SEVENTH FRAMEWORK PROGRAMME

THEME ICT-2011.4.2(a)

Language Technologies

ACCEPT
Automated Community Content Editing PorTal

www.accept-project.eu

Starting date of the project: 1 January 2012

Overall duration of the project: 36 months

Report on improved machine translation by exploiting post-editing data

Workpackage n° 4	Name: Improving SMT
Deliverable n° 4.3	Name: Report on improved machine translation by exploiting post-editing data
Due date: 31 December 2014	Submission date: 19 December 2014

Dissemination level: PU

Organisation name of lead contractor for this deliverable: University of Edinburgh

Author(s): Ulrich Germann, Barry Haddow

Internal reviewer(s): Manny Rayner

Proofreading: Violeta Seretan

Copyediting: Barry Haddow, Violeta Seretan

The research leading to these results has received funding from the European Community's Seventh Framework Programme (*FP7/2007-2013*) under *grant agreement n° 288769.*

Contents

Exploitation of Post-Editing Data

1 Overview

This document summarises the research done within the ACCEPT project in *Task 4.4: Exploitation of Usage Data* of the *Improving SMT* work package.

The aim of this work was to examine ways of improving an SMT system during post-editing. The idea is that we start with a baseline system, which proposes translations to the user. The user corrects those translations, submitting the corrected version back to the system. The system should then be able to use the corrected version to improve future translations, especially by avoiding making the same mistakes again.

We developed a general mechanism for exploiting this post-editing data, by dynamically updating the phrase table. This is described in Section 2 where we explain how phrases can be extracted on-the-fly from an aligned corpus, instead of using a batch extraction process. This means that the system can be quickly updated while it is still running, without waiting for the hours or days normally required to batch train an SMT system. As an application of this update mechanism, we show in Section 3.3 how an automatic post-editor (APE) can be built so that it learns from actual post-edits. We test this APE system on a small amount of post-editing data collecting during the ACCEPT project, showing that it can make measurable improvements to the underlying SMT system.

2 Dynamically Updating SMT Systems[1]

Conventional MT systems are fairly static in nature, as the underlying models are expensive to build, train, and update. This often restricts update cycles to, say, once per day (over-night), and precludes updating the knowledge base immediately after additional information has become available. In an interactive scenario such as integration of MT into a translation or post-editing work-flow, this is a major drawback: we would like the system to learn from post-edits as soon as possible. One of the bottlenecks is the construction of the phrase table. The conventional method of construction consists of a batch processing pipeline involving (1) word-alignment of the parallel data; (2) phrase pair extraction; (3) phrase pair scoring based on global statistics over the collection of phrase pairs (e.g., smoothed conditional phrase-level translation probabilities in both translation directions); and finally (4) storing them in a file or other data structure. This process can take hours, even days for large amounts of training data.

An alternative to the conventional pipeline is to collect and score phrase table entries on the fly by *sampling* available information sources directly rather than pre-computing a large range of conceivable requests (Callison-Burch et al., 2005). This requires indexing of all available training data, so that source phrase occurrences can be found quickly in the available parallel data.

Suffix arrays (Manber and Myers, 1990) are a space-efficient way of indexing large corpora with reasonably fast retrieval times. In the context of natural text corpora, a suffix array is an array of all word positions in the corpus, sorted in lexicographic order

[1]This section overlaps in large parts with the corresponding section in Deliverable 1.2 of the MateCat project. Dynamic phrase tables for Moses were developed with support from the European Union's 7th Framework Programme under grant agreements 287688 (MateCat), 287576 (CasMaCat), and 288769 (ACCEPT).

of the word sequence starting at the respective position (hence the name *suffix* array). The computational cost for building a suffix array for an initial corpus is $O(n_1 \log n_1)$, where n_1 is the number of tokens in the corpus; the computational cost for finding all occurrences of a given text span is $O(\log n_1)$, and the cost of adding material of length n_2 to an indexed corpus is $O((n_2 \log n_2) + n_1)$, i.e. the cost of indexing the additional material and the linear cost of merge-sorting the two indices.

Even though suffix arrays have been used in hierarchical phrase-based translation for years (Lopez, 2007; Schwartz and Callison-Burch, 2010), adoption into the Moses toolkit had been slow, due to concerns about translation speed and translation quality: speed, because sampling is often perceived as being slow, and translation quality, because certain global information that is used to compute phrase table scores in the conventional setting is not available when sampling, e.g. global count information used for Good-Turing or modified Kneser-Ney smoothing of phrase-level conditional translation probabilities.

With the support from several EU projects (MateCat (grant agreement No. 287688), CasMaCat (grant agreement No. 287576), and ACCEPT (grant agreement No. 288769)), we implemented in Moses a dynamic phrase table that uses this technology effectively. We were able to close the quality gap that resulted from smoothing in conventional phrase tables with smoothing techniques not available in the sampling scenario. Our dynamic phrase tables not only achieve the same level of performance in the static scenario, but are also considerably faster than prior phrase table implementations in Moses. Details can be found in Germann (2014), included in appendix.

3 Learning from Post-edits

3.1 Corpus

The corpus used for these experiments was produced in the process of creating an extra French-English test set for the Symantec data. To create this test set, we selected 1000 forum posts from the French section of the Symantec forums and employed two translators to translate them into English. The translators used the post-editing interface developed for the Casmacat[2] project. The initial translations for the post-editing were produced using an SMT system similar to the ACCEPT baseline systems (see D4.1).

The 1000 forum posts contained a total of 5377 French sentences, so from each one we obtained two different human translations into English. For 10% of these we did not show the translators the SMT output, in order to provide a comparison of post-editing with translation "from scratch". We also allowed the translators to enter the string "! SAME !", if they believed that they had seen a sentence before in the corpus. After excluding the sentences translated from scratch, those marked as "! SAME !", plus a small number where the SMT shown to each of the translators was different[3] and sentences where either one of the translators produced an empty translation, the resulting post-edited corpus contains 4666 tuples, each consisting of a source sentence, a machine-translated version, and two separate post-edited versions. The text was then tokenised using the Moses tokeniser. We refer to this corpus as the Edinburgh Post-editing Corpus (edinpe).

[2]`www.casmacat.eu`

[3]The translation system used a dynamic suffix array with sampling, so was not deterministic.

3.2 Analysis of Post-Edits

We first investigated the nature of the edits that the translators made, highlighting the differences between the two translators. In Table 1 we report the BLEU and TER scores of the MT sentences in edinpe, with respect to each of the post-edited versions.

	Translator A	Translator B
BLEU	70.19	52.76
TER	0.1396	0.2231

Table 1: Measures of similarity of raw MT versus the post-edited versions. Note that higher BLEU indicates greater similarity, whereas the opposite is true for TER.

From Table 1 it is notable that the translators, especially Translator A, do not make many changes to the output. Typical BLEU scores for comparing this MT system against references created from scratch in this domain are in the 40s. We also notice that there is a marked difference between the two translators. In fact, the BLEU score *between* translators (taking translator A as the reference) is 56.43, indicating that the agreement between translators is no better in many cases than the agreement between the translators and the machine translation.

In order to investigate the edits made by the translators, we used TER to align both of the post-edited references to the raw MT. From these alignments we are able to extract substitutions (where two different words are aligned), as well as insertions and deletions (where tokens are left unaligned).

Considering the comparison of raw MT output with the post-edited versions, we show in Table 2 the numbers of insertions, deletions and substitutions made by each translator.

	Translator A	Translator B
Inserts	2715 (32.5%)	4114 (29.1%)
Deletes	663 (7.9%)	1206 (8.5%)
Substitutions	4973 (59.5%)	8839 (62.4%)
Total	8351	14159

Table 2: Numbers of insertions, deletions and substitutions made by each translator, according to the TER alignment.

The main difference is observed in the total number of edits, with the proportions of the different types of edits being roughly equal between the two translators. Looking at the ratio between the number of insertions and the number of deletions, we see that the first translator (ratio = 4.1:1) tends to favour insertions more than the second (ratio = 3.4:1), but both tend to add a lot more text than they remove.

To better understand the cause of these edits, we examined the top 10 substitutions, insertions and deletions made by each of them. The insertions and deletions are listed in Table 3, whilst the substitutions are listed in Table 4.

Looking at Table 3 firstly, we notice that the most common insertions and deletions roughly line up with the most common words. That probably is not too much of a surprise – these words are difficult to get right, and of course are the most common words. The

Insertions				Deletions			
Translator A		Translator B		Translator A		Translator B	
to	(164)	.	(386)	of	(43)	!	(95)
the	(152)	the	(212)	the	(30)	the	(66)
it	(138)	to	(189)	is	(13)	of	(65)
,	(77)	,	(168)	,	(12)	to	(33)
.	(73)	it	(163)	has	(12)	is	(28)
of	(56)	a	(92)	you	(11)	have	(25)
a	(55)	'	(80)	to	(11)	that	(24)
on	(53)	for	(66)	a	(10)	,	(22)
is	(44)	on	(66)	.	(8)	it	(15)
you	(44)	of	(63)	>	(8)	?	(15)
	2715		4114		663		1206

Table 3: Most common insertions and deletions for each annotator, with absolute counts in brackets

main differences between the two translators seem to be in the treatment of punctuation and contractions, and closer inspection will suggest that these are stylistic changes.

The most common insertions of translator B are full stops, proportionally much more common than for translator A. Also, translator B has deleted 95 exclamation marks in contrast to translator A, who did not delete any. Many of the exclamation marks deleted by Translator B occur in a few sentences with multiple exclamation marks, a phenomenon which occurs in this kind of informal forum data. Furthermore, Translator B often added full stops even when they did not occur in the source, as in Example 1.

(1a)	Source:	Merci à tous
(1b)	MT:	Thank you to all
(1c)	Trans 1:	Thank you all
(1d)	Trans 2:	Thank you to all.

Substitutions			
Translator A		Translator B	
tone→your	(27)	not→'t	(133)
norton→Norton	(25)	is→'s	(49)
the→it	(20)	norton→Norton	(46)
as→have	(18)	do→don	(32)
a→an	(17)	does→doesn	(25)
not→'t	(17)	tone→your	(24)
has→to	(15)	has→to	(21)
of→to	(12)	as→have	(20)
are→is	(11)	of→from	(20)
parsing→scan	(11)	'→"	(20)

Table 4: Most common substitutions for each annotator, with absolute counts in brackets

Examining the common substitutions in Table 4 shows that capitalisation of "norton" is a frequent problem, as is the mistranslation of the French "ton" (a domain issue caused by the lack for informal text in the training data). What is different about the two translators is their handling of contractions. Translator B obviously prefers contracted forms in English (e.g. "can't" rather than "can not") whereas Translator A has corrected less of these. For example in 2, Translator B makes the change, but Translator A leaves the verb uncontracted, as in the raw MT.

(2a)	Source:	Ca ne semble pas d'etre le cas pour moi.
(2b)	MT:	CA does not seem to be the case for me.
(2c)	Trans 1:	That does not seem to be the case for me.
(2d)	Trans 2:	That doesn't seem to be the case for me.

In Tables 5 and 6 we focus on the actual disagreements between the two translators, counting the occasions where one translator inserts, deletes, or substitutes a token, but the other does not. This confirms that contractions and punctuation play a significant role in the difference between the two translators. We also see that the translators have some differences of opinion about capitalisation. This is an area where automatic processing could perhaps help to create to standardise between translators, and also to allow them to concentrate on more "interesting" corrections.

Insertions				Deletions			
Translator A		Translator B		Translator A		Translator B	
the	(92)	.	(343)	of	(35)	!	(95)
to	(90)	the	(147)	the	(21)	the	(57)
it	(83)	,	(142)	has	(11)	of	(57)
,	(53)	to	(115)	to	(9)	to	(31)
of	(43)	it	(109)	you	(9)	that	(23)
a	(42)	a	(79)	a	(9)	have	(23)
on	(40)	's	(67)	,	(9)	is	(23)
is	(32)	on	(54)	is	(8)	,	(20)
have	(31)	of	(50)	>	(8)	?	(14)
in	(28)	't	(50)	.	(7)	it	(14)

Table 5: Translator disagreements: Top 10 insertions and deletions made by one translator, but not the other. In other words, the insertions/deletions in the "Translator A" column were made by A but not B, and vice-versa for the "Translator B" column.

3.3 Automatic Post-Editing

We used the corpus from Section 3.1 and the dynamic suffix array phrase table from Section 2 to investigate building an Automatic Post-Editor (APE) using phrase-based MT.

The idea of our APE is that we create a phrase-based decoder to convert raw MT output (from the baseline system) into post-edited MT. This decoder is learnt from the user edits, which are fed back into the system after every sentence that the translator post-edits. The theory is that the APE system can learn common corrections from the translator, and apply them to future sentences, thereby improving the MT system and avoiding annoying the translator by showing them the same errors over and over again.

Substitutions			
Translator A		Translator B	
norton→Norton	(20)	not→'t	(120)
the→it	(16)	is→'s	(46)
a→an	(12)	norton→Norton	(41)
of→to	(10)	do→don	(30)
by→through	(9)	does→doesn	(23)
am→have	(9)	pc→PC	(17)
internet→Internet	(8)	this→the	(17)
parsing→scan	(8)	of→from	(15)
windows→Windows	(7)	is→has	(15)
response→answer	(7)	of→to	(14)

Table 6: Translator disagreements: Top 10 substitutions made by one translator, but not the other

The operation of our APE is as follows. The system contains two phrase tables – a foreground table and a background table – and initially the background table contains translations of words to themselves, whilst the foreground table is empty. The background table is static, but the foreground table is updated as the Translator edits. Then the first sentence is edited by the translator, who submits the edited version to the system. The APE creates a TER alignment between the machine translated sentence, and the edited version, then extracts phrases from this aligned sentence pair using the usual phrase-based MT heuristics. These extracted phrases are added to the foreground phrase table.

For the second and subsequent sentences, the output of the baseline system is passed to the APE before it is shown to the translator. The APE is really a phrase-based MT system, which uses the foreground and background phrase tables, plus the language model from the baseline system, to translate the output of the baseline system into an automatically edited version.

The implementation of the APE uses the dynamic suffix array phrase table (as described in Section 2) and the Moses server. At start-up time, the Moses server just has the background table, but after each sentence is edited by the translator, the result of this editing is passed back to the server and added to the foreground table. Since both tables are implemented with the suffix array, phrases are extracted on-the-fly, if and when they are need to post-process subsequent MT output.

In our experiments with this APE system we used the actual post-edits described in 3.1, as opposed to the simulated post-edits (SPE) typically used in previous work, such as Simard and Foster (2013). We believe that this is much more realistic, as SPE typically involves treating a from-scratch translation as if it were produced by post-editing. Nevertheless, our experiments cannot be considered as a full test of APE, since the translators did not use the APE system when they were editing. That is to say, all the MT output that they saw came from the baseline system.

We now describe our experimental setup. We took the corpus of 4666 tuples from Section 3.1, and treated each translator's portion separately. This means that we ended up with two corpora (edinpe.0 and edinpe.1) each consisting of a (source, MT, human edited MT) tuple. We then split each of these two corpora into three sections (A, B and C) of equal size, by dividing the corpora sequentually. Section B was used for tuning the parameters of the APE using the standard SMT batch-tuning paradigm – specifically we used batch mira (Cherry and Foster, 2012). Section A was optionally used for seeding the foreground corpus, i.e. providing an initial set of post-edits to the APE, and section C for testing the resulting system.

In Table 7 we report the BLEU and TER scores of the baseline MT and the APE-augmented MT, as compared to the human post-edited translation, which we treat as the reference. The table shows that the APE makes little or no difference to the overall evaluation scores. There is a slight preference for the "seeded" version as opposed to the "unseeded" version, but the score differences are too small to conclude a real effect. We examined the change of sentence BLEU over time, to see if scores improved as more post-edited output was seen, but there was no discernible pattern.

	Translator A		Translator B	
	BLEU	TER	BLEU	TER
Baseline	78.21	0.0952	51.13	0.2354
APE (noseed)	77.99	0.0960	51.15	0.2352
APE (seed)	78.19	0.0957	51.19	0.2360

Table 7: Comparison of baseline and APE systems, taking the human post-edited output as a reference. "seed" indicates whether the APE system was seeded with an initial set of post-edits.

A casual inspection of the output of the APE system, however, suggested that it was having a positive effect on translation. In order to investigate this further, we decided to do a human evaluation on all the sentences that were affected by the APE system. In fact, there were not that many sentences changed – for Translator A, only 66 (4.2%) of the test sentences were changed, whilst the corresponding figure for Translator B was 202 (13.0%). The APE system affected much more sentences for the second translator, since that translator made much more edits.

The human evaluation of the APE system was performed by a single evaluator, a bilingual French Canadian, who works in machine translation research. The evaluator inspected triples of <source, MT output, APE-augmented MT> and marked either "first better", "second better" or "equal". We used the output of the seeded APE for the human evaluation. The results, shown in Table 8, are quite encouraging for the APE system.

	Translator A	Translator B
Base better	11 (16.7%)	24 (11.9%)
APE better	47 (71.2%)	133 (65.8%)
Same	8 (12.1%)	45 (22.3%)

Table 8: Human evaluation of the APE system, comparing it to the baseline (raw MT).

For both translators, the APE system either improved or did not degrade the output in over 80% of the sentences that it changed. In the rest of this section, we examine some examples, good and bad, of the APE system at work.

The baseline MT system sometimes struggled with the poor French of the forum posts, and in particular the missing accents in crucial places. These errors were fixed by the Translators, but also were picked up by the APE system. We illustrate this in Example

3, the source sentence is missing the accent on "Ça", causing the baseline MT to fail, but the APE system fixes it.

(3a)	Source:	Ca ne marche pas au boulot avec symantec end point …
(3b)	Baseline:	CA does not work the job with symantec end point …
(3c)	APE:	That does not work the job with symantec end point …
(3d)	Reference:	It does not work at my workplace with symantec end point …

The APE system was also able to learn domain-specific terms, which were missing from the baseline system, as in Example 4. This could potentially save the editor the frustration at having to constantly correct the same errors.

(4a)	Source:	plantage de Norton lors de Liveupdate et autres
(4b)	Baseline:	Norton plantage when Liveupdate and others
(4c)	APE:	Norton crash when Liveupdate and others
(4d)	Reference:	Norton crashing when performing Live update and others

The baseline system often has poor treatment of informal French pronouns and verb forms, creating either out-of-vocabulary words or translations that are quite wrong as in Example 5. Since such examples occur multiple times, the APE is able to learn corrections.

(5a)	Source:	Tu ne parles pas de ton navigateur
(5b)	Baseline:	You do not speak of tone browser
(5c)	APE:	You do not speak of your browser
(5d)	Reference:	You're not speaking of your browser

In most cases, the APE system only updates single words, but sometimes it will alter a whole sentence. In Example 6 we see a short sentence which may be quite frequent in forum data, but translating it literally (as in the baseline system) gives poor results.

(6a)	Source:	Est-ce que le problème persiste?
(6b)	Baseline:	Is the problem persists?
(6c)	APE:	Do you still have the problem?
(6d)	Reference:	Do you still have the same problem.

The APE system is exposed to the limitations of the alignment and phrase extraction algorithms though, for example in 7 where it makes the MT output worse. This correction was learnt from an earlier example where the translator edited "I seeks" to read "I'm looking for", however a poor alignment meant that the phrase pair "I seeks → I'm" was extracted.

(7a)	Source:	Je cherche un pas à pas pour configurer …
(7b)	Baseline:	I seeks a step by step to configure …
(7c)	APE:	I'm a step by step to configure …
(7d)	Reference:	I'm looking for step by step instructions on how to configure …

One sentence that was corrected multiple times by the APE system was the one shown in Example 8. This sentence with its associated correction occurred 12 times in the corpus, and each time, the APE system was penalised by BLEU for the mismatch with the reference. The evaluator, however, preferred the APE version of this sentence. In order for the APE system to have learned this correction, the translator must at some point have inserted the “have a” into the MT output, however in subsequent cases they accepted the MT output unchanged. The APE system learnt the original correction, then applied it to all subsequent occurrences.

(8a)	Source:	Merci et bonne journée!
(8b)	Baseline:	Thank you and good day!
(8c)	APE:	Thank you and have a good day!
(8d)	Reference:	Thank you and good day!

The update in Example 8 does not affect the meaning of the text, assuming one is only interested in using translations of forum text to solve problems with Symantec software. However it could be argued that the APE output provides a more polite, and perhaps more accurate (as opposed to literal) rendering of the original.

The experiments described in this section show that an APE system based on phrase-based MT could be useful in learning from edits supplied by a translator, and re-applying them automatically. Whilst the degree of generalisation that the APE system gets from a small training set (compared to those usually employed in PBMT) may not help with complicated rewording, it could with simple lexical errors. Saving the post editor from repeatedly editing these kinds of errors could make for a less frustrating user experience for the translator.

One of the limitations of these experiments, though, is that we are not running them in a live system. The data we use is at least gathered during actual (as opposed to simulated) post-editing, but the translators only have access to the baseline MT system output during editing. If the translators edited the output of the APE system, then these edits could enable us to learn whether the translators accepted or rejected the output of the APE system. It could also potentially increase the quality of the post-editing by dealing with the mundane errors in the MT output before it is viewed by translators.

It would also interesting to determine how APE interacts with the other method of feeding the user edits back to the MT system; inserting the edited texts as additional training data. Like APE, this type of system update is also enabled by dynamic suffix array based phrase tables, but unlike APE it modifies the underlying MT system, instead of treating it as a black box, and makes use of the source text. Only experimentation will determine whether the two update methods provide overlapping, or complementary sources of information.

4 Conclusion

We have shown how an SMT system can incorporate user edits by replacing the usual batch training paradigm with a phrase table that can be updated dynamically. Initial experiments on a small amount of post-editing data show promising results, however more experimentation within live systems would be necessary in order to find the best methods of incorporating user updates.

References

Callison-Burch, C., C. Bannard, and J. Schroeder (2005). Scaling phrase-based statistical machine translation to larger corpora and longer phrases. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pp. 255–262. Association for Computational Linguistics.

Cherry, C. and G. Foster (2012). Batch Tuning Strategies for Statistical Machine Translation. In *Proceedings of NAACL.*

Germann, U. (2014, October). Dynamic Phrase Tables for Machine Translation in an Interactive Post-editing Scenario. In *AMTA 2014 Workshop on Interactive and Adaptive Machine Translation*, Vancouver, BC, Canada, pp. 20–31. Association for Machine Translation in the Americas.

Lopez, A. (2007). Hierarchical phrase-based translation with suffix arrays. In *EMNLP-CoNLL*, pp. 976–985. ACL.

Manber, U. and G. Myers (1990). Suffix arrays: A new method for on-line string searches. In *Proceedings of the First Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '90, Philadelphia, PA, USA, pp. 319–327. Society for Industrial and Applied Mathematics.

Schwartz, L. and C. Callison-Burch (2010). Hierarchical Phrase-Based Grammar Extraction in Joshua: Suffix Arrays and Prefix Tree. *The Prague Bulletin of Mathematical Linguistics 93*, 157–166.

Simard, M. and G. Foster (2013). Pepr: Post-edit propagation using phrase-based statistical machine translation. In *Proceedings of MT Summit.*

Dynamic Phrase Tables for Machine Translation in an Interactive Post-editing Scenario

Ulrich Germann
University of Edinburgh
ugermann@inf.ed.ac.uk

Abstract

This paper presents a phrase table implementation for the *Moses* system that computes phrase table entries for phrase-based statistical machine translation (PBSMT) on demand by sampling an indexed bitext. While this approach has been used for years in hierarchical phrase-based translation, the PBSMT community has been slow to adopt this paradigm, due to concerns that this would be slow and lead to lower translation quality. The experiments conducted in the course of this work provide evidence to the contrary: without loss in translation quality, the sampling phrase table ranks second out of four in terms of speed, being slightly slower than hash table look-up (Junczys-Dowmunt, 2012) and considerably faster than current implementations of the approach suggested by Zens and Ney (2007). In addition, the underlying parallel corpus can be updated in real time, so that professionally produced translations can be used to improve the quality of the machine translation engine immediately.

1 Introduction

In recent years, there has been an increasing interest in integrating machine translation (MT) into the professional translator's work flow. With translation memories (TM) firmly established as a productivity tool in the translation industry, it is a conceptually obvious extension of this paradigm to include machine translation engines as virtual TMs in the set-up.

One major obstacle to this integration is the static nature of most machine translation systems that are currently available for use in production. They cannot adapt easily to feedback from the post-editor, or integrate new data into their knowledge base on short notice. In other words, they do not learn interactively from corrections to their output. Their models and knowledge bases were originally developed and designed for a batch translation scenario, where resources are first built and then used to translate in a fully automatic fashion without further intervention. Training the model parameters is still a slow and computationally very expensive process.

This paper presents *dynamic* phrase tables as an alternative, implemented within the open-source statistical machine translation (SMT) system *Moses* (Koehn et al., 2007).[1] Rather than simply looking up pre-computed entries from a database, they construct their entries on the fly by sampling word-aligned parallel data. The underlying corpus can be amended dynamically with low latency, for example by feeding post-edited output back to the translation server. New additions to the corpus can be exploited for future translations immediately.

While the underlying mechanisms are not new (cf. Callison-Burch et al., 2005; Lopez, 2007), the work reported here eliminates two major concerns about the use of bitext sampling for phrase table entry construction on demand: translation speed and translation quality. The experimental evaluation shows that in terms of speed, the sampling phrase table clearly outperforms current implementations of the work by Zens and Ney (2007). It comes close to the translation speed achievable with the hash-based compact phrase table implementation of Junczys-Dowmunt (2012). It should be noted that if translation speed is a serious concern, it is easy to pre-compute and store or cache phrase table entries for frequently occurring phrases. In terms of translation quality, the performance of the sampling phrase table is on par with conventional phrase tables for phrase-based SMT. Among the phrase table implementations that were evaluated for this work, the sampling phrase table is the only one that allows dynamic updates to its knowledge base in real time.

2 Conventional phrase tables vs. bitext sampling

2.1 Background

Most machine translation systems used in production today follow the paradigm of phrase-based statistical machine translation (PBSMT; Koehn et al., 2003). PBSMT systems typically rely on three distinct models: a language model that judges target-language fluency of a proposed translation; a translation model that gauges the quality of the elementary translation pairs that the final translation is composed of; and a distortion model that models changes in word order between source text and translation.

The units of translation in PBSMT are contiguous sequences of words in the source text ("*phrases*") that are translated into contiguous sequences of words on the target side. Producing the translation hypothesis left-to-right in the target language, the translation algorithm selects non-overlapping phrases in arbitrary order from the source and concatenates the corresponding translations (i.e., target phrases) to produce a translation hypothesis. Jumps between the source phrases are modelled by the distortion model.

Translation options for source phrases are conventionally stored in a pre-computed table, which is called the phrase table. Phrase translation scores are computed via a (log-)linear model over a number of features values associated with the phrase pair $\langle \mathbf{s}, \mathbf{t} \rangle$ in question. In the typical set-up, phrase table entries are evaluated by four feature

[1] The code has been added to the *Moses* master branch at https://github.com/moses-smt/mosesdecoder.

functions. In the formulas below, $\mathcal{A}_{\mathbf{s},\mathbf{t}}$ is the phrase-internal word alignment between $\mathbf{s}$ and $\mathbf{t}$. The four feature functions are as follows.

- the conditional phrase-level 'forward' translation probability $p\,(\mathbf{t} \mid \mathbf{s})$
- the conditional phrase-level 'backward' translation probability $p\,(\mathbf{s} \mid \mathbf{t})$
- the joint 'lexical forward' probability of all target words, given the source phrase (and possibly a word alignment between the two phrases): $\prod_{k=0}^{|\mathbf{t}|} p\,(t_k \mid \mathbf{s}, \mathcal{A}_{\mathbf{s},\mathbf{t}})$.
- the corresponding joint 'lexical backward' probability $\prod_{k=0}^{|\mathbf{s}|} p\,(s_k \mid \mathbf{t}, \mathcal{A}_{\mathbf{s},\mathbf{t}})$.

In order to achieve better translations, phrase-level probabilities are typically smoothed by Good-Turing or Kneser-Ney smoothing (Foster et al., 2006). The underlying counts and smoothing parameters are computed based on a complete list of phrase pairs extracted from the word-aligned parallel training corpus.

2.2 Bitext sampling

Except for toy examples, pre-computed phrase tables are typically very large, with the exact size of course depending on the maximum phrase length chosen and the size of the underlying corpus. The phrase table used for the timing experiments reported in Section 3.2, for example, consists of over 90 million distinct pairs of phrases of up to 7 words extracted from a moderately sized parallel corpus of fewer than 2 million parallel sentences of German-English text.

The large sizes of phrase tables make it impractical to fully load them into memory at translation time. Fully loaded into memory in the *Moses* decoder, the phrase table of the aforementioned system requires well over 100 GB of RAM and takes far beyond an hour to load. Therefore, phrase tables are usually converted to a disk-based representation, with phrase table entries retrieved from disk when needed. There are several such representations (Zens and Ney, 2007; Germann et al., 2009; Junczys-Dowmunt, 2012), two of which (Zens and Ney, 2007; Junczys-Dowmunt, 2012) have been integrated into the *Moses* system.

As an alternative to pre-computed phrase tables, Callison-Burch et al. (2005) suggested to compute phrase table entries on the fly at runtime by extracting and scoring a sample of source phrase occurrences and their corresponding translations from a pre-indexed bitext. For indexing, they use *suffix arrays* (Manber and Myers, 1990). A suffix array is an array of all token positions in a given linear sequence of tokens (e.g., a text or a DNA sequence), sorted in lexicographic order of the sub-sequence of tokens starting at the respective position. The use of suffix-array-based bitext sampling in the context of MT has been explored at length by Lopez (2007) as well as Schwartz and Callison-Burch (2010), especially with respect to Hierarchical Phrase-based Translation (HPBSMT; Chiang, 2005, 2007).

A great advantage of the suffix-array-based approach is that it is relatively cheap and easy to augment the underlying corpus. To add a pair of sentences to the parallel corpus, all we need to do is to construct a suffix array for the added material ($\mathrm{O}(n \log n)$, where n is the number of tokens in the added material), and then merge-sort the original suffix array (of length m) with the new suffix array ($\mathrm{O}(n + m)$).

While corpus sampling is common practice in other branches of MT research (especially HPBSMT, due to the prohibitive size of pre-computed, general-purpose, wide-coverage rule bases), adoption in the PBSMT community has been slow, apparently[2] due to concerns about translation speed and quality.

In the following, I intend to dispel these concerns by presenting experimental results obtained with an implementation of suffix-array-based phrase tables that sample the underlying bitext at run time, yet outperform existing disk-based implementations of conventional phrase tables by a wide margin in terms of speed (despite the greater computational effort), without any loss in translation quality.

Much of the speed benefit is related to RAM vs. disk access. Word-aligned parallel corpora are much more compact than fully expanded phrase tables, so we can afford to keep more of the information in memory, benefiting from access times that can be several orders of magnitude faster than random access to data stored on disk (Jacobs, 2009).

Moreover, the data structures are designed to be mapped directly into memory, so that we can rely on the system's virtual memory manager to transfer the data efficiently into memory when needed. This is much faster than regular file access. Two of the four implementations evaluated here store all the data on disk by default and load them on demand (PhraseDictionaryBinary, PhraseDictionaryOnDisk); the other two (PhraseDictionaryCompact and PhraseDictionaryBitextSampling (this work)) use memory-mapped files to ensure the fastest transfer possible between disk and memory. I attribute most of the speed benefits to these implementational choices (see also Sec. 3.2).

Last but not least, one can alleviate the impact of the computational overhead on overall translation time by caching frequently occurring entries, so that they must be computed only once, and perform phrase table look-up in parallel for all source phrases in a sentence submitted for translation, subject to the number of CPUs available.

The issue of translation quality is less obvious. Despite common misconceptions, it is not so much a matter of missing translation options due to sampling the bitext instead of taking into account every single source phrase occurrence. The vast majority of phrases occur so rarely that we can easily investigate every single occurrence. More frequent words and phrases will often be contained in longer, rarer phrases whose instances we also fully explore. And if there is a rare translation of a very frequent word that escapes our sampling, it is highly unlikely that this translation would survive the

[2] I base this statement on numerous conversations with practitioners in the field.

system's hypothesis ranking process.

On the contrary, it is the rarity of most phrases that causes problems, as maximum likelihood estimates based on low counts are less reliable — they tend to over-estimate the true translation probability. As Foster et al. (2006) have shown, smoothing phrase-level conditional phrase probabilities improves translation performance. My experiments confirm this finding (Table 2).

Both standard methods for smoothing phrase-level translation probabilities in the phrase table, Good-Turing and Kneser-Ney, require global information about the entire set of phrasal translation relations contained in the parallel corpus. This information is not available when we sample. To take the amount of evidence available into account when estimating phrase translation probabilities, we therefore compute the lower bound of the confidence interval[3] over the true translation probability, at some confidence level α, based on the observed counts. The more evidence is available, the narrower the confidence interval.

Another issue is the computation of the useful backward phrase-level translation probabilities $p\,(\textit{source phrase} \,|\, \textit{target phrase})$. Omitting this feature function seriously hurts performance (see Line 5 in Table 2). One could, of course, perform a full reverse look-up for each translation candidate to obtain the inverse translation probability. This would increase the number of full phrase look-ups operations necessary to construct a phrase table entry from scratch by a factor equal to the number of translation options considered for each source phrase (although again, these look-up operations could be cached). In practice, this is not necessary. To determine the denominator for the backward phrase-level translation probability, we simply scale the number of occurrences of each translation candidate in the bitext by the ratio of the source phrase sample size to the total number of source phrase occurrences in the corpus. Retrieving the total number of occurrences of the translation candidate in the corpus is trivial if we also index the target side of the corpus with a suffix array: we only need to measure the distance between the first and the occurrence of the phrase in the suffix array. Since the suffix array is sorted in lexicographic order of the corresponding suffixes, this distance is the total number of phrase occurrences.

3 Experiments

Two sets of experiments were conducted to compare bitext sampling to conventional phrase tables in terms of static performance (without updates), and a third one to asses the benefits of dynamically updating the phrase table as interactive translation progresses. The first experiment aimed at determining the quality of translation achievable with bitext sampling and the best parameter settings; the second focused on translation speed and resource requirements. Training, tuning and test data for these two experiments were taken from the data sets for the WMT 2014 shared translation task (cf. Table 1). The language model was a standard 5-gram model with Kneser-Ney smooth-

[3] Specifically, the Clopper-Pearson interval (Clopper and Pearson, 1934) as implemented in the Boost C++ library.

Table 1: Corpus statistics for the training, development and test data. All corpora were part of the official data for the shared translation task at WMT 2014 and true-cased for processing.

			# of tokens	
	corpus	**# of sentences**	**German**	**English**
LM train	Europarl-v7	2,218,201		60,502,373
	News-Commentary-v9	304,174		7,676,138
TM train	Europarl-v7	1,920,209	50,960,730	53,586,045
	News-Commentary-v9	201,288	5,168,511	5,151,459
	total after alignment[a]	2,084,594	53,863,321	56,351,895
Tuning	Newstest-2013	3,000	64,251	65,602
Testing	Newstest-2014	3003	64,498	68,940

[a] Some sentence pairs were discarded during word alignment

ing; the distortion model was a simple distance-based model without lexicalisation. The phrase table limit (i.e., the limit on the number of distinct translation hypotheses that will be considered during translation) was set to 20; the distortion limit to 6. Sampling was performed without replacement.

3.1 Translation Quality

Table 2 shows the the quality of translation achieved by the various system configurations, as measured by the BLEU score Papineni et al. (2002). The system configurations were identical except for the method used for construction and scoring of phrase table entries.

Each system was tuned 10 times in independent tuning runs to gauge the influence of parameter initialisation on overall performance (cf. also Clark et al., 2011). The 95% confidence interval in the second-but-last column was computed with bootstrap resampling for the median system within the respective group.

The first four systems rely on conventional phrase tables with four feature functions as described in Sec. 2.1: forward and backward phrase-level conditional probabilities as well as forward and backward joint lexical translation probabilities. They differ in the smoothing method used, except for the system in Line 3, which shows that filtering the phrase table to include only the top 100 entries (according to the forward phrase-level probability $p(\mathbf{t} \mid \mathbf{s})$) has no effect on translation quality.

Lines 5 and below are based on bitext sampling. The poor performance in Line 5 illustrates the importance of the phrase-level backward probability. Without it, the performance suffers significantly. Lines 4 and 6 show the benefits of smoothing.

The parameter α in Lines 7 to 9 is the confidence level for which the Clopper-Pearson interval was computed. Notice the minuscule difference between lines 2/3

Table 2: BLEU scores with different phrase score computation methods.

#	method	low	high	median	mean	95% conf. interval[a]	runs
1	**precomp., Kneser-Ney smoothing**	18.36	18.50	18.45	18.43	17.93 – 18.95	10
2	**precomp., Good-Turing smoothing**	18.29	18.63	18.54	18.52	18.05 – 19.05	10
3	**precomp., Good-Turing smoothing, filtered**[b]	18.43	18.61	18.53	18.53	18.04 – 19.08	10
4	**precomp., no smoothing**	17.86	18.12	18.07	18.05	17.58 – 18.61	10
5	**max. 1000 smpl., no smoothing, no bwd. prob.**	16.70	16.92	16.84	16.79	16.35 – 17.32	10
6	**max. 1000 smpl., no smoothing, with bwd. prob.**	17.61	17.72	17.69	17.68	17.14 – 18.22	8
7	**max. 1000 smpl., $\alpha = .05$, with bwd. prob.**[c]	18.35	18.43	18.38	18.38	17.86 – 18.90	10
8	**max. 1000 smpl., $\alpha = .01$, with bwd. prob.**	18.43	18.65	18.53	18.52	18.03 – 19.12	10
9	**max. 100 smpl., $\alpha = .01$, with bwd. prob.**	18.40	18.55	18.46	18.46	17.94 – 19.00	10

[a] Confidence intervals were computed via bootstrap resampling for the median system in the group.
[b] Top 100 entries per source phrase selected according to $p(\mathbf{t} \mid \mathbf{s})$.
[c] The parameter α is the one-sided confidence level of the Clopper-Pearson interval for the observed counts.

and 8! By replacing plain maximum likelihood estimates with the lower bound of the confidence interval over the respective underlying translation probability, we can make up for the lack of global information necessary for Good-Turing or Kneser-Ney smoothing.

3.2 Speed

Table 3 shows average translation times[4] per sentence for four phrase table implementations in the *Moses* system. PhraseDictionaryBinary and PhraseDictionaryOnDisk are implementations of the method described in Zens and Ney (2007). PhraseDictionaryCompact (Junczys-Dowmunt, 2012) is a compressed phrase table that relies on a perfect minimum hash for look-up. PhraseDictionaryBitextSampling is the suffix array-based phrase table presented in this paper. Each system was run with 8 threads as the only processes on an 8-core machine with locally mounted disks, translating 3003 sentences from the WMT 2014 test set. Prior to each run, all file system caches in RAM were dropped.

When the pre-computed phrase tables are not filtered, the bitext sampler outperforms even the hash-based phrase table of Junczys-Dowmunt (2012). This is due to the cost of ranking very long lists of translation candidates for very frequent source phrases. Filtering the phrase table off-line to include only the 100 most likely translation candidates for each phrase (based on $p(\mathbf{t} \mid \mathbf{s})$) leads to a significant speed-up without impact on translation quality (cf. Line 3 in Table 2).[5] Similarly, the speed of the bitext sampler

[4] The times shown were computed by dividing the total wall time of the system run by the number of sentences translated. Translations were performed in 8 parallel threads, so that the average actual translation time for a single sentence is about 8 times the time shown. Since the bitext sampler is inherently multi-threaded, the fairest form of comparison was to run the systems in a way that exhausts the host computer's CPU capacity.

[5] I thank M. Junczys-Dowmunt for pointing out to me that phrase tables must be filtered for optimal performance.

Table 3: Translation speed (wall time) with different phrase table implementations. The implementation names correspond to *Moses* configuration options. Translations were performed in multi-threaded mode with 8 parallel threads.

type	**implementation**	**ave. sec./snt**
static	PhraseDictionaryBinary (Zens and Ney, 2007)	0.879
static	PhraseDictionaryOnDisk (Zens and Ney, 2007)	0.717
static	PhraseDictionaryCompact (Junczys-Dowmunt, 2012)	0.366
static	PhraseDictionaryCompact (Junczys-Dowmunt, 2012), filtered[a]	0.214
dynamic	PhraseDictionaryBitextSampling, max. 1000 samples (this work)	0.256
dynamic	PhraseDictionaryBitextSampling, max. 100 samples (this work)	0.228

[a] max 100 entries per source phrase

can be improved by reducing the maximum number of samples considered, although this slightly (but not significantly) reduces translation quality as measured by BLEU (cf. Line 9 in Table 2). Phrase table filtering has no impact on the speed of the other phrase table implementations.

3.3 Simulated Post-editing

The main goal of this work was to develop a phrase table that can incorporate user edits of raw machine translation output into its knowledge base at runtime. Since experiments involving real humans in the loop are expensive to conduct, I simulated the process by translating sentences from an earlier post-editing field trial in English-to-Italian translation in the legal domain. The training corpus consisted of ca. 2.5 million sentence pairs (English: ca. 44.6 million tokens, Italian: ca. 45.9 million). Due to the nature of such studies, the amount of data available for tuning and testing was fairly small: 564 sentence pairs with 17,869 English and 18,528 Italian tokens for tuning, and 472 segments with 10,829 tokens of English source text and 11,595 tokens of post-edited translation into Italian.

Several feature functions were added for use with dynamic updates to the underlying bitext. In the following, "background data" means parallel data available prior to the translation of the first sentence, and "foreground data" the parallel data that is successively added to the parallel corpus.

- Separate vs. pooled phrase-level conditional translation probabilities (forward and backward), i.e. the use of distinct feature functions for these probability estimates based on counts obtained *separately* from the background and the foreground corpus separately, or feature functions based on *pooled* counts from two corpora. Because of the small size of our tuning and test sets, counts were pooled in the experiments for this work.

- A provenance feature $\frac{n}{x+n}$, where n is the number of occurrences in the corpus

Table 4: Simulated post-editing vs. batch translation for English-to-Italian translation in the legal domain. For simulated post-editing, counts were pooled.

method	low	high	median	mean	95% conf. interval[a]	runs
conventional, Good-Turing smoothing	29.97	30.93	30.74	30.67	29.16 – 32.37	10
sampled, no updates, no smoothing, rarity pen.	29.84	30.97	30.52	30.43	28.97 – 32.25	10
simulated post-editing, pooled counts, no smoothing, rarity, provenance	30.63	33.05	31.96	31.88	30.19 – 33.77	10

[a] Confidence intervals were computed via bootstrap resampling for the median system in the group.

and $x > 1$ an adjustable parameter that determines the slope of the provenance reward. The purpose of this feature is to boost the score of phrase pairs that occur in the foreground corpus.

- A global rarity penalty $\frac{x}{x+n}$ (where x and n mean the same as above) that can penalise phrase pairs that co-occur only rarely overall.

Results are shown in Table 4. None of the differences are statistically significant. In light of the small size of the test set, this is hardly surprising. In general, we should expect the benefit of adding post-edited data immediately to the knowledge base of the SMT system to vary widely depending on the repetitiveness of the source text, and on how well the translation domain is already covered by the background corpus.

4 Related Work

User-adaptive MT has received considerable research interest in recent years. Due to space limitations, we can only briefly mention a few closely related efforts here. A survey of recent work can be found, for example, in the recent journal article by Bertoldi et al. (2014b). Ortiz-Martínez et al. (2010), Bertoldi et al. (2014b), and Denkowski et al. (2014) all present systems that can be updated incrementally. Ortiz-Martínez et al. (2010) present a system that can trained be incrementally from scratch with translations that are produced in an interactive computer-aided translation scenario. The work by Bertoldi et al. (2014b) relies on cache-based models that keep track of how recently phrase pairs in the translation model and n-grams in the language models have been used in the translation pipeline and give higher scores to recently used items. They also augment the phrase table with entries extracted from post-edited translations. The work by Denkowski et al. (2014) is the closest to the work presented in this paper.[6] Working with the *cdec* decoder (Dyer et al., 2010), they also use suffix arrays to construct phrase table entries on demand. In addition, they provide mechanisms to update the language model and re-tune the system parameters.

Focusing on dynamic adjustment of system parameters (feature function values and combination weights), Martínez-Gómez et al. (2012) investigate various online learning algorithms for this purpose. Blain et al. (2012) and Bertoldi et al. (2014a) describe

[6] Incidentally, Denkowski (personal communication) is using the implementation presented here to port the work of Denkowski et al. (2014) to the *Moses* framework.

online word alignment algorithms that can produce the word alignments necessary for phrase extraction.

5 Conclusions

I have presented a new phrase table for the *Moses* system that computes phrase table entries on the fly. It outperforms existing phrase table implementations in *Moses* in terms of speed, without sacrificing translation quality. This is accomplished by a new way of computing phrase-level conditional probabilities that takes the amount of evidence available into account and discounts probabilities whose estimates are based on little evidence. Unlike static conventional phrase tables, sampling-based phrase tables allow for rapid updates of the underlying parallel corpus and therefore lend themselves to use in an interactive and dynamic machine translation scenario.

Acknowledgements

This work was supported by the European Union's 7th Framework Programme (FP7/2007-2013) under grant agreements 287576 (CasMaCat), 287688 (MateCat), and 288769 (ACCEPT). I thank the anonymous reviewers for numerous helpful suggestions.

References

Bertoldi, Nicola, Amin Farajian, and Marcello Federico. 2014a. "Online word alignment for online adaptive machine translation." *Workshop on Humans and Computer-assisted Translation*, 84–92. Gothenburg, Sweden.

Bertoldi, Nicola, Patrick Simianer, Mauro Cettolo, Katharina Wäschle, Marcello Federico, and Stefan Riezler. 2014b. "Online adaptation to post-edits for phrase-based statistical machine translation." *Machine Translation*. Accepted for publication. Preprint.

Blain, Frédéric, Holger Schwenk, and Jean Senellart. 2012. "Incremental adaptation using translation information and post-editing analysis." *9th International Workshop on Spoken Language Translation*, 229–236. Hong Kong.

Callison-Burch, Chris, Colin Bannard, and Josh Schroeder. 2005. "Scaling phrase-based statistical machine translation to larger corpora and longer phrases." *43rd Annual Meeting of the Association for Computational Linguistics (ACL '05)*, 255–262. Ann Arbor, Michigan.

Chiang, David. 2005. "A hierarchical phrase-based model for statistical machine translation." *43rd Annual Meeting of the Association for Computational Linguistics (ACL '05)*, 263–270. Ann Arbor, Michigan.

Chiang, David. 2007. "Hierarchical phrase-based translation." *Computational Linguistics*, 33(2):1–28.

Clark, Jonathan H., Chris Dyer, Alon Lavie, and Noah A. Smith. 2011. "Better hypothesis testing for statistical machine translation: Controlling for optimizer instability." *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Short Papers - Volume 2*, 176–181. Stroudsburg, PA, USA.

Clopper, C.J. and E.S. Pearson. 1934. "The use of confidence or fiducial limits illustrated in the case of the binomial." *Biometrika*.

Denkowski, Michael, Chris Dyer, and Alon Lavie. 2014. "Learning from post-editing: Online model adaptation for statistical machine translation." *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, 395–404. Gothenburg, Sweden.

Dyer, Chris, Adam Lopez, Juri Ganitkevitch, Jonathan Weese, Ferhan Ture, Phil Blunsom, Hendra Setiawan, Vladimir Eidelman, and Philip Resnik. 2010. "cdec: A decoder, alignment, and learning framework for finite-state and context-free translation models." *Proceedings of the ACL 2010 System Demonstrations*, 7–12. Uppsala, Sweden.

Foster, George F., Roland Kuhn, and Howard Johnson. 2006. "Phrasetable smoothing for statistical machine translation." *EMNLP*, 53–61.

Germann, Ulrich, Eric Joanis, and Samuel Larkin. 2009. "Tightly packed tries: How to fit large models into memory, and make them load fast, too." *Workshop on Software Engineering, Testing, and Quality Assurance for Natural Language Processing (SETQA-NLP 2009)*, 31–39. Boulder, CO, USA.

Jacobs, Adam. 2009. "The pathologies of big data." *Queue*, 7(6):10:10–10:19.

Junczys-Dowmunt, Marcin. 2012. "Phrasal rank-encoding: Exploiting phrase redundancy and translational relations for phrase table compression." *Prague Bull. Math. Linguistics*, 98:63–74.

Koehn, Philipp, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. "Moses: Open source toolkit for statistical machine translation." *45th Annual Meeting of the Association for Computational Linguistics (ACL '07): Demonstration Session*. Prague, Czech Republic.

Koehn, Philipp, Franz Josef Och, and Daniel Marcu. 2003. "Statistical phrase-based translation." *Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL '03)*, 48–54. Edmonton, AB, Canada.

Lopez, Adam. 2007. "Hierarchical phrase-based translation with suffix arrays." *EMNLP-CoNLL*, 976–985.

Manber, Udi and Gene Myers. 1990. "Suffix arrays: A new method for on-line string searches." *Proceedings of the First Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '90, 319–327. Philadelphia, PA, USA.

Martínez-Gómez, Pascual, Germán Sanchis-Trilles, and Francisco Casacuberta. 2012. "Online adaptation strategies for statistical machine translation in post-editing scenarios." *Pattern Recognition*, 45(9):3193–3203.

Ortiz-Martínez, Daniel, Ismael García-Varea, and Francisco Casacuberta. 2010. "Online learning for interactive statistical machine translation." *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, 546–554. Stroudsburg, PA, USA.

Papineni, Kishore, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. "Bleu: a method for automatic evaluation of machine translation." *ACL 2002*, 311–318. Philadelphia, PA.

Schwartz, Lane and Chris Callison-Burch. 2010. “Hierarchical phrase-based grammar extraction in joshua: Suffix arrays and prefix tree.” *The Prague Bulletin of Mathematical Linguistics*, 93:157–166.

Zens, Richard and Hermann Ney. 2007. “Efficient phrase-table representation for machine translation with applications to online MT and speech translation.” *Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL ’07)*, 492–499. Rochester, New York.