Unsupervised Feature Selection for Effective Parallel Corpus Filtering

Mikko Aulamo, Ona de Gibert, Sami Virpioja, Jörg Tiedemann
Department of Digital Humanities
University of Helsinki, Helsinki / Finland
{name.surname}@helsinki.fi

Abstract

This work presents an unsupervised method of selecting filters and threshold values for the OpusFilter parallel corpus cleaning toolbox. The method clusters sentence pairs into noisy and clean categories and uses the features of the noisy cluster center as filtering parameters. Our approach utilizes feature importance analysis to disregard filters that do not differentiate between clean and noisy data. A randomly sampled subset of a given corpus is used for filter selection and ineffective filters are not run for the full corpus. We use a set of automatic evaluation metrics to assess the quality of translation models trained with data filtered by our method and data filtered with OpusFilter's default parameters. The trained models cover English-German and English-Ukrainian in both directions. The proposed method outperforms the default parameters in all translation directions for almost all evaluation metrics.

1 Introduction

Neural machine translation (NMT) is dependent on large parallel text corpora. Available training data can often be noisy, especially if the data is retrieved by the common method of extracting bitexts from web crawls (Esplà-Gomis et al., 2019; Schwenk et al., 2021; Bañón et al., 2020). Training NMT on noisy data can be detrimental to the translation models. Ensuring that the training examples are clean sentence pairs leads to better translation quality and more efficient training (Khayrallah and Koehn, 2018). If clean parallel corpora are not readily available, a common practice is to refine a noisy corpus by filtering out low quality training examples. The amount and type of noise varies between different corpora. Selecting the kind of filters that are optimal for cleaning a specific parallel corpus can take a lot of trial and error. Several methods and tools for corpus cleaning have been proposed and developed (Taghipour et al., 2011; Carpuat et al., 2017; Ramírez-Sánchez et al., 2020). OpusFilter (Aulamo et al., 2020) is one such toolkit. It provides a selection of configurable filters, but suffers from the same issue of having to manually choose the filters and their parameters. In this work, we propose an unsupervised method of selecting effective filters and filtering thresholds based on the properties of a given corpus. Our method automatically generates a filtering configuration file which serves as a solid starting point for finding the optimal settings for an OpusFilter corpus cleaning pipeline. We assess the proposed method by comparing the translation quality of models trained with data filtered with default parameters from OpusFilter and data filtered with autogenerated parameters. Our implementation of the filter selection method is available at https://github.com/Helsinki-NLP/OpusFilter.

2 Related work

Corpus cleaning has been a part of training pipelines since the statistical machine translation (SMT) era. Some of the most common and most straightforward methods include sentence length based methods, for example removing too short and too long sentences and sentence pairs where

the ratio of source and target lengths is above a given threshold. The Moses toolkit (Koehn et al., 2007) offers commonly used scripts for this purpose. Taghipour et al. (2011) map sentence pairs into an N-dimensional space and filter out the outliers. Cui et al. (2013) propose a graph-based random walk filtering method which is based on the idea that better sentence pairs lead to better phrase extraction and that good sentence pairs contain more frequent phrase pairs. The Zipporah data cleaning system (Xu and Koehn, 2017) maps sentence pairs into a feature space and uses logistic regression to classify good and bad data. As the features, they use bag-of-word translation scores and n-gram language model scores.

Training data quality has a strong effect on NMT performance. Khayrallah and Koehn (2018) study several types of noise and their impact on translation quality. They report that NMT is less robust against noisy data than SMT. Rikters (2018) points out common problems in parallel corpora that can result in low quality NMT and provides filters to overcome these issues. These problems include mismatch of non-alphabetic characters between source and target segments, wrong language and repeating tokens.

Ramírez-Sánchez et al. (2020) present two tools for more careful corpus cleaning with NMT in mind: Bifixer and Bicleaner. Bifixer is a restorative cleaner; it only removes sentence pairs with either side being empty but otherwise it fixes text-related issues in place. Bifixer corrects character encoding and orthography issues, conducts re-splitting of the sentences and identifies duplicates. Bicleaner consists of filtering rules, language model scoring and a classification part. The filtering rules are predefined, but other steps of Bicleaner require training a language model and a classifier. However, pretrained models are provided for many language pairs.

OpusFilter (Aulamo et al., 2020) is a configurable parallel corpus cleaning toolbox. OpusFilter provides a variety of data selection, text processing, filtering and classification features that can be combined into a reproducible corpus cleaning pipeline. An important step in constructing this pipeline is to choose which filters to use and with what parameters. The filters work by producing a score for a sentence pair and checking whether the score exceeds a threshold value. OpusFilter defines default threshold values for each filter, but there is no guarantee that these values are optimal for a given corpus and language pair.

We propose an unsupervised method to choose filters that are useful in differentiating between clean and noisy sentence pairs and to initialize threshold values based on features extracted from a parallel corpus. The approach consists of clustering sentence pairs into noisy and clean categories and using the features of the noisy cluster center as the threshold values. This method is especially useful in setting initial OpusFilter parameters that are adapted to the characteristics of a given corpus.

3 Method

Our proposed method of selecting relevant filters and useful threshold values for OpusFilter is based on clustering sentence pairs into clean and noisy categories and using the features of the noisy cluster center as our filtering parameters. To select the filters that are actually useful in detecting noisy sentence pairs, we convert the clustering task into a classification task and find the features that affect classification accuracy the most. For clustering, classification and feature importance inspection, we use the `scikit-learn` Python package (Pedregosa et al., 2011).

3.1 Filter scores as features

In order to extract features from a parallel corpus, we select a set of filters and use them to produce scores for sentence pairs with OpusFilter's score function. We conduct this procedure on a randomly sampled subset of 100k sentence pairs from the training corpus in order to keep the configuration generation reasonably fast even for large corpora. In this work, we use the following filter scores as features:

- AlphabetRatioFilter: The proportion of alphabetic characters in the segments.
- CharacterScoreFilter: The proportion of characters in a valid script.
- LanguageIdFilter: A confidence score from cld2 language identifier.[1]
- LengthRatioFilter: The ratio between the source and target segment lengths. We use two versions of this score: one with characters and one with tokens as the length unit.

[1] `https://github.com/CLD2Owners/cld2`

- NonZeroNumeralsFilter: The similarity of numerals in the source and target segments (Vázquez et al., 2019).

- TerminalPunctuationFilter: A penalty score for terminal punctuation co-occurrence in the source and target segments (Vázquez et al., 2019).

These features are chosen as they are inexpensive to produce and easy to interpret, but our approach can be expanded to use any filter that produces scores ranging from noisy to clean.

3.2 Clustering

We train k-means clustering with the filter scores as features and we cluster the sentence pairs into two categories: noisy and clean. We use the k-means++ algorithm for centroid initialization (Arthur and Vassilvitskii, 2007). All feature scores are standardized by removing the mean and scaling to unit variance before clustering. After training the clustering algorithm, we look at the centroids of each cluster to recognize the two categories. The cluster center which has lower mean feature score represents the noisy cluster. For some filters, low values represent clean sentence pairs and in those cases we use the value's additive inverse when calculating the mean. The features of the noisy cluster center are used as the generated filtering threshold parameters.

3.3 Feature importance

Not all features are useful in differentiating between noisy and clean sentence pairs. The k-means clustering algorithm does not directly indicate which of the features are important. In order to determine the feature importance, we convert the unsupervised clustering task into a supervised classification task similarly to Ismaili et al. (2014). We train a random forest classifier with the same features as extracted for clustering, and as the labels we use the categories assigned to each sentence pair by the clustering step.

Once the classifier is trained, we find the important features using permutation feature importance scores which show how much the classification accuracy is affected by shuffling the values of a given feature (Breiman, 2001). In order to determine which features are important enough to keep in the filtering configuration, we compare the importance value of each feature to the mean of all importance values. The importance threshold that each feature has to cross is the mean multiplied by a rejection coefficient. This coefficient is used to lower the threshold in order to accept all features in cases where all importance values are close to the mean. In our preliminary experiments, we found using 0.1 as the coefficient to work in rejecting features that do not differentiate between noisy and clean sentence pairs. The default value for the coefficient is 0.1 but it can be set to other values. Finding the optimal value is not trivial as this would require examining the results of running the filters on full datasets and possibly training MT systems to assess the datasets. Finding a more robust approach for rejecting filters remains for future work.

	Noisy	**Clean**	**Importance**
AlphabetRatio.src	0.74	0.82	0.086
AlphabetRatio.trg	0.76	0.84	0.104
CharacterScore.src	1.0	1.0	0.0
CharacterScore.trg	0.99	1.0	0.010
LanguageID.src	0.94	0.92	0.001
LanguageID.trg	0.91	0.92	0.001
LengthRatio.char	1.18	1.17	0.001
LengthRatio.word	1.21	1.21	0.001
NonZeroNum	0.67	0.99	0.088
TerminalPunctuation	-0.67	-0.05	0.063

Table 1: Feature selection for English-Ukrainian. The table shows the feature values of the noisy and clean cluster centers. The rightmost column shows the importance values determined by the random forest classification task. The mean importance is 0.036 and rejection coefficient is set to 0.1. Thus, the threshold to be considered an important feature is 0.0036. Five of the features are rejected as they do not cross this threshold. Rejected importance values have a grey background.

Table 1 shows an example of feature selection for the English-Ukrainian training set used in our translation experiments in Section 4. Five of the ten features are rejected as they do not cross the importance score threshold. The features that are rejected appear to have similar values in both the noisy and clean cluster centers. On the other hand, the character score on the target side is not rejected despite having values very close to each other in both clusters. This can be explained by the fact that the importance values take into account the whole distribution of feature scores, while the cluster centers only represent the means of each feature.

4 Translation experiments

In order to assess the impact of our data filtering method, we train translation models for English-German (en-de) and English-Ukrainian (en-uk) in

	Default		Autogen				Default		Autogen	
			en-de		en-uk		en-de	en-uk	en-de	en-uk
AlphabetRatio	0.75,	0.75	0.73,	0.76	0.74,	0.76	13.5%	16.2%	10.6%	15.0%
CharacterScore	1,	1	–,	–	–,	0.99	0.1%	14.1%	–	11.1%
LanguageId	0,	0	–,	0.85	–,	–	8.5%	10.6%	8.7%	–
LengthRatio.char		3		–		–	0.0%	0.0%	–	–
LengthRatio.word		3		–		–	0.0%	0.0%	–	–
NonZeroNumeral		0.5		0.60		0.67	7.9%	7.8%	9.6%	11.9%
TerminalPunctuation		-2		-0.66		-0.67	0.8%	0.7%	19.1%	14.9%

Table 2: The left side shows the default thresholds and the generated thresholds for each filter. The default thresholds are the same for both language pairs. AlphabetRatio, CharacterScore and LanguageId filters each have two threshold values: one for the source and one for the target sentence. The right side shows the proportions of data that each filter would remove with these thresholds if ran individually. The hyphens indicate filters that have been rejected by the autogeneration method.

both translation directions. These language pairs are chosen as the latest WMT shared translation task (Kocmi et al., 2022) provides development and test data for them and there is available ParaCrawl data for both language pairs (Esplà-Gomis et al., 2019; Bañón et al., 2020). We train models with three different training datasets: one unfiltered set, one cleaned with the default parameters from OpusFilter, and one cleaned with filters and parameters selected by our proposed configuration generation method. We compare the translation quality of the resulting models with automatic metrics.

4.1 Experiment setting

For our experiments, we use ParaCrawl v9 data, which has been previously shown to contain a good amount of noise (Kreutzer et al., 2022). To conduct basic initial cleaning on our training datasets, we remove duplicates and filter out sentences by length (we remove sentences shorter than 3 words and longer than 100 words). The en-uk training set has 12,605,229 sentence pairs after the initial filtering. For en-de, we take a sample of 30M sentence pairs from the initially filtered set to serve as the training data.

Our translation models, trained using the MarianNMT toolkit (Junczys-Dowmunt et al., 2018), are transformer-base with an encoder and decoder depth of 6. We train SentencePiece (Kudo and Richardson, 2018) unigram tokenizers for each model and restrict the vocabulary size to 32k following Gowda and May (2020). For en-de we choose a shared vocabulary, while for en-uk we choose to have separate vocabularies of 32k for each script. All models are trained until convergence with early-stopping on development data, for which we use Flores-101 (Goyal et al., 2022). Flores-101 is the only development set for en-uk in WMT22 and we aim to create consistent training conditions for all our experiments. Therefore, we use Flores-101 development data for en-de as well. We use 1 single NVIDIA Volta V100 GPU for training.

We train models in both translation directions for each language pair based on three different data filtering methods:

- `baseline`: raw data deduplicated and filtered by length.
- `default`: data filtered with OpusFilter's default parameters.
- `autogen`: data filtered with OpusFilter configuration files produced with the proposed autogeneration method.

4.2 Corpus filtering

We filter the training sets for both language pairs with two different methods: using the default parameters from OpusFilter and using automatically generated parameters. In both methods, we use the filters defined in Section 3.1. Table 2 shows the default thresholds for each filter as well as the thresholds generated by the autogeneration method. Many filtering thresholds are rejected as the configuration generation procedure does not consider them useful for differentiating between noisy and clean sentence pairs. For example, the length ratio score distributions are similar in the noisy and clean clusters for both language pairs and consequently, the length ratio filters are dropped for both language pairs. Language identification scores are not found important for en-uk but for the en-de training set, the threshold for the German side is kept. All character score thresholds are rejected except for the Ukrainian side of the en-uk set.

Table 2 also shows how much data each filter would remove with default and autogenerated parameters if each filter was run individually. The

	BLEU				chrF				COMET			
	en-uk	uk-en	en-de	de-en	en-uk	uk-en	en-de	de-en	en-uk	uk-en	en-de	de-en
Baseline	11.1	21.3	24.6	24.1	35.3	45.8	52.6	49.6	-0.395	-0.177	0.198	0.152
Default	15.8	28.9	*b*24.6	**24.6**	43.4	53.2	*b*52.5	**50.9**	0.027	0.108	*b*0.201	0.202
Autogen	**16.3**	**29.9**	**25.5**	*d***24.6**	**44.2**	**54.4**	**53.7**	*d*50.8	**0.065**	**0.164**	**0.230**	*d***0.212**

Table 3: Results of the translation experiments. When the results from default parameters or autogenerated parameters are not significantly different from the baseline results, we prefix them with *b*. When the results from autogenerated parameters are not significantly different from the default parameter results, we prefix them with *d*.

proportion of sentence pairs removed by the four length ratio filters with default thresholds ranges from none at all to 0.0005%. This supports the hypothesis that length ratio values are not useful for finding noisy data in these training sets. Similarly, the character score filter with default parameters removes only 0.1% of the en-de set and the filter is not present in the generated configuration. On the other hand, the language identification score for the en-uk set does not follow this trend: the default thresholds filter out a substantial portion of the data, 10.6%, but it is still rejected by the autogeneration method.

In total, filtering with default values keeps 22,586,611 (75.3%) sentence pairs for the en-de set and 8,069,599 (64.0%) for the en-uk set. In turn, after filtering with the autogenerated threshold parameters, the dataset size for en-de is 19,417,755 (64.7%) and for en-uk 8,316,491 (66.0%) sentence pairs. The en-de training sets have 19,031,231 overlapping sentence pairs which is 84.3% of the default set and 98.0% of the autogeneration set. For en-uk, the number of overlapping sentence pairs is 7,280,959 which is 90.2% of the default set and 87.5% of the autogeneration set.

4.3 Results

The trained translation models are evaluated with three evaluation metrics: BLEU (Papineni et al., 2002), chrF (Popović, 2015) and COMET (Rei et al., 2020). We use SacreBLEU (Post, 2018) to calculate BLEU and chrF. COMET is computed with the `unbabel-comet` Python package[2] using evaluation model wmt20-comet-da. Additionally, we conduct significance testing by using paired bootstrap resampling (Koehn, 2004) to compare the filtered training sets to the baseline, and to compare the default and autogeneration methods to each other. Results are shown in Table 3 for the WMT22 general test sets (Kocmi et al., 2022).

Autogeneration performs better than the baseline for all metrics and language pairs. The performance gains are especially noticeable for the en-uk and uk-en translation pairs. Default filtering scores are higher than the baseline in all translation directions except en-de where the scores are not significantly different from the baseline by any metric. Autogeneration outperforms default filtering in all language pairs except de-en for which there are no significant performance differences between the two approaches.

These results suggest that the proposed method is able to improve the translation quality of models trained on parallel corpora that are filtered by extracting and clustering corpus-specific features. Additionally, our method makes the corpus filtering phase more efficient. We select the filters and their thresholds based on a 100k sentence pair sample of a much larger corpus. This allows us to avoid unnecessarily running filters that do not remove noisy sentence pairs on the whole corpus. In our experiments, running the filters with default parameters took 1h3m12s for en-de and 31m21s for en-uk. Using the generated configurations, the filtering times were 47m4s (25.5% faster) for en-de and 18m35s (40.7% faster) for en-uk. Generating the filtering parameters takes one to two minutes. The filters used in this work are quite inexpensive and fast to run but our method can be easily expanded to more demanding cleaning.

5 Conclusion

We propose an unsupervised method for selecting filters and filtering thresholds for OpusFilter. We evaluate our method in translation tasks where we train models on data filtered with the default parameters of OpusFilter and another set of models trained on data filtered with generated filtering configuration files. The autogeneration method outperforms the default parameters in almost all cases. Additionally, our method makes corpus filtering more efficient as we only run useful filters with appropriate parameters on the full training set.

In future work, we will evaluate our method in a

[2] `https://github.com/Unbabel/COMET`

larger variety of corpus cleaning scenarios to confirm our findings. One point of interest is to test the method for corpora with different proportions of noisy data. We will also conduct tests in low-resource language settings. Additionally, we will evaluate the effects of expanding our approach by integrating a larger range of different filters. In order to improve the autogeneration method, more careful analysis of the feature selection process will be performed, for example manual evaluation of sentence pairs in noisy and clean categories in order to assess the clustering accuracy. We will also explore using statistical inference (e.g. Welch's t-test) for finding effective filters as an alternative for the feature importance analysis. Relying on statistical significance could be a more robust approach for discarding filters than the current rejection coefficient method.

Acknowledgements

This work was supported by the HPLT project which has received funding from the European Union's Horizon Europe research and innovation programme under grant agreement No 101070350. The contents of this publication are the sole responsibility of its authors and do not necessarily reflect the opinion of the European Union.

This work was also supported by the FoTran project, funded by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme under grant agreement No 771113.

References

Arthur, David and Sergei Vassilvitskii. 2007. K-means++: The advantages of careful seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '07, page 1027–1035, USA. Society for Industrial and Applied Mathematics.

Aulamo, Mikko, Sami Virpioja, and Jörg Tiedemann. 2020. OpusFilter: A configurable parallel corpus filtering toolbox. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 150–156, Online, July. Association for Computational Linguistics.

Bañón, Marta, Pinzhen Chen, Barry Haddow, Kenneth Heafield, Hieu Hoang, Miquel Esplà-Gomis, Mikel L Forcada, Amir Kamran, Faheem Kirefu, Philipp Koehn, et al. 2020. Paracrawl: Web-scale acquisition of parallel corpora. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4555–4567.

Breiman, Leo. 2001. Random forests. *Machine learning*, 45:5–32.

Carpuat, Marine, Yogarshi Vyas, and Xing Niu. 2017. Detecting cross-lingual semantic divergence for neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*, pages 69–79, Vancouver, August. Association for Computational Linguistics.

Cui, Lei, Dongdong Zhang, Shujie Liu, Mu Li, and Ming Zhou. 2013. Bilingual data cleaning for SMT using graph-based random walk. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 340–345, Sofia, Bulgaria, August. Association for Computational Linguistics.

Esplà-Gomis, Miquel, Mikel L 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: Translator, Project and User Tracks*, pages 118–119.

Gowda, Thamme and Jonathan May. 2020. Finding the optimal vocabulary size for neural machine translation. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3955–3964, Online, November. Association for Computational Linguistics.

Goyal, Naman, Cynthia Gao, Vishrav Chaudhary, Peng-Jen Chen, Guillaume Wenzek, Da Ju, Sanjana Krishnan, Marc'Aurelio Ranzato, Francisco Guzmán, and Angela Fan. 2022. The Flores-101 Evaluation Benchmark for Low-Resource and Multilingual Machine Translation. *Transactions of the Association for Computational Linguistics*, 10:522–538, 05.

Ismaili, Oumaima Alaoui, Vincent Lemaire, and Antoine Cornuéjols. 2014. A supervised methodology to measure the variables contribution to a clustering. In *Neural Information Processing: 21st International Conference, ICONIP 2014, Kuching, Malaysia, November 3-6, 2014. Proceedings, Part I 21*, pages 159–166. Springer.

Junczys-Dowmunt, Marcin, 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, July. Association for Computational Linguistics.

Khayrallah, Huda and Philipp Koehn. 2018. On the impact of various types of noise on neural machine translation. In *Proceedings of the 2nd Workshop on*

Neural Machine Translation and Generation, pages 74–83, Melbourne, Australia, July. Association for Computational Linguistics.

Kocmi, Tom, Rachel Bawden, Ondřej Bojar, Anton Dvorkovich, Christian Federmann, Mark Fishel, Thamme Gowda, Yvette Graham, Roman Grundkiewicz, Barry Haddow, Rebecca Knowles, Philipp Koehn, Christof Monz, Makoto Morishita, Masaaki Nagata, Toshiaki Nakazawa, Michal Novák, Martin Popel, and Maja Popović. 2022. Findings of the 2022 conference on machine translation (WMT22). In *Proceedings of the Seventh Conference on Machine Translation (WMT)*, pages 1–45, Abu Dhabi, United Arab Emirates (Hybrid), December. Association for Computational Linguistics.

Koehn, Philipp, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, ACL '07, page 177–180, USA. Association for Computational Linguistics.

Koehn, Philipp. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 388–395, Barcelona, Spain, July. Association for Computational Linguistics.

Kreutzer, Julia, Isaac Caswell, Lisa Wang, Ahsan Wahab, Daan van Esch, Nasanbayar Ulzii-Orshikh, Allahsera Tapo, Nishant Subramani, Artem Sokolov, Claytone Sikasote, Monang Setyawan, Supheakmungkol Sarin, Sokhar Samb, Benoît Sagot, Clara Rivera, Annette Rios, Isabel Papadimitriou, Salomey Osei, Pedro Ortiz Suarez, Iroro Orife, Kelechi Ogueji, Andre Niyongabo Rubungo, Toan Q. Nguyen, Mathias Müller, André Müller, Shamsuddeen Hassan Muhammad, Nanda Muhammad, Ayanda Mnyakeni, Jamshidbek Mirzakhalov, Tapiwanashe Matangira, Colin Leong, Nze Lawson, Sneha Kudugunta, Yacine Jernite, Mathias Jenny, Orhan Firat, Bonaventure F. P. Dossou, Sakhile Dlamini, Nisansa de Silva, Sakine Çabuk Ballı, Stella Biderman, Alessia Battisti, Ahmed Baruwa, Ankur Bapna, Pallavi Baljekar, Israel Abebe Azime, Ayodele Awokoya, Duygu Ataman, Orevaoghene Ahia, Oghenefego Ahia, Sweta Agrawal, and Mofetoluwa Adeyemi. 2022. Quality at a Glance: An Audit of Web-Crawled Multilingual Datasets. *Transactions of the Association for Computational Linguistics*, 10:50–72, 01.

Kudo, Taku 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, November. Association for Computational Linguistics.

Papineni, Kishore, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA, July. Association for Computational Linguistics.

Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Popović, Maja. 2015. chrF: character n-gram F-score for automatic MT evaluation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 392–395, Lisbon, Portugal, September. Association for Computational Linguistics.

Post, Matt. 2018. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Belgium, Brussels, October. Association for Computational Linguistics.

Ramírez-Sánchez, Gema, Jaume Zaragoza-Bernabeu, Marta Bañón, and Sergio Ortiz-Rojas. 2020. Bifixer and Bicleaner: two open-source tools to clean your parallel data. In *Proceedings of the 22nd Annual Conference of the European Association for Machine Translation*, pages 291–298, Lisboa, Portugal, November. European Association for Machine Translation.

Rei, Ricardo, Craig Stewart, Ana C Farinha, and Alon Lavie. 2020. COMET: A neural framework for MT evaluation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2685–2702, Online, November. Association for Computational Linguistics.

Rikters, Matīss. 2018. Impact of corpora quality on neural machine translation. In *Human Language Technologies–The Baltic Perspective*, pages 126–133. IOS Press.

Schwenk, Holger, Guillaume Wenzek, Sergey Edunov, Edouard Grave, Armand Joulin, and Angela Fan. 2021. CCMatrix: Mining billions of high-quality parallel sentences on the web. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 6490–6500, Online, August. Association for Computational Linguistics.

Taghipour, Kaveh, Shahram Khadivi, and Jia Xu. 2011. Parallel corpus refinement as an outlier detection algorithm. In *Proceedings of Machine Translation*

Summit XIII: Papers, Xiamen, China, September 19-23.

Vázquez, Raúl, Umut Sulubacak, and Jörg Tiedemann. 2019. The University of Helsinki submission to the WMT19 parallel corpus filtering task. In *Proceedings of the Fourth Conference on Machine Translation (Volume 3: Shared Task Papers, Day 2)*, pages 294–300, Florence, Italy, August. Association for Computational Linguistics.

Xu, Hainan and Philipp Koehn. 2017. Zipporah: a fast and scalable data cleaning system for noisy web-crawled parallel corpora. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2945–2950.