Simultaneous translation and paraphrase generation

Abhinav Gupta
Language Technology Institute
agupta6@andrew.cmu.edu

Anmol Jagetia
Language Technology Institute
ajagetia@andrew.cmu.edu

Ashutosh Baghel
Heinz College of Information Systems
abaghel@andrew.cmu.edu

Abstract

Our work explores the idea of generating multiple paraphrases in the target language as required in the STAPLES-2020 shared task by Mayhew et al. [2020]. In addition to that, we explore ways of improving the quality of the paraphrases by augmenting data, and we propose approaches of weighted loss function that takes into account the likelihood of different paraphrases at the training time. We achieve an improvement of 1 BLEU score over our baseline for English-Hungarian language pair. Further, we analyzed Duolingo's strategy for calculating the BLEU score and proposed an alternate strategy that provides a more holistic view.

1 Introduction

Existing approaches on translation optimize on a single translated sentence in the target language. A good reference covering all the basic techniques is shown by Neubig [2017]. This is also evident from the various data-sets available for the machine translation tasks as shown by Tiedemann [2012]. Some popular common data-sets are the data-sets used by the annual WMT conferences organized by Barrault et al. [2019] which are often parallel translations from the European parliament proceedings or news articles. Further, the state of the art approaches for NMT task for high resource language pairs like English - German or English - Romanian too optimize on generating one single output vs generating n-best outputs or diversifying the n-best outputs.

Given the nature of the languages, we know that for most common language pairs, there can be several ways of expressing the same idea. This idea is reinforced by the Duolingo as an introduction to the shared task. Since Duolingo is aimed at teaching beginners a new language through the use of translation, there are many ways through which any sentences an be represented in the target language. Some of these are more likely to be used by native speakers than the others, but there are usually more than one correct possibility. The difference could range from difference in synonyms, difference in ordering in certain languages, to use of different parts-of-speech (for example, use of different pronouns). Languages are also known to evolve and change with time. The second challenge is of scoring these sentences by the order of importance as demonstrated by native speakers. Duolingo position helps then collect this data and provide a dataset which tries to capture this relationship by the use of weights assigned to targets by their use as demonstrated by native speakers.

Due to this nature of the language, evaluating an Neural Machine Translation model, with a single gold standard as is, is not the most effective way of evaluation because the generated translation could be good but differ significantly from the gold standard and these outcomes greatly depend on the target language. Generating multiple paraphrases of high quality in the target language can be helpful in a number of situations. One of them could be the Duolingo's use-case where the idea is to evaluate a user's answers with a list of weighted translations, to give a qualitative score of the translation while also exposing learners to different ways native speakers can say the same sentence

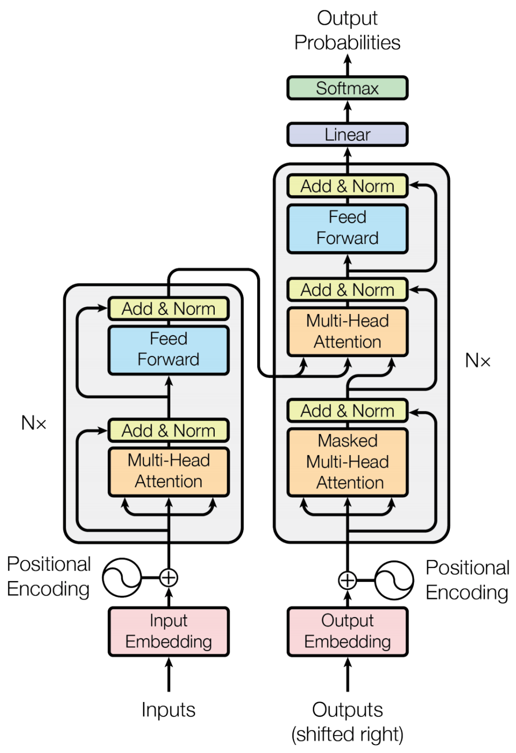

Figure 1: Architecture of transformer model Vaswani et al. [2017]

to curate a better learning experience and exposing the student to more real-world examples. Another use-case could be to learn paraphrases during translation for a better language model. Paraphrases can be used to improve the quality of statistical machine translation by addressing the problem of coverage and introducing a degree of generalization into the models. Sennrich et al. [2015a] shows how back translation can be useful in training better, more generalized language models. Paraphrases can also be used for a similar task.

Our work uses the Duolingo dataset provided by Mayhew et al. [2020] that provides several translation for a given input sentence along with a weighted score based on the user response rates. We explore the idea of modifying the loss function in present state of the art models for Neural Machine Translation leads to better learning of paraphrased output in target language.

2 Baseline

Our baseline models use Fairseq's implementation of Transformer based translation models. We use the fairseq library by Ott et al. [2019] for our experiments. Our baseline model architecture uses the transformer model by Vaswani et al. [2017].

2.1 Model Architecture of the baseline

Recurrent networks are a natural choice for temporal or sequential data like language. The primary disadvantage of such recurrent networks is their inability to learn long range dependencies. A signal at the beginning of the sentence needs to pass through the length of the sequence, which can cause the signal to diminish and vanish. A solution to this problem is to use the convolution network where each signal can be computed to the encoded representation in $O(logn)$ computations, where n is the sequence length.

The novel factor of the transformer model is that it solely applies attention on the input embeddings. This allows the model to learn dependencies between the words in a constant number of operations.

The model can be divided into an encoder module and a decoder module.

2.2 Word Embeddings

In the original work by [Vaswani et al., 2017], the authors had defined 512-dimensional vectors for representing input tokens. The dense representations of the word are learned during training and the model is not provided with pre-trained embeddings.

The fact that embeddings are learned during the training is advantageous as finding good quality representations of the target language may be a difficult task in some situations.

2.3 Positional Encoding

Unlike recurrent networks, attention operation does not take into consideration the position of the words in the sentence. This is overcome by adding positional embedding to each word in the sentence, which adds to the embedding the information of the relative position of the word in the sentence.

While there are several ways to make positional embeddings, in this work [Vaswani et al., 2017] authors have used sinusoidal functions, where each position of the sentence is encoded by a sine wave. This leads to unique frequency of the sine wave for each word.

2.4 Multi-head attention

The intuition behind multi-head attention is to calculate the self-attention scores. For each token in the sequence, the self-attention operation determines its correlation with every other token in the sequence. In the original paper, the authors have defined Query, Key, and Value vectors.

For the encoder layer, the three vectors are obtained by multiplying the input sequence with query weights, key weights, and value weights. Each query is multiplied with every key, which results in the self-attention score. This is then multiplied with the value vector to suppress the irrelevant words and retain the words that need emphasis.

The decoder layer differs slightly, where each query is the encoded input sequence and each key is a token from encoded target.

2.5 Encoder

In the original paper of transformer models by [Vaswani et al., 2017], the encoder module consists of six identical layers stacked. Each layer of the encoder module consists of two sub-modules.

The first sub-layer consists of the self-attention mechanism that is applied to the input. Here, the input is a fixed length vector which is defined as the maximum length of the sequence that can be input to the model.

If the length of the input sequence is lesser than the max-length, the input is padded with padding tokens. The input sequence is also appended with a $< eos >$ token to indicate the length of each sequence.

Before the operation of self-attention is applied, a mask for the input sequence is created. The idea behind this mask is to avoid computing attention model for the padding tokens. Next, dropout is applied to the output of self-attention and it is added with a residual connection.

The second sub-layer takes in input from the first sub-layer. This layer consists of a feedforward network which is similar to a convolution operation with stride one and window size of one. Each token of dimension 512 is passed through a MLP network with 1024 hidden cells. The weights for this layer are shared for all token in the sequence.

Similar to the first sub-layer, on the output of the second sub-layer dropout is applied. Then residual connection is added and finally, normalization is applied.

2.6 Decoder

The decoder module, too, consists of six identical layers with each layer consisting of three sub-layers.

Input to the decoder module is the target sequence, which is pre-processed like the encoder input. Additionally, a mask is generated for the input target sequence that is a lower triangular matrix where each row represents a time step. And positions till the current time step are unmasked.

All three sub-layers are followed by a dropout layer, a residual connection, and the final output is normalized.

The first sub-layer of the decoder applies multi-head attention to the input of the decoder.

Next sub-layer then applies multi-head attention with query as the output of the first sub-layer which encodes the target sequence. The key and value are the encoded source sequence. The intuition behind this self-attention layer is to understand at each time step, what part of the source sequence needs to be focused.

The last sub-layer is a feedforward layer, again, similar to the feedforward layer of the endcoder.

The output of the decoder layer is passed through a linear layer that has a dimension equal to that of the target vocab. Finally, softmax is applied on the output of the linear layer that gives us the probability of each token in the target vocab.

3 Dataset

We used the dataset provided by Duolingo's 2020 shared task (STAPLE) from [Mayhew et al., 2020]. The dataset provides a prompt in English and multiple translations in the target language. The number of paraphrases in the target language can vary from 2-3 to as high as 27.A complete list of the available data is shown in 1. Along with the paraphrase translation, the dataset also provides weights to each translation, which corresponds to the user response rates. A higher weight indicates that the particular translation is seem more frequently amongst the native speakers and is judged to be a good proxy for being a "better translation".

3.1 Dataset Preprocessing

We preprocess the Duolingo dataset to constructed a parallel dataset in the style of many machine translation tasks for training, which consists of tuples where the first element is in English, and the second element is in the target language. We also have a third field that reflects the weights to capture the popularity of a particular paraphrased translation. Duolingo provides data for five languages, Hungarian, Portuguese, Korean, Vietnamese, Japanese.

Duolingo provides a pre-split version of the dataset comprising of a train-set, a blind dev-set and a blind test-set which doesn't provide the target weights, so we split the original train-set into into train/dev/test splits in the 80/10/10 ratio.

The traditional translation dataset contains single mapping in the target language for each sentence in the source language. We preprocess our dataset by adding multiple rows for each sentence in the source language, each with a different corresponding paraphrased translation in the target language along with the weights to replicate a similar setup to the traditional machine translation datasets.

For our implementation, we use a TabularDataset data-structure implementation from the torchtext library that returns a single string, containing the source sentence, target sentence, and the corresponding weight, all separated by a delimiter. Training and Validation data is randomly shuffled. Though we also tried experimenting with more complex data-loader that does not randomly shuffle the entire dataset, instead just shuffles the mini-batches, we didn't say a major improvement and it wasn't optimal from a standpoint of training since shuffling by clubbing sentences of similar lengths together is shown to be better for effective training. The results of contextual shuffling shown by related literature is also mixed. Kocmi and Bojar [2017] show very marginal improvement in their experiments with a much larger WMT dataset whereas Press [2019] show significant improvement using partial shuffling.

For test and validation sets, prompt was used only once, with the corresponding top translation. All splits were then cleaned and tokenized using the **moses-tokenizer** by Koehn et al. [2007], and byte-pair encoding were generated using the **subword-nmt** library based on the paper by Sennrich et al. [2015b]. Further, we used the fairseq-preprocessing step on each language pairs.

Table 1: Dataset with number of source and target sentences from Duolingo's STAPLE Task

Language Pair	Data Size (prompts / total accepted)
en_hu	4000 / 251442
en_pt	4000 / 526466
en_ja	2500 / 855941
en_ko	2500 / 700410
en_vi	3500 / 194720

Table 2: Additional pairs from Tatoeba Corpus

Language Pair	Data-pairs from Tatoeba Corpus
en_hu	85851
en_pt	157975
en_ja	48955
en_ko	3621
en_vi	3369

3.2 Data Augmentation

Since we started with very little data, we augmented our training data with more data from the Tatoeba Corpus. We particularly like this dataset since the style of sentences is quite similar to the Duolingo dataset and there are lots of paraphrases as well. The number of language pairs that we added to the training data were as shown by the table 2.

We observed that the number of examples available for Korean and Vietnamese are too few. We decide to run training with augmented data with an unweighted baseline since this dataset doesn't have weights. We saw very marginal improvements over the unweighted in terms of F-1 score for Hungarian and for Portuguese the model trained with the augmented data performs worse than the model trained with the just the Duolingo dataset.

We believe that might be the case due to the lack of direct correlation in the domain of the data-sets. We were unable to use the augmented data with our new loss function due to the lack of weights available for paraphrases.

3.3 Baseline Results

To achieve our baseline results, we run the transformer model from the fairseq library with label smoothed cross entropy criterion and Adam optimizer by Kingma and Ba [2014] with Adam-betas '(0.9, 0.98)'. The baseline results obtained are close to those mentioned of the Duolingo SharedTask webpage.

The baseline results were obtained through running fairseq-train and fairseq-generate on the staple dataset. The reported Weighted Macro F1 scores resemble those reported on the Duolingo shared-task website.The results are shown in the table 6.

Baseline scores are interesting as they show a great disparity between the different language pairs, trained on the same model architecture with identical model parameters. We also find the high scores of Vietnamese surprising, compared to Portuguese and Hungarian, intuitively we estimated European languages to have more diverse parallel data available, compared to south-east asian languages. This also goes against our belief that sharing Latin script with English would have provided an advantage, especially during sub-word tokenization.

3.4 Shortcoming of the Metrics

The baseline experiment primarily uses two metrics to evaluate the system. Firstly, BLEU scores are used to determine if the model has started to overfit during training. However, BLEU scores for the validation is calculated using only the top translation result and the top gold paraphrase. We analyzed

Table 3: **SacreBLEU** scores for our experiments.

SacreBLEU	**Portuguese**	**Hungarian**	**Vietnamese**
Top Reference	20.8	8.9	39.8
Top 2 References	25.2	11.7	46.7
Top 3 References	29.1	13.9	51.8
Top 4 References	31.6	17.4	54.6

Table 4: **Levenshtein Distance** calculations were done using the top translation compared to each of the gold paraphrases. This table shows that many times the top translation was a closer to one of the gold paraphrases. Further, the top translation was infact a perfect string match with one of the gold paraphrases (other than the top weighted paraphrase) with a edit-distance of 0.

Levenshtein Distance	**Portuguese**	**Hungarian**	**Vietnamese**
Top translation had min distance from another reference	89.22 %	90.81 %	77.81 %
Top translation had distance = 0 from another reference	27.06 %	17.85 %	38.33 %

in Table 3 that this technique can undermine the translation system. It was observed that instead of just taking the top -weighted gold reference, if we instead take top-n weighted gold paraphrases (n = 1,2,3,4) and calculate the corpus-bleu, we achieve a much better evaluation of our system. It may well be the case that the best translation may not be the best rated reference. Through this approach, we do not penalize our system for predicting the second-best reference, and so on.

We confirmed this hypothesis in Table 4. We looked at how many times our model had predicted a translation that was closer to a reference which was not top-rated. Levenshtein distance was used to compare each of the reference paraphrases with our translation. We noted that, for all language-pairs, our prediction was closer to one of the references which was not the top reference.

We also noted that perfect string match between our translation and another reference was frequent. This significantly went up when we looked at translations with small edit-distances (less than 5) from one of the references.

This brings forth the challenge with using a Weighted Macro F1-score as the primary metric. Many of the translations were less than 5 characters away from being a perfect match. However, F-1 score does not award anything to this prediction, and further negatively impacts the weighted precision as a missed paraphrase.

4 Loss Function

Usually, the cost function with a negative log-likelihood loss over a batch is defined as

$$J(\theta) = E_{x,y\sim P(x,y)} \log P(y|x,\theta) \tag{1}$$

or

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} L(x^{(i)}, y^{(i)}) \tag{2}$$

Where, θ are the model parameters. x, y are input features and labels, respectively.

Our hypothesis is that for our dataset that has multiple labels y, for a given data x, this definition of our loss function 4 would converge the model to mean of all paraphrases.

To test our hypothesis, our goal is to analyze if by tweaking the loss function to include paraphrase weights and improve the quality of n-best translations.

By training the model on paraphrases, the hope is to expose the model to the diversity which helps finding n-best examples. And by tweaking the loss function, the hope is to understand the likelihood of that diversity.

4.1 Class Imbalance

One of the typical applications of weighted loss in our knowledge is for dealing with class imbalance. In such a situation, weight is assigned to each class. Authors Cui et al. [2019] propose an effective number of samples E_{n_i} for each class i. Effective number of samples can be imagined as the actual volume that will be covered by n samples where the total volume N is represented by total samples as shown by the work of Jain [2019].

$$CB(p,y) = \frac{1}{E_{n_y}} L(p,y) = \frac{1-\beta}{1-\beta^{n_y}} L(p,y) \tag{3}$$

While this approach defines a cost function as a weighted sum of some loss function, the primary difference here is that our predictions are sequences and not classes. Defining the effective number of samples for a sequence is not clear.

4.2 Weighted Loss

Our proposed solution for including the weights for each paraphrase is to define our cost function $J(\theta)$ as,

$$J(\theta) = E_{x,y \sim P(x,y)} P(y|x) \log P(y|x, \theta) \tag{4}$$

The difference here is that the negative log likelihood loss is now weighted. The user response rates provided by Duolingo are normalized, and therefore can be inferred as $P(y|x)$, i.e. the probability of a translation in the target set, given an input sentence.

We implement the loss function as given by the function equation:

$$L(y, \hat{y}) = \frac{1}{m} \sum_{i=0}^{m} w_i * NLL(y, \hat{y}) \tag{5}$$

The weight w_i can be defined as a function of the provided $P(y|x)$ in couple of ways, depending on how we reason the weights for the loss function.

- The first line of thinking was that the paraphrases with higher probability should be encouraged more. By defining $w_i = P(y|x)$, paraphrases with less probability would contribute lesser in the the overall loss function. Therefore the model would be biased towards learning paraphrases with higher probability.

  One of the major drawback of this approach in our experience was that a significant percentage of paraphrases in the training dataset have a very low probability (of the order $1e^{-2}$). As a result the loss of mini-batch reduces by an order of $1e^{-2}$ and we observed a very low loss at each epoch of the order $1e^{-3}$.

  But still this approach got us our best performance, Table 6.1 compares the different approaches. We achieved the best BLEU of **11.169** which is 1 BLEU point more than the best result we achieved with FairSeq's baseline.
- Another approach could be to define $w_i = (1 - p(y|x))$. This would reduce the contribution of the higher probability paraphrases in the loss function, meanwhile, in relative terms, increase the importance of lower probability paraphrases.

  Our hypothesis is that this would encourage model to also focus on paraphrases that have low probability.

5 Metric

NMT tasks are usually measured with BLEU introduced by Papineni et al. [2002]. Although papers like Wieting et al. [2019] show that BLEU is not the most effective method to evaluate translation, we still use BLEU due to it's ubiquitousness and simplicity. There is also clear evidence that there is a strong correlation between BLEU scores and human judgments of the same MT output. We do a more descriptive analysis in the results section. We also do analysis of the scoring mechanism used the Duolingo for the shared task and highlight some of the limitations of the way scoring is proposed for the original competition. We propose a more fair and effective method of scoring when compared to the way competition is scoring results by evaluating levenshtein distance to measure lexical similarity between the gold standard outputs and the paraphrases. A detailed analysis of the Levenshtein distance is shown in the table 4.

5.1 F-1 Score

Duolingo's Shared Task uses Weighted Macro F_1 score as the main scoring metric. This evaluates how well the translation system can return all human-curated acceptable translations, weighted on the likelihood of a human learner's response for each translation.

Below is an example prompt with assigned likelihood for each translation in Vietnamese:

prompt123 | is he going to remember the street?
anh y s nh đng ch? | 0.75
anh y s nh con đng ch? | 0.20
anh ta s nh đng ch? | 0.05

For each prompt s, weighted F_1 score is calculated. Precision is calculated in an unweighted fashion, and only recall is weighted. Specifically, weighted true positives (WTP) is calculated using generated translations that were present in the response set. Weighted false negatives (WFN) are calculated for all responses that were missing in the generated translations.

$$WTP_s = \sum_{t \in TP_s} weight(t) \tag{6}$$

$$WFN_s = \sum_{t \in FN_s} weight(t) \tag{7}$$

$$WeightedRecall(s) = \frac{WTP_s}{WTP_s + WFN_s} \tag{8}$$

Then, F_1 scores are averaged over all prompts S.

$$WeightedMacroF_1 = \sum_{s \in S} \frac{WeightedF_1(s)}{|S|} \tag{9}$$

5.2 BLEU

BLEU scores get greatly affected by the number of references passed for BLEU calculation. Duolingo demonstrated using the top reference. However, we noticed that many top predictions actually mapped well to references other than the top reference. We recommend using top 4 references when calculating BLEU. Our results for BLEU are shown in the table 3.

6 Results

6.1 Compare-MT

We use the compare-MT library by Neubig et al. [2019] to evaluate the performance of the various models that we trained.

Given the limited time frame we couldn't complete all the experiments we had planned. But we strongly believe our proposed approach of weighted loss function has more potential.

Table 5: **Weighted Loss Function Results** Compared to the Fairseq's best checkpoint, the model with modified loss function got an improvement of 1 BLEU point, when evaluated with Compare NMT library

Baseline	**Weighted Loss Function** $w_i = 1 - P(y\|x)$	**Weighted Loss Function** $w_i = P(y\|x)$
10.1693	6.8890	**11.1639**

Table 6: Baseline Results

Metric (%)	**Portuguese**	**Hungarian**	**Korean**	**Vietnamese**
Precision	25.80	15.97	13.85	28.40
Recall	1.98	2.47	0.54	4.99
Weighted Recall	9.87	9.89	2.66	25.73
Micro F1	3.67	4.29	1.04	8.49
Macro F1	7.72	8.76	3.00	16.72
Weighted Micro F1	14.28	12.22	4.47	27.00
Weighted Macro F1	11.63	10.30	3.93	22.59

References

L. Barrault, O. Bojar, M. R. Costa-jussÃ, C. Federmann, M. Fishel, Y. Graham, B. Haddow, M. Huck, P. Koehn, S. Malmasi, C. Monz, M. MÃller, S. Pal, M. Post, and M. Zampieri. 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, August 2019. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/W19-5301`.

Y. Cui, M. Jia, T.-Y. Lin, Y. Song, and S. Belongie. Class-balanced loss based on effective number of samples. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9268–9277, 2019.

V. Jain. Handling class imbalanced data using a loss specifically made for it, 2019. URL `https://towardsdatascience.com/handling-class-imbalanced-data-using-a-loss-specifically-made-for-it-6e58fd65ffab`.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization, 2014.

T. Kocmi and O. Bojar. Curriculum learning and minibatch bucketing in neural machine translation, 2017.

P. Koehn, H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, C. Dyer, O. Bojar, A. Constantin, and E. Herbst. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180, Prague, Czech Republic, June 2007. Association for Computational Linguistics. URL `https://www.aclweb.org/anthology/P07-2045`.

S. Mayhew, K. Bicknell, C. Brust, B. McDowell, W. Monroe, and B. Settles. Simultaneous translation and paraphrase for language education. In *Proceedings of the ACL Workshop on Neural Generation and Translation (WNGT)*. ACL, 2020.

G. Neubig. Neural machine translation and sequence-to-sequence models: A tutorial, 2017.

G. Neubig, Z. Dou, J. Hu, P. Michel, D. Pruthi, X. Wang, and J. Wieting. compare-mt: A tool for holistic comparison of language generation systems. *CoRR*, abs/1903.07926, 2019. URL `http://arxiv.org/abs/1903.07926`.

M. Ott, S. Edunov, A. Baevski, A. Fan, S. Gross, N. Ng, D. Grangier, and M. Auli. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*, 2019.

K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. Bleu: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, page 311–318, USA, 2002. Association for Computational Linguistics. doi: 10.3115/1073083.1073135. URL `https://doi.org/10.3115/1073083.1073135`.

O. Press. Partially shuffling the training data to improve language models, 2019.

R. Sennrich, B. Haddow, and A. Birch. Improving neural machine translation models with monolingual data, 2015a.

R. Sennrich, B. Haddow, and A. Birch. Neural machine translation of rare words with subword units, 2015b.

J. Tiedemann. Parallel data, tools and interfaces in opus. In N. C. C. Chair), K. Choukri, T. Declerck, M. U. Dogan, B. Maegaard, J. Mariani, J. Odijk, and S. Piperidis, editors, *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey, may 2012. European Language Resources Association (ELRA). ISBN 978-2-9517408-7-7.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

J. Wieting, T. Berg-Kirkpatrick, K. Gimpel, and G. Neubig. Beyond bleu:training neural machine translation with semantic similarity. *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019. doi: 10.18653/v1/p19-1427. URL `http://dx.doi.org/10.18653/v1/p19-1427`.