PBML

The Prague Bulletin of Mathematical Linguistics
NUMBER 108 JUNE 2017 233–244

Continuous Learning from Human Post-Edits for Neural Machine Translation

Marco Turchi,[a] Matteo Negri,[a] M. Amin Farajian,[ab] Marcello Federico[a]

[a] Fondazione Bruno Kessler, Trento, Italy
[b] University of Trento, Italy

Abstract

Improving machine translation (MT) by learning from human post-edits is a powerful solution that is still unexplored in the neural machine translation (NMT) framework. Also in this scenario, effective techniques for the continuous tuning of an existing model to a stream of manual corrections would have several advantages over current batch methods. First, they would make it possible to adapt systems at run time to new users/domains; second, this would happen at a lower computational cost compared to NMT retraining from scratch or in batch mode. To attack the problem, we explore several online learning strategies to stepwise fine-tune an existing model to the incoming post-edits. Our evaluation on data from two language pairs and different target domains shows significant improvements over the use of static models.

1. Introduction

In the last couple of years, after more than a decade of supremacy in shared evaluation campaigns like WMT (Bojar, 2016) and IWSLT (Cettolo et al., 2015), phrase-based SMT approaches have been significantly outperformed by neural solutions. However, despite the impressive progress of the so-called encoder-decoder NMT architectures (Bahdanau et al., 2014), MT is still far from being a solved problem. Together with the high computational training costs, one of the downsides of NMT is that performance can be significantly affected by situations in which training and testing are performed on heterogeneous data (*e.g.* coming from different domains or featuring different vocabulary and sentence structure). In this challenging condition, the availability of

Corresponding author: turchi@fbk.eu
Cite as: Marco Turchi, Matteo Negri, M. Amin Farajian, Marcello Federico. Continuous Learning from Human Post-Edits for Neural Machine Translation. The Prague Bulletin of Mathematical Linguistics No. 108, 2017, pp. 233–244.
doi: 10.1515/pralin-2017-0023.

task-specific (*e.g.* in-domain) data makes it possible to reduce the performance drops by means of fine-tuning procedures that are much faster than full model retraining.

Fine-tuning is usually applied in "batch" conditions, in which a general out-of-domain NMT model is further trained on in-domain data *before testing*. This paper investigates its adoption at the core of an "online" learning approach in which, *at test stage*, the NMT model is continuously adapted to a stream of incoming parallel sentence pairs. This scenario is relevant for the so-called computer-assisted translation (CAT) framework, which now represents the standard operating environment in the translation industry. Given a source sentence to translate (*src*), translators working with a CAT tool operate on machine-derived suggestions (*tgt*) correcting them, when necessary, into post-edited (*pe*) translations of the desired level of quality. Even if not perfect, MT suggestions normally require less post-editing effort compared to manual translation from scratch. In this "translation as post-editing" process, new data in the form of *(src, tgt, pe)* triples are continuously generated, thus providing a wealth of material to tune and adapt existing NMT models to specific users and domains.

The exploitation of human post-edits in a continuous learning NMT framework represents an ideal scenario for deploying online learning techniques. In machine learning, online learning is defined as the task of using data that becomes available in a sequential order to stepwise update a predictor for future data. The new points used for the update often consist in labeled instances provided as external feedback (*i.e.* a "true" label representing the expected response for each given input). At each step, the difference between a prediction $p(x_i)$ and the corresponding true label $\hat{p}(x_i)$ is used by the learner to refine the next prediction $p(x_{i+1})$. In this way, a general model can evolve over time by integrating external feedback in order to reduce the distance between its predictions and the expected output. Such evolution can result in a general performance improvement but also, depending on the working scenario, in an adaptation to the specificities of the target application domain.

Cast as an online learning problem, our task consists in leveraging a stream of human post-edited data for continuous NMT adaptation. Along this direction, our contributions can be summarized as follows: (1) we define and explore for the first time an application-oriented framework for continuous NMT adaptation from human feedback, which is suitable for deployment in the CAT framework; (2) we propose different strategies to approach the problem; (3) we evaluate them in two different scenarios (different target languages, domains and levels of training/test data mismatch).

2. Related work

Previous work on online MT adaptation is motivated by the problem of performance degradation when training and testing on heterogeneous data. In phrase-based MT, this problem has been widely explored. The proposed solutions include the use of incremental expectation-maximization (EM) and suffix arrays to update the statistics of a generic model (Ortiz-Martínez et al., 2010, Ortiz-Martínez, 2016,

Germann, 2014), cache-based models (Bertoldi et al., 2013), discriminative approaches based on structured perceptrons (Wäschle et al., 2013), incremental Bayesian language models (Denkowski et al., 2014), and hierarchical methods (Wuebker et al., 2015).

In NMT, online methods have not been explored yet. In fact, adaptation approaches mostly rely on batch fine-tuning procedures that are carried out on a small corpus of "in-domain data".[1] To cope with training/test heterogeneity, fine-tuning consists in exploiting the availability of in-domain data representative of the test set to perform a focused additional training step (Luong and Manning, 2015). Despite some risk of overfitting to the small size of the in-domain data, this practice often results in significant performance gains. An interesting variant, closer to our approach, is proposed in (Li et al., 2016). It presents an on-the-fly local adaptation method which, for each incoming test sentence, performs fine-tuning on source-reference pairs extracted from the parallel corpus used to train the general model. This solution, however, does not take into account human feedback (post-edits), as the retrieval step is carried out on a static pool of parallel training data. In contrast, in our online scenario we explore different strategies for continuous NMT model update by fine-tuning on a dynamic pool that incorporates a stream of human post-edited data from a given (possibly new) domain. Different from (Li et al., 2016), moreover, our retrieval step is based on faster and more powerful information retrieval techniques (ngram-based search with Lucene), which reward longer matches of relevant terms (as opposed to Levenshtein distance, and the other similarity methods proposed in (Li et al., 2016), which treat all the matching terms equally).

Finally, among the strategies explored in this paper, we also consider the case in which the general model evolves over time (*i.e.* the updated model for sentence *n* becomes the starting model for sentence *n+1*). In (Li et al., 2016), instead, the original model is always restored before processing each incoming sentence.

3. Integrating User Feedback

In order to exploit human feedback for continuous NMT model update, we explore three possible strategies, in which post-edited data are respectively used: *i)* for global model improvements after translating an input sentence (§ 3.1), *ii)* as additional knowledge for local improvements before translating the input sentence (§ 3.2), or *iii)* for both global and local improvements (§ 3.3).

3.1. Adaptation "a posteriori"

This strategy makes a direct use of user feedback for updating a general model as in any standard online MT framework, that is by using human feedback in the form

[1] With the expression "in-domain" we broadly refer to data that differ from those used for training the model. This mismatch can be due to an actual difference in terms of semantic domain, but also to other discrepancies in terms of style, vocabulary, sentence structure, etc.

of *(src,pe)* pairs to stepwise update the MT model *after translating* each segment. After receiving the human post-edit (*pe*) of the translation (*tgt*) of a given segment (*src*), the goal is to learn from the *(src,pe)* pair and induce the model to better translate the next input segment. This is done by performing a further fine-tuning step of the original model, which consists of one (or more) training iterations over the *(src,pe)* pair.

Overall, adaptation a posteriori is rather conservative since, at each step, the changes of the general model are induced only from a parallel sentence pair consisting of a source segment coming from the target domain and its human post-edit.

3.2. Adaptation "a priori"

This strategy, inspired by the approach of Li et al. (2016), makes an indirect use of user feedback. It relies on an update step to locally adapt the model to each incoming segment *before translating* it. Given an input sentence (*src*), parallel sentence pairs in which the source side is similar to *src* are retrieved from the data used to train the general model. The retrieved material is used to fine-tune the general model, which results in a local model that will be used to translate the input sentence. Although in the approach of Li et al. (2016) the starting general model is the same for each input sentence, nothing prevents to take advantage from new *(src,pe)* pairs as long as they become available. To this aim, instead of keeping fixed the pool of parallel data accessed for the local update, we experiment with a pool that is continuously populated with the previously collected *(src,pe)* instances. Differently from (Li et al., 2016), in which the data for local adaptation are retrieved by computing similarities based on Levenshtein distance, word embeddings and the NMT encoder's hidden states, we adopt standard information retrieval techniques. In particular, we use the Shingle[2] filter of Lucene (McCandless et al., 2010), which performs ngram-based searches that reward at the same time relevant and longer matches. In our experiments, for each query (*i.e.* input sentence), the top matching source sentence retrieved from the pool and the corresponding translation are used to perform the local fine-tuning step.

Though more focused compared to the previous approach, adaptation a priori is potentially more risky. Indeed, at each step, the local adaptation of the general model is based on similar (but not necessarily relevant/useful) sentence pairs.

3.3. Double adaptation

The two previous approaches can be combined in the general scheme depicted in Figure 1. In this case, given an input sentence (*src*), the general NMT model (GM1) is first adapted locally by performing a fine-tuning step on similar data retrieved from the parallel data pool (training pairs + previously collected (*src,pe*) pairs). Then, the resulting adapted model (LM1) is used to translate the sentence. After receiving the

[2] goo.gl/HzeSAI

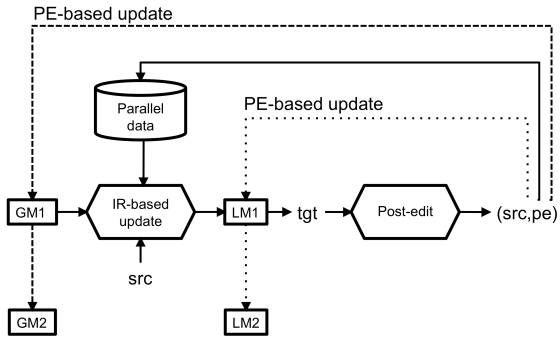

Figure 1. Double adaptation process.

human post-edit (*pe*) of the output translation (*tgt*), two options are possible. One is to exploit the (*src,pe*) pair to update the general model, which will be used as the starting model (GM2) for the next input segment. The second option is to exploit the (*src,pe*) pair to update the local model (LM1), which will be used as the starting model (LM2) for the next input segment. By adopting the first option, the translation process will rely on a chain of continuously evolving generic models (GM1, GM2, ..., GMn). By adopting the second option, the translation process will rely on a chain of models (GM1, LM1, LM2, ..., LMn) that, starting from the initial general model, evolves through local adaptations.

4. Experimental Setup

4.1. Approaches

Figure 2 illustrates the approaches compared in our experiments. The first one (a) is our baseline. It consists of a static NMT model (GM1), which is used to process the entire stream of data without changing over time (*i.e.* without learning from the *(src,pe)* sentence pairs obtained as human feedback). The second approach (b) is the adaptation "a posteriori" described in § 3.1, in which a general model is continuously fine-tuned to each incoming *(src,pe)* segment (GM1 for the first segment, GM2 for the second, and so on). The third and fourth approaches (c and d) represent the adaptation "a priori" described in § 3.2. In one case (c), for the first input segment to translate, a general model (GM1) is locally fine-tuned to similar sentence pairs retrieved from the pool of parallel data (recall that similarity is computed between the input segment and the source side of the instances in the pool). After translation, the same initial model (GM1) is used for the second input segment, and so on. In the other case (d), the locally-adapted model (LM1) is kept after translating the first input segment and used as starting model for the second one. The fifth and sixth so-

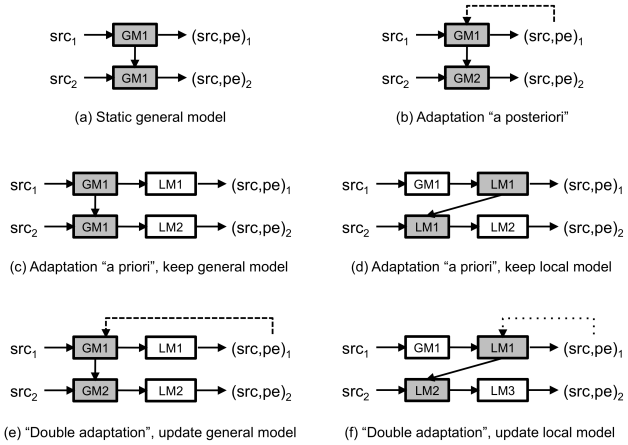

Figure 2. Static (a) vs online (b, c, d, e, f) NMT approaches.

lutions (e and f) represent the double adaptation method described in § 3.3. In (e), the general model (GM1) is locally fine-tuned (LM1) to translate the first input segment. Then, after translation and post-editing, human feedback is used to fine-tune again the general model, which will become the starting model (GM2) for the second input segment, and so on. In (f), the second fine-tuning step is applied to the local model.

4.2. Data

Our evaluation is carried out on two different language pairs and domains. The first scenario consists in translating information technology (IT) English sentences into German using a large set of heterogeneous parallel data to train the NMT system. In the second scenario, the NMT is trained on a small quantity of domain-specific data and it is used to translate medical English segments into Latvian. The two conditions pose different challenges. In the En_De setting, the initial NMT model is trained on a large general dataset that scarcely represents the target domain. *Hence, continuous learning mainly acts as a domain-adaptation process.* For En_Lv, the model is trained on domain-specific data, but in limited quantity. *Hence, the goal is to improve the overall translation quality by leveraging the new incoming data.* Regarding the target languages, Latvian is a Baltic language that is much more inflected that German. This results in a more sparse vocabulary that can affect translation performance.

For training the En_De NMT system, we merged the Europarl v7 (Koehn, 2005) and Common Crawl datasets released for the translation task at the 2016 Workshop on Statistical Machine Translation (WMT'16 (Bojar, 2016)) and random sampled 3.5 million sentence pairs. As domain-specific test set, we randomly selected 3k instances

from the training data released for the automatic post-editing task at WMT'16 (ibid.). This dataset consists of 12k *(src,tgt,pe)* triples, in which the source sentences come from an IT manual and the post-edits are generated by professional translators. In our experiments, the source sentences are translated by our NMT system and we assume that the existing post-edits are corrections of the NMT output.

For training the En_Lv NMT system, we used a subset of the EMEA parallel corpus proposed in (Pinnis et al., 2016). The test set is obtained by extracting 3k consecutive segments from randomly selected EMEA documents. Post-edits were generated by professional translators who corrected the output of our NMT system. Some data statistics are reported in Table 1.

	En_De			En_Lv		
	train	dev	test	train	dev	test
Number of sentence pairs	3.5M	2K	3K	385K	2K	3K
Source language tokens	63M	18K	50K	60.5M	20K	54K
Target language tokens	7.5M	37K	55K	6.8M	34K	50K

Table 1. Statistics about the En_De and En_Lv training, dev and test corpora.

4.3. NMT System

All the experiments are conducted with an in-house developed and maintained branch of the Nematus toolkit,[3] which is an implementation of the attentional encoder-decoder architecture (Bahdanau et al., 2014). Models were trained by splitting words into sub-word units using byte pair encoding (BPE), which Sennrich et al. (2016) indicates as an effective way to handle large vocabularies (*e.g.* to deal with rare words and highly inflected languages). Word segmentation was carried out by combining the source and target side of the training set and setting the number of merge rules to 40,000 for both language pairs. We used mini-batches of size 100, word embeddings of size 1024, and hidden layers of size 1024. The maximum sentence length was set to 50. The models were trained using Adam (Kingma and Ba, 2015) with an initial learning rate of 0.001, reshuffling the training corpora at each epoch. In both language pairs, the training of the generic systems was stopped after 20 epochs. Dropout is disabled.

5. Impact of gradient descent optimization algorithms

Following Li et al. (2016), in all the scenarios proposed in § 3 our NMT models are always updated using one single sentence pair. This is quite unusual for the NMT training common practice, in which batches containing dozens of sentence pairs are

[3] https://github.com/rsennrich/nematus

normally used. Leaving for future work the investigation on how to exploit larger sets of retrieved sentences, we run several experiments to measure the impact on performance of different optimization algorithms when using only one sentence pair.

The most used family of optimization algorithms is based on gradient descent, which is a way to minimize an objective function $J(\Theta)$, where $\Theta \in \mathbb{R}^d$, by updating the Θ parameters in the opposite direction of the gradient of the objective function $\nabla_\Theta J(\Theta)$. The learning rate η determines the size of the steps we take to reach a (local) minimum. Among the several optimizers proposed in literature, in our experiments we test: *i)* stochastic gradient descent (Sgd) (Bottou, 2010); *ii)* Adagrad (Duchi et al., 2011), *iii)* Adadelta (Zeiler, 2012) and Adam (Kingma and Ba, 2015). The main differences between these methods lie on the use of gradient information from the past time steps and on the way learning rates are updated. Sgd performs a parameter update for each training example ignoring the past gradient information and using a fix η chosen a priori. Differently from Sgd, Adagrad adapts the learning rate to the parameters using the past information and by performing larger updates for infrequent parameters and smaller updates for the frequent ones. Adadelta extends Adagrad by restricting the window of accumulated past gradients to some fixed size in order to mitigate the problem of a too fast monotonic decrease of the learning rate, which rapidly gets close to zero when all past gradients are retained. Adam introduces a bias correction mechanism and a better handling of moment information to induce faster parameter variations in the right direction.

	adam	adagrad	adadelta	sgd 1	sgd 0.1	sgd 0.01	sgd 0.001
En_De	39.1	30.1	44.4	37.2	**50.2**	47.5	44.0
En_Lv	28.3	16.2	38.7	34.5	**47.9**	47.3	46.8

Table 2. Results (BLEU) of different optimization algorithms.

In this set of experiments, we only consider the "a posteriori" adaptation strategy, which uses reliable in-domain *(src,pe)* pairs (see Figure 2(b)). In contrast with "a priori" adaptation, which operates on similar but potentially noisy retrieved instances, we believe that reliable insights will more likely come from this setting. Several learning rates are tested for Sgd (*i.e.* 1, 0.1, 0.01 and 0.001), while the other optimizers are initialized with a learning rate of 0.01. The BLEU results for both language pairs are reported in Table 2. Sgd generally performs better than the other optimizers that result in significantly lower scores. Looking at the different values of the Sgd learning rate, the performance improves when a larger η is used. For both languages, this is valid up to η equal to 0.1 while, for larger values, also Sgd results in poor translations.

The superiority of Sgd in our scenario contrasts with the results achievable in NMT when learning from a batch of sentence pairs, which usually favor dynamic optimizers. Our explanation is that gradients computed on a batch are more stable and less

affected by differences between sentence pairs. For this reason, optimizers that can leverage past gradient information are usually more reliable. When working with only one sentence pair, segments from the same document may have different structure, words and length, which makes gradient information from the past potentially misleading and causing instability in the optimizer. Since this problem would likely be exacerbated when adapting to diverse and potentially noisy data in the "a priori" setting, Sgd seems to be a safer solution for our case. In the remainder of the paper, all the experiments are run using Sgd with learning rate of 0.1.

6. Analysis of continuous learning strategies

Table 3 reports the results of a comparison between the adaptation strategies discussed in §4.1 and two baselines that do not exploit human feedback. The first one (Static) is an NMT model that is kept unchanged during the processing of the entire test set. The second one ("a priori w/o PE") is our re-implementation of (Li et al., 2016), which locally adapts the original NMT model to each test sentence by finding the most similar instance in the training data. After each translation, the locally-updated model is replaced by the initial general model, which is used as a starting point for the next sentence. This approach resembles our "a priori – keep general model" adaptation strategy (method (c) in Figure2) with the exception that human post-edits are not added to the pool of data accessed by Lucene. The reported results are obtained by iterating for 1 and 5 epochs over each sentence pair during updating.

By comparing the BLEU scores of the static and our re-implementation of (Li et al., 2016), it becomes evident that simple local adaptation has a marginal impact on the results (even negative for En_De with 5 epochs, with a drop of ~2 BLEU points). Although this contrasts with the results of Li et al. (2016), what is interesting to note here (more than comparing similar approaches on different language directions and data) is the scarce contribution, in our testing conditions, of retrieving instances from the static pool of training data. More visible improvements are in fact yielded by the application of the "a priori with PE" strategy, which takes advantage of a data pool that constantly grows by integrating human post-edits. With 1 fine-tuning epoch, the new domain-specific information results in slight improvements, on both language pairs, both over the baseline and over the "a priori w/o PE" adaptation. The gain is small (and not significant) on En_Lv, probably due to the fact that the original NMT model is domain-specific, hence already adapted to the target domain. In this case, performance is almost identical either if we Keep the General model after processing each sentence (K.G., which corresponds to method (c) in Figure 2) or if we Keep the Local model (K.L., method (d)). For En_De, improvements are significant in both conditions, especially when keeping the local model (+1.7 for K.L. vs. +0.5 for K.G.). This suggests that, despite the risks inherent to the "a priori" strategy, which adapts the NMT model to the retrieved sentence pairs independently from their degree of similarity with the sentence to translate, the locally-adapted model can be useful also for

		Static	a priori w/o PE	a priori with PE		a post.	Double	
				K. G.	K. L.		U. G.	U. L.
En_De	1 epoch	42.7	42.8	43.3*	44.5*	**50.2***	50.0*	48.4*
	5 epochs		40.9†	41.7†	41.8†	**49.2***	47.9*	46.3*
En_Lv	1 epoch	46.8	46.8	46.9	47.0	**47.9***	47.8*	47.4*
	5 epochs		46.9	47.2	47.3*	**48.3***	48.0*	47.5*

*Table 3. Results of different adaptation strategies. * and † respectively indicate statistically significant improvements/degradations with respect to the static system. Significance tests are performed with paired bootstrap resampling (Koehn, 2004).*

the next incoming sentences. With 5 fine-tuning epochs, instead, we observe mixed results. On En_De, in which training and test are heterogeneous, local adaptation overfits to sentences that can feature low similarity with the input and is definitely harmful (both K.G. and K.L. results are significantly below the baseline). On En_Lv, for which training and test data are homogeneous, we observe slight improvements, which are significant when keeping the local model (+0.5 BLEU with K.L.).

The use of post-edits a posteriori ("a post." column) results in a significant improvement over the baseline on both language pairs (+7.5 for En_De and +1.1 for En_Lv). We interpret these coherent gains as an indication that continuous NMT adaptation to reliable domain-specific sentence pairs reinforces the model capability to translate the incoming sentences. Again, overfitting by running more epochs yields mixed results. On En_De (heterogeneous data) performance drops but is still significantly better compared to all previous methods, while on En_Lv (homogeneous data), more epochs yield the best result. The difference between "a priori" and "a posteriori" adaptation also emerges when combining them together ("Double" column). In general, updating the General model (U.G, method (e) in Figure 2) achieves better results than Updating the Local model (U.L. method (f)), though slightly worse than "a posteriori" adaptation.

7. Conclusion and Future Work

We addressed the problem of improving an existing NMT model by continuously learning from human feedback. As opposed to batch learning techniques, continuous learning from a stream of incoming post-edits represents a promising solution for cutting the costs of resource/time-demanding routines to periodically retrain NMT models from scratch. Moreover, it would make it possible to adapt systems' behavior to users and domains while the system is in use, thus making the improvements visible in a short time and reducing the human post-editing workload. To achieve these objectives we explored different strategies, in which an NMT model is fine tuned: *i)* a posteriori (*i.e.* after receiving the human post-edit of a translated sentence), *ii)* a priori (*i.e.* locally, before translation, by learning also from previous feedback), or *iii)*

both (*i.e.* before and after translation). We experimented in different settings, with two language combinations and two target domains, either homogeneous or diverse with respect to the data used to train the initial NMT model. Our best results reveal significant gains both over a static NMT model used as a baseline and over an adaptive solution (the most similar to our *a priori* adaptation strategy), which does not exploit human feedback. Several interesting aspects have not been discussed and deserve attention in future work. From the technical side, our initial exploration of the impact of using different parameter optimizers and running different numbers of fine-tuning epochs can be extended and complemented with the analysis of: *i)* alternative instance selection techniques (*e.g.* similarity thresholds applied to the retrieved data), *ii)* dynamic, instance-specific ways to set the learning rate and the number of epochs depending on the similarity of the retrieved material with respect to an input sentence, and *iii)* the impact of fine-tuning on more than one sentence at a time. From the application side, the evaluation with multiple target domains, possibly involving professional translators operating in a computer-assisted translation environment is the first step in our agenda.

Acknowledgements

This work has been partially supported by the EC-funded H2020 projects QT21 (grant no. 645452) and ModernMT (grant no. 645487).

Bibliography

Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio. "Neural Machine Translation by Jointly Learning to align and translate". *arXiv preprint arXiv:1409.0473*, 2014.

Bertoldi, Nicola, Mauro Cettolo, and Marcello Federico. Cache-based Online Adaptation for Machine Translation Enhanced Computer Assisted Translation. In *Proc. of the XIV Machine Translation Summit*, pages 35–42, Nice, France, September 2013.

Bojar, Ondřej et al. Findings of the 2016 Conference on Machine Translation. In *Proc. of the First Conference on Machine Translation*, pages 131–198, Berlin, Germany, August 2016.

Bottou, Léon. "Large-Scale Machine Learning with Stochastic Gradient Descent". In *Proc. of COMPSTAT'2010*, pages 177–187, Paris, France, August 2010. Springer.

Cettolo, Mauro, Jan Niehues, Sebastian Stüker, Luisa Bentivogli, Roldano Cattoni, and Marcello Federico. The IWSLT 2015 Evaluation Campaign. In *Proc. of the 12th International Workshop on Spoken Language Translation (IWSLT 2015)*, Da Nang, Vietnam, 2015.

Denkowski, Michael, Chris Dyer, and Alon Lavie. Learning from Post-Editing: Online Model Adaptation for Statistical Machine Translation. In *Proc. of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, Gothenburg, Sweden, April 2014.

Duchi, John, Elad Hazan, and Yoram Singer. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *Journal of Machine Learning Research*, 2011.

Germann, Ulrich. Dynamic Phrase Tables for Machine Translation in an Interactive Post-editing Scenario. In *Proc. of the Workshop on interactive and adaptive machine translation*, pages 20–31, Vancouver, BC, Canada, 2014.

Kingma, Diederik P. and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *Proc. of the 3rd Int. Conference on Learning Representations*, pages 1–13, San Diego, USA, May 2015.

Koehn, Philipp. Statistical Significance Tests for Machine Translation Evaluation. In *Proceedings of the Empirical Methods on Natural Language Processing*, pages 388–395, 2004.

Koehn, Philipp. Europarl: A Parallel Corpus for Statistical Machine Translation. In *Proc. of the tenth Machine Translation Summit*, pages 79–86, Phuket, Thailand, 2005.

Li, Xiaoqing, Jiajun Zhang, and Chengqing Zong. "One Sentence One Model for Neural Machine Translation". *arXiv preprint arXiv:1609.06490*, 2016.

Luong, Minh-Thang and Christopher D. Manning. Mixture-Model Adaptation for SMT. In *Proc. of the 12th International Workshop on Spoken Language Translation*, pages 76–79, Da Nang, Vietnam, December 2015.

McCandless, Michael, Erik Hatcher, and Otis Gospodnetic. *Lucene in Action*. Manning Publications Co., Greenwich, CT, USA, 2010.

Ortiz-Martínez, Daniel. Online Learning for Statistical Machine Translation. *Computational Linguistics*, 42(1):121–161, 2016.

Ortiz-Martínez, Daniel, Ismael García-Varea, and Francisco Casacuberta. Online Learning for Interactive Statistical Machine Translation. In *Proc. of NAACL-HLT 2010*, pages 546–554, Los Angeles, California, June 2010.

Pinnis, Marcis, Rihards Kalnins, Raivis Skadins, and Inguna Skadina. What Can We Really Learn from Post-editing? In *Proc. of AMTA 2016 vol. 2: MT Users' Track*, pages 86–91, Austin, Texas, November 2016.

Sennrich, Rico, Barry Haddow, and Alexandra Birch. Neural Machine Translation of Rare Words with Subword Units. In *Proc. of the 54th Annual Meeting on Association for Computational Linguistics*, pages 1715—1725, Berlin, Germany, August 2016. Association for Computational Linguistics.

Wäschle, Katharina, Patrick Simianer, Nicola Bertoldi, Stefan Riezler, and Marcello Federico. Generative and Discriminative Methods for Online Adaptation in SMT. In *Proc. of Machine Translation Summit XIV*, pages 11–18, Nice, France, September 2013.

Wuebker, Joern, Spence Green, and John DeNero. Hierarchical Incremental Adaptation for Statistical Machine Translation. In *Proc. of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1059–1065, Lisbon, Portugal, September 2015.

Zeiler, Matthew D. "ADADELTA: An Adaptive Learning Rate Method". *arXiv preprint arXiv:1212.5701*, 2012.

Address for correspondence:
Marco Turchi
turchi@fbk.eu
Via Sommarive 18, Povo, 38123 Trento, Italy