NLU+ Coursework 2: Neural Machine Translation

s1817624 S1866666

1. Understanding the Baseline Model

1.1. A

In the case of bi-directional LSTM, the final hidden/cell states of the two directions are concatenated (forming state vectors twice as long as before). The function `combine_directions` entangles the interleaved layers by their direction and concatenate them along the state vector dimension. The rest of the code performs the direction combining on both the hidden and cell states at the final step, `final_hidden_states` represents the hidden states of all layers at the final step, and contains the information of output words at all time steps. `final_cell_states` represents the cell states of all layers at the final step. Cell state is the key to tackling the vanishing gradient problem.

1.2. B

`src_mask` is a binary tensor with size `[batch_size, src_time_steps]` where the 1's indicate padding tokens. The code adds an extra dimension to match the shape of `attn_scores` so as to constrain the attention scores of the paddings tokens to be negative infinity (minimal attention). The attention weights are obtained by soft-maxing attention scores as the weights has to sum up to 1. We then compute the batch matrix-matrix product of the weights and encoder context vector and squeeze it. The product is then concatenated with the target hidden states at $t-1$. This concatenated vector is finally fed to a linear layer followed by a tanh non-linearity. The attention output has shape `[batch_size, output_dims]` and consists of the attention context vectors for a batch.

1.3. C

We use the the bilinear function to obtain general attention scores (Luong et al., 2015), which is computed by 1.

$$\text{score}(h_t, \bar{h}_s) = h_t^T W_a \bar{h}_s \tag{1}$$

where h_t is an encoder output vector, W_a is a learnable linear transformation, and $\bar{h}_s$ is the decoder output vector. The linear transformation is applied so that the source and target vectors are not required to be in the same space (and shape). The matrix multiplication is used here so that the attention score can be seen as the inner product of the target context vector h_t and the linearly transformed source encoding $W_a\bar{h}_s$ at each step. Hence, to get a high attention score, the two vectors must be close in space. By doing so, we perform a "spacial alignment" on the context representation of the encoder and decoder.

1.4. D

The variable `cached_state` stores the encoder and decoder states at previous step in auto-regressive generation (i.e. at test time). `cached_state` is `None` at training time and the first step in the generation (as there is no previous states). `input_feed` feeds the output at previous step into the current step. If the attention is used, the connection also makes the decoder aware of the previous alignment choice (Luong et al., 2015). At $t = 0$, the decoder hidden states and cell states are initialized to $\mathbf{0}$ for all layers. The `input_feed` is initialized to $\mathbf{0}$ as well.

1.5. E

At each time step t, the output of the decoder's final layer is feed into the attention layer, together with the encoder outputs. The output of the attention layer at time step t is used to compute the decoder output, and is also auto-regressively fed to the decoder at the next time step $t+1$. The target hidden states of the final layer are given to the attention layer because the alignment between source and target context relies on information from both the encoder and decoder, and the decision made by the decoder is conditioned on this alignment. The target hidden states carries encoded target contextual information and is thus required in the alignment decision. Dropouts are inserted between layers to encourage better generalization.

1.6. F

The code performs a single iteration of training. First we make a forward pass of the model, and then compute the cross-entropy loss of this batch. We then back-propagate the gradients of the parameters w.r.t. the loss. To prevent exploding gradients, we clip the gradient norm before updating the parameters according to the gradients. Finally, the gradients are cleared for the next batch.

2. Understanding the Data

2.1. Plot

The histogram of the English and Japanese sentence lengths and their correlation plot are shown in figure 1. And some statistics of the distribution of sentence lengths are as Table 1.

The least square regression fit of the correlation can be represented as $L_{jp} = 1.102L_{en} + 3.43$, where L_{jp} denotes Japanese sentence length and L_{eg} denotes English sentence length. It can be inferred that the Japanese sentences are

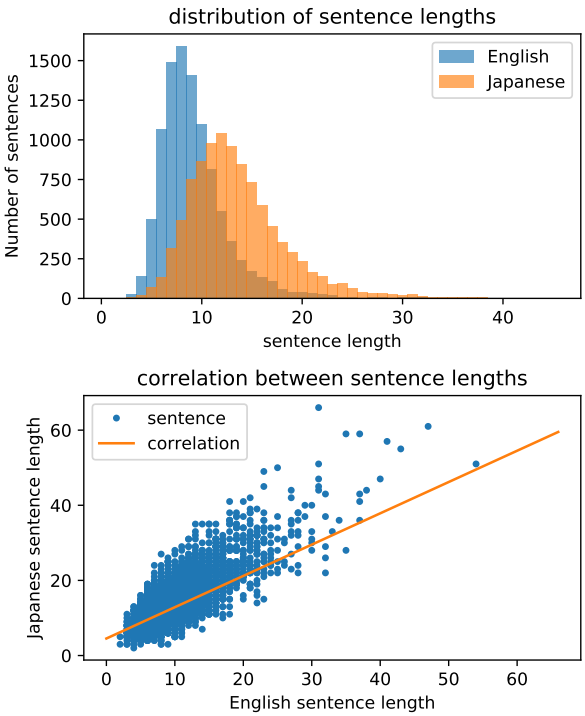


Figure 1. Distribution of sentence lengths

Language	mean	median	std dev.
English	9.309	9	3.597
Japanese	13.690	13	5.161

Table 1. Statistics of the distribution of sentence lengths

generally longer (in number of tokens) than the corresponding English sentences. This means in the alignment of English-Japanese pairs, each English token aligns with more than one Japanese token on average.

2.2. Statistics of word tokens

There are 93086 tokens in the English data in total.

There are 136899 tokens in the Japanese data in total.

2.3. Statistics of word types

There are 7040 types in the English data.

There are 8058 types in the Japanese data.

2.4. Statistics of <UNK>s

3331 tokens in the English data will be replaced.

4113 tokens in the Japanese data will be replaced.

2.5. Impact of data distribution

The difference in sentence lengths can cause one-to-many alignment in translation, and thus results in a more complicated task. The type-token ratio (TTR) (0.076 for English and 0.059 for Japanese) measures the lexical variability of the data. The higher TTR of the English sentences indicates English words can represent more specific meanings than Japanese. This can be explained by a richer use of morphology in English than Japanese. For example, 'am' 'is', 'are' in English all translate to Japanese word 'は', and Japanese words usually don't have plural forms. Thus there may be more than one possible English translations to a Japanese word, causing more uncertainty and degraded quality when translating Japanese to English. Concerning the difference in the number of words replaced by <UNK>, the effect can be evaluated in terms of information. Technically, the lower the frequency of a word, the more information it carries. Therefore, if we assume all the words that appears once carries the same amount of information, because there are more Japanese words replaced than English, there is more information loss in the Japanese data than in English. This asymmetry is likely to make it more difficult when translating from Japanese to English.

3. Improved Decoding

3.1. Problems of greedy search

Choosing argmax $p(e_i|F, e_{0:i-1})$ (Neubig, 2017) at each timestep is sub-optimal, meaning that it doesn't necessarily lead to the solution that maximize $p(E|F)$. From a linguistic perspective, this can be problematic as it's prone to producing nonsense. As an extreme example, a decoder may fall into an eternal loop generating "There is a (Noun) of the (Noun) of the (Noun) of ..." by choosing argmax $p(e_i|F, e_{0:i-1})$ all the time.

3.2. Beam search

Instead of always looking at one hypothesis, beam search (Neubig, 2017) consider k best candidate sequences at each timestep. That is, suppose k=5 and a vocabulary size of $|V| = 10$, in the first timestep, the expanded hypotheses include the top 5 out of 10 words that maximizes $p(e_1|F, e_0)$. In the second timestep, we look into all the possible words e_2 that expands the 5 hypotheses retained from the previous step, and choose the top 5 out of $k * |V| = 50$ possible hypotheses that maximizes $p(e_2|F, e_{0:1})$. We continue this process until all the hypotheses reach the end of sentence. The hypothesis with the highest probability is the result.

3.3. Length normalization

Greedy or beam search decoders favor short sentences because short sentences generally have high probability. In other words, every word added to the sentence reduces the probability of the whole sentence by multiplying a conditional probability (< 1). Normalizing the sentence probability by the length of the sentence can effectively solve the length bias, however, this may result in producing exceedingly long sentences.

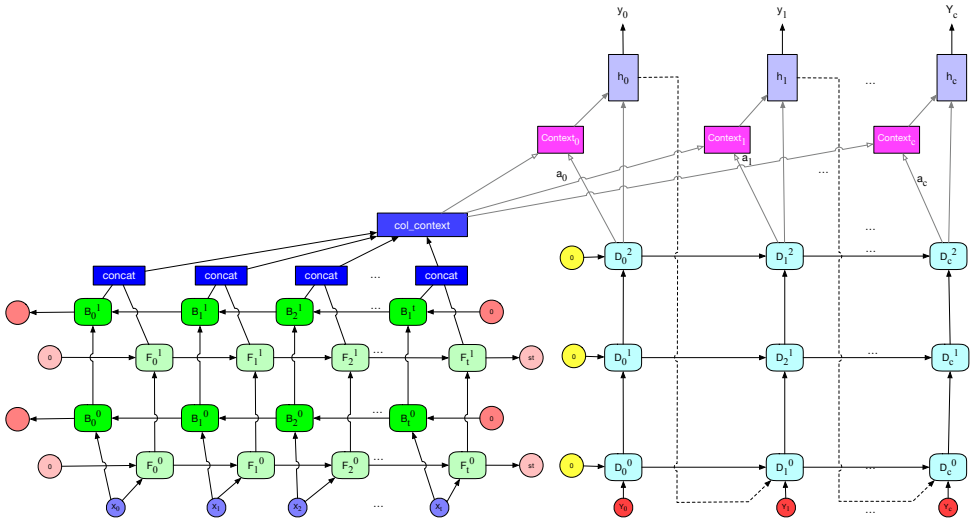


Figure 2. Diagram of new architecture with 2 encoder layers and 3 decoder layers. Encoder cells are in green, decoder cells are in cyan, and attention cells in pink.

4. MOAR Layers

4.1. Command of changing the number of layers

To train a model with more layers, we changed the number of layers in the encoder = 2, decoder = 3. The Command we used is:

```
python train.py --encoder-num-layers 2
--decoder-num-layers 3
```

4.2. Diagram with more layers

The diagram of new architecture with 2 encoder layers and 3 decoder layers is shown in figure 2.

4.3. Comparison with baseline

The results are shown in the Table 2

model	baseline	layers+
training loss	2.334	2.368
dev perplexity	22.4	27.0
test BLEU	7.98	5.44

Table 2. Results of multi-layer model, compared with baseline

We can see the results from the Table 2, the training loss and dev-set perplexity of the model with more layers are higher than the baseline, and test BLEU score on the model with more layers is lower than the baseline. It shows the performance of model with more layers is worse than the baseline, and the lower training loss indicates an underfitting problem. To ensure the convergence of the training, we try to use the parameter we trained to train the model again to see if we can get the lower loss after early stop(60 epochs). The result shows it stops again after 5 epochs (at 65 epochs), and finally yields the same training loss.

Generally, adding more layers to the model will increase the representational power of the model and thus lead to a better training loss, however, it makes the model harder to train and converge. The worse result of this deeper model is likely due to the more complicated manifold, and therefore the optimizer is trapped into a sub-optimal solution. It could also be the vanishing gradient problem induced by the deepened model that obstructs the effective training.

5. Implementing the Lexical Model

See the implementation of the lexical model (Nguyen & Chiang, 2018) is shown in code *lstm.py*.

6. Effects on Model Quality

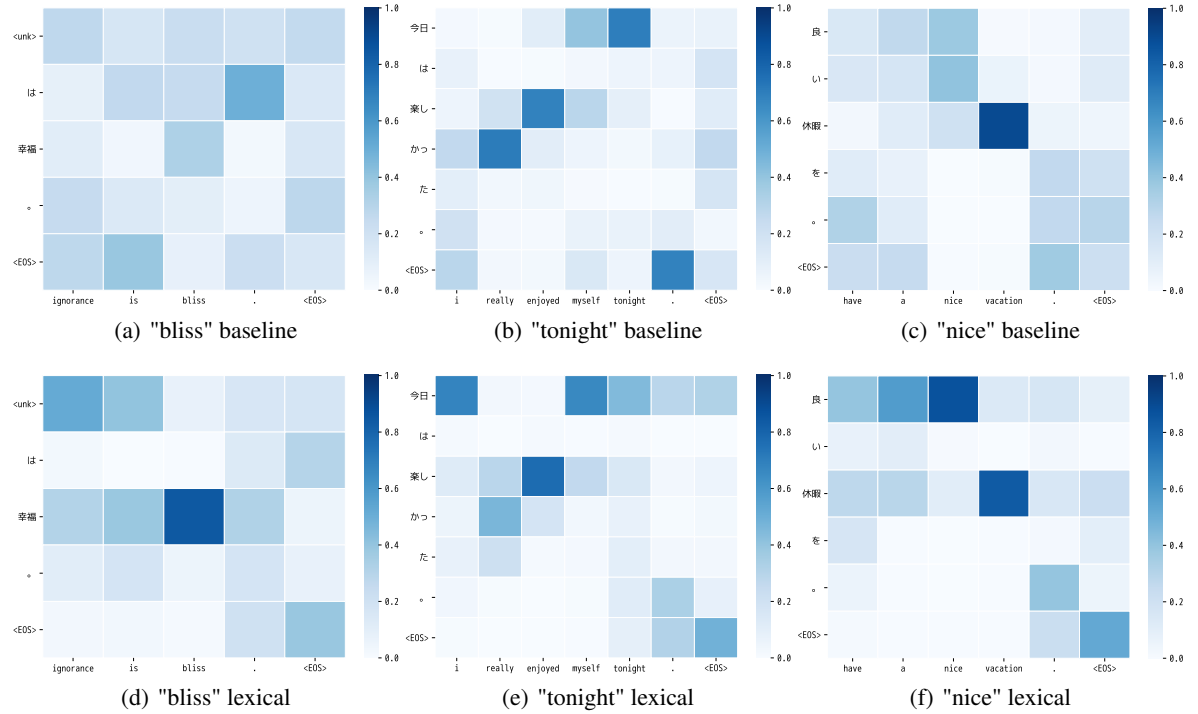
model	baseline	lexical
training loss	2.334	1.957
dev perplexity	22.4	20.2
test BLEU	7.98	10.09

Table 3. Results of lexical model, compared with baseline

The results of the lexical model are shown in figure 3. It shows that the lexical model makes a significant improvement on both training loss and development set perplexity, indicating that the model fits the data better than the baseline. It also yields a higher test set BLEU score, meaning the translation quality is improved as well. The contribution of the lexical mechanism is obvious, and it is proved that considering lexical information in addition to the contextual aspects of the source text is beneficial to the performance of NMT systems.

7. Effects on Attention

Max value of the attention weights across all source words for each target words in lexical model are generally higher than in baseline model. This is because the lexical model



(a) "bliss" baseline (b) "tonight" baseline (c) "nice" baseline

(d) "bliss" lexical (e) "tonight" lexical (f) "nice" lexical

Figure 3. Comparison between alignments of the two models.

takes into account the source word which the decoder attends on when modelling the target word, whereas the baseline model prioritizes the context of the target sentence, in order to produce fluent outcomes. As a result, the lexical model is more sensible to the co-occurrence of the words and is therefore more accurate and certain about the lexical alignments as long as it has seen such co-occurrence. More specifically, we hypothesize that the lexical model assigns the attention based on the relevance of words. Positive point-wise mutual information (PPMI)(Church & Hanks, 1990) can be used to measure the relevance/similarity between a pair of source and target words. The PPMI of the two words w_t and w_s can be computed by

$$\text{PPMI}(w_t, w_s) = \max(\frac{C(w_t, w_s) * N}{C(w_t)C(w_s)}, 0) \qquad (2)$$

where N is the total sentence pairs in the training set, $C(\cdot)$ is the occurrence or co-occurrence count.

A typical example is shown in the figure 3(a) and figure 3(d), the attention of the word 'bliss' on its correct translation '幸福' is almost definite (with a high weight of 0.9) modeled by the lexical model, whereas according to the baseline model, the attention is roughly uniformly distributed over all the source words. This is the case as the words 'bliss' and '幸福' have a high PPMI of 8.48 in the training set. On the other hand, the lexical model tends not to assign a high weight on an alignment if the PPMI is low. As shown in the figure 3(b) and figure 3(e), the attention of the word 'tonight' on '今日'(today) is lower in the lexical model than in the baseline. Looking at the training set, we found no co-occurrence (thus 0 PPMI) of the two words, which complies with our theory.

As shown in the figure 3(c), we noticed in the alignment of the lexical model that the word 'have' slightly inclines to an apparently wrong translation '良'(nice), as there is no proper Japanese counterpart of 'have' (which is probably habitually omitted) in the source sentence. However, we still found that the lexical model yields better translation than the baseline:

```
reference: Have a nice vacation .
lexical: I am a good vacation .
baseline: I have a good time .
```

Although not being able to come up with the word 'have', the lexical model precisely translates the word 'vacation'. This shows that the lexical model may be vulnerable when there is no obvious word to attend on, however, if the attention is heavily put on a word, it's very likely to produce the exact translation.

It is also worth noticing from the above example that the lexical model makes a significant trade-off between lexical precision and sentence fluency, as pointed out by Nguyen & Chiang (2018). Despite the precise translation, some word (especially those with high attention) can be disruptive to the syntactic flow of the sentence.

8. Analyse Effects on Translation Quality

As elaborated in section 7, the lexical model is able to assigning attention more accurately and certainly than the baseline model, regardless of the word's frequency in the training set. Therefore, we argue that sentences with less

frequent words benefit more from the lexical information. In other words, the lexical mechanism enables more accurate translation of rare words than of frequent words.

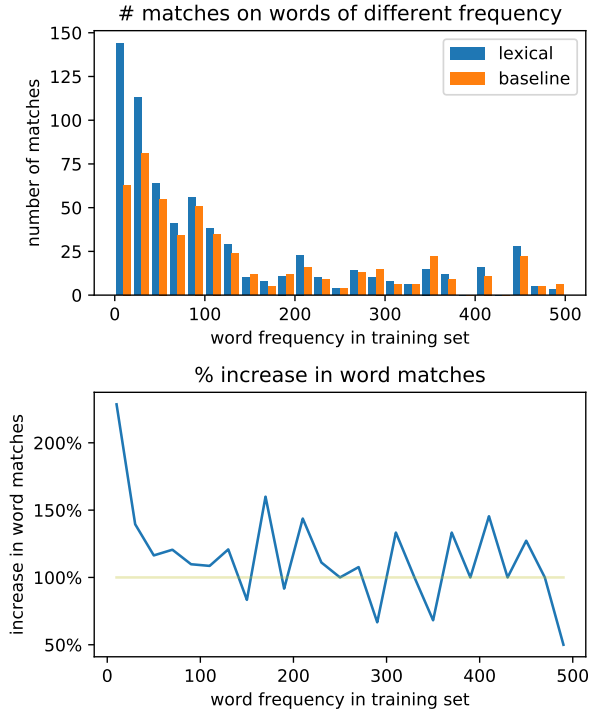


Figure 4. Top: Number of correctly translated words of different frequency under the baseline and lexical models. Bottom: the difference in number of matches of the two model.

We base our argument on the Japanese-English translation performed on the test set. Take the following translations as example:

```
reference: I have a picture of an airport .
lexical: I have a picture of the airport .
baseline: I have a picture of my mind .
```

The baseline translation is clearly worse than that of the lexical model, and the reason is likely that the English word 'airport' appears 9 times in the training data, while the word 'picture' appears 25 times. The baseline model is able to translate only the more frequent 'picture' but the less frequent 'airport', whereas the lexical model correctly translates both.

To evaluate our hypothesis quantitatively, for each model, we count how many translated words matches the reference text and compute their frequency in the training set. Note if there are repeated words in a model translation but only one such word in the reference text, we treat this as one match. Shown in the figure 4 (top) is the histogram of number of word-matches by the lexical and baseline under different intervals of word frequencies. In the figure 4 (bottom) is the percent difference in the matches of the models. It can be noticed that the lexical model has a greater lead in terms of the correct matches in the low-frequency area, while as the word frequency goes up, the difference between the two models becomes less significant. This observation is in accordance with our argument that the less frequent words benefit more from the lexical information.

From a probabilistic perspective, it is reasonable to assume that the rarest word in a sentence contains the most information. Thus on the sentence level, we base on this intuition and use the translation of the rarest word to estimate the amount of information delivered. For each reference sentence, we check if the rarest word is translated correctly, and compare the number of such matches by the two models over several frequency intervals, as shown in figure 5, On the top is the histogram showing the number of rarest-word matches, and on the bottom is the percent increase in the number of matches. The graph shows that the lexical model translates 5 times more rarest words than baseline when the rarest-word frequency is less than 10. Such improvement fades dramatically as the frequency increase.

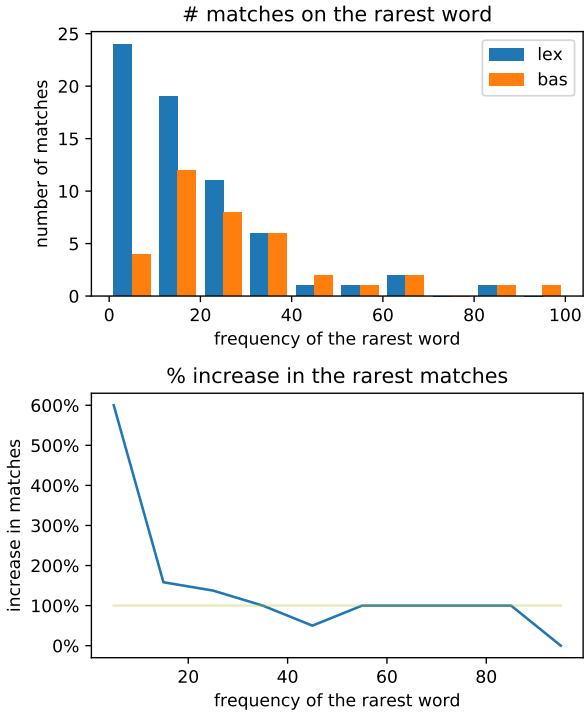


Figure 5. Number of matches on the most rare words by the two models

In the bigger picture, despite a noticeable degradation on intelligibility as alluded in section 7, the improvement on translating rare words still means a convincing improvement on the overall translation quality in terms of the information delivery rate. Future work can aim on achieving more fluent translation while preserving the lexical precision. A network dedicated for syntactic rectification can be a promising, feasible augmentation to complement the lexical model.

References

Church, Kenneth Ward and Hanks, Patrick. Word association norms, mutual information, and lexicography. *Computational linguistics*, 16(1):22–29, 1990.

Luong, Minh-Thang, Pham, Hieu, and Manning, Christopher D. Effective approaches to attention-based neural machine translation. 2015.

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

Nguyen, Toan Q. and Chiang, David. Improving lexical choice in neural machine translation. In *NAACL-HLT*, 2018.