Expand and Filter: CUNI and LMU Systems for the WNGT 2020 Duolingo Shared Task

Jindřich Libovický[1] and **Zdeněk Kasner**[2] and **Jindřich Helcl**[2] and **Ondřej Dušek**[2]

[1]Center for Information and Language Processing, LMU Munich, Germany
[2]Faculty of Mathematics and Physics, Charles University, Prague, Czech Republic
libovicky@cis.lmu.de, {kasner,helcl,odusek}@ufal.mff.cuni.cz

Abstract

We present our submission to the Simultaneous Translation And Paraphrase for Language Education (STAPLE) challenge. We used a standard Transformer model for translation, with a crosslingual classifier predicting correct translations on the output *n*-best list. To increase the diversity of the outputs, we used additional data to train the translation model, and we trained a paraphrasing model based on the Levenshtein Transformer architecture to generate further synonymous translations. The paraphrasing results were again filtered using our classifier. While the use of additional data and our classifier filter were able to improve results, the paraphrasing model produced too many invalid outputs to further improve the output quality. Our model without the paraphrasing component finished in the middle of the field for the shared task, improving over the best baseline by a margin of 10–22% weighted F1 absolute.

1 Introduction

The usual goal of machine translation (MT) is to generate a single correct translation of a source sentence. Neural machine translation (NMT; Bahdanau et al., 2015; Vaswani et al., 2017) models a conditional distribution over possible target sentences given a source sentence, and uses beam-search decoding as a heuristic to get one or more translations. However, the number of *possible* correct translations is often vast in comparison (Bojar et al., 2013).

The STAPLE challenge (Mayhew et al., 2020) poses the problem of MT slightly differently. Here, the goal is to generate as many correct translations as possible. Knowing many correct translations can be useful e.g. for automatic scoring in tools for language education, such as Duolingo.[1] On one hand, the learners should be guided to use the more common formulations, on the other hand, they should not be penalized for providing a correct but unusual answer.

We present a pipeline of two systems in our submission to the STAPLE challenge. As the first step, we use a standard NMT model trained with additional, carefully filtered data. The NMT output *n*-best lists are filtered using a classifier. Second, we use a Levenshtein Transformer model (Gu et al., 2019) to generate paraphrases of the outputs of the first model. Again, the outputs of the Levenshtein Transformer are filtered using another classifier.

The paper is structured as follows. Our training datasets are described in detail in Section 2. We describe the two models in Sections 3 and 4 respectively. We conduct experiments with all five target languages in the challenge, i.e. Hungarian, Japanese, Korean, Vietnamese and Portuguese. The source language is English in all setups. The experiment settings are shown in Section 5. The results (Section 6) show that the Levenshtein Transformer paraphrase generator cannot easily improve on the filtered NMT output *n*-best list.

2 Data

2.1 STAPLE Dataset

The data in the STAPLE shared task comes from the Duolingo language learning platform. This represents a specific domain with a limited number of mostly simple sentences targeted at learners, using a limited vocabulary.

Each source (English) sentence comes with a list of valid translations in the target language, ranging from a few up to hundreds of paraphrases. All of the valid translations are further annotated with a probability score indicating how frequent a given variant is. Statistics of the data are given in Table 1.

We held out 200 source sentences with all their

[1]https://www.duolingo.com

Language	Prompts	Trans.	Avg. P/S
Hungarian	3,800	238,467	62.75
Korean	2,300	646,410	281.05
Japanese	2,300	788,591	342.87
Portuguese	3,800	503,839	132.59
Vietnamese	3,300	183,339	55.57

Table 1: STAPLE training data statistics (target language set, number of source sentences (prompts), number of paraphrases (translations), average number of paraphrases per source).

translations as our internal validation dataset. We use this dataset for validating the translation models and for estimating the filtering thresholds.

2.2 Additional data for MT

For training the translation model, we obtained out-of-domain parallel corpora from the OPUS collection (Tiedemann, 2012) for all target languages, ParaCrawl (Esplà et al., 2019) for Portuguese and Hungarian, and JParaCrawl (Morishita et al., 2019) for Japanese. We applied FastText language identifier (Joulin et al., 2016b,a) to clean the corpora. Furthermore, we filtered out sentence pairs with a length ratio that differs from the estimated mean ratio by more than 2.5 times the standard deviation.

As in-domain training data, we mix data from the STAPLE training dataset and the Tatoeba[2] corpus (part of the OPUS collection). To balance the underrepresentation of the in-domain data in the training dataset, we oversample both the STAPLE dataset (200×) and the Tatoeba dataset (10×).

We use the combined mixed-domain parallel corpora for training backtranslation models (Sennrich et al., 2016) and an XLM-R-based domain classifier. The classifier is trained to predict whether a target sentence came from the STAPLE training dataset, conditioning on the source sentence.

The monolingual data consists of Wikipedia, WMT NewsCrawl (Barrault et al., 2019) for Hungarian, Japanese and Portuguese, Leipzig Corpora NewsCrawl (Goldhahn et al., 2012) for all languages and the jpWaC corpus for Japanese (Erjavec et al., 2008).

We filtered both the monolingual and parallel data using the domain classifier. The classifier has over 99% accuracy on balanced data. We set a permissive threshold for keeping the sentence pair to 10^{-5}.

[2] https://tatoeba.org/

Based on preliminary experiments, we include only a single correct translation from the STAPLE training set into the machine translation training data. This had a slightly positive effect on translation quality. Also, the n-best lists obtained by machine-translating the STAPLE dataset are more representative and thus more suitable for training the classifier for n-best list filtering than if we included all translations from the training set.

3 Translation and Filtering Model

3.1 Translation Model

Our pipeline starts with the Transformer model (Vaswani et al., 2017) trained on the provided dataset enriched with additional data (see Section 2). This provides initial translations of the source sentence on the output n-best list, which are further filtered.

3.2 Filtering Classifier

We train a crosslingual classifier which predicts whether a translation in the MT output n-best list is correct (given the source sentence). Using the trained translation model, we first generate large n-best lists for all English sentences in the original training data. Next, we label each generated sentence whether it is a positive or a negative sample (based on the reference data). Finally, we create a balanced mix of negative and positive samples. Since the n-best lists contained much more incorrect translations, we oversample the list of correct translations.

We use XLM-RoBERTa as our sentence classifier model (XLM-R; Conneau et al., 2020), specifically the pretrained variant available in the HuggingFace Transformers library[3] (Wolf et al., 2019). We finetune this model on the balanced mix of the correct and incorrect translations for a given sentence.

During inference, we generate an n-best list for a given source sentence, and we apply the classifier to filter out the sentences which are labeled as incorrect using a threshold value. The n-best list size and the threshold are hyperparameters of the method.

4 Paraphrasing Model

As an additional step to increase the number of valid translations produced, we train a target-

[3] https://github.com/huggingface/transformers

	All parallel	+ lang. & length filter	+ domain filtering	Monolingual	+ domain filtering	Total (unique)
Hungarian	56.1M	47.9 M	22.1M	54.3M	7.0M	27.1M
Japanese	14.7M	14.0 M	1.9M	44.2M	2.8M	4.7M
Korean	2.2M	2.0 M	0.6M	9.3M	0.8M	1.4M
Portuguese	63.5M	53.7 M	21.3M	30.0M	4.5M	25.8M
Vietnamese	3.9M	3.4 M	1.5M	9.7M	1.4M	2.8M

Table 2: Number of sentences in the parallel and monolingual data used for training the MT systems. The data in the second column were used for training the backtranslation systems, the last column corresponds to final translation systems.

language paraphrasing model. Rather than generating a translation directly from the source sentence, the model refines existing translations in order to produce new ones.

The model is based on the Levenshtein Transformer (LevT; Gu et al., 2019), which is a sequence generation model based on the Transformer (Vaswani et al., 2017) architecture. Instead of left-to-right autoregressive generation, LevT generates sequences in an arbitrary order using two basic operations – insertion and deletion. Using an initial sequence as a starting point, LevT is able to perform sequence refinement.

4.1 Training

LevT iteratively applies three policies represented by fully-connected neural network layers on top of the last layer of a Transformer decoder:

1. *Deletion* policy π^{del} removes tokens from the sequence;
2. *Placeholder* policy π^{plh} inserts placeholders into the sequence;
3. *Insertion* policy π^{ins} replaces placeholders with tokens from the vocabulary.

The policies are trained to follow oracle policies. Given a source sequence X and a target sequence Y, $L(X,Y)$ is a minimum sequence of edit operations (delete and insert) that transform X to Y. Its length is equal to the Levenshtein distance (Levenshtein, 1966) between X and Y. The operations in $L(X,Y)$ define oracle policies for π^{del}, π^{plh} and π^{ins}.

There are two other possible training strategies: Either training the insertion policy to repair a target sentence with randomly dropped tokens, or training the deletion policy to refine the output from the insertion policy. However, we do not use these strategies for our model. In the first case, we did not find it beneficial for the model performance. In the second case, the option does not fit together with our inference scheme as described in Section 4.2.

To train the model to gradually produce more diverse, but still valid paraphrases, we provided the model with training paraphrase pairs with minimum edit distance. We represent the set of paraphrases as a complete graph with edges weighted by the Levenshtein distance (see Figure 1). We construct a minimum spanning tree of this graph and use the sentence pairs from the spanning tree edges as training examples.

Formally, a training example for LevT is a tuple (E, X_0, X, Y) where E is the original English sentence, X_0 is the gold translation, X is the source node and Y is the target node. (E, X_0) is processed by the encoder, X is used as the initial sequence for the decoder and Y acts as the ground truth. We do not use any additional data for LevT.

4.2 Inference

In the original LevT formulation, inference is done by applying the model over the initial sequence for several iterations. This approach aims to produce a single output translation and the intermediate translations are deemed to be incorrect. We redefine the generation process as *state-space search*, considering translations in each step as potentially correct and accepting the translations based on the classifier score. We also repurpose the deletion policy as *paraphrasing policy*, which gives us the possibility to generate multiple translations in each step. Similarly to the original LevT architecture, the encoder output is grounding the translations in the source sentence throughout the inference process.

The process can start from an arbitrary number of initial translations – in our case, the initial translations are the filtered outputs from the MT system. We put all the initial translations in a queue. In each step, we pop a translation from the queue, and we let the deletion policy mark all tokens suitable for paraphrasing. To expand our search space, we gen-

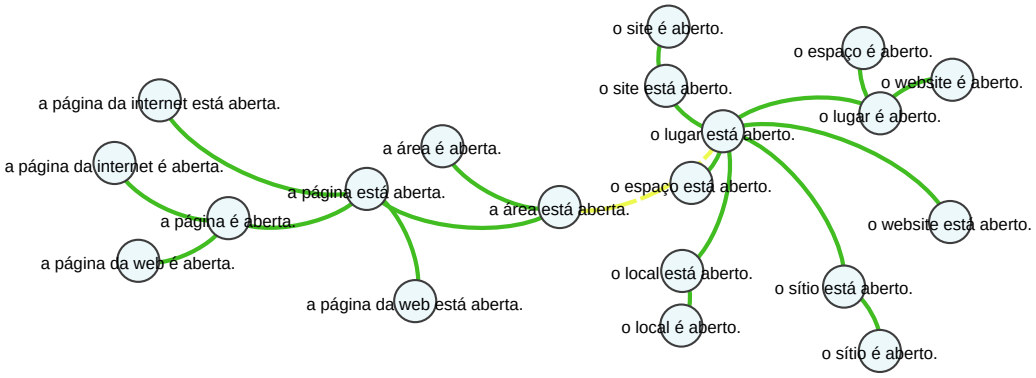

Figure 1: A minimum spanning tree of the graph of Portuguese translations for "*The site is open.*". Levenshtein distance (the sum of delete and insert operations) is 2 for solid green lines and 6 for the dashed yellow line.

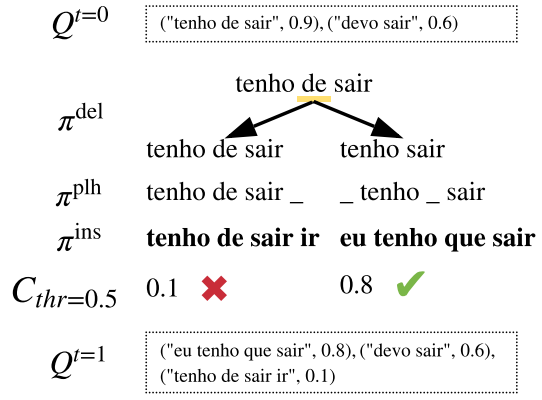

Figure 2: An example of a step of the inference algorithm of the paraphrasing model. A translation is taken from the queue $\mathcal{Q}$ and expanded using LevT policies. Results are scored by the classifier C; the translations scoring above the threshold $thr = 0.5$ are accepted. The translations are grounded in the source sentence by the encoder output.

erate multiple versions of the partially deleted sentence using all possible combinations of selected tokens, which are individually processed by the placeholder and insertion policies. We put the output translations in the queue and repeat the process until the queue is empty or we reach a preset limit on the number of generated sentences.

4.3 Filtering

Similarly to our NMT model, we filter relevant translations using a classifier built on top of XLM-R (see Section 3). In this case, the negative examples for the classifier are generated from LevT. We use the classifier's predictions to accept only translations passing a preset threshold. Moreover, we use the scores predicted by the classifier (probability of translation correctness) to define the priority of the translations in the queue, thus making it a priority queue. Figure 2 shows an example of a single step of the inference algorithm with filtering.

5 Experiments

5.1 Evaluation Metric

We use the official evaluation metric for the STAPLE challenge, which is the *weighted macro F1 score*, computed by exact match with respect to the set of all valid translations for a given source sentence. The weighted F1 is a compound of unweighted precision and weighted recall, where the weight is determined based on each translation's probability (see Section 2.1).

5.2 Translation Model

We train the translation model using the Marian toolkit[4] (Junczys-Dowmunt et al., 2018). We use the Transformer Base hyperparameters, i.e., model dimension 512, feed-forward layer dimension 2048, 8 attention heads with a head dimension of 64. All models use SentencePiece-based (Kudo and Richardson, 2018) vocabulary of 32k units. Japanese was tokenized using UDPipe (Straka and Straková, 2017), other languages were processed with SentencePiece without tokenization.

The model is trained using the Adam optimizer (Kingma and Ba, 2015), with Noam learning schedule (Vaswani et al., 2017) with 8,000 warmup steps and initial learning rate $3 \cdot 10^{-4}$, dropout rate 0.1, label smoothing 0.1 and gradient clipping at 5.0. We set the training batch size to 4,096 tokens.

5.3 Paraphrasing Model

We base our paraphrasing model on the Levenshtein Transformer as implemented in fairseq[5] (Ott

[4] https://marian-nmt.github.io/
[5] https://github.com/pytorch/fairseq

et al., 2019). We replace the Transformer encoder with the pretrained XLM-R Base with 12 layers and 8 attention heads, keeping the vanilla Transformer decoder with 6 layers and 8 attention heads. We use the output layer of XLM-R as the decoder output layer and finetune it together with the last four layers of XLM-R (freezing the rest of XLM-R parameters). We train a separate decoder for each policy and employ the early exit as described by Gu et al. (2019) by using only the features from the third layer of the decoder for the deletion and placeholder policies.

The model is optimized using the Adam optimizer, with fixed learning rate 10^{-5}, batch size 1,500 tokens, dropout rate 0.5 and label smoothing 0.1. We set the maximum number of placeholders for each position at 3 instead of 256.

We experiment with various decoding strategies. For deletion and placeholder policies, we introduce a penalty parameter preventing the policy from producing the zero (unchanged) outcome. This proved beneficial in particular for the deletion policy, as it frequently did not mark any tokens for deletion, thus limiting the search process. Alternatively, we force the policy to produce an outcome by selecting top k results with the highest score. In both cases, we find that limiting the number of placeholders generated at the same position to 1 helps to prevent excessive sentence length.

For the insertion policy, we give up the non-autoregressivity and replace the placeholders in the left-to-right order, re-running the decoder in each step. This acts as a supplement for the fact that the insertion policy cannot repeatedly interact with the deletion policy in a single state-space search step.

Our experiments show that it is difficult to find a set of decoding parameters which would consistently produce meaningful output. For generating paraphrases, we find it useful to use the *penalty* strategy described above and tune the penalty separately for each language. Producing less paraphrases generally leads to better results, as it tends to limit the amount of incorrect output. On the contrary, we use the *top-k* strategy for generating the classifier training data, as it produces negative samples more robustly.

6 Results

The results on the blind development and test data are shown in Table 3. Our translation model was able to bring considerable improvements over the provided baseline models – 10–22% weighted F1 absolute due to increase in both precision and recall. We observed the highest improvements for Portuguese. Overall, our model tends to finish in the middle of the field.

Experiments with training data filtering showed that a smaller training set with better selected sentences leads to better trasnslation quality than using a larger general domain corpus (see Table 4), although the general corpus was double in the number of sentences (see Table 2). The optimal beam size for filtering only roughly corresponds to the average number of paraphrases in the data (see Table 1). Figure 3 shows the effect of beam size on the output quality. Without filtering, the precision quickly drops with increasing beam size. The filtering can partially compensate for the precision loss, however, at the expense of decreasing recall, too.

Despite our extensive efforts, the paraphrasing model did not bring substantial improvements over the translation model. Increase in recall typically came with lower precision and lower weighted F1 score. Even with the specific classifier training and its overall accuracy over 99%, the LevT output was too noisy to be precisely filtered. The only improvement was achieved for Portuguese, where the final paraphrasing and filtering setting resulted in slightly higher precision and similar recall. However, the improvement was mostly influenced by the second round of filtering (on the paraphrasing output). Therefore, we did not include the paraphrasing output in our primary submission.

We suppose that the inefficiency of the LevT model is caused by a mismatch between model training criteria and its application: the individual policies are trained separately, but need to complement each other to achieve good results. Moreover, the loss computed independently for each paraphrase may prevent the model from learning to generate multiple paraphrases. The shortage of useful paraphrases may be also caused by the lack of additional training data.

7 Conclusions

We presented our submission to the 2020 STAPLE translation and paraphrasing shared task. Our model is based on the Transformer architecture, used additional carefully selected training data, a XLM-R-based classifier to filter MT output beams, and an optional paraphrasing component based on the Levenshtein Transformer approach. The MT

Target	System	Development						Test					
		Rk	Pre	W-R	W-F1	Δbase	Δbest	Rk	Pre	W-R	W-F1	Δbase	Δbest
Hungarian	MT	4/9	50.65	50.66	46.25	+16.40	-13.58	4/7	51.04	45.77	43.49	+15.39	-11.91
	+Para	-	49.73	50.57	45.75	+15.90	-14.08	-	49.91	45.66	43.01	+14.91	-12.39
Japanese	MT	5/10	39.02	19.36	21.50	+17.25	-10.09	5/8	36.70	19.93	21.28	+16.97	-10.49
	+Para	-	38.27	19.32	21.34	+17.09	-10.25	-	35.76	19.87	21.08	+16.77	-10.69
Korean	MT	4/8	40.05	21.57	22.21	+16.76	-19.16	4/6	38.94	19.35	20.58	+15.71	-19.77
	+Para	-	39.65	21.06	21.87	+16.42	-19.50	-	38.42	18.92	20.29	+15.42	-20.06
Portuguese	MT	8/14	49.75	46.78	42.74	+21.59	-13.00	6/10	49.88	43.81	40.84	+19.54	-14.26
	+Para	-	50.72	46.60	43.22	+22.07	-12.52	-	50.94	43.44	41.18	+19.88	-13.92
Vietnamese	MT	3/6	52.27	37.36	38.26	+11.47	-16.47	3/5	51.59	36.84	37.71	+12.32	-17.85
	+Para	-	53.17	36.39	37.68	+10.89	-17.05	-	52.09	36.35	37.34	+11.95	-18.22

Table 3: Results for both system variants and all languages. "MT" denotes the variant without paraphrasing (final official result), "+Para" is the system with paraphrasing. Metrics: *Rk* = official competition rank (with the number of valid submissions), *Pre* = precision (%), *W-R* = weighted recall (%), *W-F1* = weighted F1 (%), Δ*base* = weighted F1 difference w. r. t. the best baseline model (% absolute), Δ*best* = w. r. t. the overall winner (% absolute).

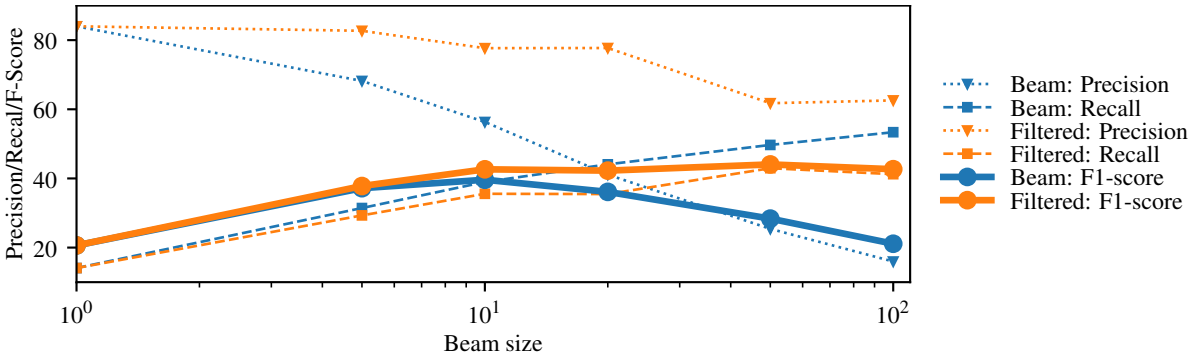

Figure 3: Precision, weighted recall and weighted F1 score for NMT decoding with different beam sizes and with and without beam filtering.

	AWS	Gen.	Dom.	Best	
Hungarian	29.2	27.4	40.2	49.1	100
Japanese	4.8	2.8	11.9	18.6	1000
Korean	4.9	6.5	11.5	22.0	1000
Portuguese	23.1	29.9	39.6	44.0	50
Vietnamese	20.3	33.0	38.5	42.1	5

Table 4: Translation quality measured by the weighted F1 score for general training (Gen.) data and domain-specific (Dom.) training data measured on our validation set with beam size 10 compared with the task baseline (AWS) and best filtered beam with beam size.

model with the filter was able to gain substantial increases over the baseline, but did not reach the top places in the challenge. The paraphrasing component's output proved too noisy to bring any substantial benefits – we only observed minor improvements in Portuguese. Therefore, the paraphrasing was not included in our primary submission.

Improving the paraphrasing model could be an interesting direction of future work. The amount of incorrect output could be reduced by better accounting for deletion and insertion policy interplay. However, computing the loss independently for each paraphrase may still hinder the ability of the model to generate multiple paraphrases for a single sentence. It may be thus necessary to rethink the training objective and tie it together with the inference process.

Acknowledgements

Work conducted at LMU was supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 640550).

Work conducted at CUNI was supported by the Charles University grant No. 140320, the SVV project No. 260575, the European Union's Horizon 2020 research and innovation programme under grant agreement No. 825303 (Bergamot), Czech Science Foundation grant No. 19-26934X (NEUREM3), and the Charles University project PRIMUS/19/SCI/10.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations (ICLR2015)*, San Diego, CA, USA.

Loïc Barrault, Ondřej Bojar, Marta R. Costa-jussà, Christian Federmann, Mark Fishel, Yvette Graham, Barry Haddow, Matthias Huck, Philipp Koehn, Shervin Malmasi, Christof Monz, Mathias Müller, Santanu Pal, Matt Post, and Marcos Zampieri. 2019. Findings of the 2019 conference on machine translation (WMT19). In *Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1)*, pages 1–61, Florence, Italy. Association for Computational Linguistics.

Ondřej Bojar, Matouš Macháček, Aleš Tamchyna, and Daniel Zeman. 2013. Scratching the surface of possible translations. In *Proc. of TSD 2013*, Lecture Notes in Artificial Intelligence, Berlin / Heidelberg. Západočeská univerzita v Plzni, Springer Verlag.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of ACL 2020*. ArXiv: 1911.02116.

Tomaž Erjavec, Kristina Hmeljak Sangawa, and Yoshiko Kawamura. 2008. Japanese web corpus with difficulty levels jpWaC-l 1.0. Slovenian language resource repository CLARIN.SI.

Miquel Esplà, Mikel Forcada, Gema Ramírez-Sánchez, and Hieu Hoang. 2019. ParaCrawl: Web-scale parallel corpora for the languages of the EU. In *Proceedings of Machine Translation Summit XVII Volume 2: Translator, Project and User Tracks*, pages 118–119, Dublin, Ireland. European Association for Machine Translation.

Dirk Goldhahn, Thomas Eckart, and Uwe Quasthoff. 2012. Building large monolingual dictionaries at the leipzig corpora collection: From 100 to 200 languages. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 759–765, Istanbul, Turkey. European Language Resources Association (ELRA).

Jiatao Gu, Changhan Wang, and Junbo Zhao. 2019. Levenshtein transformer. In *Advances in Neural Information Processing Systems*, pages 11179–11189.

Armand Joulin, Edouard Grave, Piotr Bojanowski, Matthijs Douze, Hérve Jégou, and Tomas Mikolov. 2016a. Fasttext.zip: Compressing text classification models. *arXiv preprint arXiv:1612.03651*.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016b. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.

Marcin Junczys-Dowmunt, Roman Grundkiewicz, Tomasz Dwojak, Hieu Hoang, Kenneth Heafield, Tom Neckermann, Frank Seide, Ulrich Germann, Alham Fikri Aji, Nikolay Bogoychev, André F. T. Martins, and Alexandra Birch. 2018. Marian: Fast neural machine translation in C++. In *Proceedings of ACL 2018, System Demonstrations*, pages 116–121, Melbourne, Australia. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Taku Kudo and John Richardson. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium. Association for Computational Linguistics.

Vladimir I Levenshtein. 1966. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710.

Stephen Mayhew, Klinton Bicknell, Chris Brust, Bill McDowell, Will Monroe, and Burr Settles. 2020. Simultaneous translation and paraphrase for language education. In *Proceedings of the ACL Workshop on Neural Generation and Translation (WNGT)*. ACL.

Makoto Morishita, Jun Suzuki, and Masaaki Nagata. 2019. Jparacrawl: A large scale web-based english-japanese parallel corpus. *CoRR*, abs/1911.10668.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. 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)*, pages 86–96, Berlin, Germany. Association for Computational Linguistics.

Milan Straka and Jana Straková. 2017. Tokenizing, POS tagging, lemmatizing and parsing UD 2.0 with UDPipe. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 88–99, Vancouver, Canada. Association for Computational Linguistics.

Jörg Tiedemann. 2012. Parallel data, tools and interfaces in OPUS. In *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, pages 2214–2218, Istanbul, Turkey. European Language Resources Association (ELRA).

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30*, pages 6000–6010, Long Beach, CA, USA. Curran Associates, Inc.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv preprint arXiv:1910.03771*.