliminary results based on which we setup new experiments. The *test* set is used only once, in order to obtain the final experimental results reported here.

The rest of the data constitutes the *training* set, which is split in two. We sampled a subset of 1000 blocks as *train-2*, which is used for optimiz-

[4]Note that, in a post-editing context, translation direction is crucially important. It is not possible to use the same corpus in both directions.

Corpus	English-to-French				French-to-English			
		words:				words:		
	blocks	source	baseline	reference	blocks	source	baseline	reference
train-1	28577	310k	382k	410k	36005	485k	501k	456k
train-2	1000	11k	14k	14k	1000	13k	14k	12k
validation	881	10k	13k	13k	966	13k	14k	12k
test	899	10k	12k	13k	953	13k	13k	12k

Table 1: Data and split used in our experiments, (in thousand words). 'baseline' is the output of the commercial rule-based MT system and 'reference' is the final, post-edited text.

ing the log-linear model parameters used for decoding and rescoring. The rest is the *train-1* set, used for estimating IBM translation models, constructing phrasetables and estimating a target language model.

The composition of the various sets is detailed in Table 1. All data was tokenized and lowercased; all evaluations were performed independent of case. Note that the *validation* and *test* sets were originally made out of 1000 blocks sampled randomly from the data. These sets turned out to contain blocks identical to blocks from the training sets. Considering that these would normally have been handled by the translation memory component (see the HRSDC workflow description in Section 2.1), we removed those blocks for which the source part was already found in the *training* set (in either *train-1* or *train-2*), hence their smaller sizes.

In order to check the sensitivity of experimental results to the choice of the *train-2* set, we did a run of preliminary experiments using different subsets of 1000 blocks. The experimental results were nearly identical and highly consistent, showing that the choice of a particular *train-2* subset has no influence on our conclusions. In the experiments reported below, we therefore use a single identical *train-2* set.

We initially performed two sets of experiments on this data. The first was intended to compare the performance of the Portage PBMT system as an alternative to the commercial rule-based MT system on this type of data. In these experiments, English-to-French and French-to-English translation systems were trained on the source and reference (manually post-edited target language) slices of the training set. In addition to the target language model estimated on the *train-1* data, we used an external contribution, a trigram target language model trained on a fairly large quantity of data from the Canadian Hansard.

Language	TER	BLEU
English-to-French		
Baseline	53.5	32.9
Portage *translation*	53.7	36.0
Baseline + Portage APE	**47.3**	**41.6**
French-to-English		
Baseline	59.3	31.2
Portage *translation*	43.9	41.0
Baseline + Portage APE	**41.0**	**44.9**

Table 2: Experimental Results: For TER, lower (error) is better, while for BLEU, higher (score) is better. Best results are in bold.

The goal of the second set of experiments was to assess the potential of the Portage technology in automatic post-editing mode. Again, we built systems for both language directions, but this time using the existing rule-based MT output as source and the reference as target. Apart from the use of different source data, the training procedure and system configurations of the translation and post-editing systems were in all points identical.

4.2 Experimental results

The results of both experiments are presented in Table 2. Results are reported both in terms of the TER and BLEU metrics; *Baseline* refers to the commercial rule-based MT output.

The first observation from these results is that, while the performance of Portage in translation mode is approximately equivalent to that of the baseline system when translating into French, its performance is much better than the baseline when translating into English. Two factors possibly contribute

to this result: first, the fact that the baseline system itself performs better when translating into French; second, and possibly more importantly, the fact that we had access to less training data for English-to-French translation.

The second observation is that when Portage is used in automatic post-editing mode, on top of the baseline MT system, it achieves better quality than either of the two translation systems used on its own. This appears to be true regardless of the translation direction or metric. This is an extremely interesting result, especially in light of how little data was actually available to train the post-editing system.

One aspect of statistical MT systems is that, contrary to rule-based systems, their performance (usually) increases as more training data is available. In order to quantify this effect in our setting, we have computed learning curves by training the Portage translation and Portage APE systems on subsets of the training data of increasing sizes. We start with as little as 1000 blocks, which corresponds to around 10-15k words.

Figure 2 (next page) compares the learning rates of the two competing approaches (Portage translation vs. Portage APE). Both approaches display very steady learning rates (note the logarithmic scale for training data size). These graphs strongly suggest that both systems would continue to improve given more training data. The most impressive aspect is how little data is necessary to improve upon the baseline, especially when translating into English: as little as 8000 blocks (around 100k words) for direct translation and 2000 blocks (around 25k words) for automatic post-editing. This suggests that such a post-editing setup might be worth implementing even for specialized domains with very small volumes of data.

4.3 Extensions

Given the encouraging results of the Portage APE approach in the above experiments, we were curious to see whether a Portage+Portage combination might be as successful: after all, if Portage was good at correcting some other system's output, could it not manage to correct the output of another Portage translator?

We tested this in two settings. First, we actually use the output of the Portage translation system obtained above, i.e. trained on the same data. In our second experiment, we use the output of a Portage translator trained on different domain data (the Canadian Hansard), but with much larger amounts of training material (over 85 million words per language). In both sets of experiments, the Portage APE system was trained as previously, but using Portage translations of the Job Bank data as input text.

Language	TER	BLEU
English-to-French		
Portage *Job Bank*	**53.7**	36.0
+ Portage APE	**53.7**	**36.2**
Portage *Hansard*	76.9	13.0
+ Portage APE	64.6	26.2
French-to-English		
Portage *Job Bank*	**43.9**	41.0
+ Portage APE	**43.9**	**41.4**
Portage *Hansard*	80.1	14.0
+ Portage APE	57.7	28.6

Table 3: Portage translation - Portage APE system combination experimental results.

The results of both experiments are presented in Table 3. The first observation in these results is that there is nothing to be gained from post-editing when both the translation and APE systems are trained on the same data sets (Portage *Job Bank* + Portage APE experiments). In other words, the translation system is apparently already making the best possible use of the training data, and additional layers do not help (but nor do they hurt, interestingly).

However, when the translation system has been trained using distinct data (Portage *Hansard* + Portage APE experiments), post-editing makes a large difference, comparable to that observed with the rule-based MT output provided with the *Job Bank* data. In this case, however, the Portage translation system behaves very poorly in spite of the important size of the training set for this system, much worse in fact than the "baseline" system. This highlights the fact that both the *Job Bank* and *Hansard* data are very much domain-specific, and that access to appropriate training material is crucial for phrase-based translation technology.

In this context, combining two phrase-based sys-

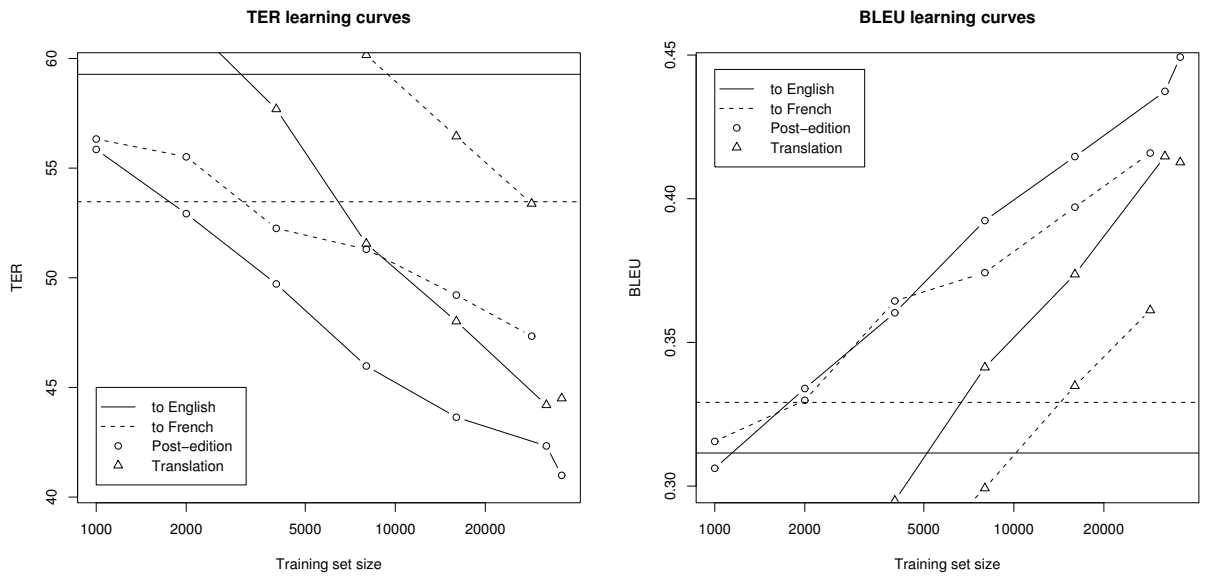

Figure 2: TER and BLEU scores of the phrase-based post-editing models as the amount of training data increases (log scale). The horizontal lines correspond to the performance of the baseline system (rule-based translation).

tems as done here can be seen as a way of adapting an existing MT system to a new text domain; the APE system then acts as an "adapter", so to speak. Note however that, in our experiments, this setup doesn't perform as well as a single Portage translation system, trained directly and exclusively on the *Job Bank* data.

Such an adaptation strategy should be contrasted with one in which the translation models of the old and new domains are "merged" to create a new translation system. As mentioned earlier, Portage allows using multiple phrase translation tables and language models concurrently. For example, in the current context, we can extract phrase tables and language models from the *Job Bank* data, as when training the "Portage *Job Bank*" translation system, and then build a Portage translation model using both the *Hansard* and *Job Bank* model components. Loglinear model parameters are then optimized on the *Job Bank* data, so as to find the model weights that best fit the new domain.

In a straightforward implementation of this idea, we obtained performances almost identical to those of the Portage translation system trained solely on *Job Bank* data. Upon closer examination of the model parameters, we observed that *Hansard* model components (language model, phrase tables, IBM translation models) were systematically attributed negligeable weights. Again, the amount of training material for the new domain may be critical in chosing between alternative adaptation mechanisms.

5 Conclusions and Future Work

We have proposed using a phrase-based MT system to automatically post-edit the output of another MT system, and have tested this idea with the Portage MT system on the *Job Bank* data set, a corpus of manually post-edited French-English machine translations. In our experiments, not only does phrase-based APE significantly improve the quality of the output translations, this approach outperforms a standalone phrase-based translation system.

While these results are very encouraging, the learning curves of Figure 2 suggest that the output quality of the PBMT systems increases faster than that of the APE systems as more data is used for training. So while the combination strategy clearly performs better with limited amounts of training data, there is reason to believe that, given sufficient training data, it would eventually be outperformed

by a direct phrase-based translation strategy. Of course, this remains to be verified empirically, something which will obviously require more data than is currently available to us. But this sort of behavior is expectable: while both types of system improve as more training data is used, inevitably some details of the source text will be lost by the front-end MT system, which the APE system will never be able to retrieve.[5] Ultimately, the APE system will be weighted down by the inherent limitations of the front-end MT system.

One way around this problem would be to modify the APE system so that it not only uses the baseline MT output, but also the source-language input. In the Portage system, this could be achieved, for example, by introducing feature functions into the log-linear model that relate target-language phrases with the source-language text. This is one research avenue that we are currently exploring.

Alternatively, we could combine these two inputs differently within Portage: for example, use the source-language text as the *primary* input, and use the raw MT output as a secondary source. In this perspective, if we have multiple MT systems available, nothing precludes using *all* of them as providers of secondary inputs. In such a setting, the phrase-based system becomes a sort of *combination MT system*. We intend to explore such alternatives in the near future as well.

Acknowledgements

The work reported here was part of a collaboration between the National Research Council of Canada and the department of Human Resources and Social Development Canada. Special thanks go to Souad Benayyoub, Jean-Frédéric Hübsch and the rest of the *Job Bank* team at HRSDC for preparing data that was essential to this project.

References

Jeffrey Allen and Christofer Hogan. 2000. Toward the development of a post-editing module for Machine Translation raw output: a new productivity tool for processing controlled language. In *Third International Controlled Language Applications Workshop (CLAW2000)*, Washington, USA.

Jeffrey Allen. 2004. Case study: Implementing MT for the translation of pre-sales marketing and post-sales software deployment documentation. In *Proceedings of AMTA-2004*, pages 1–6, Washington, USA.

Peter F Brown, Stephen A Della Pietra, Vincent J Della Pietra, and Robert L Mercer. 1993. The Mathematics of Statistical Machine Translation: Parameter Estimation. *Computational Linguistics*, 19(2):263–311.

Jakob Elming. 2006. Transformation-based corrections of rule-based MT. In *Proceedings of the EAMT 11th Annual Conference*, Oslo, Norway.

George Foster, Roland Kuhn, and Howard Johnson. 2006. Phrasetable Smoothing for Statistical Machine Translation. In *Proceedings of EMNLP 2006*, pages 53–61, Sydney, Australia.

Kevin Knight and Ishwar Chander. 1994. Automated Postediting of Documents. In *Proceedings of National Conference on Artificial Intelligence*, pages 779–784, Seattle, USA.

Philipp Koehn, Franz J. Och, and Daniel Marcu. 2003. Statistical Phrase-Based Translation. In *Proceedings of HLT-NAACL 2003*, pages 127–133, Edmonton, Canada.

Philipp Koehn. 2004. Pharaoh: a Beam Search Decoder for Phrase-Based Statistical Machine Translation Models. In *Proceedings of AMTA 2004*, pages 115–124, Washington, USA.

Daniel Marcu and William Wong. 2002. A Phrase-Based, Joint Probability Model for Statistical Machine Translation. In *Proceedings of EMNLP 2002*, Philadelphia, USA.

Franz Josef Och. 2003. Minimum error rate training in Statistical Machine Translation. In *Proceedings of ACL-2003*, pages 160–167, Sapporo, Japan.

Fatiha Sadat, Howard Johnson, Akakpo Agbago, George Foster, Roland Kuhn, Joel Martin, and Aaron Tikuisis. 2005. PORTAGE: A Phrase-Based Machine Translation System. In *Proceedings of the ACL Workshop on Building and Using Parallel Texts*, pages 129–132, Ann Arbor, USA.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A Study of Translation Edit Rate with Targeted Human Annotation. In *Proceedings of AMTA-2006*, Cambridge, USA.

[5] As a trivial example, imagine an MT system that "deletes" out-of-vocabulary words.