The JHU-Microsoft Submission for WMT21 Quality Estimation Shared Task

Shuoyang Ding†* **Marcin Junczys-Dowmunt**‡
Matt Post†‡ **Christian Federmann**‡ **Philipp Koehn**†
† Center for Language and Speech Processing, Johns Hopkins University ‡ Microsoft
{dings, phi}@jhu.edu {marcin.junczysdowmunt, mattpost, chrife}@microsoft.com

Abstract

This paper presents the JHU-Microsoft joint submission for WMT 2021 quality estimation shared task. We only participate in Task 2 (post-editing effort estimation) of the shared task, focusing on the target-side word-level quality estimation. The techniques we experimented with include Levenshtein Transformer training and data augmentation with a combination of forward, backward, round-trip translation, and pseudo post-editing of the MT output. We demonstrate the competitiveness of our system compared to the widely adopted OpenKiwi-XLM baseline. Our system is also the top-ranking system on the MT MCC metric for the English-German language pair.

1 Introduction

In the machine translation (MT) literature, quality estimation (QE) refers to the task of evaluating the translation quality of a system without using a human-generated reference. There are several different granularities as to the way those quality labels or scores are generated. Our participation in the WMT21 quality estimation shared task focuses specifically on the *word-level* quality labels (word-level subtask of Task 2), which are helpful for both human (Lee et al., 2021) and automatic (Lee, 2020a) post-editing of translation outputs. The task asks the participant to predict one binary quality label (`OK`/`BAD`) for each target word and each gap between target words, respectively.[1]

Our approach closely follows our contemporary work (Ding et al., 2021), which focuses on en-de and en-zh language pairs tested in the 2020 version of the shared task. The intuition behind our idea is that translation knowledge is very useful for predicting word-level quality labels of translations. However, usage of machine translation models is limited in the previous work mainly due to (1) the difficulties in using both the left and right context of an MT word to be evaluated; (2) the difficulties in making the word-level reference labels compatible with subword-level models; and (3) the difficulties in enabling translation models to predict gap labels. To resolve these difficulties, we resort to Levenshtein Transformer (LevT, Gu et al., 2019), a model architecture designed for non-autoregressive neural machine translation (NA-NMT). Because of its iterative inference procedure, LevT is capable of performing post-editing on existing translation output even just trained for translation. To further improve the model performance, we also propose to initialize the encoder and decoder of the LevT model with those from a massively pre-trained multilingual NMT model (M2M-100, Fan et al., 2020).

Starting from a LevT translation model, we then perform a two-stage finetuning process to adapt the model from translation prediction to quality label prediction, using automatically-generated pseudo-post-editing triplets and human post-editing triplets respectively. All of our final system submissions are also linear ensembles from several individual models with weights optimized on the development set using the Nelder-Mead method (Nelder and Mead, 1965).

2 Method

Our system building pipeline is consisted of three different stages:

- **Stage 1**: Training LevT for translation
- **Stage 2 (Optional)**: Finetuning LevT on synthetic post-editing triplets
- **Stage 3**: Finetuning LevT on human post-editing triplets

Stage 1: Training LevT for Translation We largely follow the same procedure as Gu et al.

* Shuoyang Ding had a part-time affiliation with Microsoft at the time of this work.

[1] While there is another sub-task for predicting source-side quality labels, we do not participate in that task.

Proceedings of the Sixth Conference on Machine Translation (WMT), pages 904–910
November 10–11, 2021.

(LevT, 2019) to train the LevT translation model, except that we initialize the embedding, the encoder, and decoder of LevT with those from the small M2M-100-small model (418M parameters, Fan et al., 2020) to take advantage of large-scale pretraining. Because of that, we also use the same sentencepiece model and vocabulary as the M2M-100 model.

For to-English language pairs, we explored training multi-source LevT model. According to the results on devtest data, this is shown to be beneficial for the QE task for ro-en, ru-en and ne-en, but not for other language pairs.

Stage 2: Synthetic Finetuning During both finetuning stages, we update the model parameters to minimize the NLL loss of word quality labels and gap quality labels, for the deletion and insertion head, respectively. To obtain training targets for finetuning, we need *translation triplet data*, i.e., the aligned triplet of source, target, and post-edited segments. Human post-edited data naturally provides all three fields of the triplet, but only comes in a limited quantity. To further help the model to generalize, we conduct an extra step of finetuning on synthetic translation triplets, similar to some previous work (Lee, 2020b, *inter alia*). We explored five different methods for data synthesis, namely:

1. `src-mt-tgt`: Take the source side of a parallel corpus (`src`), translate it with a MT model to obtain the MT output (`mt`), and use the target side of the parallel corpus as the pseudo post-edited output (`tgt`).

2. `src-mt1-mt2`: Take a corpus in the source language (`src`) and translate it with two different MT systems that have clear system-level translation quality orderings. Then, take the worse MT output as the MT output in the triplet (`mt1`) and the better as the pseudo post-edited output in the triplet (`mt2`).

3. `bt-rt-tgt`: Take a corpus in the target language (`tgt`) and back-translate it into the source langauge (`bt`), and then translate again to the target language (`rt`). We then use `rt` as the MT output in the triplet and `tgt` as the pseudo post-edited output in the triplet.

4. `src-rt-ft`: Take a parallel corpus and translate its source side and use it as the pseudo post-edited output (`ft`), and round-trip translate its target side (`rt`) as the MT output in the translation triplet.

5. `Multi-view Pseudo Post-Editing (MVPPE)`: Same as Ding et al. (2021), we take a parallel corpus and translate the source side (`src`) with a multilingual translation system (`mt`) as the MT output in the triplet. We then generate the pseudo-post-edited output by ensembling two different *views* of the same model: (1) using the multilingual translation model as a translation model, with `src` as the input; (2) using the multilingual translation model as a paraphrasing model, with `tgt` as the input. The ensemble process is the same as ensembling standard MT models, and we perform beam search on top of the ensemble. Unless otherwise specified, we use the same ensembling weights of $\lambda_t = 2.0$ and $\lambda_p = 1.0$ as Ding et al. (2021).

Stage 3: Human Post-editing Finetuning We follow the same procedure as stage 2, except that we finetune on the human post-edited dataset provided by the shared task organizers for this stage.

Compatibility With Subwords As pointed out before, since LevT predicts edits on a subword-level starting from translation training, we must construct reference tags that are compatible with the subword segmentation done for both the MT and the post-edited output. Specifically, we need to: (1) for inference, convert subword-level tags predicted by the model to word-level tags for evaluation, and (2) for both finetuning stages, build subword-level reference tags. We follow the same heuristic subword-level tag reference construction procedure as Ding et al. (2021), which was shown to be helpful for task performance.

Label Imbalance Like several previous work (Lee, 2020a; Wang et al., 2020; Moura et al., 2020), we also observed that the translation errors are often quite scarce, thus creating a skewed label distribution over the `OK` and `BAD` labels. Since it is critical for the model to reliably predict both classes of labels, we introduce an extra hyperparameter μ in the loss function that allows us to upweight the words that are classified with `BAD` tags in the reference.

$$\mathcal{L} = \mathcal{L}_{OK} + \mu\mathcal{L}_{BAD}$$

Ensemble For each binary label prediction made by the model, the model will give a score $p(\texttt{OK})$,

Language Pair	Data Source	Sentence Pairs
English-German	WMT20 en-de parallel data	44.2M
English-Chinese	shared task en-zh parallel	20.3M
Romanian-English	shared task ro-en parallel	3.09M
Russian-English	shared task ru-en parallel	2.32M
Estonian-English	shared task et-en parallel	880K
Estonian-English	shared task et-en parallel + NewsCrawl 14-17	3.42M
Nepalese-English	shared task ne-en parallel	498K
Pashto-English	WMT20 Parallel Corpus Filtering Task	347K

Table 1: Source and statistics of parallel datasets used in our experiments.

which are translated into binary labels in post-processing. To ensemble predictions from k models $p_1(\texttt{OK}), p_2(\texttt{OK}), \ldots, p_k(\texttt{OK})$, we perform a linear combination of the scores for each label:

$$p_E(\texttt{OK}) = \lambda_1 p_1(\texttt{OK}) + \lambda_2 p_2(\texttt{OK}) + \cdots + \lambda_k p_k(\texttt{OK})$$

To determine the optimal interpolation weights, we optimize towards target-side MCC on the development set. Because the target-side MCC computation is not implemented in a way such that gradient information can be easily obtained, we experimented with two gradient-free optimization method: Powell method (Powell, 1964) and Nelder-Mead method (Nelder and Mead, 1965), both as implemented in `SciPy` (Virtanen et al., 2020). We found that the Nelder-Mead method finds better optimum on the development set while also leading to better performance on the devtest dataset (not involved in optimization). Hence, we use the Nelder-Mead optimizer for all of our final submissions with ensembles. We set the initial points of Nelder-Mead optimization to be the vertices of the standard simplex in the k-dimensional space, with k being the number of the models.

We find that it is critical to build ensembles from models that yield diverse yet high-quality outputs. Specifically, we notice that ensembles from multiple checkpoints of a single experimental run are not helpful. Hence, for each language pair, we select 2-8 models with different training configurations that also have the highest performance to build our final ensemble model for submission.

3 Experiments

3.1 Data Setup

LevT Training We used the same parallel data that was used to train the MT system in the shared task, except for the en-de, et-en, and ps-en language pairs. For en-de language pair, we use the larger parallel data from the WMT20 news translation shared task. For et-en language pair, we experiment with augmenting with the News Crawl Estonian monolingual data from 2014 to 2017, which was inspired by Zhou and Keung (2020). For ps-en language pair, because there is no MT system provided, we take the data from the WMT20 parallel corpus filtering shared task and applied the baseline LASER filtering method. For the multi-source LevT model, we simply concatenate the data from ro-en, ru-en, es-en (w/o monolingual augmentation) and ne-en. The resulting data scale is summarized in Table 1.

Following the setup in Gu et al. (2019), we conduct sequence-level knowledge distillation during training for all language pairs except for ne-en and ps-en[2]. For en-de, the knowledge distillation data is generated by the WMT19 winning submission for that language pair from Facebook (Ng et al., 2019). For en-zh, we train our own en-zh autoregressive model on the parallel data from the WMT17 news translation shared task. For the other language pairs, we use the decoding output from M2M-100-mid (1.2B parameters) model to perform knowledge distillation.

Synthetic Finetuning We always conduct data synthesis based on the same parallel data that was used to train the LevT translation model. For the only language pair (en-de) where we applied the `src-mt1-mt2` synthetic finetuning for shared task submission, we again use the WMT19 Facebook's winning system (Ng et al., 2019) to generate the higher-quality translation `mt2`, and the system provided by the shared task to generate the MT output in the pseudo translation triplet `mt1`. For all other combinations of translation directions, language pairs and MVPPE decoding, we use the M2M-100-

[2]The exception was motivated by the poor quality of the translation we obtained from the M2M-100 model.

Configuration	Stage 2	Stage 3	Target MCC
en-de OpenKiwi	N	default	0.337
en-de bilingual best	src-mt1-mt2	$\mu = 1.0$	0.500
en-de ensemble	N/A	N/A	0.504
en-zh OpenKiwi	N	default	0.421
en-zh bilingual best	mvppe	$\mu = 1.0$	0.459
en-zh ensemble	N/A	N/A	0.466
ro-en OpenKiwi	N	default	0.556
ro-en bilingual best	src-rt-ft	$\mu = 1.0$	0.604
ro-en multilingual best	N	$\mu = 1.0$	0.612
ro-en ensemble	N/A	N/A	0.633
ru-en OpenKiwi	N	default	0.279
ru-en bilingual best	src-rt-ft	$\mu = 3.0$	0.316
ru-en multilingual best	N	$\mu = 3.0$	0.339
ru-en ensemble	N/A	N/A	0.349
et-en OpenKiwi	N	default	0.503
et-en bilingual best	N	$\mu = 3.0$	0.556
et-en bilingual best (w/ aug)	N	$\mu = 3.0$	0.548
et-en multilingual best	N	$\mu = 3.0$	0.533
et-en ensemble	N/A	N/A	0.575
ne-en OpenKiwi	N	default	0.664
ne-en bilingual best	N	$\mu = 3.0$	0.677
ne-en multilingual best	N	$\mu = 3.0$	0.681
ne-en ensemble	N/A	N/A	0.688

Table 2: Target MCC results on `test20` dataset for all language pairs we submitted systems for (except for ps-en which is not included in `test20`). Stage 2 stands for synthetic finetuning (where N stands for not performing this stage). Stage 3 stands for human annotation finetuning. μ stands for the label balancing factor.

	Target MCC	F1-OK	F1-BAD
N	0.489	0.955	0.533
src-mt-ref	0.493	0.955	0.537
src-mt1-mt2	**0.500**	0.956	**0.544**
bt-rt-tgt	0.490	0.956	0.534
src-rt-ft	0.494	0.956	0.538
mvppe	**0.500**	**0.960**	0.540

Table 3: Analysis of different data synthesis methods on en-de language pair. All models here are initialized with M2M-100-small.

mid (1.2B parameters) model.

Human Annotation Finetuning We follow the data split for human post-edited data as determined by the task organizers and use `test20` as the devtest for our system development purposes.

Reference Tag Generation We implemented another TER computation tool[3] to generate the word-level and subword-level tags that we use as the reference for finetuning, but stick to the original reference tags in the test set for evaluation to avoid potential result mismatch.

[3] `https://github.com/marian-nmt/moses-scorers`

3.2 Model Setup

Our LevT-QE model is implemented based on Fairseq (Ott et al., 2019). All of our experiments uses Adam optimizer (Kingma and Ba, 2015) with linear warmup and `inverse-sqrt` scheduler. For stage 1, we use the same hyperparameters as Gu et al. (2019) for LevT translation training, but use a smaller learning rate of 2e-5 to avoid overfitting for all to-English language pairs. For stage 2 and beyond, we stick to the learning rate of 2e-5 and perform early-stopping based on the loss function on the development set. For stage 3, e also experiment with label balancing factor $\mu = 1.0$ and $\mu = 3.0$ for each language pair and pick the one that works the best on devtest data, while for stage 2 we keep $\mu = 1.0$ because early experiments indicate that using $\mu = 3.0$ at this stage is not helpful.

For pre-submission developments, we built OpenKiwi-XLM baselines (Kepler et al., 2019) following their `xlmroberta.yaml` recipe. Keep in mind due to the fact that this baseline model is initialized with a much smaller XLM-Roberta-base model (281M parameters) compared to our M2M-100-small initialization (484M parameters), the performance comparison is not a strict one.

3.3 Devtest Results

Our system development results on `test20` devtest data are shown in Table 2[4]. In all language pairs, our systems can outperform the OpenKiwi baseline based upon the pre-trained XLM-RoBERTa-base encoder. Among these language pairs, the benefit of LevT is most significant on the language pairs with a large amount of available parallel data. Such behavior is expected, because the less parallel data we have, the less knowledge we can extract from the LevT training process. Furthermore, the lack of good quality knowledge distillation data in the low-resource language pairs also expands this performance gap. To our best knowledge, this is also the first attempt to train non-autoregressive translation systems under low-resource settings, and we hope future explorations in this area can enable us to build a better QE system from LevT.

In terms of comparison between multilingual and bilingual models for to-English language pairs, the results are mixed, with the multilingual model per-

[4] Note that the results on en-zh also reflect a crucial bug fix on our TER computation tool that we added after the system submission deadline. Hence the results shown here are from a different system as in the official shared task results. The bug fix should not affect the results of the other language pairs.

Configuration	Stage 2	Stage 3	Target MCC	F1-OK	F1-BAD
ro-en multilingual	N	$\mu = 1.0$	**0.612**	0.949	**0.659**
ro-en multilingual	mvppe	$\mu = 1.0$	0.611	**0.951**	**0.659**
ro-en multilingual	src-mt1-mt2 (Bing mt2)	$\mu = 1.0$	0.585	0.936	0.630
ro-en bilingual (Bing KD)	N	$\mu = 1.0$	0.581	0.949	0.632
ro-en bilingual (Bing KD)	src-mt1-mt2 (Bing mt2)	$\mu = 1.0$	0.568	0.938	0.619
et-en bilingual	N	$\mu = 3.0$	0.548	0.914	0.622
et-en bilingual	mvppe	$\mu = 3.0$	0.544	**0.929**	0.615
et-en bilingual	src-mt1-mt2 (Bing mt2)	$\mu = 3.0$	**0.563**	0.919	**0.634**
et-en bilingual (Bing KD)	N	$\mu = 3.0$	0.557	0.918	0.629
et-en bilingual (Bing KD)	src-mt1-mt2 (Bing mt2)	$\mu = 3.0$	0.559	0.916	0.631

Table 4: Analysis of `src-mt1-mt2` and `mvppe` method on ro-en and et-en language pair.

forming significantly better for ru-en language pair, but significantly worse for et-en language pair. Finally, our Nelder-Mead ensemble further improves the result by a small but steady margin.

3.4 Analysis

Ding et al. (2021) already conducted comprehensive ablation studies for techniques such as the effect of LevT training step, heuristic subword-level reference tag, as well as the effect of various data synthesis methods. In this section, we extend the existing analyses by studying if the synthetic finetuning is still useful with M2M initialization, and if it is universally helpful across different languages. We also examine the effect of label balancing factor μ and take a detailed look at the prediction errors.

Synthetic Finetuning We redo the analysis on en-de synthetic finetuning with the smaller 2M parallel sentence samples from Europarl, as in Ding et al. (2021), but with the updated `test20` test set and models with M2M-100-small initialization. The results largely corroborate the trend in the other paper, showing that `src-mt1-mt2` and `mvppe` being the most helpful two data synthesis methods. We then extend those two most helpful methods to ro-en and et-en, using the up-to-date Bing Translator production model as the stronger MT system (a.k.a. `mt2`) in the `src-mt1-mt2` synthetic data. The result is mixed, with `mvppe` failing to improve performance for both language pairs, and `src-mt1-mt2` only being helpful for et-en language pair. We also trained two extra ro-en and et-en LevT models using the respective Bing Translator models to generate the KD data, which are neither helpful for improving performance on their own nor working better with `src-mt1-mt2` synthetic data.

We notice that the `mvppe` synthetic data seems to significantly improve the F1 score of the `OK` label in general, for which we don't have a good explanation yet.

Configuration	Target MCC	F1-OK	F1-BAD
ro-en $\mu = 1.0$	**0.612**	**0.949**	**0.659**
ro-en $\mu = 3.0$	0.577	0.930	0.619
ru-en $\mu = 1.0$	0.267	**0.960**	0.284
ru-en $\mu = 3.0$	**0.339**	0.943	**0.390**
et-en $\mu = 1.0$	0.478	**0.933**	0.511
et-en $\mu = 3.0$	**0.512**	0.925	**0.587**
ne-en $\mu = 1.0$	0.660	**0.885**	0.774
ne-en $\mu = 3.0$	**0.681**	0.855	**0.788**

Table 5: Analysis of different label balancing factors initialized on to-English language pairs. All results are based on the multilingual model and not performing synthetic finetuning step.

Label Balancing Factor We find the QE task performance to be quite sensitive to the label balancing factor μ, but there is also no universally optimal value for all language pairs. Table 5 shows this behavior for all to-English language pairs. Notice that while for most of the cases μ simply controls a trade-off between the performance of `OK` and `BAD` outputs, there are also cases such as ro-en where a certain choice of μ hurts the performance of both classes. This might be due to a certain label class being particularly hard to fit, thus creating more difficulties with learning when the loss function is designed to skew to this label class.

It should be noted that this label balancing factor does not correlate directly with the ratio of the `OK` vs. `BAD` labels in the training set. For example, to obtain the best performance, ne-en requires $\mu = 3.0$ while en-de requires $\mu = 1.0$, while the `OK` to `BAD` ratio for ne-en (2.14:1) is much less skewed

Lang.	Tgt. MCC	MT MCC	MT BAD (P/R/F1)			MT OK (P/R/F1)			GAP MCC	GAP BAD (P/R/F1)			GAP OK (P/R/F1)		
en-de	0.504	0.503	0.476	0.731	0.576	0.950	0.863	0.904	0.280	0.366	0.238	0.288	0.980	0.989	0.984
en-zh	0.466	0.381	0.467	0.787	0.586	0.879	0.633	0.736	0.146	0.276	0.099	0.145	0.965	0.990	0.977
ro-en	0.612	0.645	0.729	0.709	0.719	0.922	0.929	0.926	0.164	0.411	0.073	0.125	0.973	0.997	0.985
ru-en	0.349	0.329	0.296	0.675	0.411	0.945	0.775	0.852	0.167	0.265	0.123	0.168	0.978	0.991	0.985
et-en	0.575	0.553	0.676	0.681	0.679	0.875	0.873	0.874	0.251	0.426	0.169	0.242	0.967	0.991	0.979
ne-en	0.694	0.434	0.760	0.918	0.832	0.746	0.454	0.564	0.192	0.444	0.098	0.161	0.955	0.994	0.974

Table 6: Detailed evaluation metric breakdown of all submitted ensemble system on `test20` test set.

compare to en-de (10.2:1).

Detailed Error Breakdown We found it hard to develop an intuition for the model performance from the MCC metric. To further understand which label categories our models struggle with the most, we breakdown the target-side metric into a cross product of {`MT`, `GAP`} tags and {`OK`, `BAD`} classes and compute precision, recall and F1-score for each category. The breakdown is shown in Table 6. It can be seen that our model is making the most mistakes with the `GAP BAD` category, while the category with the least mistakes is the `GAP OK` category. Also, note that for MT word tags, the models often seem to suffer more from low precision rather than low recall, while for gaps it is the opposite.

Overall, we see that the highest F1 scores we can achieve for detecting bad MT words or gaps are rarely higher than 0.8, which indicates that there should be ample room for improvement. It would also be interesting to measure the inter-annotator agreement of these word-level quality labels, in order to get a sense of the human performance we should be aiming for.

4 Conclusion

In this paper, we present our WMT21 word-level QE shared task submission based on Levenshtein Transformer training and a two-step finetuning process. We also explore various ways to create synthetic data to build more generalizable systems with limited human annotations. We show that our system outperforms the OpenKiwi+XLM baseline for all language pairs we experimented with. Our official results on the blind test set also demonstrate the competitiveness of our system. We hope that our work can inspire other applications of Levenshtein Transformer beyond the widely studied case of non-autoregressive translation.

References

Shuoyang Ding, Marcin Junczys-Dowmunt, Matt Post, and Philipp Koehn. 2021. Levenshtein training for word-level quality estimation.

Angela Fan, Shruti Bhosale, Holger Schwenk, Zhiyi Ma, Ahmed El-Kishky, Siddharth Goyal, Mandeep Baines, Onur Celebi, Guillaume Wenzek, Vishrav Chaudhary, Naman Goyal, Tom Birch, Vitaliy Liptchinsky, Sergey Edunov, Edouard Grave, Michael Auli, and Armand Joulin. 2020. Beyond english-centric multilingual machine translation. *CoRR*, abs/2010.11125.

Jiatao Gu, Changhan Wang, and Junbo Zhao. 2019. Levenshtein transformer. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 11179–11189.

Fabio Kepler, Jonay Trénous, Marcos Treviso, Miguel Vera, and André F. T. Martins. 2019. OpenKiwi: An open source framework for quality estimation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 117–122, Florence, Italy. 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*.

Dongjun Lee. 2020a. Cross-lingual transformers for neural automatic post-editing. In *Proceedings of the Fifth Conference on Machine Translation*, pages 772–776, Online. Association for Computational Linguistics.

Dongjun Lee. 2020b. Two-phase cross-lingual language model fine-tuning for machine translation quality estimation. In *Proceedings of the Fifth Conference on Machine Translation*, pages 1024–1028, Online. Association for Computational Linguistics.

Dongjun Lee, Junhyeong Ahn, Heesoo Park, and Jaemin Jo. 2021. IntelliCAT: Intelligent machine translation post-editing with quality estimation and translation suggestion. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: System Demonstrations*, pages 11–19, Online. Association for Computational Linguistics.

João Moura, Miguel Vera, Daan van Stigt, Fabio Kepler, and André F. T. Martins. 2020. IST-unbabel participation in the WMT20 quality estimation shared task. In *Proceedings of the Fifth Conference on Machine Translation*, pages 1029–1036, Online. Association for Computational Linguistics.

John A. Nelder and R. Mead. 1965. A simplex method for function minimization. *Comput. J.*, 7(4):308–313.

Nathan Ng, Kyra Yee, Alexei Baevski, Myle Ott, Michael Auli, and Sergey Edunov. 2019. Facebook FAIR's WMT19 news translation task submission. In *Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1)*, pages 314–319, Florence, Italy. Association for Computational Linguistics.

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 the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Demonstrations*, pages 48–53. Association for Computational Linguistics.

M. J. D. Powell. 1964. An efficient method for finding the minimum of a function of several variables without calculating derivatives. *Comput. J.*, 7(2):155–162.

Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. 2020. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272.

Minghan Wang, Hao Yang, Hengchao Shang, Daimeng Wei, Jiaxin Guo, Lizhi Lei, Ying Qin, Shimin Tao, Shiliang Sun, Yimeng Chen, and Liangyou Li. 2020. HW-TSC's participation at WMT 2020 quality estimation shared task. In *Proceedings of the Fifth Conference on Machine Translation*, pages 1056–1061, Online. Association for Computational Linguistics.

Jiawei Zhou and Phillip Keung. 2020. Improving non-autoregressive neural machine translation with monolingual data. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1893–1898, Online. Association for Computational Linguistics.