Training and Inference Methods for High-Coverage Neural Machine Translation

Michael Yang **Yixin Liu** **Rahul Mayuranath**

Language Technologies Institute, Carnegie Mellon University, Pittsburgh PA, U.S.A.
{myang2,yixinl2,rmayuran}@cs.cmu.edu

Abstract

In this paper, we introduce a system built for the Duolingo Simultaneous Translation And Paraphrase for Language Education (STAPLE) shared task at the 4th Workshop on Neural Generation and Translation (WNGT 2020). We participated in the English-to-Japanese track with a Transformer model pretrained on the JParaCrawl corpus and fine-tuned in two steps on the JESC corpus and then the (smaller) Duolingo training corpus. First, during training, we find it is essential to deliberately expose the model to higher-quality translations more often during training for optimal translation performance. For inference, encouraging a small amount of diversity with Diverse Beam Search to improve translation coverage yielded marginal improvement over regular Beam Search. Finally, using an auxiliary filtering model to filter out unlikely candidates from Beam Search improves performance further. We achieve a weighted F1 score of 27.56% on our own test set, outperforming the STAPLE AWS translations baseline score of 4.31%.

1 Introduction

Currently, state of the art machine translation systems generally produce a single output translation. However, human evaluators of translation tasks will often accept multiple translations as correct. We introduce a neural machine translation (NMT) system that generates high-coverage translation sets for a single given prompt in the source language.

Our system was prepared for the English-to-Japanese track[1] of the Duolingo Simultaneous Translation And Paraphrase for Language Education (STAPLE) shared task (Mayhew et al., 2020) at the 4th Workshop on Neural Generation and Translation (WNGT 2020). The shared task datasets consist of English prompts and a weighted set of target language translations for each prompt. The task requires systems to produce translation sets for given English prompts that are evaluated on weighted F1 score, defined in Appendix A. We have made our code publicly available.[2]

We experimented with models trained and fine-tuned on the provided Duolingo English-Japanese prompt-translation data (Mayhew et al., 2020), the JParaCrawl web-crawled corpus (Morishita et al., 2019), as well as the Japanese-English Subtitle Corpus (JESC) (Pryzant et al., 2018). The sizes of each dataset are summarized in Table 1.

Our system uses a Transformer-based (Vaswani et al., 2017) NMT model and we began with weights pretrained on the large JParaCrawl corpus (Morishita et al., 2019). Section 4 describes in detail how the model was pretrained. Our system's NMT model was then obtained by fine-tuning first on the Japanese-English Subtitle Corpus (JESC) (Pryzant et al., 2018) before further fine-tuning on the Duolingo training set (Mayhew et al., 2020). We outline these datasets in more detail in Section 2.

Given the small size of the Duolingo data, this multi-step fine-tuning helped the model generalize and outperformed single-step fine-tuning and no fine-tuning. High-coverage translation bitext data is not easy to mine or create, so we expect that in other settings, the size of such available training data will also be small. Therefore, it is very likely that adopting a multi-step fine-tuning method may be advantageous more generally. The fine-tuning procedure is described in Section 6.

Outputting the entire beam of candidates from 150-width Beam Search, scored on per token log likelihood, this two-step fine-tuned system produced the translations that we submitted to the

[1] There were five tracks of target languages in total, the others being Hungarian, Korean, Portuguese and Vietnamese.

[2] Our code can be found at https://github.com/michaelzyang/high-coverage-translation.

Dataset	English Sentences	Japanese Sentences
JParaCrawl	8,763,995	8,763,995
JESC	2,801,388	2,801,388
Duolingo	2,500	855,940

Table 1: Number of sentence-pairs in the datasets (Duolingo pairs have a one-to-many correspondence)

shared task leaderboard. It achieved 25.69 % weighted F1 score on the shared task blind development set and 26.0% on the blind test set.

After the leaderboard closed, we conducted further experiments and discovered several notable optimizations.

The most effective optimization was using the ground truth weights that indicate variations in translation quality during training. We find that it is essential to deliberately expose the model to higher-quality translations more often during training. Otherwise, overexposure to low-quality translations harms the model's translation performance.

Secondly, Diverse Beam Search with a very small penalty outperformed Beam Search. However, too much diversity begins to introduce minor semantic shifts that deviate from correct translations.

We also explored introducing an auxiliary filtering model for post-processing candidates. Our proposed filtering model is able to refine the candidates generated by the NMT model, which improved the system's performance with respect to the weighted F1 score.

We share our results in Section 7. Our best result was a weighted F1 score of 27.56% on our own test set of 200 prompts randomly selected from the training data.

2 Corpora

2.1 Duolingo High-coverage Translations

Duolingo provided training, development and test sets (Mayhew et al., 2020). However, the development and test datasets were 'blind' and did not contain ground truth translations, so we did not use these for training or development.

The training set consists of 2,500 English prompts, each of which are paired with a variable number of Japanese translations (Table 1). Duolingo provides weights for each translation, which can be interpreted as a quality score. For our experiments, we randomly split the the 2,500 prompts into 2,100, 200 and 200-prompt training, development and test sets respectively. For the shared task submission, we retrained a model over all 2,500 prompts with our best hyperparameters.

2.2 JParaCrawl

As our base model, we use a model pre-trained on the JParaCrawl corpus (Morishita et al., 2019). This corpus contains over 8.7 million sentence pairs which were crawled from the web and then automatically aligned, similar to European corpora in the ParaCrawl project[3]. Though noisy due to an imperfect alignment method, this is currently the largest publicly-available English-Japanese bitext corpus.

2.3 Japanese-English Subtitle Corpus

The Japanese-English Subtitle Corpus (JESC) (Pryzant et al., 2018), is a large parallel training corpus that contains 2.8 million pairs of TV and movie subtitles. With an average length of 8, the corpus mostly consists of short sentences, which is similar to the data present in the Duolingo training corpus. Even though JESC contains some noise, it captures sufficient information that is useful for downstream NMT tasks.

3 Related work

Machine Translation Machine translation (MT) involves finding a target sentence $\mathbf{y} = y_1, ...y_m$ with the maximum probability conditioned on a source sentence $\mathbf{x} = x_1, ...x_n$, i.e $\underset{\mathbf{y}}{\text{argmax}}\ P(\mathbf{y}|\mathbf{x})$. There are various neural approaches to tackle machine translation. These include utilizing recurrent neural networks (Cho et al., 2014b), convolutional neural networks (Kalchbrenner et al., 2016), attention-based models (Luong et al., 2014; Bahdanau et al., 2015) and transformer networks (Vaswani et al., 2017). Sequence to sequence models deal with the task of mapping an input sequence to an output sequence. These were first introduced by Sutskever et al. (2014) and typically use an RNN

[3] https://paracrawl.eu/

based encoder-decoder architecture, where the encoder outputs a fixed length representation of the input which is fed into the decoder to get a target translation. RNN and LSTM based approaches struggle to handle long sequences and long-range dependencies since the encoder network is tasked with encoding all relevant information in a fixed-length hidden state vector. Bahdanau et al. (2015) overcome this by utilizing attention, an alignment model that can attend to important parts of the input during translation. Luong et al. (2014) used the attention mechanism to great effect, observing gains of 5.0 BLEU over non-attention based techniques for NMT.

The Transformer Architecture For our experiments, we used the the Transformer architecture proposed by Vaswani et al. (2017). It is a self-attention based model that produces superior results for machine translation tasks compared to CNN and LSTM based models. By stacking multiple layers of multi-head self-attention blocks, they demonstrate that the attention mechanism by itself is very powerful for sequence encoding and decoding. Recently, Transformer-based models that are pre-trained on large-scale datasets have produced superior performance on various Natural Language Processing (NLP) tasks (Rajpurkar et al., 2016; Talmor and Berant, 2019; Mayhew et al., 2019). In Section 4 we further describe the transformer architecture and our pretraining procedure.

Domain Adaptation Domain adaptation involves making use of out-of-domain data in situations where high quality in-domain data are scarce. This fine tuning approach has been shown to be effective for NMT (Luong and Manning, 2015; Sennrich et al., 2015; Freitag and Al-Onaizan, 2016). Morishita et al. (2019) show that pre-training with JParaCrawl vastly improves in-domain performance for English-Japanese translations. We make use of these ideas in our multi-step fine-tuning experiments.

Inference with Beam Search Beam Search is an approximate search algorithm used for finding high likelihood sequences from sequential decoders. At every time step, the top k outputs are traversed and the rest are discarded. A common issue with beam search is that it generates similar outputs that only differ by a few words or minor morphological variations (Li and Jurafsky, 2016). Vijayakumar et al. (2016) propose Diverse Beam Search, a method that reduces redundancy during decoding in NMT models to generate a wider range of candidate outputs. This is achieved by splitting the beam width into evenly-sized groups and adding a penalty term for the presence of similar candidates across groups. The authors find most success with the Hamming Diversity penalty term, which penalizes the selection of tokens used in previous groups proportionally to the number of times it was selected before. We detail our experiments using both search strategies in Section 6.

Post-processing in NLP For tasks that require sets of outputs rather than single outputs, post-processing or reranking methods are often used as a downstream step after a model generates an initial set. They have proven to be useful techniques for various NLP tasks, such as Question Answering (Kratzwald et al., 2019), Named Entity Recognition (Yang et al., 2017) and Neural Summarization (Cao et al., 2018). The basic methodology is to first generate an initial candidate set and rerank or prune these candidates to generate the final set. This set up reduces reliance on generators by introducing an auxiliary discriminator to refine the outputs of the generator. Section 6 describes our experiments with pruning or filtering Beam Search candidates during decoding.

4 Pretrained Base Model

As our base model, we used a model pretrained by Morishita et al. (2019) on the JParaCrawl data using the `fairseq` framework (Ott et al., 2019).

Data Preprocessing Morishita et al. (2019) preprocessed the JParaCrawl English and Japanese text using `sentencepiece` (Kudo and Richardson, 2018) to obtain 32,000-token vocabularies on both the English and Japanese sides.

Architecture The pretrained model follows the Transformer 'base' architecture (Vaswani et al., 2017), with a dropout probability of 0.3 (Srivastava et al., 2014).

Transformer is a multi-layer self-attention model. Both its encoder and its decoder contain multiple similar sub-modules which include a multi-head attention layer (MultiHead) and a position-wise feed-forward network (FFN).

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V \quad (1)$$

$$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \quad (2)$$

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_h)W^O \quad (3)$$

$$\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2 \quad (4)$$

Here, Q, K, V are the matrix representation of the query, key, and value separately. W and b denote the weights and biases of the linear layers. d_k denotes the dimension of the key matrix.

Optimizer The pretrained model was trained using the Adam optimizer (Kingma and Ba, 2014) with the hyperparameters $\beta_1 = 0.9$, $\beta_2 = 0.98$, $\alpha = 10^{-3}$ and $\epsilon = 10^{-9}$. The loss function used was cross entropy loss with $\epsilon_{ls} = 0.1$ loss smoothing (Szegedy et al., 2016). To improve update stability, gradients were clipped to a maximum norm of 1.0 (Pascanu et al., 2013).

Learning rate scheduling The learning rate schedule adopted for the pretrained model was the so-called 'Noam' schedule (Vaswani et al., 2017). This schedule linearly increases the learning rate for 4000 'warm-up' steps from a starting learning rate of 10^{-7} to the target learning rate of 10^{-3}, then decreases it from that point proportionally to the inverse square root of the step number.

5 Filtering Model

Apart from the NMT model, we additionally introduce a neural filtering model to post-process the NMT model's candidates. Instead of designing a model that will assign a real-value score to each of the candidates, we simplify the task by formulating it as a binary classification problem. Namely, the filtering model is trained to classify a given candidate sentence as a valid sample (in the gold-standard list) or an invalid sample. The intuition is that the gold-standard candidate list contains a small number of high-quality sentences (with larger weights) and a large number of lower-quality sentences. Thus it is more important to distinguish the hits from misses than high-quality hits from low-quality hits.

To construct the dataset for the filtering model, we augmented the Duolinguo dataset with the results of NMT model. Specifically, we labeled those result sentences that appear in the gold-standard list as *True* and labeled others as *False*.

As for the model architecture, we encode the source sentence and the candidate sentence separately with a one-layer bidirectional LSTM model. The encoding is the concatenation of the hidden vectors in both directions after complete traversal of the sequence, along with a (learned) positional embedding vector. This embedding encodes the position of the candidate sentence in the candidate list generated by the NMT model, which is sorted by descending score order.[4] Lastly, we use a multilayer perception (MLP) to classify the concatenated vector.

$$v_s = \text{LSTM}_s(s) \quad (5)$$

$$v_{c_i} = \text{LSTM}_c(c_i) \quad (6)$$

$$p_i = \text{MLP}(\text{Dropout}([v_s : v_{c_i} : v_i])) \quad (7)$$

Here, s denotes the source sentence, c_i denoted the i-th candidate, v_i denotes the positional encoding, and p_i denotes the predicted likelihood. The filtering model is optimized with binary cross-entropy loss.

6 Experiments

6.1 Multi-step Fine-tuning

We experiment with several different fine-tuning scenarios, each time evaluating the models using the Weighted F1 metric on our 200-prompt Duolingo test set. First as a baseline, we directly evaluate the JParaCrawl pretrained model without fine-tuning. Then we evaluate the performance of models fine-tuned on either JESC or on all English-Japanese pairs in our 2,100-prompt Duolingo training set.[5] Finally, we experiment with first fine-tuning on the JESC data and then on the Duolingo training set.

Before training, we preprocessed the JESC and Duolingo data using the same 32,000-token English and Japanese `sentencepiece` models as Morishita et al. (2019) used on the JParaCrawl data.

[4] In our experiments, we scored candidates using per token log likelihood (see Section 6.2 for further details.)

[5] The given English and Japanese sentences are unbalanced as there are multiple reference Japanese translations per English prompt. We balanced the training data by repeating the corresponding prompts over all reference translations to create English-Japanese pairs.

Training procedure We adopted the same optimizer settings as they used for the pretrained model, described in Section 4. Using mini-batches of up to 5,000 tokens, we made an update step every 16 mini-batches with mixed precision computation for increased training speed (Micikevicius et al., 2018). While the pretrained model was trained for 24,000 steps, each time we fine-tuned the model, we did so for 2,000 steps, continuing the inverse square root learning rate schedule from the pretraining. We saved the model parameters every 100 steps and for each fine-tuning experiment, we averaged the last eight parameter checkpoints to obtain our final model weights. For the model with two-step fine-tuning, we use the averaged checkpoint from the JESC fine-tuning experiment as the starting point for further fine-tuning on the Duolingo dataset.

6.2 Decoding Strategies

For producing multiple translations for each prompt, we output the entire beam width of candidates from the Beam Search or Diverse Beam Search (Vijayakumar et al., 2016) algorithms. Our motivation for experimenting with using Diverse Beam Search is to improve the coverage of our translation sets. In all our experiments, we capped the generated sequence length at 200 tokens.

Beam Search scoring Beam Search using sequence log likelihood (or likelihood) as scores results in a well-known length bias towards shorter sequences, with worsening bias for wider beams (Murray and Chiang, 2018). To address this, we scored beam candidates based on the mean log likelihood per token (Cho et al., 2014a). Further work could involve the use of more complex adjustments for length bias and including a coverage penalty over the source prompt (Wu et al., 2016).

6.3 Training Data Augmentation

Aligning data distributions The ground truth weights of the Duolingo reference translations invariably follow skewed distributions, with long tails of low weight translations (Figure 1). Consequently, one drawback of training with all English-Japanese pairs in the Duolingo data is that each pair is essentially provided to the model with equal weight. In other words, the distribution over reference translations at training time is uniform, whereas the distribution when evaluating weighted F1 score is skewed.

To address this, we sampled the training data

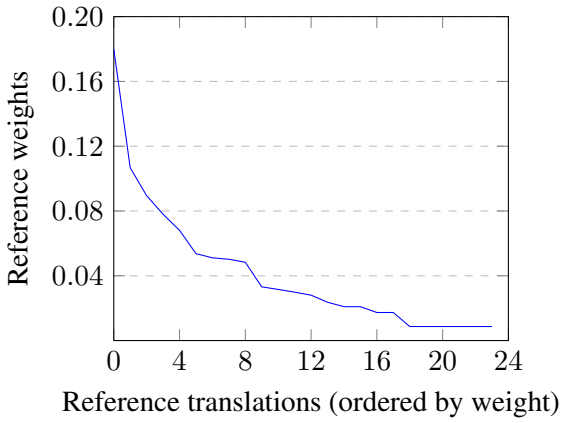

Figure 1: Typical distribution of ground truth weights

such that the model was trained on prompts with equal probability but for each prompt, reference translations were sampled according to the distribution given by the ground truth weights. In effect, this aligns the distribution over reference translations during training time and evaluation time.

Loss smoothing to improve coverage Aside from helping NMT models generalize, Müller et al. (2019) show that use of loss smoothing also better calibrates NMT models, preventing them from becoming over-confident. To encourage our NMT model to produce high-coverage translations, we hypothesize that increasing loss smoothing to decrease the model's confidence will improve its performance in producing a wider variety of correct translation candidates.

6.4 Filtering Model

Since our filtering model is trained with the results of the NMT model, we trained two filtering models with two different decoding strategies of the NMT model, namely, Regular and Diverse Beam Search with beam widths set such that approximately 100 unique candidates are output for each prompt. The NMT model is trained with the best hyper-parameters we found with the weighted sampling technique. We use the same train/dev/test splits as the NMT model and select the checkpoint with the best classification accuracy on the development set.

We used the Adam optimizer with initial learning rate 0.0001 and halved the learning rate when the validation accuracy plateaued for 2 epochs. The word embedding dimension, positional embedding dimension, the hidden dimension of the LSTM and MLP are all set to 128. The dropout rate was 0.2.

Fine-tuning	Precision	Recall	Weighted Recall	Weighted F1
None (JParaCrawl only)	18.57%	4.62%	16.97%	14.23%
JESC	13.37%	3.49%	13.12%	10.69%
Duolingo	34.59%	10.48%	26.90%	24.96%
JESC + Duolingo	35.98%	10.89%	27.85%	**25.92**%

Table 2: Results of different fine-tuning methods. Metrics evaluated on Beam Search beams of width 100 on our 200-prompt test set.

Beam	Mean # Cands	Precision	Recall	Weighted Recall	Weighted F1
50	49.6	46.27%	7.04%	21.29%	24.75%
100	98.6	35.98%	10.89%	27.85%	**25.92**%
150	147.1	29.98%	13.54%	31.33%	24.98%
200	195.4	26.01%	15.61%	33.92%	23.98%

Table 3: Results of tuning Beam (beam width). Mean # Cands refers to the mean number of unique candidates remaining after detokenizing subword tokens back into raw text and then removing duplicates. Metrics evaluated on our 200-prompt test set.

The post-processing procedure involved pruning all candidates with predicted likelihood less than 0.5.

7 Results

We conducted our experiments sequentially and generally used the best results so far as a baseline for subsequent experiments.

7.1 Multi-step Fine-tuning Results

Our best performing model was the one trained using multi-step fine-tuning, as shown in Table 2. The performance of this model was superior to the other fine-tuning settings on every metric, suggesting this result was not simply a matter of imbalance between precision and recall. This result provides strong evidence that the first fine-tuning step on the JESC data helped the model generalize to the Duolingo test set. In contrast, the model only fine-tuned on the Duolingo training set may not have generalized as well due to the training set's small size.

In order to balance precision and (weighted) recall appropriately to maximize the weighted F1 metric, we experimented with tuning the number of Beam Search candidates to output and found that 100 was optimal (Table 3). Note that the number of unique candidates returned can be fewer than the beam width as Beam Search searches over sequences of subword tokens and sometimes detokenization results in duplicates.

7.2 Diverse Beam Search Results

Our experiments with Diverse Beam Search show that using 3 beam groups with a very low Hamming diversity penalty can result in marginal performance improvement (Table 4). The algorithm evenly divides the total beam width between the groups and although the algorithm penalizes duplicate sequences, high scoring candidates are still often duplicated across groups. As such, we varied the total beam widths so that the mean number of unique candidates per prompt were approximately 100.[6] We conclude that encouraging a small amount diversity can allow the model to capture a wider range of variations without sacrificing too much precision.

We found that performance deteriorates when increasing the diversity penalty or the number of groups further. These results suggest that standard beam search by itself is relatively good at producing high-coverage translations and that acceptable variations of translations are rather homogeneous rather than diverse. To illustrate, Table 5 contains some examples of error candidates produced by Diverse Beam Search. Even though they would backtrackslate to the English prompt correctly, they nevertheless introduce a minor semantic variation that

[6]This duplication makes the number of outputs from Diverse Beam Search more variable. Our result with beam width 225 outputted 62-182 unique results per prompt with a standard deviation of 19.6, compared to 72-100 unique results with standard deviation of 4.0 from 100-width Regular Beam Search.

Groups	Penalty	Beam	Mean # Cands	Precision	Recall	Weighted Recall	Weighted F1
1	-	100	98.6	35.98%	10.89%	27.85%	**25.92**%
2	0.01	170	99.4	36.04%	11.00%	27.57%	26.18%
3	0.01	225	99.2	35.72%	10.88%	27.18%	**26.29**%
4	0.01	276	100.6	35.26%	10.90%	27.02%	26.23%
5	0.01	315	100.0	35.04%	10.76%	26.76%	26.21%
2	0.1	116	99.6	32.76%	10.03%	16.00%	24.42%
3	0.1	129	100.3	29.52%	9.10%	23.77%	22.59%

Table 4: Results of Diverse Beam Search on the test set. Beam refers to the beam width. Groups refers to the number of Diverse Groups (use of 1 group is equivalent to regular Beam Search). Penalty refers to the Hamming Diversity penalty in the Diverse Beam Search algorithm. Mean # Cands refers to the mean number of unique candidates remaining after detokenizing subword tokens back into raw text and then removing duplicates. Metrics evaluated on our 200-prompt test set.

Prompt	my parents have money
Incorrect	僕の両親はお金を持ってる
Diverse	僕の両親はお金を持ってます
Candidates	僕の両親には金があります

Table 5: Example incorrect candidates from Diverse Beam Search with 3 groups and 0.1 Hamming Diversity penalty. While the candidates would correctly back-translate to 'my parents have money', the first character of each candidate sentence indicates that the speaker / subject must be male (a restriction that is absent in the prompt).

Sampling	Smoothing	1-best BLEU	Precision	Recall	Weighted Recall	Weighted F1
Weighted	0	43.2	36.28%	10.95%	27.88%	26.88%
Weighted	0.05	42.5	37.41%	11.30%	28.31%	**27.43**%
Weighted	0.10	43.2	37.00%	11.27%	28.14%	**27.21**%
Unweighted	0.10	27.0	35.72%	10.88%	27.18%	**26.29**%
Weighted	0.15	41.8	36.84%	11.07%	28.01%	27.06%
Weighted	0.20	42.3	36.86%	11.09%	27.96%	27.04%

Table 6: Results of weighted sampling of input translation pairs and different loss smoothing rates on the test set. 1-best BLEU refers to corpus BLEU-4 score between the single highest-scoring Diverse Beam Search candidate and the single highest weighted reference translation for each prompt, smoothed with the NIST method (Chen and Cherry, 2014). The other metrics were evaluated over Diverse Beam Search with 225-width beams split across 3 groups and Hamming diversity penalty of 0.01. Metrics evaluated on our 200-prompt test set.

makes them unacceptable translations.

7.3 Training Data Augmentation Results

Sampling training data according to the ground truth weights meaningfully improves performance, as shown in Table 6. Our previous best weighted F1 score using Diverse Beam Search was 26.29%, and this improved to 27.21%. Moreover, evaluating the model on the standard machine translation metric of BLEU-4 score between the single best candidates and the single best ground truth translations, we observe a remarkable increase in BLEU score if weighted sampling is used during training. From this result, we conclude that unweighted sampling of training data overexposes the model to poorer translations, which significantly reduces the model's effectiveness as a general-purpose NMT model.

As for loss smoothing, contrary to our hypothesis, increasing the loss smoothing rate was detrimental. and, in fact, decreasing the rate from 0.1 to 0.05 even improved the weighted F1 score slightly from 27.21% to 27.43%. This suggests that the effect of loss smoothing on the high-coverage translation task is not necessarily different to the usual machine translation task.

Filtering	Decoding method	Precision	Recall	Weighted Recall	Weighted F1
None	Regular Beam Search	37.49%	**11.35**%	**28.93**%	27.00%
Filtered	Regular Beam Search	**38.89**%	10.87%	27.67%	**27.43**%
None	Diverse Beam Search	37.41%	**11.30**%	**28.31**%	27.43%
Filtered	Diverse Beam Search	**38.08**%	10.86%	26.94%	**27.56**%

Table 7: Results of filtering methods on our 200-prompt test set. Candidates were generated by the NMT models fine-tuned on JESC then Duolingo data with weighted sampling technique. Regular Beam Search used beam width 100 and Diverse Beam Search used beam width 225 over 3 groups with Hamming diversity penalty of 0.01 to yield approximately 100 candidates per prompt after deduplication. Candidates that have likelihoods greater than 0.5 assigned by the filtering model are selected as the results.

7.4 Beam Filtering Results

Table 7 shows the results of the filtering algorithm. The filtering model can improve the weighted F1 score with both the diverse beam search and regular beam search, especially with the regular beam search. This improvement results from a larger gain in precision from filtering than the loss in recall.

One thing to note is that our filtering model suffers from over-fitting. For example, with Regular Beam Search, our filtering model improves the weighted F1 score by 0.43% on the test set (Table 7). However, using the same technique on the training set results in an improvement of 6.25%.[7] This may result from the limited size of Duolinguo dataset, and the fact that over-fitting introduced by the NMT model would be amplified since the filtering model is trained on the results of the NMT model.

8 Conclusions and Future Work

Our machine translation system produces high-coverage sets of target language translations from single source language prompts.

We used multi-step fine-tuning to train a robust NMT model. This involved first training or fine-tuning a model on a large bitext dataset, then fine-tuning on the bitext dataset with high coverage sets of target language translations, which is likely to be small. In our experiments, we find that fine-tuning a pretrained model first on a corpus similar to our intended domain and then fine-tuning further on our smaller in-domain dataset produced the best results.

During training, we find that if the ground truth translations come with weights that indicate variations in their quality / likelihood, it is essential to expose the model to higher-quality translations more often during training. One way to do this is to to sample the training data with probabilities commensurate to the ground truth weights. Doing so will prevent overexposure to low-quality translations that ultimately harm the model's translation performance.

For decoding, we find that Beam Search scored on per token log likelihood finds very good translation candidates on its own. Nevertheless, instead using Diverse Beam Search with a very small penalty improves coverage.

We observed a further performance boost from post-processing the translation candidates. This was achieved by training an auxiliary filtering model on the results of the NMT model to prune unlikely candidates as a final step.

One idea for future work is to directly optimize the weighted F1 score during training using reinforcement learning. As the weighted F1 score is not a differentiable function, it is impossible to train directly on this metric using maximum likelihood estimation. Instead, one may use policy gradients under a reinforcement learning paradigm to do so.

[7] On the training set, the filtering algorithm improves the weighted F1 score from 56.78% to 63.03%.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Ziqiang Cao, Wenjie Li, Sujian Li, and Furu Wei. 2018. Retrieve, rerank and rewrite: Soft template based neural summarization. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 152–161, Melbourne, Australia. Association for Computational Linguistics.

Boxing Chen and Colin Cherry. 2014. A systematic comparison of smoothing techniques for sentence-level bleu. In *WMT@ACL*.

Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014a. On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111, Doha, Qatar. Association for Computational Linguistics.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014b. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078.

Markus Freitag and Yaser Al-Onaizan. 2016. Fast domain adaptation for neural machine translation. *CoRR*, abs/1612.06897.

Nal Kalchbrenner, Lasse Espeholt, Karen Simonyan, Aäron van den Oord, Alex Graves, and Koray Kavukcuoglu. 2016. Neural machine translation in linear time. *CoRR*, abs/1610.10099.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Bernhard Kratzwald, Anna Eigenmann, and Stefan Feuerriegel. 2019. Rankqa: Neural question answering with answer re-ranking. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6076–6085.

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

Jiwei Li and Dan Jurafsky. 2016. Mutual information and diverse decoding improve neural machine translation. *CoRR*, abs/1601.00372.

Minh-Thang Luong and Christopher D. Manning. 2015. Stanford neural machine translation systems for spoken language domain. In *International Workshop on Spoken Language Translation*, Da Nang, Vietnam.

Minh-Thang Luong, Ilya Sutskever, Quoc V Le, Oriol Vinyals, and Wojciech Zaremba. 2014. Addressing the rare word problem in neural machine translation. *arXiv preprint arXiv:1410.8206*.

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

Stephen Mayhew, Nitish Gupta, and Dan Roth. 2019. Robust named entity recognition with truecasing pretraining.

Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, and Hao Wu. 2018. Mixed precision training. In *International Conference on Learning Representations*.

Makoto Morishita, Jun Suzuki, and Masaaki Nagata. 2019. JParaCrawl: A large scale web-based japanese-english parallel corpus. *arXiv preprint arXiv:1911.10668*.

Rafael Müller, Simon Kornblith, and Geoffrey E Hinton. 2019. When does label smoothing help? In H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 4694–4703. Curran Associates, Inc.

Kenton Murray and David Chiang. 2018. Correcting length bias in neural machine translation. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 212–223, Brussels, Belgium. 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 NAACL-HLT 2019: Demonstrations*.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, ICML'13, page III–1310–III–1318. JMLR.org.

R. Pryzant, Y. Chung, D. Jurafsky, and D. Britz. 2018. JESC: Japanese-English Subtitle Corpus. *Language Resources and Evaluation Conference (LREC)*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100, 000+ questions for machine comprehension of text. *CoRR*, abs/1606.05250.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Improving neural machine translation models with monolingual data.

Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15:1929–1958.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. *CoRR*, abs/1409.3215.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Alon Talmor and Jonathan Berant. 2019. Multiqa: An empirical investigation of generalization and transfer in reading comprehension. *CoRR*, abs/1905.13453.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Ashwin K. Vijayakumar, Michael Cogswell, Ramprasaath R. Selvaraju, Qing Sun, Stefan Lee, David J. Crandall, and Dhruv Batra. 2016. Diverse beam search: Decoding diverse solutions from neural sequence models. *CoRR*, abs/1610.02424.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144.

Jie Yang, Yue Zhang, and Fei Dong. 2017. Neural reranking for named entity recognition. In *Proceedings of the International Conference Recent Advances in Natural Language Processing, RANLP 2017*, pages 784–792, Varna, Bulgaria. INCOMA Ltd.

A Appendices

To evaluate the result, the weighted macro F_1 (equation 8) with respect to the accepted translations is the metric of interest. This is the average weighted F_1 score (equation 12) over all prompts s in the corpus, where weighted F_1 is calculated with (unweighted) precision and weighted recall.

$$\textbf{Weighted Macro } \mathbf{F_1} = \sum_{s \in S} \frac{\textbf{Weighted } \mathbf{F_1}(s)}{|S|} \tag{8}$$

Calculating the weighted recall requires the use of weights included in the dataset. These weights are associated with each human-curated acceptable translation, which represent the likelihood that an English learner would respond with that translation.

For each prompt s, the weighted true positives (WTP) and weighted false negatives (WFN) are:

$$\mathbf{WTP}_s = \sum_{t \in \mathbf{TP}_s} weight(t) \tag{9}$$

$$\mathbf{WFN}_s = \sum_{t \in \mathbf{FN}_s} weight(t) \tag{10}$$

With these, the weighted recall for each s can be calculated as follows

$$\textbf{Weighted Recall}(s) = \frac{\mathbf{WTP}_s}{\mathbf{WTP}_s + \mathbf{WFN}_s} \tag{11}$$

Precision is calculated in the usual way, so the weighted F_1 score, $\textbf{Weighted } \mathbf{F_1}(s)$, for a particular input s is given by

$$2 \cdot \frac{\textbf{Precision}(s) \cdot \textbf{WeightedRecall}(s)}{\textbf{Precision}(s) + \textbf{WeightedRecall}(s)} \tag{12}$$