FBK's Neural Machine Translation Systems for IWSLT 2016

M. Amin Farajian[1,2], *Rajen Chatterjee*[1,2], *Costanza Conforti*[3], *Shahab Jalalvand*[1,2], *Vevake Balaraman*[2], *Mattia A. Di Gangi*[1,2], *Duygu Ataman*[1,2], *Marco Turchi*[1], *Matteo Negri*[1], *Marcello Federico*[1]

[1]Fondazione Bruno Kessler, Trento, Italy
[2]ICT Doctoral School, University of Trento, Italy
[3]Ludwig-Maximilian University of Munich, Germany

federico@fbk.eu

Abstract

In this paper, we describe FBK's neural machine translation (NMT) systems submitted at the International Workshop on Spoken Language Translation (IWSLT) 2016. The systems are based on the state-of-the-art NMT architecture that is equipped with a bi-directional encoder and an attention mechanism in the decoder. They leverage linguistic information such as lemmas and part-of-speech tags of the source words in the form of additional factors along with the words. We compare performances of word and subword NMT systems along with different optimizers. Further, we explore different ensemble techniques to leverage multiple models within the same and across different networks. Several reranking methods are also explored. Our submissions cover all directions of the MSLT task, as well as *en*-{*de*, *fr*} and {*de*, *fr*}-*en* directions of TED. Compared to previously published best results on the TED 2014 test set, our models achieve comparable results on *en-de* and surpass them on *en-fr* (+2 BLEU) and *fr-en* (+7.7 BLEU) language pairs.

1. Introduction

This paper reports on our first participation at IWSLT with neural machine translation (NMT) systems. This activity was carried out during a Summer project involving both members of FBK's HLT-MT research unit and internship students.[1]

The main goal of the project was to gain knowledge on the Theano library and the sequence-to-sequence translation model by [1], also implemented in the Nematus toolkit [2]. During the project we explored different alternatives related to data selection, pre-/post-processing of the network's input/output data, training modalities of the network and, finally, decoding and re-scoring methods applied during the inference step. In this paper, we summarise our experience by reporting both on approaches that showed to be beneficial to the system performance, as well as on methods that apparently did not work as we expected.

[1]The first two authors contributed equally. The following authors contributed while attending a Summer internship with the HLT-MT research unit of FBK: Costanza Conforti, Ludwig-Maximilian University of Munich (Germany), Mattia di Gangi, University of Palermo (Italy), Vevake Balaraman, University of Trento (Italy).

For IWSLT, we developed systems for the TED and the Microsoft Speech Language (MSLT) tasks of the MT track, in particular for the English-{German, French} and {German, French}-English directions. For each language pair we first developed generic systems that we trained on a mix of in-domain and out-domain data. Successively, generic systems were adapted to each task by using in-domain training data. Out-domain training data was actually selected in order to improve training effectiveness both in terms of translation quality and time. We employed word and factored models (lemma, POS) to represent the source text. We also trained a NMT network generating translations from right to left that we used to re-score the conventional left-to-right model. We optimized our networks with the Adagrad optimizer [3] as this provided faster convergence compared to Adadelta in our preliminary experiments (see § 5). Finally, we applied ensemble decoding both intra- and inter-network.

This paper is arranged as follows. We first describe in general the followed NMT approach. Then, we summarize the data selection and pre-processing steps applied on the task data, as well as the overall experimental set-up for our NMT systems. Finally, we present the experimental results of the submitted systems, followed by a section surveying the approaches we tried but that did not show to improve systems' performance.

2. Neural Machine Translation

We develop our NMT systems around the sequence-to-sequence encoder-decoder architecture proposed in [1] and further developed by [4, 5]. The architecture is summarized in the time-unfolded representation shown in Figure 1. In the left-bottom rectangle we see the encoder network, which reads one input word at a time (we omitted the one-hot vector representation for the sake of space), encodes it into an embedding vector and further encodes it together with its left (right) context into an forward (backward) hidden state. Once the full input has been read, the decoder network (on the right side, in lighter gray) starts generating the translation. The first word is generated by initializing the output sequence with the start symbol <s>, which is mapped to an embedding, and further encoded into a (recursive) hid-

den state, which also depends on a linear combination of the bidirectional hidden states of the encoder. The weights of this combination are computed by another network, called attention model, which is not shown in the picture.

The hidden state is mapped into a probability distribution over the output vocabulary, via a scoring matrix and a softmax operator. The most probable word (`il`) is selected as the first output and is used as input for the next inference step. The process continues until the sentence boundary symbol (`<s>`) is emitted. In fact, the real picture is slightly more complicated than the one shown in Figure 1: a beam of k top probable words is picked and used to infer the top k words of the next output word.

In the following we provide a mathematical definition of the computations carried out by the network. Neural machine translation aims to optimize the parameters of the model to maximize the likelihood of the training data. The ultimate goal is to estimate a conditional probability model $p_\Theta(y|x)$, where Θ is the parameter set of the model (the weights and biases of the network), y is a target sentence and x is a source sentence. Thus, the objective function is:

$$\underset{\Theta}{argmax}\frac{1}{N}\sum_{n=1}^{N} log(p_\Theta(y_n|x_n)); \tag{1}$$

where N is the total number of sentence pairs in the training corpus. The conditional probability is computed as:

$$p_\Theta(y|x) = \prod_{t=1}^{T_y} p_\Theta(y_t|y_{<t}, x) \tag{2}$$

where T_y is the number of words in the target sentence. The probability of target word y_t, given all the previous target words $y_{<t}$ and the source x, is modelled by the decoder network as follows:

$$p_\Theta(y_t|y_{<t}, x) = g(\dot{y}_{t-1}, s_t, c_t) \tag{3}$$

where $\dot{y}_{t-1}$ is the word embedding of the previous target word, s_t is the hidden state of the decoder, and c_t the source context vector (encoding of the source sentence x) at time t. The decoder state s_t is computed by a gated recurrent unit (GRU) [6] in two steps: first, *i)* the previous hidden state and the previous target word embedding are used to compute an intermediate hidden state by a GRU unit:

$$s'_t = f'(s_{t-1}, \dot{y}_{t-1}) \tag{4}$$

then, *ii)* the intermediate hidden state and the source context vector are passed to another GRU to compute the final hidden state of the decoder. In short:

$$s_t = f(s'_{t-1}, c_t) \tag{5}$$

The source context vector is a weighted sum of all the hidden states of a bi-directional encoder [1].[2]

$$c_t = \sum_{j=1}^{T_x} a_{tj} h_j \tag{6}$$

[2]In rest of the paper, by encoder we mean bi-directional encoder

where a_{tj} is the attention weight given to the j-th encoder hidden state at decoding time t, and T_x is the number of words in the source sentence. The attention weight represents the importance of the j-th hidden state of the encoder in generating the target word of time t. It is drawn from a probability distribution over all the hidden states of the encoder, which is computed by applying a $softmax$ operator over all the scores of the hidden units of the encoder.

$$a_{tj} = \frac{exp(e_{tj})}{\sum_{k=1}^{T_x} exp(e_{tk})} \tag{7}$$

where e_{tj} and e_{tk} are the score of the j-th and k-th hidden units of the encoder at time step t, which is a function of the intermediate hidden state of the decoder (as mentioned in Equation 4) and the hidden state of the encoder, as shown below:

$$e_{tj} = a(s'_{t-1}, h_j) \tag{8}$$

The hidden state h_j of the j-th source word is a concatenation of the hidden states of the forward and backward encoders:

$$h_j = [\overrightarrow{h_j}; \overleftarrow{h_j}] \tag{9}$$

where $\overrightarrow{h_j}$ and $\overleftarrow{h_j}$ are the hidden state of the forward and backward encoders, respectively. These hidden states are computed by the GRU unit that takes previous/next hidden state and the word embedding of the j-th source word.

$$\overrightarrow{h}_j = f(\dot{x}_j, \overrightarrow{h}_{j-1}) \tag{10}$$

$$\overleftarrow{h}_j = f(\dot{x}_j, \overleftarrow{h}_{j+1}) \tag{11}$$

3. Data Selection and Pre-processing

It has been shown that deep neural networks can take advantage of large training corpora [4, 7]. However, finding a large amount of in-domain training data is very challenging, if not impossible, while it is much easier to find large generic parallel corpora. In practice, a NMT system can be first trained on a large generic corpus and then adapted on the available in-domain/task-specific data. We applied this two-stage training procedure for all our submissions.

In order to build a generic system, we *pooled* all the permissible data sets, including the in-domain data. We pre-processed this data to normalize punctuations, remove special characters, tokenize, truecase, remove empty lines as well as sentences with lengths greater than 80 and also the ones with length ratio greater than (1:9). The pooled data was further filtered by removing sentence pairs for which the out-of-vocabulary word rate of either the source or the target sentence was higher than a given threshold (10% for English-German, and 5% for English-French). The reference vocabularies used for this step will be explained later in this section. This resulted in generic domain corpora of size 14M and 54M sentence pairs for the English-German and English-French language pairs, respectively. Unfortunately, training

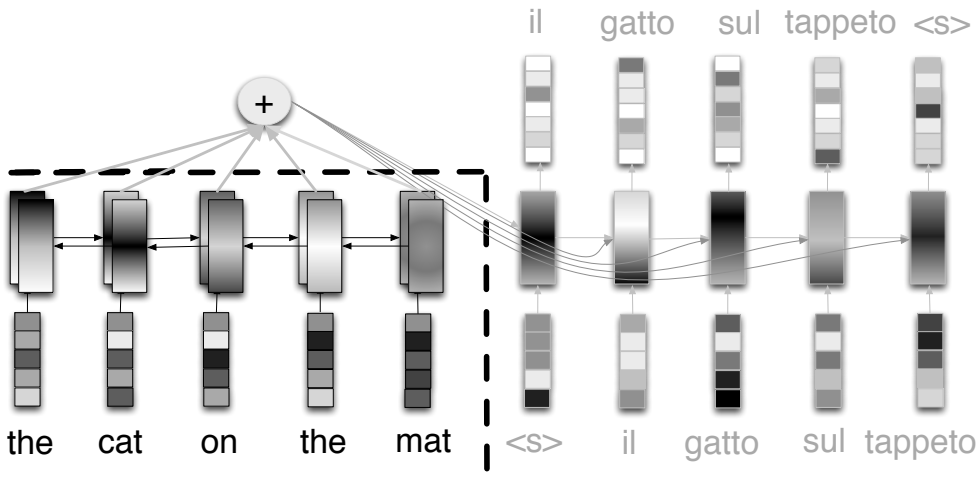

Figure 1: Time unfolded representation of the neural MT encoder-decoder architecture with bi-directional hidden states and attention model.

English-French NMT systems with such a large data set was not considered feasible given our time constraints. Hence, a further data selection step was applied for this language pair, in which 15.5M sentence pairs were selected using the bilingual cross-entropy approach[3] [8]. The statistics of the generic corpora are presented in Table 1.

While for the TED task both training and development data was supplied to the participants, for the MSLT task only development data was provided. Hence, we decided to use data from the OpenSubtitles corpus[4], a large collection of translated movie subtitles, as in-domain training data for the MSLT task. From this corpus we removed sentences longer than 20 words as well as sentence pairs with length ratio larger than 1:1.2. These choices are motivated by the statistics of the development corpora, which contain short sentences with average length of $\sim$10 words and source/target length ratio of $\sim$1.2.

The development sets of the TED task consist of the previously released development and test corpora, except for `tst2014` which the organisers reserved for progress testing purposes. For the MSLT task, a subset of 1,000 sentences was randomly selected from the released development corpora to decide about the early stopping, and the rest was used to compare the performance of the systems for each language direction [9].

Vocabularies for each direction were created by using both in-domain and out-domain corpora. First, the vocabularies were initialized with all the words occurring more than three times in the in-domain corpora, resulting in vocabularies of size 26k, 32k, and 30k for English, German, and French languages, respectively. Then, the most frequent new words found in the out-domain corpora were added, until the vocabulary reached the desired size.

[3] `https://github.com/rousseau-lium/XenC`

[4] `http://opus.lingfil.uu.se/OpenSubtitles2016.php`

		Segments	Tokens	Types
English-German	En	14.1M	175.9M	527K
	De		166.1M	1,027K
English-French	En	15.5M	233.3M	352K
	Fr		247.4M	347K

Table 1: Statistics of the parallel corpora used to train generic NMT systems.

4. Experimental Setup

All the experiments reported in this paper were conducted with the Nematus toolkit[5], which is an enhanced version of the code provided in the DL4MT tutorial.[6] It is based on the Theano deep learning framework [10], and contains all the necessary scripts to train and test NMT systems. We used the modified version of the byte pair encoding (BPE) compression algorithm [11] that works at the character level,[7] to perform experiments with subwords as proposed in [5]. This algorithm iteratively merges the most frequent pair of symbols (in this case character or sequence of characters) into a single symbol. Thus, the most frequent words in the corpus remain intact whereas the rare words are segmented into subunits. Our preliminary experiments on words and subwords were performed on *pool-balanced* data, which contain out-domain data and multiple copies of the in-domain data, so as to reach a size comparable to that of the out-domain data. The reason behind this choice is twofold: *i)* to avoid the network being biased to the large out-domain corpus, and *ii)* to obtain more realistic learning curves in shorter time, compared to the case of training a generic system for several weeks and then adapting it to the task. These experiments were performed on *en-de* direction with both $Adadelta$ and

[5] `https://github.com/rsennrich/nematus`

[6] `https://github.com/nyu-dl/dl4mt-tutorial`

[7] `https://github.com/rsennrich/subword-nmt`

Adagrad optimizers with learning rates set to 0.01, to investigate which combination gives the best performance. Since we participated in two tasks, it was not feasible to train systems with pool-balanced data for each of them separately. Rather, we decided to train a generic system for each direction that was later adapted to a task using in-domain data.

Domain adaptation to the TED task was performed from two different checkpoints of the generic systems (using the best model of the second and third epoch), whereas for MSLT only the single best model was used. For the TED task the batch size was set to 100 with max sentence length to 75, whereas for MSLT the max sentence length was set to 20 and the batch size to 600. In addition to word level systems, we built factored NMT systems as proposed in [2] to *explicitly* provide linguistic knowledge to the network. The factors (lemmas and part-of-speech tags) were generated by tagging the source corpora with Treetagger [12].[8] The common experimental configurations are provided in Table 2.

The models of the general and in-domain systems were saved and evaluated at intervals of 10,000 and 2,000 updates, respectively. Evaluation was performed over the dev sets with the BLEU score [13]. The best model of each configuration was further evaluated on tst2014 using the IWSLT evaluation server,[9] and compared with the best results of the previous years.

Type	Value
source vocab size	40,000
target vocab size	40,000
word embedding	1024
hidden units	1024
learning rate	0.01
optimizer	Adagrad
shuffle	True
source dropout	0.1
target dropout	0.1
hidden dropout	0.2
embedding dropout	0.2

Table 2: Training configuration used for all our systems.

5. Experiments and Results

In this section, we first discuss the preliminary experiments that were performed in order to compare *i)* subword versus word-based NMT, and *ii)* Adagrad versus Adadelta optimization. We then discuss further experiments that were performed using different input/output configurations.

[8] http://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/

[9] http://hlt-services5.fbk.eu/iwslt/Eval.html

5.1. Words versus sub-words

One of the most challenging problems in NMT is to deal with large source and target vocabularies. In practice, the vocabulary size in NMT is often set to 30K-50K words to bound training time and memory consumption. This problem has been addressed in several recent works, in which different word segmentation methods were explored, at character level [14], at subword level [5], and at hybrid levels [15]. These solutions are especially useful when working with broad domains, such as news, where the vocabulary growth is faster and the coverage given by a 50K vocabulary is rather low. Actually, for TED and MSLT tasks we found that the top most frequent 40,000 words provide a coverage of 96% - 99% on the training data for the two considered translation directions. This raises the question of whether segmenting words into sub-word units is worth or not. To answer this question, we performed preliminary experiments and compared word-level and sub-word-level NMT systems, using Adadelta and Adagrad optimization techniques. In the experiments with sub-words, 40K merge rules were learned and applied on the training data. All the experiments were run with pool-balanced data and evaluated every 10K updates up to one training epoch ($\sim$140K updates, which took $\sim$4 days on a Tesla K80 GPU).[10] Figure 2 shows the learning curves of these experiments.

We noticed that the word level NMT systems performed better than the sub-word ones, with both optimizers. It is possible that, in long run after many epochs, both word and subword models reach similar performance but based on these observations (on 1 epoch) we decided to go for the word models for all the following experiments. It is also evident from Figure 2 that Adagrad outperforms Adadelta, both in the word and sub-word experiments, showing also faster convergence. This motivates our choice of always applying this method henceforth.

5.2. Word and factored models

Previous work has shown that enhancing a NMT system with linguistic features can improve its performance [2]. So, we trained several factored NMT systems along with one word-level system for each translation direction, as detailed below:

- **Word:** The input consist of only sequence of words, and the distributed representation of the word in this category has the highest degree of freedom to leverage all the dimensions (1024) to encode the word in the vector space.
- **Factor-1:** Each word of the input sequence is annotated by its lemma and POS tag. The word embedding (1024 dimensions) is *exclusively* shared by all the factors, where word, lemma, and POS occupy 904, 110, and 10 dimensions, respectively.

[10] Experiment on sub-word units with Adagrad crashed after 80k updates so we can report only scores up to 80k updates

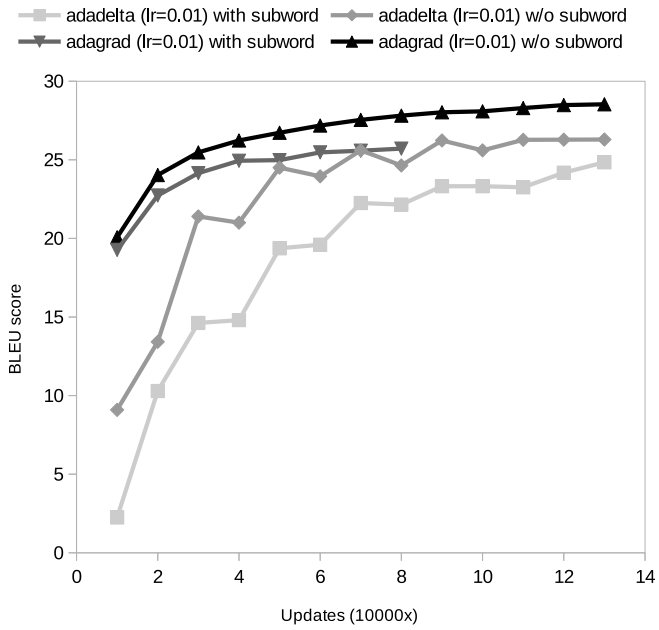

Figure 2: Learning curve of word and sub-word NMT systems (en-de) when trained with Adadelta and Adagrad optimizers

- **Factor-2:** In Factor-1, lemma uses a small portion ($\sim$11%) of the embedding vector, which gives a low importance to this feature. However, lemmas can be very useful for translating rare words or even unknown words (if their lemma is known), since they generally are more frequent than the words. So, to give more emphasis to lemmas, we alter the dimensions to 599, 415, and 10 for word, lemma, and POS factors, respectively. We increased the dimension of lemma by borrowing it from words because we wanted to study the impact of having different dimensions for word and lemma when the total embedding dimension is fixed to 1024. It might be better to have additional components for the lemma instead of borrowing them from the word but this increases the total number of parameters of the network, which eventually increases the training time and memory requirements.

- **Factor-2-rev:** All the above models generate the translation hypothesis in left-to-right (L2R) order. To rerank the n-best list obtained from these models, a reversed model is trained following the approach proposed in [7], which produces the translation output in reverse order (R2L). In this experiment, we used the same configuration as Factor-2, because in our preliminary experiments we observed this setting had slightly better performance than Factor-1 for the *en-de* direction.

Results of the experiments with different word/factor representations are reported in Table 3. To have a fair comparison, we evaluated on development set of the TED task the first 10 models, that were saved every 2,000 updates in the adaptation stage. We observe that Factor-1 consistently performs better than other representations across all the language directions. Increasing the embedding size of lemma at the cost of word embedding (in Factor-2) did not yield improvement over Factor-1. However, we observed that when training the generic system for *en-de*, Factor-2 had slightly better performance than Factor-1, and that was the motivation of training a reverse model with Factor-2. The reverse model performs slightly better than the word model when translating into English. It might be possible that generating text in reverse order is more difficult for German and French. As a

	en-de	de-en	en-fr	fr-en
Word	29.43	34.89	40.82	39.84
Factor-1	**29.51**	**35.49**	**41.33**	**40.85**
Factor-2	29.47	35.00	40.74	40.46
Factor-2-rev	28.09	35.19	39.98	40.19

Table 3: Performance on development set for TED task

term of comparison with previously published best results, we selected the best model across all the representations and evaluated them on test 2014 (results reported in Table 4). Our single best model significantly outperforms previously published best results on *en-fr* and *fr-en* directions, however, for the *en-de* and *de-en* directions our model still remains below state-of-the-art results.

5.3. Ensemble decoding

As shown in the literature, ensemble decoding typically improves performance over the single models [4, 7, 15]. In this work, we investigated different ensemble decoding techniques:

- **Intra-network:** uses multiple models of the same network. To evaluate this technique, we performed two independent runs using the last 4 models, as well as the best 4 models of the same network. Both runs showed marginal improvements of 0.1 - 0.3 BLEU score over the single models. This can be due to the fact that models of the same network are less diverse, and do not provide any additional information to the decoder.

- **Inter-network:** uses models from different networks. Unlike intra-network decoding, the models used in this approach are selected from different networks, which leads in having more diverse models. In our experiments, we observed 0.5 - 1.0 BLEU score improvement by selecting the best model from each network (that is ensemble of 3 models). However, using multiple models from each network (4 from each, so a total of 12 models) did not yield further improvements. This technique is also evaluated on test 2014 of TED

task, in which 2 best models of each network (one from each checkpoint) are used in ensemble. As expected, this technique outperforms the intra-network decoding with improvements of 0.5 - 1.0 BLEU score over the single models, across different language directions. We observe that using this technique helps to reach the performance of the best system of 2015 in *en-de* direction, which used an ensemble of 8 models trained with different attention mechanisms [4].

5.4. Re-ranking with Right-To-Left model

Sennrich et al. report significant improvements by re-scoring the n-best hypotheses of L2R models using R2L models, followed by re-ranking [7]. Following their approach, we trained a reverse model for each direction and used it to re-score the n-best hypotheses of the ensemble models. The re-scored list was then re-ranked by linear interpolation of the model scores (equally weighing both L2R and R2L models) to select the best hypothesis. Due to time constraints, the R2L models were trained for 2 epochs on the pool data and later adapted to the TED task only. Although, the performance of R2L models is lower than their L2R counterparts, they showed consistent improvements across all language directions when used in re-ranking, as reported in Table 4.

	en-de	de-en	en-fr	fr-en
Previous best	**27.58**	**26.18**	36.99	32.92
Single best	26.44	23.02	37.78	39.77
Ensemble	27.17	23.55	38.74	40.44
Re-rank	27.32	23.75	**39.04**	**40.64**

Table 4: Performance on IWSLT 2014 TED test set

5.5. Official submissions

The official submissions of each task are described below, and the results are reported in Table 5:

- **TED:** As reported in § 5.4, the best system is the one that uses re-ranking on the n-best list of ensemble models, so we use it as our primary submission. To summarize, our primary submission for each language direction consists of an ensemble of 6 L2R models (2 best from each Word, Factor-1, and Factor-2) that is re-scored by the best R2L model and then re-ranked by equally weighing all the model scores. Contrastive submissions use only ensemble of L2R models without any re-scoring/re-ranking. As expected, the primary submission always outperformed the contrastive one as reported in the official results.

- **MSLT:** Our primary submissions in this task are ensemble of 4 models (2 best from each Word and Factor-1 networks) with equal weights to all the models. Due to time constraint we were not able to perform adaptation on Factor-2 and R2L models, so we discard the re-ranking step for this task.

	TED				MSLT
	2015		2016		2016
	P	C	P	C	P
en-de	30.05	29.70	26.56	26.27	38.78
de-en	32.38	32.11	30.30	29.84	35.06
en-fr	39.71	39.49	36.77	36.75	42.98
fr-en	38.44	38.33	37.19	36.75	-

Table 5: Official results of our primary (P) and contrastive (C) submissions.

6. What did not work

In this section we report experimental results of methods that were not considered for the final submissions. We believe that also negative results can be in fact useful to share, especially when documenting the efforts done to build a competitive system for an evaluation campaign.

6.1. Re-scoring and re-ranking with large language models (LM)

In § 5.4, we showed that n-best list re-scoring using R2L models helps to improve the performance of the systems. These models require parallel data for training and thus can not leverage the large amount of monolingual data available. To overcome this limitation, we investigated the use of large language models (for LM re-scoring) and also machine-learned ranking techniques that can benefit from the additional monolingual data.

LM re-scoring is performed using two language models: *i)* a 4-gram statistical LM trained with KenLM toolkit [16] on a monolingual corpus containing ∼3B words selected from news commentary corpus, and *ii)* a neural LM trained with faster-rnnlm toolkit[11] on the target side of the pool data (see Table 1 for statistics). Re-ranking is performed with TranscRater toolkit[12] [17] using the features extracted from NMT decoder along with the LM scores. These features include the posterior score, mean/min/max/std of word probabilities and alignment probabilities. These features are used along with the LM scores to build a feature vector for each hypothesis in the n-best list. Ranking machines are trained to learn the rank of different hypotheses corresponding to each source sentence using a pairwise comparison techniques. Note that the ranks are derived from the sentence-level translation error rate (TER) scores.

Table 6 shows the results of re-scoring and re-ranking approaches on two directions: *en-de* and *de-en*. The preliminary results on the development set show that R2L+LM re-scoring improves the performance up to 0.8% BLEU score

[11] https://github.com/yandex/faster-rnnlm
[12] https://github.com/hlt-mt/TranscRater

rescoring/reranking method	en-de		de-en	
	dev	test2014	dev	test2014
baseline	30.51	27.17	36.36	23.55
R2L rescoring	30.64	27.33	36.75	23.74
LM rescoring	30.71	27.52	36.90	23.42
R2L+LM rescoring	**30.81**	**27.54**	**37.14**	**23.78**
TranscRater	–	27.22	–	23.73

Table 6: The results of re-scoring and re-ranking approaches on 12-best lists.

over the baseline, and up to 0.4% over R2L re-scoring. On the test set, however, the improvement of R2L+LM re-scoring was marginal in both directions (*en-de* and *de-en*). By applying TranscRater as the re-ranking strategy, we first observed slight improvements on the tokenized output, however after normalization and detokenization, we observed that the BLEU scores reduce. We are still investigating what is the reason behind this inconsistency.

Finally, since the achievable improvements using LM, R2L+LM and TranscRater were not consistent, we decided not to use these approaches for our final submissions.

6.2. Using back-translation for model adaptation

Back-translation is an approach to generate new training instances for MT, where a monolingual corpus usually in the target language [18, 19], is translated to the source language and added to the parallel corpora. Different than [18, 19], however, in this study, we investigated ways of adapting the translation model by adding artificially-created in-domain data. The TED track is an ideal opportunity to evaluate this approach as the task data contain translations of many talks into both German and French.

In order to evaluate our approach, we trained NMT systems in the *en-de* direction using out-domain parallel training corpora and after 3 updates started adapting these systems using different in-domain data. In *tuning 1*, we adapted the system only on the original in-domain parallel corpus. In *tuning 2*, we also added parallel training instances generated using the following two steps: *i)* the target sides of the English-German and English-French in-domain corpora are translated into English; *ii)* once obtained the translation, the the target sides is selected from the original German corpus. This approach can be seen as a way of generating paraphrases to the source sentences in the test domain. Some example outputs are given below.

Reference: *Marcel Proust said, "The true voyage of discovery is not so much in seeking new landscapes as in having new eyes."*

Back-Translation (from DE): *Marcel Proust once said that, "The real voyage of discovery consists not in that you're looking for new countries, but that you have new eyes."*

Back-Translation (from FR): *Well, Marcel Proust has this saying, "The real voyage of discovery consists not looking for new landscape, but to have new eyes."*

We used a phrase-based system for building MT systems to generate the back-translations through cross-validation. As shown in Figure 3, our preliminary experiments did not yield significant differences in the two fine-tuning strategies. This might be due to the different interpretation of the back-translated data compared to [18, 19], in terms of *i)* the side of enriched representation (source instead of target) and *ii)* the characteristics of new data (only synthetic instead of natural and synthetic corpora).

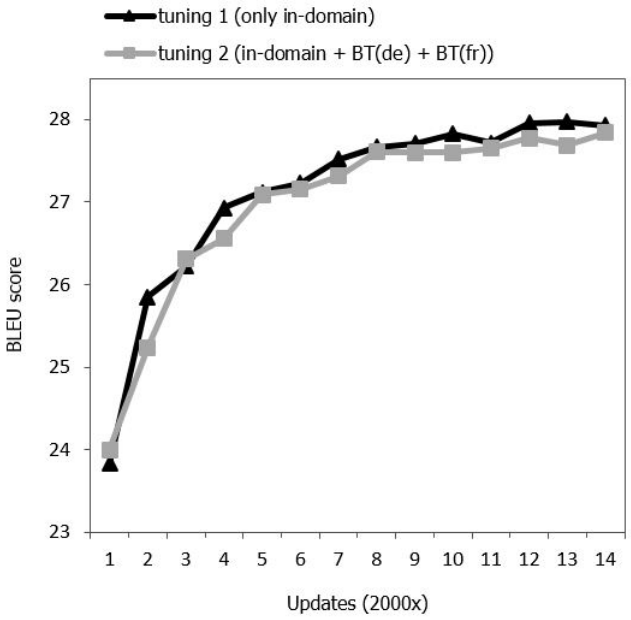

Figure 3: The results of using back-translation for model adaptation on the IWSLT 2014 TED test set. (BT: Back-translated data)

7. Conclusions

In this paper, we described FBK's NMT systems submitted at IWSLT 2016. Our submissions comprised of 4 language directions (en-de, de-en, en-fr, and fr-en) for TED and 3 directions (en-de, de-en, and en-fr) for MSLT tasks. All the systems were trained on a pool data and then adapted to each task using in-domain data. From our preliminary experiments on TED task for en-de direction, we learned that word based NMT system had better performance compared to subwords. This possibly indicates that subwords might be more suitable for developing general domain translation systems, where the out-of-vocabulary rate is higher and the coverage of the training data is lower compared to narrow domains such as TED and MSLT. Also, we learned that optimizing the network parameters using Adagrad leads to faster convergence with higher performance compared to Adadelta, indicating it to be a better option to train NMT systems. The

performance of the word-based models further improved by the addition of linguistic knowledge in the form of factors. Moreover, performing inter-network decoding with word and factored models resulted in higher performance compared to both single model as well as ensemble of multiple models of the same network (intra-network decoding). This indicates that the use of diverse models in decoding helps to leverage the strengths of each single model while mitigating their limitations. All of our submissions are hence based on inter-network decoding, with an additional step of re-ranking in the primary submissions to the TED task.

8. Acknowledgements

This work has been partially supported by the EC-funded project ModernMT (H2020 grant agreement no. 645487).

9. References

[1] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[2] R. Sennrich and B. Haddow, "Linguistic Input Features Improve Neural Machine Translation," in *Proceedings of the First Conference on Machine Translation*. Berlin, Germany: Association for Computational Linguistics, August 2016, pp. 83–91.

[3] J. Duchi, E. Hazan, and Y. Singer, "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization," *Journal of Machine Learning Research*, 2011.

[4] M.-T. Luong and C. D. Manning, "Stanford Neural Machine Translation Systems for Spoken Language Domains," in *Proceedings of the International Workshop on Spoken Language Translation*, 2015.

[5] R. Sennrich, B. Haddow, and A. Birch, "Neural Machine Translation of Rare Words with Subword Units," in *Proceedings of the 54th Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, 2016.

[6] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[7] R. Sennrich, B. Haddow, and A. Birch, "Edinburgh Neural Machine Translation Systems for WMT 16," in *Proceedings of the First Conference on Machine Translation*. Berlin, Germany: Association for Computational Linguistics, August 2016, pp. 371–376.

[8] A. Rousseau, "XenC: An Open-Source Tool for Data Selection in Natural Language Processing," *The Prague Bulletin of Mathematical Linguistics*, no. 100, pp. 73–82, 2013.

[9] C. Federmann and W. D. Lewis, "Microsoft speech language translation (mslt) corpus: The iwslt 2016 release for english, french and german," 2016.

[10] F. Bastien, P. Lamblin, R. Pascanu, J. Bergstra, I. J. Goodfellow, A. Bergeron, N. Bouchard, and Y. Bengio, "Theano: new features and speed improvements," Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop, 2012.

[11] P. Gage, "A New Algorithm for Data Compression," *C Users J.*, vol. 12, no. 2, pp. 23–38, Feb. 1994. [Online]. Available: http://dl.acm.org/citation.cfm?id=177910.177914

[12] H. Schmid, "Probabilistic part-of-speech tagging using decision trees," in *New methods in language processing*. Routledge, 2013, p. 154.

[13] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "BLEU: a method for automatic evaluation of machine translation," in *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, 2002, pp. 311–318.

[14] M. R. Costa-Jussà and J. A. Fonollosa, "Character-based Neural Machine Translation," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, 2016.

[15] M.-T. Luong and C. D. Manning, "Achieving open vocabulary neural machine translation with hybrid word-character models," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, 2016.

[16] K. Heafield, "KenLM: Faster and smaller language model queries," in *Proceedings of the Sixth Workshop on Statistical Machine Translation*. Association for Computational Linguistics, 2011, pp. 187–197.

[17] S. Jalalvand, M. Negri, M. Turchi, J. G. de Souza, D. Falavigna, and M. R. Qwaider, "TranscRater: a Tool for Automatic Speech Recognition Quality Estimation," *ACL 2016*, p. 43, 2016.

[18] N. Bertoldi and M. Federico, "Domain adaptation for statistical machine translation with monolingual resources," in *Proceedings of the Fourth Workshop on Statistical Machine Translation*. ACL, 2009, pp. 182–189.

[19] R. Sennrich, B. Haddow, and A. Birch, "Improving neural machine translation models with monolingual data," in *Proceedings of the The 54th Annual Meeting of the Association for Computational Linguistics*. Berlin, Germany: Association for Computational Linguistics, August 2016, pp. 86–96.