Stream-based Translation Models for Statistical Machine Translation

Abby Levenberg
School of Informatics
University of Edinburgh
a.levenberg@ed.ac.uk

Chris Callison-Burch
Computer Science Department
Johns Hopkins University
ccb@cs.jhu.edu

Miles Osborne
School of Informatics
University of Edinburgh
miles@inf.ed.ac.uk

Abstract

Typical statistical machine translation systems are trained with static parallel corpora. Here we account for scenarios with a continuous incoming stream of parallel training data. Such scenarios include daily governmental proceedings, sustained output from translation agencies, or crowd-sourced translations. We show incorporating recent sentence pairs from the stream improves performance compared with a static baseline. Since frequent batch retraining is computationally demanding we introduce a fast incremental alternative using an online version of the EM algorithm. To bound our memory requirements we use a novel data-structure and associated training regime. When compared to frequent batch retraining, our online time and space-bounded model achieves the same performance with significantly less computational overhead.

1 Introduction

There is more parallel training data available today than there has ever been and it keeps increasing. For example, the European Parliament[1] releases new parallel data in 22 languages on a regular basis. Project Syndicate[2] translates editorials into seven languages (including Arabic, Chinese and Russian) every day. Existing translation systems often get 'crowd-sourced' improvements such as the option to contribute a better translation to GoogleTranslate[3]. In these and many other instances, the data can be viewed as an incoming *unbounded stream* since the corpus grows continually with time. Dealing with such unbounded streams of parallel sentences presents two challenges: making retraining efficient and operating within a bounded amount of space.

Statistical Machine Translation (SMT) systems are typically batch trained, often taking many CPU-days of computation when using large volumes of training material. Incorporating new data into these models forces us to retrain from scratch. Clearly, this makes rapidly adding newly translated sentences into our models a daunting engineering challenge. We introduce an adaptive training regime using an online variant of EM that is capable of incrementally adding new parallel sentences without incurring the burdens of full retraining.

For situations with large volumes of incoming parallel sentences we are also forced to consider placing space-bounds on our SMT system. We introduce a dynamic suffix array which allows us to add and delete parallel sentences, thereby maintaining bounded space despite processing a potentially high-rate input stream of unbounded length.

Taken as a whole we show that online translation models operating within bounded space can perform as well as systems which are batch-based and have no space constraints thereby making our approach suitable for stream-based translation.

2 Stepwise Online EM

The EM algorithm is a common way of inducing latent structure from unlabeled data in an unsupervised manner (Dempster et al., 1977). Given a set of unlabeled examples and an initial, often uniform guess at a probability distribution over the latent variables, the EM algorithm maximizes the marginal

[1] http://www.europarl.europa.eu
[2] http://www.project-syndicate.org
[3] http://www.translate.google.com

Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the ACL, pages 394–402,
Los Angeles, California, June 2010.

log-likelihood of the examples by repeatedly computing the expectation of the conditional probability of the latent data with respect to the current distribution, and then maximizing the expectations over the observations into a new distribution used in the next iteration. EM (and related variants such as variational or sampling approaches) form the basis of how SMT systems learn their translation models.

2.1 Batch vs. Online EM

Computing an expectation for the conditional probabilities requires collecting the *sufficient statistics* $\mathbf{S}$ over the set of n unlabeled examples. In the case of a multinomial distribution, $\mathbf{S}$ is comprised of the counts over each conditional observation occurring in the n examples. In traditional *batch* EM, we collect the counts over the entire dataset of n unlabeled training examples via the current 'best-guess' probability model $\hat{\theta}_t$ at iteration t (E-step) before normalizing the counts into probabilities $\bar{\theta}(\mathbf{S})$ (M-step)[4]. After each iteration all the counts in the sufficient statistics vector $\mathbf{S}$ are cleared and the count collection begins anew using the new distribution $\hat{\theta}_{t+1}$.

When we move to processing an incoming data stream, however, the batch EM algorithm's requirement that all data be available for each iteration becomes impractical since we do not have access to all n examples at once. Instead we receive examples from the input stream incrementally. For this reason online EM algorithms have been developed to update the probability model $\hat{\theta}$ incrementally without needing to store and iterate through all the unlabeled training data repeatedly.

Various online EM algorithms have been investigated (see Liang and Klein (2009) for an overview) but our focus is on the *stepwise online* EM (sOEM) algorithm (Cappe and Moulines, 2009). Instead of iterating over the full set of training examples, sOEM stochastically approximates the batch E-step and incorporates the information from the newly available streaming observations in steps. Each step is called a *mini-batch* and is comprised of one or more new examples encountered in the stream.

Unlike in batch EM, in sOEM the expected counts are retained between EM iterations and not cleared.

[4]As the M-step can be computed in closed form we designate it in this work as $\bar{\theta}(\mathbf{S})$.

Algorithm 1: Batch EM for Word Alignments

Input: $\{F$(source),E (target)$\}$ sentence-pairs
Output: MLE $\hat{\theta}_T$ over alignments $\mathbf{a}$
$\hat{\theta}_0 \leftarrow$MLE initialization;
for *iteration* $k = 0, \ldots, T$ **do**
  $S \leftarrow 0$; `// reset counts`
  foreach $(f, e) \in \{F, E\}$ **do** `// E-step`
    $S \leftarrow S + \sum_{a' \in \mathbf{a}} \Pr(f, a'|e; \hat{\theta}_t)$;
  end
  $\hat{\theta}_{t+1} \leftarrow \bar{\theta}_t(S)$; `// M-step`
end

That is, for each new example we interpolate its expected count with the existing set of sufficient statistics. For each step we use a *stepsize* parameter γ which mixes the information from the current example with information gathered from all previous examples. Over time the sOEM model probabilities begin to stabilize and are guaranteed to converge to a local maximum (Cappe and Moulines, 2009).

Note that the stepsize γ has a dependence on the current mini-batch. As we observe more incoming data the model's current probability distribution is closer to the true distribution so the new observations receive less weight. From Liang and Klein (2009), if we set the stepsize as $\gamma_t = (t + 2)^{-\alpha}$, with $0.5 < \alpha \leq 1$, we can guarantee convergence in the limit as $n \rightarrow \infty$. If we set α low, γ weighs the newly observed statistics heavily whereas if γ is low new observations are down-weighted.

2.2 Batch EM for Word Alignments

Batch EM is used in statistical machine translation to estimate word alignment probabilities between parallel sentences. From these alignments, bilingual rules or phrase pairs can be extracted. Given a set of parallel sentence examples, $\{\mathbf{F}, \mathbf{E}\}$, with $\mathbf{F}$ the set of source sentences and $\mathbf{E}$ the corresponding target sentences, we want to find the latent alignments $\mathbf{a}$ for a sentence pair $(\mathbf{f}, \mathbf{e}) \in \{\mathbf{F}, \mathbf{E}\}$ that defines the most probable correspondence between words f_j and e_i such that $a_j = i$. We can induce these alignments using an *HMM-based* alignment model where the probability of alignment a_j is dependent only on the previous alignment at a_{j-1} (Vogel et al., 1996).

We can write

$$\Pr(\mathbf{f}, \mathbf{a} \mid \mathbf{e}) = \sum_{a' \in \mathbf{a}} \prod_{j=1}^{|\mathbf{f}|} p(a_j \mid a_{j-1}, |\mathbf{e}|) \cdot p(f_j \mid e_{a_j})$$

where we assume a first-order dependence on previously aligned positions.

To find the most likely parameter weights for the translation and alignment probabilities for the HMM-based alignments, we employ the EM algorithm via dynamic programming. Since HMMs have multiple local minima, we seed the HMM-based model probabilities with a better than random guess using IBM Model 1 (Brown et al., 1993) as is standard. IBM Model 1 is of the same form as the HMM-based model except it uses a uniform distribution instead of a first-order dependency. Although a series of more complex models are defined, IBM Models 2 to Model 6 (Brown et al., 1993; Och and Ney, 2003), researchers typically find that extracting phrase pairs or translation grammar rules using Model 1 and the HMM-based alignments results in equivalently high translation quality. Nevertheless, there is nothing in our approach which limits us to using just Model 1 and the HMM model.

A high-level overview of the standard, batch EM algorithm applied to HMM-based word alignment model is shown in Algorithm 1.

2.3 Stepwise EM for Word Alignments

Application of sOEM to HMM and Model 1 based word aligning is straightforward. The process of collecting the counts over the expected conditional probabilities inside each iteration loop remains the same as in the batch case. However, instead of clearing the sufficient statistics between the iterations we retain them and interpolate them with the batch of counts gathered in the next iteration.

Algorithm 2 shows high level pseudocode of our sOEM framework as applied to HMM-based word alignments. Here we have an unbounded input stream of source and target sentences $\{\mathbf{F}, \mathbf{E}\}$ which we do not have access to in its entirety at once. Instead we observe mini-batches $\{\mathbf{M}\}$ comprised of chronologically ordered strict subsets of the full stream. To word align the sentences for each mini-batch $\mathbf{m} \in \mathbf{M}$, we use the probability assigned by the current model parameters and then interpolate the newest sufficient statistics $\bar{s}$ with our full count vector $\mathbf{S}$ using an interpolation parameter γ. The interpolation parameter γ has a dependency on how far along the input stream we are processing.

Algorithm 2: sOEM Algorithm for Word Alignments

Input: mini-batches of sentence pairs $\{M : M \subset \{F(source), E(target)\}\}$
Input: stepsize weight α
Output: MLE $\hat{\theta}_T$ over alignments $\mathbf{a}$
$\hat{\theta}_0 \leftarrow$MLE initialization;
$S \leftarrow 0$; $k = 0$;
foreach *mini-batch* $\{m : m \in M\}$ **do**
  for *iteration* $t = 0, \ldots, T$ **do**
    foreach $(f, e) \in \{m\}$ **do** `// E-step`
      $\bar{s} \leftarrow \sum_{a' \in \mathbf{a}} \Pr(f, a' | e; \hat{\theta}_t)$;
    end
    $\gamma = (k+2)^{-\alpha}; k = k+1$; `// stepsize`
    $S \leftarrow \gamma \bar{s} + (1-\gamma) S$; `// interpolate`
    $\hat{\theta}_{t+1} \leftarrow \bar{\theta}_t(S)$; `// M-step`
  end
end

3 Dynamic Suffix Arrays

So far we have shown how to incrementally retrain translation models. We now consider how we might bound the space we use for them when processing (potentially) unbounded streams of parallel data.

Suffix arrays are space-efficient data structures for fast searching over large text strings (Manber and Myers, 1990). Treating the entire corpus as a single string, a suffix array holds in lexicographical order (only) the starting index of each suffix of the string. After construction, since the corpus is now ordered, we can query the suffix array quickly using binary search to efficiently find all occurrences of a particular token or sequence of tokens. Then we can easily compute, on-the-fly, the statistics required such as translation probabilities for a given source phrase. Suffix arrays can also be compressed, which make them highly attractive structures for representing massive translation models (Callison-Burch et al., 2005; Lopez, 2008).

We need to delete items if we wish to maintain

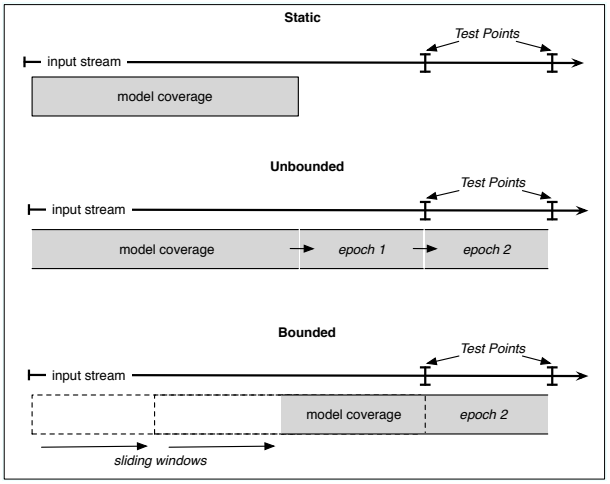

Figure 1: Streaming coverage conditions. In traditional batch based modeling the coverage of a trained model never changes. Unbounded coverage operates without any memory constraints so the model is able to continually add data from the input stream. Bounded coverage uses just a fixed window.

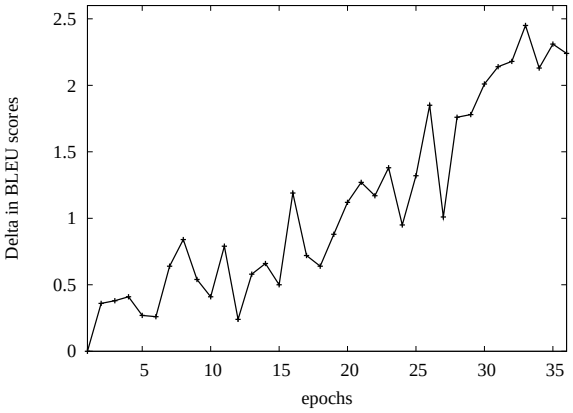

Figure 2: Recency effects to SMT performance. Depicted are the differences in BLEU scores for multiple test points decoded by a static baseline system and a system batched retrained on a fixed sized window prior to the test point in question. The results are accentuated at the end of the timeline when more time has passed confirming that recent data impacts translation performance.

constant space when processing unbounded streams. Standard suffix arrays are static, store a fixed corpus and do not support deletions. Nevertheless, a dynamic variant of the suffix array does support deletions as well as insertions and therefore can be used in our stream-based approach (Salson et al., 2009). Using a dynamic suffix array, we can compactly represent the set of parallel sentences from which we eventually extract grammar rules. Furthermore, when incorporating new parallel sentences, we simply insert them into the array and, to maintain constant space usage, we delete an equivalent number.

4 Experiments

In this section we describe the experiments conducted comparing various batch trained translation models (TMs) versus online incrementally retrained TMs in a full SMT setting with different conditions set on model coverage. We used publicly available resources for all our tests. We start by showing that recency motivates incremental retraining.

4.1 Effects of Recency on SMT

For language modeling, it is known that performance can be improved using the criterion of *recency* where training data is drawn from times chronologically closer to the test data (Rosenfeld, 1995). Given an incoming stream of parallel text, we gauged the extent to which incorporating recent data into a TM affects translation quality.

We used the Europarl corpus[5] with the Fr-En language pair using French as source and English as target. Europarl is released in the format of a daily parliamentary session per time-stamped file. The actual dates of the full corpus are interspersed unevenly (they do not convene daily) over a continuous timeline corresponding to the parliament sessions from April,1996 through October, 2006, but for conceptual simplicity we treated the corpus as a continual input stream over consecutive days.

As a baseline we aligned the first 500k sentence pairs from the beginning of the corpus timeline. We extracted a grammar for and translated 36 held out test documents that were evenly spaced along the remainder of the Europarl timeline. These test documents effectively divided the remaining training data into *epochs* and we used a *sliding window* over the timeline to build 36 distinct, overlapping training sets of 500k sentences each.

We then translated all 36 test points again using a new grammar for each document extracted from only the sentences contained in the epoch that was before it. To explicitly test the effect of recency

[5] Available at `http://www.statmt.org/europarl`

on the TM all other factors of the SMT pipeline remained constant including the language model and the feature weights. Hence, the only change from the static baseline to the epochs performance was the TM data which was based on recency. Note that at this stage we did not use any incremental retraining.

Results are shown in Figure 2 as the differences in BLEU score (Papineni et al., 2001) between the baseline TM versus the translation models trained on material chronologically closer to the given test point. The consistently positive deltas in BLEU scores between the model that is never retrained and the models that are retrained show that we achieve a higher translation performance when using more up-to-date TMs that incorporate recent sentence pairs. As the chronological distance between the initial, static model and the retrained models increases, we see ever-increasing differences in translation performance. This underlines the need to retrain translation models with timely material.

4.2 Unbounded and Bounded Translation Model Retraining

Here we consider how to process a stream along two main axes: by bounding time (batch versus incremental retraining) and by bounding space (either using all the stream seen so far, or only using a fixed sized sample of it).

To ensure the recency results reported above were not limited to French-English, this time our parallel input stream was generated from the German-English language pair of Europarl with German as source and English again as target. For testing we held out a total of 22k sentences from 10 evenly spaced intervals in the input stream which divided the input stream into 10 epochs. Stream statistics for three example epochs are shown in Table 1. We held out 4.5k sentence pairs as development data to optimize the feature function weights using minimum error rate training (Och, 2003) and these weights were used by all models. We used *Joshua* (Li et al., 2009), a syntax-based decoder with a suffix array implementation, and rule induction via the standard *Hiero* grammar extraction heuristics (Chiang, 2007) for the TMs. Note that nothing hinges on whether we used a syntax or a phrase-based system.

We used a 5-gram, Kneser-Ney smoothed language model (LM) trained on the initial segment of the target side parallel data used in the first baseline as described further in the next subsection. As our initial experiments aim to isolate the effect of changes to the TM on overall translation system performance, our in-domain LM remains static for every decoding run reported below until indicated.

Ep	From–To	Sent Pairs	Source/Target
00	04/1996–12/2000	600k	15.0M/16.0M
03	02/2002–09/2002	70k	1.9M/2.0M
06	10/2003–03/2004	60k	1.6M/1.7M
10	03/2006–09/2006	73k	1.9M/2.0M

Table 1: Date ranges, total sentence pairs, and source and target word counts encountered in the input stream for example epochs. Epoch 00 is baseline data that is also used as a seed corpus for the online models.

We used the open-source toolkit GIZA++ (Och and Ney, 2003) for all word alignments. For the online adaptation experiments we modified Model 1 and the HMM model in GIZA++ to use the sOEM algorithm. Batch baselines were aligned using the standard version of GIZA++. We ran the batch and incremental versions of Model 1 and HMM for the same number of iterations each in both directions.

4.3 Time and Space Bounds

For both batch and sOEM we ran a number of experiments listed below corresponding to the different training scenarios diagrammed in Figure 1.

1. **Static**: We used the first half of the input stream, approximately 600k sentences and 15/16 million source/target words, as parallel training data. We then translated each of the 10 test sets using the static model. This is the traditional approach and the coverage of the model never changes.

2. **Unbounded Space**: Batch or incremental retraining with no memory constraint. For each epoch in the stream, we retrained the TM using *all* the data from the beginning of the input stream until just before the present with respect to a given test point. As more time passes our training data set grows so each *batch* run of GIZA++ takes more time. Overall this is the most computationally expensive approach.

			Baseline		Unbounded		Bounded	
Epoch	Test Date	Test Sent.	Train Sent.	Rules	Train Sent.	Rules	Train Sent.	Rules
03	09/23/2002	1.0k	580k	4.0M	800k	5.0M	580k	4.2M
06	03/29/2004	1.5k	580k	5.0M	1.0M	7.0M	580k	5.5M
10	09/26/2006	3.5k	580k	8.5M	1.3M	14.0M	580k	10.0M

Table 2: Translation model statistics for example epochs and the next test dates grouped by experimental condition. *Test* and *Train Sent.* is the number of sentence pairs in test and training data respectively. *Rules* is the count of unique Hiero grammar rules extracted for the corresponding test set.

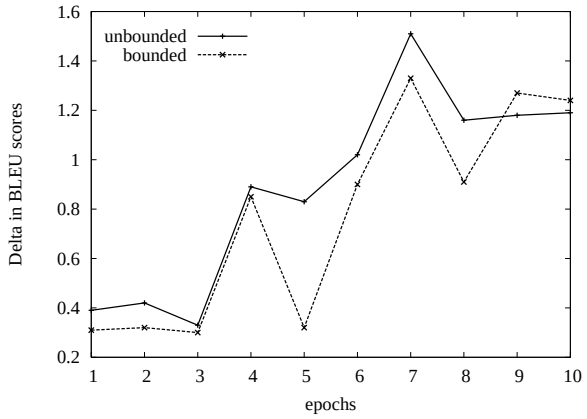

Figure 3: Static vs. online TM performance. Gains in translation performance measured by BLEU are achieved when recent German-English sentence pairs are automatically incorporated into the TM. Shown are relative BLEU improvements for the online models against the static baseline.

3. **Bounded Space**: Batch and incremental retraining with an enforced memory constraint. Here we batch or incrementally retrain using a *sliding window* approach where the training set size (the number of sentence pairs) remains constant. In particular, we ensured that we used the same number of sentences as the baseline. Each batch run of GIZA++ takes approximately the same time.

The *time* for aligning in the sOEM model is unaffected by the bounded/unbounded conditions since we always only align the mini-batch of sentences encountered in the last epoch. In contrast, for batch EM we must realign all the sentences in our training set from scratch to incorporate the new training data.

Similarly *space* usage for the batch training grows with the training set size. For sOEM, in theory memory used is with respect to vocabulary size (which grows slowly with the stream size) since we retain count history for the entire stream. To make space usage truly constant, we filter for just the needed word pairs in the current epoch being aligned. This effectively means that online EM is more memory efficient than the batch version. As our experiments will show, the sufficient statistics kept between epochs by sOEM benefits performance compared to the batch models which can only use information present within the batch itself.

4.4 Incremental Retraining Procedure

Our incremental adaptation procedure was as follows: after the latest mini-batch of sentences had been aligned using sOEM we added all newly aligned sentence pairs to the dynamic suffix arrays. For the experiments where our memory was bounded, we also *deleted* an equal number of sentences from the suffix arrays before extracting the Hiero grammar for the next test point. For the unbounded coverage experiments we deleted nothing prior to grammar extraction. Table 2 presents statistics for the number of training sentence pairs and grammar rules extracted for each coverage condition for various test points.

4.5 Results

Figure 3 shows the results of the static baseline against both the unbounded and bounded online EM models. We can see that both the online models outperform the static baseline. On average the unconstrained model that contains more sentence pairs for rule extraction slightly outperforms the bounded condition which uses less data per epoch. However, the static baseline and the bounded models both use the same number of sentence-pairs for TM training. We see there is a clear gain by incorporating recent sentence-pairs made available by the stream.

	Static Baseline	Retrained (Unbounded)		Retrained (Bounded)	
Test Date	Batch	Batch	Online	Batch	Online
09/23/2002	26.10	**26.60**	26.43	26.19	*26.40*
03/29/2004	27.40	28.33	**28.42**	28.06	*28.38*
09/26/2006	28.56	29.74	**29.75**	29.73	*29.80*

Table 3: Sample BLEU results for all baseline and online EM model conditions. The *static baseline* is a traditional model that is never retrained. The *batch unbounded* and *batch bounded* models incorporate new data from the stream but retraining is slow and computationally expensive (best results are bolded). In contrast both unbounded and bounded online models incrementally retrain only the mini-batch of new sentences collected from the incoming stream so quickly adopt the new data (best results are italicized).

Table 3 gives results of the online models compared to the batch retrained models. For presentation clarity we show only a sample of the full set of ten test points though all results follow the pattern that using more aligned sentences to derive our grammar set resulted in slightly better performance versus a restricted training set. However, for the same coverage constraints not only do we achieve comparable performance to batch retrained models using the sOEM method of incremental adaptation, we are able to align and adopt new data from the input stream orders of magnitude quicker since we only align the mini-batch of sentences collected from the last epoch. In the bounded condition, not only do we benefit from quicker adaptation, we also see that sOEM models slightly outperform the batch based models due to the online algorithm employing a longer history of count-based evidence to draw on when aligning new sentence pairs.

Figure 4 shows two example test sentences that benefited from the online TM adaptation. Translations from the online model produce more and longer matching phrases for both sentences (e.g., “creation of such a”, “of the occupying forces”) leading to more fluent output as well as the improvements achieved in BLEU scores.

We experimented with a variety of interpolation parameters (see Algorithm 2) but found no significant difference between them (the biggest improvement gained over all test points for all parameter settings was less than 0.1% BLEU).

4.6 Increasing LM Coverage

A natural and interesting extension to the experiments above is to use the target side of the incoming stream to extend the LM coverage alongside the TM.

Test Date	Static	Unbounded	Bounded
09/23/2002	26.46	27.11	26.96
03/29/2004	28.11	29.53	29.20
09/26/2006	29.53	30.94	30.88

Table 4: Unbounded LM coverage improvements. Shown are the BLEU scores for each experimental conditional when we allow the LM coverage to increase.

It is well known that more LM coverage (via larger training data sets) is beneficial to SMT performance (Brants et al., 2007) so we investigated whether recency gains for the TM were additive with recency gains afforded by a LM.

To test this we added all the target side data from the beginning of the stream to the most recent epoch into the LM training set before each test point. We then batch retrained[6] and used the new LM with greater coverage for the next decoding run. Experiments were for the static baseline and online models.

Results are reported in Table 4. We can see that increasing LM coverage is complimentary to adapting the TM with recent data. Comparing Tables 3 and 4, for the bounded condition, adapting only the TM achieved an absolute improvement of +1.24 BLEU over the static baseline for the final test point. We get another absolute gain of +1.08 BLEU by allowing the LM coverage to adapt as well. Using an online, adaptive model gives a total gain of +2.32 BLEU over a static baseline that does not adapt.

[6] Although we batch retrain the LMs we could use an online LM that incorporates new vocabulary from the input stream as in Levenberg and Osborne (2009).

Source: Die Kommission ist bereit, an der Schaffung eines solchen Rechtsrahmens unter Zugrundelegung von vier wesentlichen Prinzipien mitzuwirken.

Reference: The commission is willing to cooperate in the creation of such a legal framework on the basis of four essential principles.

Static: The commission is prepared, in the creation of a legal framework, taking account of four fundamental principles them.

Online: The commission is prepared to participate in the creation of such a legal framework, based on four fundamental principles.

Source: Unser Standpunkt ist klar und allseits bekannt: Wir sind gegen den Krieg und die Besetzung des Irak durch die USA und das Vereinigte Königreich, und wir verlangen den unverzüglichen Abzug der Besatzungsmächte aus diesem Land.

Reference: Our position is clear and well known: we are against the war and the US-British occupation in Iraq and we demand the immediate withdrawal of the occupying forces from that country.

Static: Our position is clear and we all know: we are against the war and the occupation of Iraq by the United States and the United Kingdom, and we are calling for the immediate withdrawal of the besatzungsmächte from this country.

Online: Our position is clear and well known: we are against the war and the occupation of Iraq by the United States and the United Kingdom, and we demand the immediate withdrawal of the occupying forces from this country .

Figure 4: Example sentences and improvements to their translation fluency by the adaptation of the TM with recent sentences. In both examples we get longer matching phrases in the online translation compared to the static one.

5 Related Work

5.1 Translation Model Domain Adaptation

Our work is related to domain adaptation for translation models. See, for example, Koehn and Schroeder (2007) or Bertoldi and Federico (2009). Most techniques center around using mixtures of translation models. Once trained, these models generally never change. They therefore fall under the *batch* training regime. The focus of this work instead is on incremental retraining and also on supporting bounded memory consumption. Our experiments examine updating model parameters in a single domain over different periods in time. Naturally, we could also use domain adaptation techniques to further improve how we incorporate new samples.

5.2 Online EM for SMT

For stepwise online EM for SMT models, the only prior work we are aware of is Liang and Klein (2009), where variations of online EM were experimented with on various NLP tasks including word alignments. They showed application of sOEM can produce quicker convergence compared to the batch EM algorithm. However, the model presented does not incorporate any unseen data, instead iterating over a static data set multiple times using sOEM. For Liang and Klein (2009) incremental retraining is simply an alternative way to use a fixed training set.

5.3 Streaming Language Models

Recent work in Levenberg and Osborne (2009) presented a streaming LM that was capable of adapting to an unbounded monolingual input stream in constant space and time. The LM has the ability to add or delete n-grams (and their counts) based on feedback from the decoder after translation points. The model was tested in an SMT setting and results showed recent data benefited performance. However, adaptation was only to the LM and no tests were conducted on the TM.

6 Conclusion and Future Work

We have presented an online EM approach for word alignments. We have shown that, for a SMT system, incorporating recent parallel data into a TM from an input stream is beneficial to translation performance compared to a traditional, static baseline.

Our strategy for populating the suffix array was simply to use a first-in, first-out stack. For future work we will investigate whether information provided by the incoming stream coupled with the feedback from the decoder allows for more sophisticated adaptation strategies that reinforce useful word alignments and delete bad or unused ones.

In the near future we also hope to test the online EM setup in an application setting such as a computer aided translation or crowdsourced generated streams via Amazon's Mechanical Turk.

Acknowledgements

Research supported by EuroMatrixPlus funded by the European Commission, by the DARPA GALE program under Contract Nos. HR0011-06-2-0001 and HR0011-06-C-0022, and the NSF under grant IIS-0713448.

References

Nicola Bertoldi and Marcello Federico. 2009. Domain adaptation for statistical machine translation with monolingual resources. In *WMT09: Proceedings of the Fourth Workshop on Statistical Machine Translation*, pages 182–189, Morristown, NJ, USA. Association for Computational Linguistics.

Thorsten Brants, Ashok C. Popat, Peng Xu, Franz J. Och, and Jeffrey Dean. 2007. Large language models in machine translation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 858–867.

Peter F. Brown, Vincent J. Della Pietra, Stephen A. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: parameter estimation. *Computational Linguistics*, 19(2):263–311.

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

Olivier Cappe and Eric Moulines. 2009. Online EM algorithm for latent data models. *Journal Of The Royal Statistical Society Series B*, 71:593.

David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228.

A. P. Dempster, N. M. Laird, and D. B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39:1–38.

Philipp Koehn and Josh Schroeder. 2007. Experiments in domain adaptation for statistical machine translation. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 224–227, Prague, Czech Republic, June. Association for Computational Linguistics.

Abby Levenberg and Miles Osborne. 2009. Stream-based randomised language models for SMT. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Zhifei Li, Chris Callison-Burch, Chris Dyer, Juri Ganitkevitch, Sanjeev Khudanpur, Lane Schwartz, Wren N. G. Thornton, Jonathan Weese, and Omar F. Zaidan. 2009. Joshua: an open source toolkit for parsing-based machine translation. In *WMT09: Proceedings of the Fourth Workshop on Statistical Machine Translation*, pages 135–139, Morristown, NJ, USA. Association for Computational Linguistics.

Percy Liang and Dan Klein. 2009. Online EM for unsupervised models. In *North American Association for Computational Linguistics (NAACL)*.

Adam Lopez. 2008. Tera-scale translation models via pattern matching. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 505–512, Manchester, UK, August. Coling 2008 Organizing Committee.

Udi Manber and Gene Myers. 1990. Suffix arrays: A new method for on-line string searches. In *The First Annual ACM-SIAM Symposium on Dicrete Algorithms*, pages 319–327.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51, March.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *ACL '03: Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*, pages 160–167, Morristown, NJ, USA. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2001. Bleu: a method for automatic evaluation of machine translation. In *ACL '02: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 311–318, Morristown, NJ, USA. Association for Computational Linguistics.

Ronald Rosenfeld. 1995. Optimizing lexical and n-gram coverage via judicious use of linguistic data. In *In Proc. European Conf. on Speech Technology*, pages 1763–1766.

Mikaël Salson, Thierry Lecroq, Martine Léonard, and Laurent Mouchard. 2009. Dynamic extended suffix arrays. *Journal of Discrete Algorithms*, March.

Stephan Vogel, Hermann Ney, and Christoph Tillmann. 1996. HMM-based word alignment in statistical translation. In *Proceedings of the 16th conference on Computational linguistics*, pages 836–841, Morristown, NJ, USA. Association for Computational Linguistics.