Incremental Adaptation Strategies for Neural Network Language Models

Aram Ter-Sarkisov, Holger Schwenk, Loïc Barrault and Fethi Bougares
School of Computer Science, University of Maine,
Le Mans, France
tersarkisovl@lium.univ-lemans.fr

Abstract

It is today acknowledged that neural network language models outperform back-off language models in applications like speech recognition or statistical machine translation. However, training these models on large amounts of data can take several days. We present efficient techniques to adapt a neural network language model to new data. Instead of training a completely new model or relying on mixture approaches, we propose two new methods: continued training on resampled data or insertion of adaptation layers. We present experimental results in an CAT environment where the post-edits of professional translators are used to improve an SMT system. Both methods are very fast and achieve significant improvements without over-fitting the small adaptation data.

1 Introduction

A language model (LM) plays an important role in many natural language processing applications, namely speech recognition and statistical machine translation (SMT). For a very long time, back-off n-gram models were considered to be the state-of-the-art, in particular when large amounts of training data are available.

An alternative approach is based on the use of high-dimensional embeddings of the words and the idea to perform the probability estimation in this space. By these means, meaningful interpolations can be expected. The projection and probability estimation can be jointly learned by a neural network (Bengio et al., 2003). These models, also called continuous space language models (CSLM), have seen a surge in popularity, and it was confirmed in many studies that they systematically outperform back-off n-gram models by a significant margin in SMT and speech recognition. Many variants of the basic approach were proposed during the last years, e.g. the use of recurrent architectures (Mikolov et al., 2010) or LSTM (Sundermeyer et al., 2012). More recently, neural networks were also used for the translation model in an SMT system (Le et al., 2012; Schwenk, 2012; Cho et al., 2014), and first translations systems entirely based on neural networks were proposed (Sutskever et al., 2014; Bahdanau et al., 2014).

However, to the best of our knowledge, all these systems are static, i.e. they are trained once on a large representative corpus and are not changed or adapted to new data or conditions. The ability to adapt to changing conditions is a very important property of an operational SMT system. The need for adaptation occurs for instance in a system to translate daily news articles in order to account for the changing environment. Another typical application is the integration of an SMT system in an CAT[1] tool: we want to improve the SMT systems with help of user corrections. Finally, one may also want to adapt a generic SMT to a particular genre or topic for which we lack large amounts of specific data. Various adaptation schemes were proposed for *classical SMT systems*, but to the best of our knowledge, there is only very limited works involving neural network models.

We are interested in a setting where an LM needs to be adapted to a small amount of data which is representative of a domain change, so that the overall system will perform better on this domain in the future. Our task, which corresponds to concrete needs in real-world applications, is the translation of a document by an human over several days. The human translator is assisted by an SMT system which proposes translation hypothesis to speed up his work (post editing). After one day of work, we adapt the CSLM to the transla-

[1] Computer Assisted Translation

Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality (CVSC), pages 48–56,
Beijing, China, July 26-31, 2015.

tions already performed by the human translator, and show that the SMT system performs better on the remaining part of the document.

In this paper, we use the open-source MateCat tool[2] and a closely integrated SMT system[3] which is already adapted to the task (translation of legal documents). For each source sentence, the system proposes an eventual match in the translation memory and a translation by the SMT system. The human translator can decide to either post-edit them, or to perform a new translation from scratch. After one day of work, we want to use all the post-edited sentences to adapt the SMT systems, so that the translation quality is improved for the next day. This means that the SMT system will be adapted to the specific translation project. One important particularity of the task is that we have a very small amount of adaptation data, usually around three thousand words per day.

This paper is organized as follows. In the next two sections, we summarize basic notions of statistical machine translation and continuous space language models. We then present our tasks and results. The paper concludes with a discussion and directions of future research.

2 Related work

Popular approaches to adapt the LM in an SMT system are mixture models, *e.g.* (Foster and Kuhn, 2007; Koehn and Schroeder, 2007) and data selection. In the former case, separate LMs are trained on the available corpora and are then merged into one, the interpolation coefficients being estimated to minimize perplexity on an in-domain development corpus. This is known as linear mixture models. We can also integrate the various corpus-specific LMs as separate feature functions in the usual log-linear model of an SMT system.

Data selection aims at extracting the most relevant subset of all the available LM training data. The approach proposed in (Moore and Lewis, 2010) has turned out to be the most effective one in many settings. Adaptation of the LM of an SMT models in an CAT environment was also investigated in several studies, e.g. (Bach et al., 2009; Bertoldi et al., 2012; Cettolo et al., 2014).

Adaptation to new data was also investigated in the neural network community, usually by some type of incremental training on a (subset) of the data. Curriculum learning (Bengio et al., 2009), which aims in presenting the training data in a particular order to improve generalization, could be also used to perform adaptation on some new data. There are a couple of papers which investigate adaptation in the context of a particular application, namely image processing and speech recognition. One could for instance mention a recent work which investigated how to transfer features in convolutional networks (Yosinski et al., 2014), or research to perform speaker adaptation of a phoneme classifier based on TRAPS (Trmal et al., 2010).

There are also a few publications which investigate adaptation of neural network language models, most of them very recent. The insertion of an additional adaption layer to perform speaker adaptation was proposed by Park et al. (Park et al., 2010). Earlier this idea was explored in (Yao et al., 2012) for speech recognition through an affine transform of the output layer. Adaptation through data selection was studied in (Jalalvand, 2013) (selection of sentences in out-of-domain corpora based on similarity between sentences) and (Duh et al., 2013) (training of three models: n-gram, RNN and interpolated LM on two SMT systems: in-domain data only and all-domain). Several variants of curriculum learning are explored by Shi et al. to adapt a recurrent LM to a sub-domain, again in the area of speech recognition (Shia et al., 2014). Finally, one of the early applications of RNN was in (Kombrink et al., 2011): it was used to rescore the n-best list, speed-up the rescoring process, adapt an LM and estimate the influence of history.

3 Statistical Machine Translation

In the statistical approach to machine translation, all models are automatically estimated from examples. Let us assume that we want to translate a sentence in the source language $\mathbf{s}$ to a sentence in the target language $\mathbf{t}$. Then, the fundamental equation of SMT is, applying Bayes rule:

$$\mathbf{t}^* = \arg\max_{\mathbf{t}} P(\mathbf{t}|\mathbf{s}) = \arg\max_{\mathbf{t}} P(\mathbf{s}|\mathbf{t})P(\mathbf{t}) \quad (1)$$

The translation model $P(\mathbf{s}|\mathbf{t})$ is estimated from bitexts, bilingual sentence aligned data, and the language model $P(\mathbf{t})$ from monolingual data in the target language. A popular approach are phrase-based models which translate short sequences of words together (Koehn et al., 2003; Och and

[2]https://www.matecat.com/

[3]http://www.statmt.org/moses/

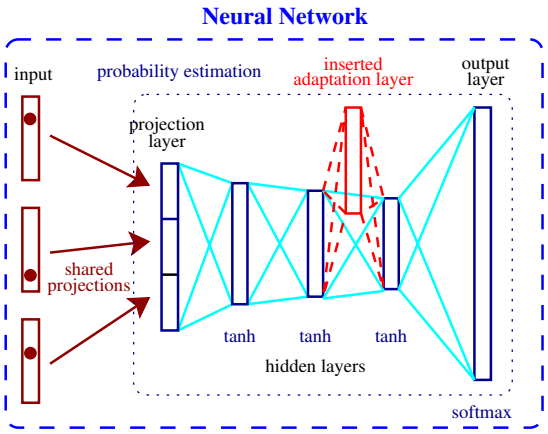

Figure 1: Basic architecture of an CSLM and insertion of an adaptation layer (dashed red).

Ney, 2003). The translation probabilities of these phrase pairs are usually estimated by simple relative frequency. The LM is normally a 4-gram back-off model. The log-linear approach is commonly used to consider more models (Och, 2003), instead of just a translation and language model:

$$\mathbf{t}^* = \arg\max_{\mathbf{t}} \sum_{m=1}^{M} \lambda_m h_m(\mathbf{s}, \mathbf{t}), \qquad (2)$$

where $h_m(\mathbf{s}, \mathbf{t})$ are so-called feature functions. The weights λ_m are optimized during the tuning stage. In the Moses system, fourteen feature functions are usually used.

Automatic evaluation of an SMT system remains an open question and many metrics have been proposed. In this study we use the BLEU score which measures the n-gram precision between the translation and a human reference translation (Papineni et al., 2002). Higher values mean better translation quality.

4 Continuous Space Language Model

The basic architecture of an CSLM is shown in Figure 1. The words are first projected onto a continuous representation, the remaining part of the network estimates the probabilities. Usually one tanh hidden and a softmax output layer are used, but recent studies have shown that deeper architecture perform better (Schwenk et al., 2014). We will use three tanh hidden and a softmax output layer as depicted in Figure 1. This type of architecture is now well known and the reader is referred to the literature for further details, e.g. (Schwenk, 2007).

All our experiments were performed with the open-source CSLM toolkit[4] (Schwenk, 2013), which was extended for our purposes. A major challenge for neural network LMs is how to handle the words at the output layer since a the softmax normalization would be very costly for large vocabularies. Various solutions have been proposed: short-lists (Schwenk, 2007), a class decomposition (Mikolov et al., 2011) or an hierarchical decomposition (Le et al., 2011). In this work, we use short-lists, but our adaptation scheme could be equally applied to the other solutions.

4.1 Adaptation schemes

As mentioned above, the most popular and most successful adaptation schemes for standard back-off LMs are data selection and mixture models. Both could be also applied to CSLMs. In practice, this would mean that we train a completely new CSLM on data selected by the adaptation process, or that we train several CSLMs, e.g. a generic and task-specific one, and combine them in linear or log-linear way. However, full training of an CSLM usually takes a substantial amount of time, often several hours or even days in function of the size of the available training data. Building several CSLMs and combining them would also increase the translation time.

Therefore, we propose and compare CSLM adaptation schemes which are very efficient: they can be performed in a couple of minutes. The underlying idea of both techniques is not to train new models, but to slightly change the existing CSLM in order to account for the new training data. In the first method, we perform **continued training** of the CSLM with a mixture of the new adaptation data and the original training data. In the second method, **adaptation layers are inserted** in the neural network as outlined in red in Figure 1. This additional layer is initialized with the identity matrix and only the weights of this layer are updated. This idea was previously proposed in framework of a speech recognition system (Park et al., 2010). We build on this work and explore different variants of this technique. An interesting alternative is to keep the original architecture of the NN and to only modify one layer, e.g. the weights between two tanh layers in Figure 1. This variant will be explored in future work.

[4]The CSLM toolkit is available at http://www-lium.univ-lemans.fr/~cslm/

Corpus	En/German	En/French
All data:		
Bitexts	129M	512M
Monolingual	643M	1300M
After data selection:		
Bitexts	49M	26M
Monolingual	44M	178M

Table 1: Statistics of the available resources (number of tokenized words)

5 Task and baselines

Our task is to improve an SMT system which is closely integrated into an open-source CAT tool with the post-edits provided by professional human translators. This tool and algorithms to update standard phrase-based SMT systems, including back-off language models, were developed in the framework of the European project MateCat (Cettolo et al., 2014). We consider the translation of legal texts from English into German and French. The available resources for each language pair are summarized in Table 1.

Each SMT system is based on the Moses toolkit (Koehn et al., 2007) and built according to the following procedure: first we perform data selection on the parallel and monolingual corpora in order to extract the data which is the most representative to our development set. In our case, we are interested in the translation of legal documents. Data selection is now a well established method in the SMT community. It is performed for the language and translation model using the methods described in (Moore and Lewis, 2010) and (Axelrod et al., 2011) respectively.

We train a 4-gram back-off LM and a phrase-based system using the standard Moses parameters. The coefficients of the 14 feature functions are optimized by MERT to maximize the BLEU score on the development data. This system is then used to create up to 1000 distinct hypotheses for each source sentence. We then add a 15th feature function corresponding to the log probability generated by CSLM for each hypothesis and the coefficients are again optimized. This is usually referred to as *n-best list rescoring*. We call this final system **domain-adapted** since it is optimized to translate legal documents. This system is then used to assist human translators to translate a large document in the legal domain.

Typically, we will process day by day: after one day of work, all the human translations (created from scratch or by post-editing the hypotheses from the SMT system) are injected into the system and we hope that SMT will perform better on the rest of the document to be translated, e.g. on the second day of work. This procedure can be repeated over several days when the document is rather large (see section 5.2). Usually humans are able to translate approximately 3 000 words per day. We call this procedure **project-adaptation**.

5.1 Results for the English/German system

The 4-gram back-off LM built on the selected data has a perplexity of 151.1 on the domain-specific development data. Given the fact that an CSLM can be very efficiently trained on long context windows, we used a 28-gram in all experiments. By these means we hope to capture the long range dependencies of German. The projection layer of the CSLM was of dimension 320, followed by three tanh hidden layers of size 1024 and a softmax output layer of 32k neurons (short-list). This shortlist accounts for around 92 % of the tokens used in the corpus. The initial learning rate was set to 0.06 and exponentially decreased over the iterations. The network converged after 7 epochs with a perplexity of 96.6, *i.e.* a 36% relative reduction. The total training time is less than 7 hours on a Nvidia K20x GPU. Table 2 (upper part) gives the BLEU score of these baseline domain-adapted systems.

To analyze our project adaptation techniques we have split another legal document into two part, *"Day 1"* and *"Day 2"*. The first part, *"Day 1"*, containing around 3.2K words, is used to adapt the SMT system and the CSLM, aiming to improve the translation performance on the second part, named *"Day 2"*. Note that the performance on *"Day 1"* itself, after adaptation, is of limited interest since we could quite easily overtrain the model on this data. On the other hand, it is informative to monitor the performance on the domain-generic development set. Ideally, we will improve the performance on *"Day 2"*, *i.e.* future text of the same project than the adaptation data, with only a slightly loss on the generic development data.

Various adaptation schemes are compared in Table 4. The network is adapted on the data from *Day 1* and we want to improve performance on *Day 2*. At the same time, we do not want to

LM		BLEU score		
Approach	Adaptation	Dev	Day 1	Day 2
Domain adapted:				
Back-off	n/a	26.18	27.53	19.31
CSLM	n/a	26.89	27.14	20.28
Project adapted;				
Back-off	data selection	25.76	(*28.45*)	20.14
CSLM	none	26.45	(*28.65*)	20.57
	continued training	26.27	(*33.10*)	21.12
	additional layers	26.39	(*31.94*)	**21.26**

Table 2: Comparative BLEU scores for the English/German systems. Italic values in parenthesis are for information only. They are biased since the reference translations are used in training.

Percentage of adaptation data	Generic data (44M words)	Day 1 data (3.2k words)	# examples per epoch	training time per epoch
Domain-adapted CSLM:				
none	19.3M (42%)	n/a	19.3M	3250 sec
Project-adapted CSLM:				
14%	19 356 (0.042%)	3 220	22 576	3.5 sec
25%	9 696 (0.021%)	3 220	12 916	2.0 sec
45%	3 899 (0.008%)	3 220	7 119	1.1 sec
62%	1 967 (0.004%)	3 220	5 187	0.6 sec
77%	1 003 (0.002%)	3 220	4 223	0.5 sec

Table 3: English/German system: number of examples (28-grams) seen by the CSLM at each epoch. For the domain adapted system, we randomly resample about 42% of the examples at each epoch. For the project-adapted system, we experimented with various mixtures between generic and project specific data (Day 1). We don't want to train on Day 1 data only since this would result in strong over-fitting.

overfit the data and keep good performance on the domain-specific Dev set. To achieve this, we continued training of the networks with a mixture of old and new data. All the adaptation data was always used (*Day 1*, 3.2k words) and small fractions of the domain-selected data were randomly sampled at each epoch, so that the adaptation data accounts for 14, 25, 45, 62 and 77 % respectively. Since the networks are trained on very small amounts of data (4 - 23k words), the overall adaptation process takes only a few minutes. The statistics of the data used at each epoch is detailed in Table 3. We will show below that it is important to perform the adaptation of the CSLM with a mixture of generic and adaptation data to prevent overfitting.

We experiment along the following lines:

1. different resampling coefficients of adaptation and generic data according to Table 3.
2. network topologies:
   a) continue training of the original network updating all the weights.
   b) insert one or two hidden layers with 1024 neurons using linear or hyperbolic tangent activation functions respectively. These additional layers are initialized with the identity matrix and only these layers are updated using backpropagation function.

We record the perplexity of the adapted CSLM on *Day 2* ($\sim 11K$ words), which is then used as a guideline for selecting the best networks to integrate into an SMT system (marked with an asterisk in the Table 4). Lowest perplexity was obtained by keeping the baseline network topology (upper part of Table 4) when *Day 1* data constituted 14 % of the incremental training data set: the perplexity on *Day 2* decreases from 126.1 to 94.6, with a minor increase on the Dev set (96.6→98.7). Using larger

Network architecture	Updated layers	Activation function	Addtl. params	Percentage of adapt. data	Perplexity Day 2	Perplexity Dev
Original network architecture:						
1024-1024-1024 without adaptation	-	Tanh	-	-	126.1	96.6
1024-1024-1024 with incremental training	All	Tanh	-	14%	**94.6***	98.7
				25%	103.7	97.3
				45%	102.9	98.9
				62%	102.7	100.2
Insertion of an adaptation layer:						
1024-**1024**-1024-1024	inserted one only	Linear	1M	14%	106.0	97.4
				25%	104.9	99.5
1024-1024-**1024**-1024	inserted one only	Linear	1M	14%	103.8	98.8
				25%	97.9	102.5
1024-1024-1024-**1024**	inserted one only	Linear	1M	14%	101.2	100.8
				25%	102.2	104.1
1024-**1024**-1024-1024	inserted one only	Tanh	1M	14%	105.7	96.8
				25%	104.6	98.9
1024-1024-**1024**-1024	inserted one only	Tanh	1M	14%	103.5	96.4
				25%	102.6	98.4
1024-1024-1024-**1024**	inserted one only	Tanh	1M	14%	101.5	**95.1***
				25%	101.3	97.4

Table 4: Perplexities of CSLMs with one new hidden layer adapted to *Day 1*. Bold values in the architecture column are the new hidden layers. Bold values in the last two columns are the best perplexities for the respective test corpora. Tanh is a shorthand notation for the hyperbolic tangent activation function. Percentage is the proportion of *Day 1* data in the total corpora (see Table 3). All networks have been trained for 50 iterations.

fractions of *Day 1* leads to over-fitting of the network: the perplexity on *Day 2* and the generic Dev set increases.

The lower part of Table 4 summarizes the results when inserting one *adaptation layer*, with a linear or tanh activation function, at three different slots respectively. For each configuration, we explored five different proportions of the baseline corpora and *Day 1* (cf. Table 3), but for clarity, we only report the most interesting results. The overall tendency was that using more than 25% of *Day 1* systematically leads to over-fitting of the network. Several conclusions can be made: a) an tanh adaptation layer outperforms a linear one; b) it is better to insert the adaptation layer at the end of the network; c) updating the weights of the inserted layer only overfits less than incremental training the whole network (comparing the last block in Table 4 with the second block): the perplexity on *Day 1* decreases substantially (126.1→101.5) and we observe a slight improvement on the Dev set (96.6→95.1).

Finally, Table 2 lower part gives the BLEU scores of the project-adapted systems. When no CSLM is used, the BLEU score on Day 2 increases from 19.31 to 20.14 (+0.83). This is achieved by adapting the translation and back-off LM (details of the algorithms can be found in (Cettolo et al., 2014)). Both CSLM adaptation schemes obtained quite similar BLEU scores: 21.12 and 21.26 respectively, the insertion of one additional tanh layer having a slight advantage. Overall, the adapted CSLM yields an improvement of 1.12 BLEU (20.14 to 21.26) while it was about 1 point BLEU for the domain-adapted system (19.31 to 20.28). This nicely shows the effectiveness of our adaptation scheme, which can be applied in a couple of minutes.

5.2 Results for the English/French system

A second set of experiments was performed to confirm the effectiveness of our adaptation proce-

dure on a different language pair: English/French. In the MT community it is well known that the translation into German is a very hard task which is reflected in the low BLEU scores around 20 (see Table 2). On the other hand, our baseline SMT system for the English/French language pair has a BLEU score well above 40. One may argue that it is more complicated to further improve such a system.

In addition, we investigate adaptation of the SMT system and the CSLM over five consecutive days: the human translator works for one day and corrects the SMT hypothesis, these corrections are used to adapt the system for the second day. Human corrections are again inserted into the system and a new system for the third day is built, and so on. With this adaptation scheme we want to verify whether our methods are robust or quickly overfit the adaptation data. The number of words for each day are about three thousand. A 16-gram CSLM for the French target language with a shortlist of 12k was used. Training was performed for 15 epochs.

Day	Day 1	Days 1-2	Days 1-3	Days 1-4
1	39 %	27.9 %	21.6 %	17.7 %
2	-	29.6 %	22.9 %	18.8 %
3	-	-	22.3 %	18.1 %
4	-	-	-	17.4 %

Table 6: English/French task: proportion of each day in the adaptation data set, *e.g.* at the end of Day 2, we create an adaptation corpus which consists of 27.9% and 29.6% of data from Day 1 and Day 2 respectively, the remaining portions are randomly resampled in the training data.

For this task, we only used the incremental learning method (see Table 4) as it yielded the lowest perplexity in the English/German experiment. The data from the five consecutive days is coming from one large document which is assumed to be from one domain only. Therefore, we decided to always use all the available data from the preceding days to adapt our models. For instance, after the third day, the data from Day 1, 2 and 3 is used to build a new system for the fourth day. The proportions of each day in the corpus used to continue the training of the CSLM are given in Table 6 (note that every day's proportion decreases, but their combined share increases from 39% to 68%). The perplexities of the various CSLMs are given in Table 7.

Data	CSLM baseline	CSLM adapted
Day 1	233.9	-
Day 2	175.6	130.3
Day 3	153.0	130.2
Day 4	189.4	169.4
Day 5	189.2	167.7

Table 7: English/French task: perplexities of baseline and adapted CSLM (on all preceding days), *e.g.* the CSLM tested on Day 4 is the baseline CSLM that had been adapted with Days 1-3.

One first observation is the rather high perplexity of the models on each day. This shows the importance of project adaptation even when domain related data is available. Adaptation allows to decrease the perplexity by more than 10% relative for each day. While the perplexities vary between the project days, they are reduced in every case, which demonstrates the effectiveness of the adaptation method.

In order to evaluate the impact of the CSLM adaptation on the SMT system, we performed various translation experiments. The results are provided in Table 5. The BLEU scores of the various systems using the baseline and the adapted CSLMs are presented. We run tests with three different human translators - for the sake of clarity, we provide detailed results for one translator only. The observed tendencies are similar for the two other translators. First of all, one can see that the CSLM improves the BLEU score of the baseline systems between 2.3 to 3.4 BLEU points, *e.g.* for Day 2 from 44.07 to 46.61. Adapting the whole SMT system to the new data improves significantly the translation quality, *e.g.* from 46.61 to 52.01 for Day 2, without changing the CSLM. The proposed adaptation scheme of the CSLM achieves additional important improvements, in average 2.6 BLEU points. This gain is relatively constant for all days.

For comparison, we also give the BLEU scores when using four reference translations: the one of the three human translators and one independent translation which was provided by the European Commission.

We still observe some small gains although three out of four translations were not used in the adaptation process. This shows that our adaptation scheme not only learns the particular style of one

Approach	Day 1	Day 2	Day 3	Day 4	Day 5
Baseline SMT system:					
back-off LM	48.84/63.69	44.07/62.13	46.88/67.14	43.22/64.74	47.77/67.07
CSLM	52.25/67.04	46.61/65.64	49.73/70.70	45.68/68.61	50.06/69.70
Adapted SMT system:					
baseline CSLM	n/a	52.01/66.68	57.35/75.31	54.99/71.88	59.11/74.49
adapted CSLM		54.61/67.97	60.23/75.90	57.19/72.05	61.83/5.21
Improvement obtained by adapted CSLM		2.60/1.29	2.88/0.56	2.20/0.17	2.72/0.72

Table 5: BLEU scores obtained by a baseline SMT (without and with an CSLM) and a project-adapted SMT with baseline (unadapted) CSLM and adapted CSLM. The first value in every cell is the BLEU score obtained with respect to the reference translation of the human translator; the second one is calculated with respect to all the 3 references created by the professional translators (*i.e.* obtained by post-edition) and an independent reference.

translator, but also achieves more `generic` improvements. This also shows that the adaptation process is beneficial for improving state-of-the-art systems which already perform very well on certain tasks.

6 Conclusions

In this paper, we presented a thorough study of different techniques to adapt a continuous space language model to small amounts of new data. In our case, we want to integrate user corrections so that a statistical machine translation system performs better on similar texts. Our task, which corresponds to concrete needs in real-world applications, is the translation of a document by an human over several days. The human translator is assisted by an SMT system which proposes translation hypothesis to speed up his work (post editing). After one day of work, we adapt the CSLM to the translations already performed by the human translator, and show that the SMT system performs better on the remaining part of the document.

We explored two adaptation strategies: continued training of an existing neural network LM, and insertion of an adaptation layer with the weight updates being limited to that layer only. In both cases, the network is trained on a combination of adaptation data (3–15k words) and a portion of similar size, randomly sampled in the original training data. By these means, we avoid overfitting of the neural network to the adaptation data. Overall, the adaptation data is very small – less than 50k words – which leads to very fast training of the neural network language model: a couple of minutes on a standard GPU.

We provided experimental evidence of the effectiveness of our approach on two large SMT tasks: the translation of legal documents from English into German and French respectively. In both cases, significantly improvement of the translation quality was observed.

References

Amittai Axelrod, Xiaodong He, and Jianfeng Gao. 2011. Domain adaptation via pseudo in-domain data selection. In *EMNLP*, pages 355–362.

Nguyen Bach, Roger Hsiao, Matthias Eck, Paisarn Charoenpornsawat, Stephan Vogel, Tanja Schultz, Ian Lane, Alex Waibel, and Alan W. Black. 2009. Incremental Adaptation of Speech-to-Speech Translation. In *NAACL*, pages 149—-152, Boulder, USCO.

D. Bahdanau, K. Cho, and Y. Bengio. 2014. Neural machine translation by jointly learning to align and translate. In *NIPS workshop on Modern Machine Learning and Natural Language Processing*.

Yoshua Bengio, Rejean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *JMLR*, 3(2):1137–1155.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *ICML*.

Nicola Bertoldi, Mauro Cettolo, Marcello Federico, and Christian Buck. 2012. Evaluating the Learning Curve of Domain Adaptive Statistical Machine-Translation Systems. In *Workshop on SMT*, pages 433–441, Montréal, Canada.

Mauro Cettolo, Nicola Bertoldi, Marcello Federico, Holger Schwenk, Loïc Barrault, and Christophe Servan. 2014. Translation project adaptation for MT-

enhanced computer assisted translation. *Machine Translation*, 28(2):127–150.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *EMNLP*, pages 1724–1734.

Kevin Duh, Graham Neubig, Katsuhito Sudoh, and Hajime Tsukada. 2013. Adaptation data selection using neural language models: Experiments in machine translation. In *ACL (2)*, pages 678–683.

George Foster and Roland Kuhn. 2007. Mixture-model adaptation for SMT. In *EMNLP*, pages 128–135.

Shahab Jalalvand. 2013. Improving language model adaptation using automatic data selection and neural network. In *RANLP*, pages 86–92.

Philipp Koehn and Josh Schroeder. 2007. Experiments in domain adaptation for statistical machine translation. In *Second Workshop on SMT*, pages 224–227, June.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based machine translation. In *HLT/NACL*, pages 127–133.

Philipp Koehn, 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. In *ACL, demonstration session*.

Stefan Kombrink, Tomas Mikolov, Martin Karafiát, and Lukás Burget. 2011. Recurrent neural network based language modeling in meeting recognition. In *INTERSPEECH*, pages 2877–2880.

Hai-Son Le, I. Oparin, A. Allauzen, J-L. Gauvain, and F. Yvon. 2011. Structured output layer neural network language model. In *ICASSP*, pages 5524–5527.

Hai-Son Le, Alexandre Allauzen, and François Yvon. 2012. Continuous space translation models with neural networks. In *NAACL*.

Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černocký, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Interspeech*, pages 1045–1048.

Tomáš Mikolov, A. Deoras, D. Povey, L. Burget, and J. Černocký. 2011. Strategies for training large scale neural network language models. In *ASRU*, pages 196–201.

Robert C. Moore and William Lewis. 2010. Intelligent selection of language model training data. In *ACL*, pages 220–224.

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

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *ACL*, pages 160–167.

K. Papineni, S. Roukos, T. Ward, and W.J. Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *ACL*, pages 311–318.

Junho Park, Xunying Liu, Mark J. F. Gales, and Phil C. Woodland. 2010. Improved neural network based language modelling and adaptation. In *Interspeech*, pages 1041–1044.

Holger Schwenk, Fethi Bougares, and Loïc Barrault. 2014. Efficient training strategies for deep neural network language models. In *NIPS workshop on Deep Learning and Representation Learning*.

Holger Schwenk. 2007. Continuous space language models. *Computer Speech and Language*, 21:492–518.

Holger Schwenk. 2012. Continuous space translation models for phrase-based statistical machine translation. In *Coling*, pages 1071–1080.

Holger Schwenk. 2013. CSLM - a modular open-source continuous space language modeling toolkit. In *Interspeech*, pages 1198–1202.

Yangyang Shia, Martha Larsona, and Catholijn M. Jonkera. 2014. Recurrent neural network language model adaptation with curriculum learning. *Computer Speech & Language*, 33(1):136–154.

Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. 2012. LSTM neural networks for language modeling. In *Interspeech*.

I. Sutskever, O. Vinyals, and Q. Le. 2014. Sequence to sequence learning with neural networks. In *NIPS*, pages 3104–3112.

Jan Trmal, Jan Zelinka, and Ludek Müller. 2010. Adaptation of a feedforward articifical neural network using a linear transform. In *Text, Speech and Dialogue*, pages 423–430.

Kaisheng Yao, Dong Yu, Frank Seide, Hang Su, Li Deng, and Yifan Gong. 2012. Adaptation of context-dependent deep neural networks for automatic speech recognition. In *Spoken Language Technology Workshop (SLT), 2012 IEEE*, pages 366–369. IEEE.

Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. 2014. How transferable are features in deep neural networks? In *NIPS*, pages 3320–3328.