UFAL Submissions to the IWSLT 2016 MT Track

Ondřej Bojar, Ondřej Cífka, Jindřich Helcl,
Tom Kocmi, and Roman Sudarikov

Charles University, Faculty of Mathematics and Physics
Institute of Formal and Applied Linguistics
surname@ufal.mff.cuni.cz

Abstract

We present our submissions to the IWSLT 2016 machine translation task, as our first attempt to translate subtitles and one of our early experiments with neural machine translation (NMT). We focus primarily on English→Czech translation direction but perform also basic adaptation experiments for NMT with German and also the reverse direction. Three MT systems are tested: (1) our Chimera, a tight combination of phrase-based MT and deep linguistic processing, (2) Neural Monkey, our implementation of a NMT system in TensorFlow and (3) Nematus, an established NMT system.

1. Introduction

In the machine translation task (MT task) at IWSLT 2016, the goal is to translate a given set of sentences using (a subset of) permitted training data and any MT system. IWSLT has covered a range of languages and recently started to include Czech in one of the tested domains, namely the "Talk" domain in 2016.

We take this opportunity to evaluate our system Chimera [1] on subtitles, and to compare it with two neural MT systems: Neural Monkey [2] and the well-known Nematus [3].

The paper is structured as follows. In Section 2, we describe our selection of the training data and note that unfortunately, the permitted resources for IWSLT this year *included an overlap* with the test sets. The subsequent sections are devoted to the three system setups targetting Czech: Chimera in Section 3, Neural Monkey in Section 4 and Nematus in Section 5. The results are presented and discussed in Section 6. Since NMT is computationally expensive and requires special hardware (large-memory GPUs), we also used Amazon EC2 resources, see Section 7 for some technical details.

2. Training and Test Data

The organizers of the MT track provided participants with cleaned development and test sets, and also with a relatively small in-domain training set of sentences. Additionally, a long list of permitted resources was given on the web page. To make a better use of the large set of permitted data, we selected sentences similar to the official in-domain training set, as described in Section 2.1.

We spotted an overlap between CzEng 1.6, one of the permitted corpora, and the official development and test sets. In order to avoid overfitting (tuning phrase-based MT on part of its training data is likely to arrive at too strong preference for long phrases), we created our own version of the development set, as detailed in Section 2.2.

We leave up to the organizers of the shared task to select the best way to handle the overlap with the official test set. As described below, the overlap is not always at the level of sentences, so it is rather difficult to identify which sentences from the test set should be disregarded (and if a reasonably big set remains). Admittedly, with the current size of available training data for some language pairs, it is genuinely possible that the test material will be partly covered. The only way of reducing such risk is to create the test set from scratch every year (as e.g. WMT does, see [4]), but one should be aware that any text that appears on-line can easily become a part of at least language model training data, so even the practice of WMT to translate some news texts is not totally safe if the participants are targetting the language.

2.1. Training Data

To select sentences similar to the official in-domain training data as provided by the organizers, we pre-processed all other data sources in the following way:

- Tokenize both the source and target side using the Moses [5] tokenizer.
- Concatenate the resulting corpora and sort them using the XenC [6] tool. XenC was used in two modes: simple source language perplexity filtering, based on perplexity scores given by an in-domain language model, and bilingual cross-entropy difference filtering. For both modes, domain training data was used to create respective language models.
- Take the top 2–5% of the sentences as sorted by each mode from the previous step and remove duplicates. For English→Czech this gives us 650,836 sentences (TOP2 in Table 1) or 5,832,340 sentences (TOP5). For German-English, we used the same procedure, taking the top 1,726,548 most similar to the provided in-

	Sentences	en Tokens	cs Tokens
TRAIN •	122,382	1,999,190	1,661,449
CZENGPRE	51,424,584	559,640,512	472,769,122
TOP2	650,836	3,541,384	3,792,587
TOP5	5,832,340	62,275,138	52,312,292
SYNTDEV+TEST	7,402	130,717	78,427
DTEST	4,774	78,427	65,734
DTESTDEDUP	3,110	59,525	49,864
ETEST	1,193	20,059	16,826
ETESTDEDUP	1,555	28,989	24,191
Official Test Set •			
IWSLT16.TED.tst2015	1,080	17,861	—
IWSLT16.TED.tst2016	1,133	19,896	—
IWSLT16.QED.tst2016	549	4,522	—

Table 1: Summary of training, development and test sets. Corpora denoted • were directly provided by task organizers.

domain training data out of 18,598,505 sentences allowed for training.

The corpora for our English→Czech translation are summarized in Table 1: TRAIN is the provided in-domain corpus, CZENGPRE is the pre-release of CzEng 1.6 [7] as used in WMT16[1] and TOPX are the top 2 or 5% sentences from CZENGPRE selected by the described procedure. Finally, SYNTDEV+TEST is the synthetic corpus created from the source side of the provided development and test set for our system Chimera, see Section 3 below.

2.2. Test Data

In order to be able to tune our systems and select which setup to use for final submission, we split the provided development set of 5,967 sentences into DTEST and ETEST. DTEST contained 4,774 sentences and was used for MERT [8] in Moses-based systems, see Section 3 below. Neural Monkey runs were internally evaluated on DTEST and Nematus did not make use of DTEST at all. ETEST contained 1,193 sentences and was used for comparing results of our systems using the standard `mteval-v13a.pl`[2] with the option `--international-tokenization`.

2.2.1. Test and Training Data Overlap

When we ran our first system setup (Moses two phrase tables extracted from CZENGPRE and TRAIN), it achieved a suspicious result of 44 BLEU [9]. When searching for an explanation, we found that CZENGPRE (as well as the full release of the corpus CzEng 1.6), while permitted by IWSLT organizers, contains sentences from the development data.

To avoid overfitting, we created restricted versions of DTEST and ETEST, called DTESTDEDUP and ETESTDEDUP, resp., removing all sentences that were found in CZENGPRE. In the sentence-level (i.e. segment-level) comparison, we disregarded letter case, all punctuation and numbers and also most possible tokenization differences. But even then our Moses systems with CZENGPRE phrase table got more than 40 BLEU.

Source	If you were to shoot a bullet straight at the Sun, it would take 17 years to get there!
Moses	Takže by to bylo 6 250 dní nebo to podělíme 365 a získáme téměř 17 let!
Gloss	So this would be 6,250 days, or we divide it by 365 and get almost 17 years!
Reference	Pokud byste vystřelili kulku na Slunce trvalo by jí 17 let tam doletět.

Figure 1: Our Moses translation of one of the sentences in the official test set provides details not seen in the source. The official IWSLT reference translation is different, indicating that perhaps a slightly different revision of the data was included in the training and in the test corpora.

One possible explation is that segmentation schemes differed. For instance the segment *"What was that about," I'm saying. What was that about?* as present in one of the subtitle files would be probably stored as two separate sentences in CzEng. No segment-level matching would be thus able to identify the overlap.

The official test sets suffer the same overlap issue, as illustrated in Figure 1. The particular example comes from a Khan Academy video[3] and it is indeed listed in the data sources of IWSLT test set as a part of QED talks.[4]

2.3. Data Pre-Processing

Table 2 summarizes the pre-processing pipelines of our setups. Our Chimera and Nematus setups reuse (parts of) existing models and we had to honor the pre-processing used for the models. As a result, the comparison of the systems is not possible at the level of tokens but only in an end-to-end evaluation of translation quality, similarly as the whole IWSLT compares participating systems. Please note also that the different systems were trained on incomparable amounts of data – e.g. our Nematus run for IWSLT is only an adaptation of the models trained for WMT [10], so effectively, Nematus was trained on much larger data than what we list here.

The pre-processing for Chimera means the Morphodita tagger[5] [11], which includes tokenization and provides case-sensitive lemmas. We then cast the lemma case (capital for names) to word forms ("supervised truecasing", stc). Morphodita keeps hyphenated words as one token but phrase-

[1] `http://www.statmt.org/wmt16/translation-task.html`

[2] `ftp://jaguar.ncsl.nist.gov/mt/resources/mteval-v13a.pl`

[3] `https://www.amara.org/en/videos/G3Mcfu0B5hUN/cs/236630/`

[4] `https://sites.google.com/site/iwsltevaluation2016/home/off-limit-ted-talks`

[5] `http://ufal.mff.cuni.cz/morphodita`

System	Tokenization and Letter Case	BPE
Chimera	Morphodita stc + split at hyphens	no
Neural Monkey	Morphodita stc + split at hyphens	yes
Nematus	Moses tokenizer + truecasing	yes

Table 2: Corpus pre-processing.

	CZENGPRE	TRAIN	SYNTDEV+TEST	DepFix
UFAL Chimera	•	•	•	•
UFAL MosesTecto	•	•	•	—
UFAL Moses	•	•	—	—

Table 3: Training data and additional modules used in Moses-based systems

based MT benefits from finer tokens, so we split such words after Morphodita processing.

Nematus models, on the other hand, rely on Moses tools `tokenizer.perl` and `truecase.perl`.

For neural approaches, we limit the size of vocabulary to a fixed number of token types by applying byte-pair encoding (BPE, [12]), created jointly for the source and target languages.

3. Primary Submission: Chimera

Since we were only starting with neural machine translation, we decided to use our complex hybrid system Chimera as the primary submission. Chimera consists of three components: transfer-based deep-syntactic system TectoMT [13], which contributes translations to a factored Moses setup. The final translations are then processed with Depfix [14], a rule-based system for correcting grammar and also recovering lost negation.

The idea for IWSLT2016 was to reuse most of our WMT16 news translation setup [15] and adapt it for the new domain. The standard Chimera setup uses Moses with two phrase tables, one trained on a large general corpus, and one coming from TectoMT. We reused the large phrase table from WMT16, which was trained on CZENGPRE, and recreated the TectoMT phrase table by translating the IWSLT development and test set source with TectoMT. Several components of TectoMT are also trainable but this retraining is done only very rarely. Depfix is applied unaltered.

We achieve domain adaptation by two means: (1) Moses is tuned on the domain-specific DTEST, and (2) we used a third phrase table extracted from TRAIN in one of our setups.

The final selection of the particular Chimera variant was based on the evaluation on ETEST. The differences between our Moses-based systems are shown in Table 3. The highest BLEU score on ETEST was reached by our our baseline ("UFAL Moses") relying only on two phrase tables: CZENGPRE and TRAIN. Nevertheless we decided to use "UFAL Chimera" as our IWSLT primary submission, based on our experience in WMT16.

4. Main Focus: Neural Monkey

Neural Monkey is a sequence-to-sequence learning system following the standard encoder-decoder architecture implemented in TensorFlow[6]. Neural Monkey was used in the WMT16 multimodal and post-editing tasks [2], because it supports an arbitrary number of sentence encoders. We only made use of one encoder for IWSLT and used a branch of NM that aims to replicate the setup of Bahdanau et al. [16].

[6] `http://www.tensorflow.org/`

4.1. Learning Curves

To our knowledge, papers describing NMT models typically report only a few hyper-parameter settings, and subsequent works follow the reported values. We would like to partially fill this gap. NMT is however notoriously computationally demanding so we had to limit our analysis to rather few setups.

Figure 2 plots the learning curves of Neural Monkey in terms of BLEU score when trained on TRAIN and evaluated on DTEST. Each of the curves corresponds to approximately 82 hours of training in 16 threads on a 32-core non-GPU machine. This time is too short for proper training of an NMT system but we hoped it would suffice for the comparison. All the runs used the batch size of 500 sentence pairs and limited both source and target sentences to 50 tokens. Contrary to common practice, we did not shuffle the corpus after every epoch. Arguably, and as also pointed by one of our reviewers, shuffling may have helped to achieve better scores through more robust training.

The left plot in Figure 2 corresponds to models using BPE vocabulary of 5k (a joint vocabulary for both the source and target languages), the right plot corresponds to the vocabulary of 50k. We see that the vocabulary size is a critical parameter for NMT performance, the 50k systems reach about double the BLEU score in the given training time and seem rather far from saturation.

Larger vocabulary size is however much more time consuming on CPUs, so only about a half of the training iterations could have been performed. Running such a setup on GPU would take no difference in time, but the GPU RAM is the bottleneck here, usually requiring to reduce the batch size.

The different curves in the plots then correspond to various sizes of word embeddings ("emb"; the decoder and encoder use separate embeddings derived from the same vocabulary) and hidden state sizes ("rnn"). A similar pattern in observed in both 5k and 50k vocabularies: in each hidden state size, larger embedding helps to reach higher BLEU score in the same number of iterations and no difference in time is observed (on 16 CPU threads). Moving to a larger RNN size takes its cost in terms of time, so fewer iterations were achieved.

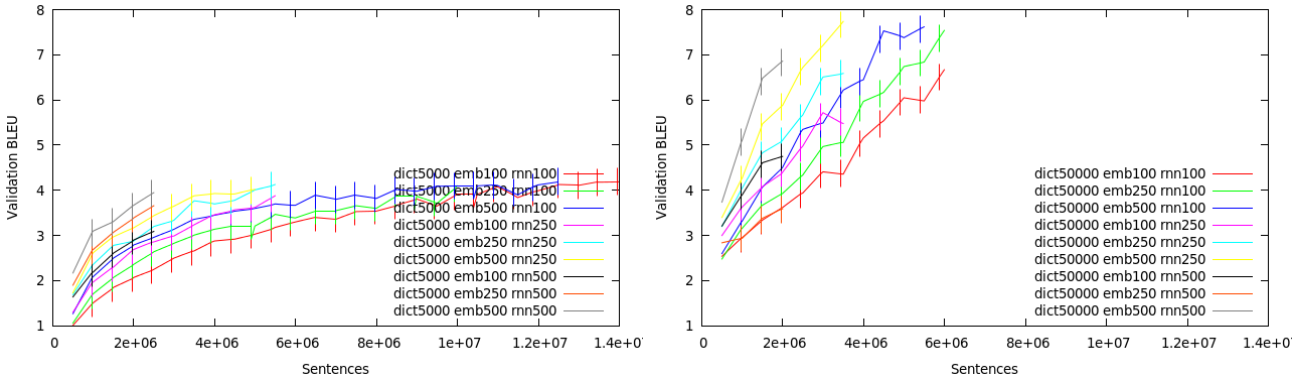

Figure 2: Learning curves of baseline setups varying vocabulary size ("dict"), embedding size ("emb") and the hidden state size ("rnn").

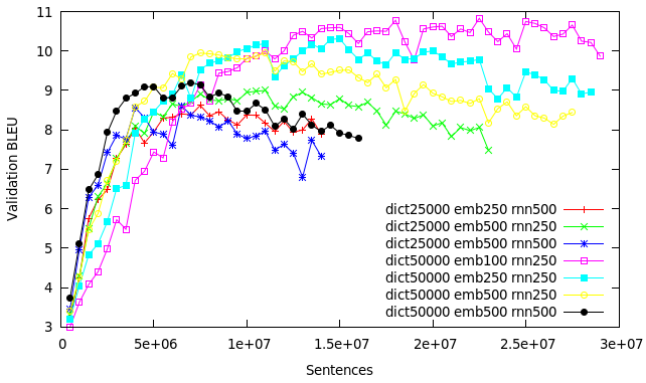

Figure 3: Learning curves of a few selected setups running until overfitting.

An unfortunate but very important observation is then illustrated in Figure 3: the performance in early stages of learning does not necessarily correspond to the final score. For example the run with dict50k and the largest hidden layers (emb500 rnn500) seemed most promising at the beginning but started overfitting at about 6M sentences at validation BLEU of 9. The less promising run "emb100 rnn250" did not overfit until 20M sentences and reached the best score of these test runs, BLEU between 10 and 11. As a consequence, the training should be run until the network starts overfitting (for large datasets this may take a very long time). For the best performance, the hyperparameters of the models have to be carefully selected to match the data, e.g. smaller training data in narrower domains may benefit from smaller embedding sizes. Computational complexity is a serious bottleneck, though. Each of the runs in Figure 3 took about 20 to 25 days on 16 CPU cores which might correspond to approximately two or three days on a powerful GPU.

Most of the learning curves for our runs have the typical slowly saturating shape. As illustrated in Figure 4, we have also seen quite a strange run where the network suddenly decided to "unlearn" everything and the validation BLEU dropped close to 0. At a later stage, the same run suddenly made a leap in validation scores. We speculate that the Adam optimizer employed in Neural Monkey may have changed some of its parameters but it is difficult to provide better insight; among other problems, this particular run took about 24 days on a GPU.

4.2. Considered Neural Monkey Runs

Figure 5 plots the learning curves of the few Neural Monkey runs we considered for submitting. All the runs are based on 50k BPE vocabulary extracted from the concatenation of TRAIN and TOP5, embedding size 300 and hidden state size 300. The epoch size in the legend indicates whether the particular run was traversing TRAIN (122k) or TOP5+TRAIN (in this order, 6M sentences).

The legend is sorted according to the score on ETESTD-EDUP at the system submission deadline, Run 1+2+3 being the best at that time.

While Run 1 was running, we were spawning other runs from its tentative saved models. The spawned runs start their learning curves at BLEU scores between 12 and 16. The system we ended up submitting was Run 1+2+3, derived from Run 1 and another intermediate Run 1+2. If we had the time to wait longer, Run 1 alone would have been probably the best system to submit, while Run 1+2+3 has deteriorated. In total, Run 1 spent more than 9 days to complete the plotted 10 epochs on a two-core GeForce GTX Titan Z with 6 GB RAM in each core. TensorFlow made use of both of the cores.

Some of the spawns were not successful, such as Run 1+5, or showed mixed performance such as Run 1+4.

With the exception of Run 1, the setups did not shuffle the training data after every epoch. This explains the periodic structure of Run 1+4.

The main difference was the batch size, which had to be limited due to lower GPU RAM on other available computers. Run 1 used the default batch size of 128, Run 2 used 20 and Run 3 used 50, Run 4 used 500 and Run 5 again only 20

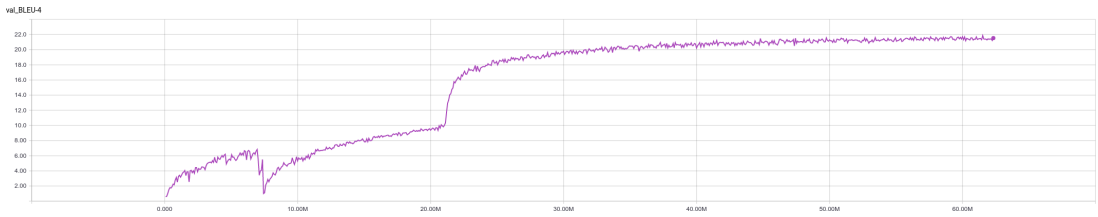

Figure 4: A rather unexpected learning curve (BLEU) of Neural Monkey on a validation set.

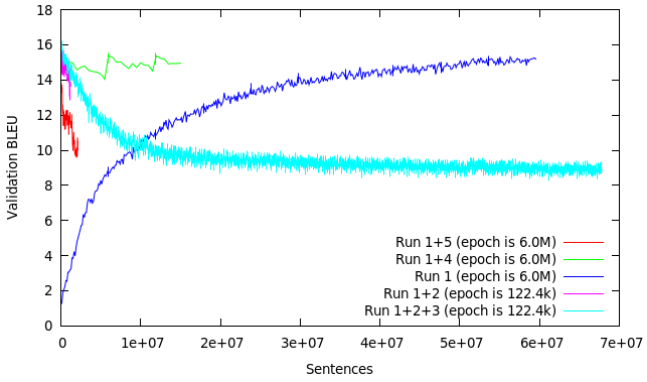

Figure 5: Learning curves of Neural Monkey runs.

parallel sentences.

It is not very clear why the performance increases in some runs and decreases in others. It is also unclear whether they would recover at a later stage. One explanation can be found in the batch size: Run 1 and Run 1+4 had access to 12 GB GPU RAM (Titan Z and Tesla K80, resp.) and could afford the batch size of 128 and 500, respectively. Larger batch size better average the gradients and make the search more stable. Moreover, Run 1 was shuffling the corpus at every epoch, increasing the stability again.

5. For Comparison: Nematus

To compare our systems with the state of the art, we also experimented with Nematus [3]. This NMT system is built upon dl4mt-tutorial[7] by Kyunghyun Cho et al.

We followed Rico Sennrich's tutorial on neural model adaptation presented at MT Marathon 2016[8], using the Nematus models which were trained for the WMT16 submissions[9]. [10]

For the adaptation process, we experimented with different training sets, learning rates and numbers of epochs. After a few experiments, we noticed that the model performance increases during first epoch over the training data but decreases as soon as in the second epoch. Therefore, we decided to run only one epoch and experiment with different training sets (TRAIN and TOP5 for Czech and similarly for German) while optimizing the learning rate to get the best results at the end of the first epoch. Each adaptation process took several hours to days on the GPU, depending on the size of the training data.

Direction	Original model	IWSLT trainset	Our trainset
en→cs	16.84	**18.29**	17.57
cs→en	25.12	25.16	**25.22**
en→de	19.89	**19.94**	19.90
de→en	27.89	**30.04**	29.99

Table 4: BLEU scores of Nematus on official development sets. Results in bold indicate models submitted to the IWSLT MT Task.

In general, the unadapted WMT16 model (news domain) performed quite well on the IWSLT data and the adaptation process was not very helpful: we got only a slight improvement on the development sets. Table 4 shows the performance of the original models and models adapted using the official training set or our selected set (TOPX). In the end, we submitted the non-adapted runs except for cs→en.

5.1. Neural Monkey vs. Nematus and Bahdanau et al.

Neural Monkey and Nematus should both implement the same model of [16]. Nematus is actually derived from Bahdanau et al.'s codebase.

Table 5 summarizes the obvious differences of our Nematus and Neural Monkey runs.

Additionally, we carefully examined the implementations and noticed the following differences:

- Nematus uses two-layer conditional GRU,[10] while in the Neural Monkey implementation, only a single GRU layer is used in the decoder.
- The initial state of the decoder state is computed differently. In [16], the initial state is the result of a linear projection of the last hidden state of the encoder backward RNN with the tanh activation function. In contrast, Nematus code uses the average of all encoder

[7] https://github.com/nyu-dl/dl4mt-tutorial
[8] http://www.statmt.org/mtm16/programme.html
[9] http://data.statmt.org/rsennrich/wmt16_systems/
[10] https://github.com/nyu-dl/dl4mt-tutorial/blob/master/docs/cgru.pdf

	Neural Monkey	Nematus
Training data	TRAIN and TOP5	CZENGPRE, mononews +adapt on: TRAIN, TOP5
BPE vocabulary	50k	85k
Embedding size	300	500
RNN size	300	1024
Batch size	20 – 500	80
Max. sent. len.	50	50
Dropout	0.8	not used
L2 regularization	1×10^{-8}	0.0
Gradient clip.	–	1.0
Optimizer	Adam	Adadelta
Learning rate	–	0.0001
Beam search	–	12

Table 5: Differences between our NMT system setups for English→Czech.

hidden states. Finally, in Neural Monkey, the decoder initial state computation is configurable. In our experiments, we used a linear projection of the final state of the encoder.

- As decoder output projection, Neural Monkey uses the deep output with one maxout hidden layer [17, 18], as described in Bahdanau et al. [16]. In contrast, Nematus applies a feed-forward neural network with one hidden tanh layer to the outputs from the conditional GRU layers.

- Nematus disables L2 regularization by setting its parameter to 0.0.

- Nematus uses gradient clipping of 1.0, Neural Monkey leaves the gradients as they are.

- Nematus does not use any dropout technique, whereas Neural Monkey employs one dropout probability parameter for the whole model and applies the dropout to the embedding layer of the encoder and decoder, the encoder projection layer to the decoder initial state, encoder hidden states in the attention mechanism, and to the hidden state-to-output transitions in the decoder.

- There is also a few differences in the parameter initialization and the used optimizer. The biggest difference is that Neural Monkey does not initialize the weight matrices in the recurrent connections as orthogonal matrices. This issue, however, is already solved in the current version.

The comparison of the implementation of Theano vs. Tensorflow internals is left as an exercise to the reader...

All these differences can easily affect the performance of the systems and we plan to carefully evaluate them in the future.

6. Results and Discussion

Table 6 lists the official IWSLT MT track results for English to Czech translation.

We see that the results vary a lot across the different test corpora. We are afraid that only very indicative conclusions can be drawn from these results, esp. due to the corpus overlap described in Section 2.2 above. Different systems may have benefited from the overlap to different degrees and such a gain has no relation to the actual translation quality of the system. The consistent win of MosesTecto over Chimera is against all our experience from WMT and suggests that BLEU reflects the exact match with the overlapping data.

This is our first participation in IWSLT, and sadly, we underestimated subtitle formatting conventions. Our systems are generally set up to translate one full sentence at a time. The IWSLT MT track, at least in the QED domain, uses segment breaks as reasonable for presenting subtitles. Unfortunately, many sentences are broken into shorter segments and on the other hand, some segments contain (parts of) several sentences.

The unexpected sentence breaks prevent our systems from seeing relevant parts of the sentence. E.g. Depfix in Chimera checks for presence of certain sentence elements in the parse of the MT output (e.g. negated verb modifiers or reflexive particles complementing verbs). The quality of the parse is going to be severely damaged and some of the checked sentence elements may appear in another segment. The 20-gram POS LM in Chimera is aimed to capture the long-range sentence structure, but we now apply it to sentence parts... And finally, neural models (both Nematus and Neural Monkey) are also trained on full sentences. Here, they will suffer from not seeing the preceding context within the sentence, and they also have a strong bias to stop producing anything after the first full stop, leading to shorter output.

7. Comments on Training on Amazon EC2

We explored the possibility of training Neural Monkey and Nematus on the GPU-enabled machines on Amazon EC2. Getting the setup running was not very straightforward at the first attempt, due to various requirements of bleeding-edge tools (e.g. Neural Monkey requires Python 3.5 but TensorFlow did not have ready-made GPU packages for that yet). Fortunately, in the meantime, Amazon released the Deep Learning AMI[11] (Amazon Machine Image), which includes Python, TensorFlow (though not compiled for Python 3.5) and setup scripts for the GPU drivers (the drivers themselves have to be downloaded manually from Nvidia), so we could abandon our fragile and complex setup. We added Neural Monkey and produced an AMI ready for NMT experiments. Nematus was easier to get running because it builds upon less recent tools.

Our training workflow then consisted of launching a new

[11] https://aws.amazon.com/marketplace/pp/B01M0AXXQB

QED.TST2016		TED.TST2015		TED.TST2016	
LIMSI primary	15.89	UFAL Nematus	25.93	UFAL Nematus	22.28
UFAL MosesTecto	14.39	LIMSI primary	19.18	LIMSI primary	16.24
UFAL Chimera	14.18	UFAL Neural Monkey	16.04	UFAL MosesTecto	12.71
UFAL Moses	13.35	UFAL MosesTecto	15.72	**UFAL Chimera**	12.71
UFAL Nematus	12.35	**UFAL Chimera**	15.71	UFAL Neural Monkey	12.60
UFAL Neural Monkey	7.89	UFAL Moses	14.63	UFAL Moses	12.30

Table 6: Tentantive official results for English→Czech. UFAL Chimera in bold is our primary submission.

machine instance based on this AMI using the AWS SDK,[12] copying over the necessary data, running the training remotely, copying back the result and terminating the instance. We also regularly monitored the training by synchronizing the working directory from the remote machine to our local disks and terminated the instance when the model started to overfit.

8. Conclusion

We presented our submission to the IWSLT MT track, covering phrase-based, hybrid and two neural MT systems. The whole exercise was very useful for our main aim of developing the Neural Monkey toolkit. A detailed analysis of Nematus and Neural Monkey has indicated several implementation details that we should change in the near future.

We also provided some general observations on NMT hyperparameters and training, hoping to help other researchers entering the field.

The overall results of the task are unfortunately not very reliable. One reason is that we paid insufficient attention to subtitle formatting customs, and the main reason is that the training and development data for Czech unfortunately had an overlap.

9. Acknowledgements

This research was supported by the grants H2020-ICT-2014-1-645452 (QT21), SVV 260 333, GAUK 8502/2016 and using language resources distributed by the LINDAT/CLARIN project of the Ministry of Education, Youth and Sports of the Czech Republic (LM2015071). We are also grateful to Amazon for their provision of EC2 credits for the purposes of MT Marathon 2016.

10. References

[1] O. Bojar, R. Rosa, and A. Tamchyna, "Chimera – Three Heads for English-to-Czech Translation," in *Proceedings of the Eighth Workshop on Statistical Machine Translation*. Sofia, Bulgaria: Association for Computational Linguistics, August 2013, pp. 92–98. [Online]. Available: http://www.aclweb.org/anthology/W13-2208

[2] J. Libovický, J. Helcl, M. Tlustý, O. Bojar, and P. Pecina, "Cuni system for wmt16 automatic post-editing and multimodal translation tasks," in *Proceedings of the First Conference on Machine Translation*. Berlin, Germany: Association for Computational Linguistics, August 2016, pp. 646–654. [Online]. Available: http://www.aclweb.org/anthology/W/W16/W16-2361

[3] 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. [Online]. Available: http://www.aclweb.org/anthology/W16-2323

[4] O. Bojar, R. Chatterjee, C. Federmann, Y. Graham, B. Haddow, M. Huck, A. Jimeno Yepes, P. Koehn, V. Logacheva, C. Monz, M. Negri, A. Neveol, M. Neves, M. Popel, M. Post, R. Rubino, C. Scarton, L. Specia, M. Turchi, K. Verspoor, and M. Zampieri, "Findings of the 2016 Conference on Machine Translation," in *Proceedings of the First Conference on Machine Translation*. Berlin, Germany: Association for Computational Linguistics, August 2016, pp. 131–198. [Online]. Available: http://www.aclweb.org/anthology/W/W16/W16-2301

[5] P. Koehn, H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, C. Dyer, O. Bojar, A. Constantin, and E. Herbst, "Moses: Open source toolkit for statistical machine translation," in *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, ser. ACL '07. Stroudsburg, PA, USA: Association for Computational Linguistics, 2007, pp. 177–180. [Online]. Available: http://dl.acm.org/citation.cfm?id=1557769.1557821

[6] A. Rousseau, "Xenc: An open-source tool for data selection in natural language processing," *The Prague Bulletin of Mathematical Linguistics*, vol. 100, pp. 73–82, 2013.

[7] O. Bojar, O. Dušek, T. Kocmi, J. Libovický, M. Novák, M. Popel, R. Sudarikov, and D. Variš, "Czeng 1.6: en-

[12] https://aws.amazon.com/sdk-for-python/

larged czech-english parallel corpus with processing tools dockered," in *International Conference on Text, Speech, and Dialogue*. Springer, 2016, pp. 231–238.

[8] F. J. Och, "Minimum error rate training in statistical machine translation," in *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, ser. ACL '03. Stroudsburg, PA, USA: Association for Computational Linguistics, 2003, pp. 160–167. [Online]. Available: http://dx.doi.org/10.3115/1075096.1075117

[9] 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 on Association for Computational Linguistics*, ser. ACL '02. Stroudsburg, PA, USA: Association for Computational Linguistics, 2002, pp. 311–318.

[10] R. Sennrich, B. Haddow, and A. Birch, "Improving neural machine translation models with monolingual data," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany: Association for Computational Linguistics, August 2016, pp. 86–96. [Online]. Available: http://www.aclweb.org/anthology/P16-1009

[11] J. Straková, M. Straka, and J. Hajič, "Open-Source Tools for Morphology, Lemmatization, POS Tagging and Named Entity Recognition," in *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. Baltimore, Maryland: Association for Computational Linguistics, June 2014, pp. 13–18. [Online]. Available: http://www.aclweb.org/anthology/P/P14/P14-5003.pdf

[12] R. Sennrich, B. Haddow, and A. Birch, "Neural machine translation of rare words with subword units," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany: Association for Computational Linguistics, August 2016, pp. 1715–1725. [Online]. Available: http://www.aclweb.org/anthology/P16-1162

[13] O. Dušek, Z. Žabokrtský, M. Popel, M. Majliš, M. Novák, and D. Mareček, "Formemes in english-czech deep syntactic MT," in *Proceedings of NAACL 2012 Workshop on Machine Translation*. Montréal, Canada: Association for Computational Linguistics, 2012, pp. 267–274.

[14] R. Rosa, D. Mareček, and O. Dušek, "DEPFIX: A system for automatic correction of Czech MT outputs," in *Proceedings of the Seventh Workshop on Statistical Machine Translation*, ser. WMT '12. Stroudsburg, PA, USA: Association for Computational Linguistics, 2012, pp. 362–368.

[15] A. Tamchyna, R. Sudarikov, O. Bojar, and A. Fraser, "Cuni-lmu submissions in wmt2016: Chimera constrained and beaten," in *Proceedings of the First Conference on Machine Translation, Berlin, Germany. Association for Computational Linguistics*, 2016.

[16] D. Bahdanau, K. Cho, and Y. Bengio, "Neural Machine Translation by Jointly Learning to Align and Translate," in *Proceedings of the Third International Conference on Learning Representations (ICLR 2015)*, 2015. [Online]. Available: http://arxiv.org/abs/1409.0473

[17] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. C. Courville, and Y. Bengio, "Maxout Networks," in *Proceedings of The 30th International Conference on Machine Learning (ICML 2013)*, 2013, pp. 1319 – 1327.

[18] R. Pascanu, C. Gulcehre, K. Cho, and Y. Bengio, "How to Construct Deep Recurrent Neural Networks," in *Proceedings of the Second International Conference on Learning Represenations (ICLR 2014)*, 2014. [Online]. Available: http://arxiv.org/abs/1312.6026