The University of Helsinki Submission to the WMT19 Parallel Corpus Filtering Task

Raúl Vázquez, Umut Sulubacak and **Jörg Tiedemann**

University of Helsinki
{name.surname}@helsinki.fi

Abstract

This paper describes the University of Helsinki Language Technology group's participation in the WMT 2019 parallel corpus filtering task. Our scores were produced using a two-step strategy. First, we individually applied a series of filters to remove the *'bad'* quality sentences. Then, we produced scores for each sentence by weighting these features with a classification model. This methodology allowed us to build a simple and reliable system that is easily adaptable to other language pairs.

1 Introduction

Data-driven methodologies define the state of the art in a wide variety of language processing tasks. The availability of well-formed, clean data varies from language to language, and finding such data in sufficient amounts can prove challenging for some of the lower-resourced languages. In particular, the increasingly common neural machine translation systems are highly sensitive to the quality as well as the quantity of training data (Khayrallah and Koehn, 2018), which creates an impediment to achieving good-quality translations in a low-resource scenario.

The web is a massive resource for text data in a wide array of languages. However, it is costly to manually extract high-quality parallel samples from the web, and automatically-crawled datasets such as the ParaCrawl Corpus[1] are typically quite noisy. Designing automatic methods to select high-quality aligned samples from noisy parallel corpora can therefore make crawling the web a more viable option for compiling useful training data.

To emphasize this untapped potential, Koehn et al. (2018) proposed the Shared Task on Parallel Corpus Filtering as part of WMT in 2018. We participated in this year's task with three sets of quality scores. Each score is a different aggregation of a shared set of features, with each feature representing a local quality estimate focusing on a different aspect. Section 2 contains a brief discussion of this year's shared task. We present our scoring system in Section 3, discussing the filters we used for feature extraction in Section 3.2, and the aggregate scorers in Section 3.3. Finally, we report our contrastive results in Section 4.

2 Task Description

This year, the corpus filtering task organizers decided to pose the problem under more challenging conditions by focusing on low-resource scenarios, as opposed to previous year German–English (Koehn et al., 2018). In particular, two parallel corpora are to be scored for filtering: Nepali–English and Sinhala–English. The task for each participating team is to provide a quality score for each sentence pair in either or both of the corpora. The scores do not have to be meaningful, except that higher scores indicate better quality. The computed scores are then evaluated under four scenarios: training SMT and NMT systems, on samples of 5 million and 1 million words each, where the samples are obtained from the corresponding corpus using the quality scores.

Participants are provided with raw corpora to score, which were crawled using the ParaCrawl pipeline, and consist of 40.6 million (English) words for Nepali–English, and 59.6 million for Sinhala–English. Additionally, some parallel and monolingual corpora were provided for each language pair. We used the parallel datasets to train some of our scoring systems[2]. Some descriptive

[1]ParaCrawl can be downloaded from https://paracrawl.eu/

[2]En–Si: OpenSubtitles and GNOME/KDE/Ubuntu; En–Ne: Bible (two translations), Global Voices, Penn Treebank, GNOME/KDE/Ubuntu, and Nepali Dictionary.

corpus	lang. pair	sent. pairs	EN words
ParaCrawl	EN–NE	2.2M	40.6M
additional	EN–NE	543K	2.9M
ParaCrawl	EN–SI	3.4M	45.5M
additional	EN–SI	647K	3.7M

Table 1: Statistics on the ParaCrawl data and the used parallel data. Only English word counts reported.

statistics of the data we have used can be found in Table 1.

3 Scoring system

We first independently applied a series of filters to the data and computed relevant numerical features with them. We have previously corroborated the filters' effectiveness, since we have used them to clean the noisy datasets provided for this year's news translation task at WMT with satisfactory results. Then, we selected a cut-off value for each filter and trained a classifier over the features to compute a global score for each sentence pair, which we used to rank them.

3.1 Cleaning up the clean training data

Some of our filters require clean data for training. We observed that the provided parallel data still contained quite a lot of noise, and therefore, we applied some additional heuristic filters to clean it further. In particular, we used the following heuristics to remove pairs with characteristics that indicate likely problems in the data:

- Removing all sentence pairs with a length ratio above 3 between the source and the target.
- Removing pairs with very long sentences containing more than 100 words.
- Removing sentences with extremely long words, *i.e.* excluding all sentence pairs with words of 40 or more characters.
- Removing sentence pairs that include HTML or XML tags.
- Removing sentence pairs that include characters outside of the decoding table of Devanagari (for Nepalese) and Sinhala characters besides punctuation and whitespace.
- Removing sentence pairs that include Devanagari or Sinhala characters in English.

The procedure above discarded around 23% of the data for Nepali–English, and we kept around 440k parallel sentences from the original data. For Sinhala–English, we removed about 19% of the data and kept 522k sentence pairs for training.

3.2 Filters

Word alignment. Our first filter applies statistical word alignment models to rank sentence pairs. Word alignment models implement a straightforward way of estimating the likelihood of parallel sentences. In particular, IBM-style alignment models estimate the probability $p(f|a,e)$ of a foreign sentence f given an "emitted" sentence e and an alignment a between them.

We used *eflomal*[3] (Östling and Tiedemann, 2016) for word-level alignment, as it provides significant benefits. First, it is an efficient algorithm based on Gibbs sampling, as opposed to the slower expectation maximization methods commonly used for training. This method is thus able to train and align large quantities of data in a small amount of time. Second, this software allows us to load model priors, a feature we use to initialize the aligner with previously stored model parameters. This is handy for our filtering needs, as we can now train a model on clean parallel data and apply that model to estimate alignment probabilities of noisy data sets.

For obtaining model priors, we use the cleaned training data described above, tokenized with the generic tokenizer from the Moses toolkit (Koehn et al., 2007). We cut all words at 10 characters to improve statistics and training efficiency. With this, we train for both language pairs a Bayesian HMM alignment model with fertilities in both directions, and estimate the model priors from the symmetrized alignment. We then use those priors to run the alignment of the noisy datasets using only a single iteration of the final model to avoid a strong influence of the noisy data on alignment parameters. As it is intractable to estimate a fully normalized conditional probability of a sentence pair under the given higher-level word alignment model, *eflomal* estimates a score based on the maximum unnormalized log-probability of links in the last sampling iteration. In practice, this seems to work well, and we take that value to rank sentence pairs by their alignment quality.

[3]Software available from https://github.com/robertostling/eflomal

Language model filter. The second filter applies language models for source and target languages. In our approach, we opt for a combination of source and target language models, and focus on the comparison between scores coming from both models. The idea with this filter is to prefer sentence pairs for which the cross-entropy with the clean monolingual language models is low for both languages, and that the absolute difference between the cross-entropy of aligned sentences is low as well. The intuition is that both models should be roughly similarly surprised when observing sentences that are translations of each other. In order to make the values comparable, we trained our language models on parallel data sets.

As both training data sets are rather small, and as we aim for an efficient and cheap filter, we chose a traditional n-gram language model. To further avoid data sparseness and to improve comparability between source and target languages, we also base our language models on BPE-segmented texts (Sennrich et al., 2016) using a BPE model trained on the cleaned parallel data set with 37k merge operations per language. *VariKN*[4] (Siivola et al., 2007b,a) is the perfect toolkit for the purpose of estimating n-gram language models with subword units. It implements Kneser-Ney growing and revised Kneser-Ney pruning methods with the support of n-grams of varying size and the estimation of word likelihoods from text segmented into subword units. In our case, we set the maximum n-gram size to 20, and a pruning threshold of 0.002. Finally, we compute cross-entropies for each sentence in the noisy parallel training data, and store five values as potential features for filtering: the source and target language cross-entropy, $H(S, q_s)$ and $H(T, q_t)$, as well ad the average, max and absolute difference between them, i.e., $avg(H(S, q_s), H(T, q_t))$, $abs(H(S, q_s) - (T, q_t))$ and $max(H(S, q_s), H(T, q_t))$.

Language identifiers. A third filter applies off-the-shelf language identifiers. In particular, we use the Python interface of the Compact Language Detector[5] version 2 (CLD2) from the Google Chrome project, and the widely used `langid.py` package (Lui and Baldwin, 2012), to classify each sentence in the datasets.

We generate 4 features from these classifiers. For each language, we use the reliability score by CLD2 only if the predicted language was correct, and zero otherwise; and we use the detection probability of `langid.py` only if the language was classified correctly, and zero otherwise.

Character scores. Another simple filter computes the proportion of Devanagari, Sinhala and Latin–1 characters in Nepali, Sinhala and English sentences, respectively. For this computation, we ignore all whitespace and punctuation characters using common Unicode character classes.

Terminal punctuation. This heuristic filter generates a penalty score with respect to the co-occurrence of terminal punctuation marks ('.', '…', '?', '!') in a pair of sentences. In order to have a finer granularity than $\{0, 1\}$, we penalize both asymmetry (to catch many-to-one alignments) and large numbers of terminal punctuation (to cover very long sentences, URLs and code). For a given source and target sentence pair, we initialize a score as the absolute difference between source and target terminal punctuation counts. Then, we increment this score by the number of terminal punctuation beyond the first occurrence in both source and target sentences.

The intended effect is for the ideal sentence pair to contain either no terminal punctuation or a single terminal punctuation on either side ($score = 0$). In practice, many sentences are very far from the ideal ($score \gg 100$), and it is counter-intuitive to use a larger positive value to represent a higher penalty. To address both problems, we finally make the following update:

$$score = -log(score + 1)$$

Non-zero numerals. This filter assumes that numerals used to represent quantities and dates will be typically translated in the same format, and penalizes sentence pairs where numerals do not have a one-to-one correspondence or do not occur in the same sequence.

Sinhala uses the same Western Arabic numerals used in the Latin alphabet. Nepali uses Devanagari numerals, following the same decimal system as Western Arabic numerals. This filter takes that into account, and first converts those to digits between $[0, 9]$. After numeric normalization, the filter extracts sequences of numerals from each

[4]VariKN is available from https://vsiivola.github.io/variKN/

[5]The Python implementation of CLD2 is available at https://github.com/aboSamoor/pycld2

pair of sentences, preserving their relative order. Considering that a leading zero can be omitted in some numeric sequences such as in dates and numbered lists, the digit '0' is ignored. Finally, the score is calculated as a similarity measure between the extracted sequences in the range $[0, 1]$ using `SequenceMatcher.ratio()` from Python's `difflib`.

Clean-corpus filter Finally, we use the well-proven *clean-corpus-n* script from Moses to produce a binary feature augmented by a feature that marks sentences including HTML or XML tags.

All in all, we obtain 15 potential features from these filters. However, some of them are to be considered redundant and the information they provide is already encoded in some other variable. For instance, using the reliability score produced by CLD2 together with the prediction probability from `langid.py` would not provide crucial additional information to a model. Table 2 summarizes the filters we used to train our scoring models.

№	Feature	Definition
1	word-align	$\sim p(f\|a, e)$
2 3	lang-model	$H(S, q_s)$ $H(T, q_t)$
4 5	lang-id	src reliability score tgt reliability score
6 7	char-score	English chars % Ne/Si chars %
8	term-punct	penalty for asymmetric & excessive term. punct.
9	non-zero	similarity between non-zero digit seq.
10	clean-corpus	1, if kept 0, otherwise

Table 2: List of features extracted from the filters.

3.3 Scorers

We trained a logistic regression classifier and a random forest classifier to score each sentence pair using the features presented in Section 3.2. We trained three independent binary classifiers under the following settings:

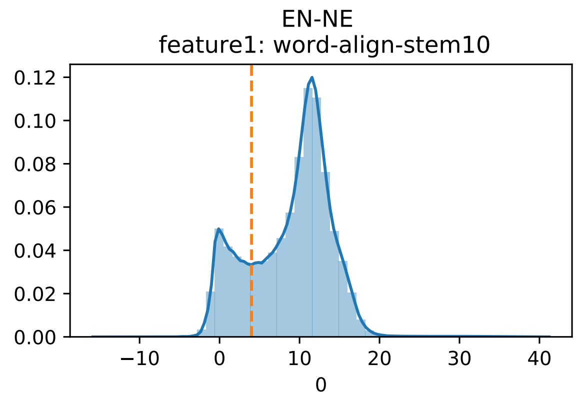

Figure 1: Distribution and cutoff value of feature 1 (word alignment) in the English–Nepali ParaCrawl corpus.

1. Applying all filters to the additional parallel corpora, and using filtered data as positive examples, and filtered-out data as negative examples.

2. Applying all filters to the corresponding ParaCrawl corpus, and using filtered data as positive examples, and a sample of 600k filtered-out examples as negative examples.

3. Applying all filters to both the ParaCrawl and the additional parallel corpora, and using these as positive examples, and a sample of 1M filtered-out examples as negative examples.

	lang.	RF		LR	
	pair	AIC	BIC	AIC	BIC
PC	en-ne	17.8	-1.0e+7	-1.3	-2.5e+6
PC+BIC	en-ne	16.8	-1.1e+7	-0.9	-2.9e+6
PC	en-si	15.4	-9.4e+6	-1.5	-2.3e+6
PC+BIC	en-si	15.6	-1.1e+7	-1.4	-2.9e+6

Table 3: AIC and BIC obtained with random forest (RF) and logistic regression (LR) models. Comparison between the first chosen thresholds for ParaCrawl (PC) data and the model that optimizes the information criteria (PC+BIC).

For each filter under the first two scenarios, we adjusted thresholds based on score distributions, attempting to keep a balance between having restrictive thresholds that limited the amount of positive examples, and having lax thresholds

data	langpair	word-align	lang-model (*src*)	lang-model (*tgt*)	lang-id (*src*)	lang-id (*tgt*)	char-score (*%En*)	char-score (*%Ne/Si*)	term-punct	non-zero	clean-corpus
additional clean	ne-en	1	5	0	—	0	0	0	–2	0.5	0
ParaCrawl	ne-en	4	10	9	0	0	0	0	–2	0.5	0
ParaCrawl bestBIC	ne-en	—	—	—	0	0	0	0	–2	0.5	0
additional clean	ne-si	2	6	5	0	0	0	0	–1.5	0.5	0
ParaCrawl	ne-si	3	10	10	0	0	0	0	–1	0.5	0
ParaCrawl bestBIC	ne-si	—	10	10	0	0	0	0	–2	0.5	0

Table 4: Selected threshold value for each feature.

that classified many low-quality examples as positive. In some cases the score distributions were clearly bi-modal, making it easy to determine cutoff values (*e.g.* see Figure 1); while in other cases, we had to opt for a more empirical approach. For this reason, we have a second model that optimizes the Akaike Information Criterion (AIC) (Akaike, 1974) and the Bayes Information Criterion (BIC) (Schwarz et al., 1978) under scenario 2. This model was chosen from among 7 models trained with different reasonable combinations of the features. In Table 3, we compare the information criteria for both models. Finally, under the third scenario we chose to combine the data using the defined cutoff values from the previous two to include a significant amount of examples from both data sets.

Table 4 summarizes the threshold values used for each feature. After applying the filters, we kept 240k sentences (≈ 11% of the total) from the ParaCrawl EN–NE, 230k sentences (≈ 7%) from ParaCrawl EN–SI; 239k (≈ 44%) from the additional clean EN–NE data, and 231k (≈ 36%) from the additional clean EN–SI data. This means that, when combining them for scenario 3, we get 419k sentences (≈ 15%) for EN–NE, and 537k for EN–SI (≈ 14%). In order to avoid overfitting to the negative examples in scenarios 2 and 3, which vastly outnumber the positive ones, we performed stratified sampling of the negative examples where we selected 600K and 1M negative examples, respectively. We then randomly split the data into train (70%) and test (30%) sets.

4 Results

We report the accuracy on the test set achieved by the aforementioned models in Table 5. We do not report the accuracy of the random forest classifiers since they are all ≈ 99.99%. This is likely because the algorithm "cuts" through the variables in a similar way to how we chose the threshold values. For the same reason, they are unsuitable for the scoring task at hand. The output produced is a sharp classification that does not help rank the sentences. In contrast, the logarithmic regression model softens the output probabilities, emulating the creation of a composite index when used in combination without the threshold selection procedure.

	lang. pair	accuracy
additional	en-ne	78.21%
ParaCrawl	en-ne	96.09%
ParaCrawl+BIC	en-ne	96.46%
All data	en-ne	86.55%
additional	en-si	78.82%
ParaCrawl	en-si	95.26%
ParaCrawl+BIC	en-si	95.26%
All data	en-si	91.14%

Table 5: Accuracy values on the test data for the trained logistic regression models. Additional refers to the additional parallel clean data provided, ParaCrawl+BIC to the model that optimized the BIC, and All data to scenario 3.

In a final step, we also combined the score given by the regression model with two heuristic features that we deemed to be important for the ranking. One of them is the character score that we introduced earlier, which computes the proportion of language-specific characters in the string ignoring punctuation and whitespace. With this factor, we heavily penalize sentence pairs that contain large portions of foreign text. The second factor is based on the heuristics that translated sentences should exhibit similar lengths in terms of characters. This feature is proven to be efficient for common sentence alignment algorithms, and hence, we add the character length ratio as another factor in the final score. For simplicity, we just multiply the three values without any extra weights to obtain the final ranking score. The system that applies those additional factors is marked with *char-length* in Table 6 with the SMT results on the development test set.

model	NE–EN	SI–EN
baseline	4.22	4.77
logreg	4.91	5.06
+char-length	4.82	5.32
bestBIC	4.63	4.91

Table 6: BLEU scores using SMT on 5 million sampled training examples. The *baseline* refers to the Zipporah model reported by the organizers of the shared task.

We only ran experiments with the provided SMT model. We do not present results from the NMT model, since we encountered complications while running the pre-processing script in the provided development pack for the task. We believe it might be due to character encoding and noise in the data. However, we did not further investigate the source of said problem. The SMT scores are listed in Table 6. We can see that we indeed outperform the baseline model, but the scores are still so low that we deem the resulting models to be essentially useless. The performance for our three attempts are rather similar, with the plain logistic regression model having a slight advantage, and a small improvement provided by the char-length filter for the case of Sinhala–English. For that reason, we selected that model as our final submission, with the plain logreg model as a contrastive run to be evaluated.

By inspecting the provided data we draw the conclusion that the low quality of the final MT models is mainly due to the overall poor quality of the data, rather than solely an issue of the scoring algorithms. The final results of the shared task suggest that it has not been possible to squeeze much more out of the data. As seen in Table 7, submissions for this year demonstrate a narrow range of scores, and our primary submissions rank above average despite their poor performance.

	model	1M	5M	10M
EN–NE	best	4.21	4.62	4.74
	UHel (1)	3.19	3.87	4.31
	average	3.03 ± 1.22	3.60 ± 1.12	3.96 ± 0.89
	UHel (2)	1.29	2.05	3.83
EN–SI	best	4.27	4.76	4.94
	UHel (1)	3.26	3.84	4.12
	average	3.00 ± 1.13	3.43 ± 1.09	3.92 ± 0.87
	UHel (2)	2.28	3.24	3.96

Table 7: An overview of the relative performance (in BLEU scores) of our (1) primary and (2) contrastive SMT models trained on 1, 5, and 10 million samples. The *best* and *average* rows represent the highest score and the mean ± standard deviation among this year's submissions, respectively.

5 Conclusions

In this paper, we presented our rescoring system for the WMT 2019 Shared Task on Parallel Corpus Filtering. Our system is based on contrastive scoring models using features extracted from different kinds of data-driven and heuristic filters. We used these models to assign quality scores to each sentence pair. This methodology allowed us to build a simple and reliable system that is easily adapted to other language pairs. The machine translation quality indeed improves, however, BLEU scores remain particularly low. This raises questions about the general quality of the data. More detailed analyses of the data sets seem to be necessary to draw further conclusions.

Acknowledgments

This work is part of the FoTran project, funded by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement № 771113). as well as the MeMAD project, funded by the European Union's Horizon 2020 Research and Innovation Programme (grant № 780069).

References

Hirotugu Akaike. 1974. A new look at the statistical model identification. In *Selected Papers of Hirotugu Akaike*, pages 215–222. Springer.

Huda Khayrallah 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. Association for Computational Linguistics.

Philipp Koehn, 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, pages 177–180, Stroudsburg, PA, USA. Association for Computational Linguistics.

Philipp Koehn, Huda Khayrallah, Kenneth Heafield, and Mikel L. Forcada. 2018. Findings of the WMT 2018 shared task on parallel corpus filtering. In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 726–739, Belgium, Brussels. Association for Computational Linguistics.

Marco Lui and Timothy Baldwin. 2012. langid.py: An off-the-shelf language identification tool. In *Proceedings of the ACL 2012 System Demonstrations*, pages 25–30, Jeju Island, Korea. Association for Computational Linguistics.

Robert Östling and Jörg Tiedemann. 2016. Efficient word alignment with Markov Chain Monte Carlo. *Prague Bulletin of Mathematical Linguistics*, 106:125–146.

Gideon Schwarz et al. 1978. Estimating the dimension of a model. *The annals of statistics*, 6(2):461–464.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. 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)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Vesa Siivola, Mathias Creutz, and Mikko Kurimo. 2007a. Morfessor and VariKN machine learning tools for speech and language technology. In *8th Annual Conference of the International Speech Communication Association (Interspeech 2007), Antwerp, Belgium, August 27-31, 2007*, pages 1549–1552. ISCA.

Vesa Siivola, Teemu Hirsimäki, and Sami Virpioja. 2007b. On growing and pruning Kneser-Ney smoothed n-gram models. *IEEE Trans. Audio, Speech & Language Processing*, 15(5):1617–1624.