Natural Language Understanding, Generation and Machine Translation (2018-2019)
Coursework 2: Neural Machine Translation
(S1873947, S1884908)

Question 1.

Q1.a.

When self.bidirectional is set to True, it makes the network behave as a bidirectional LSTM encoder.This effect is achieved by concatenating the final hidden states and the final cell states in each direction.

In LSTM, the output of LSTM cell is called the Hidden state and internal state of a cell is called the cell state. The final_hidden_states are obtained by using the outputs from the hidden layers, whereas final_cell_states exist for each of the cells in the layer.

Q1.b.

The attention context vector is calculated by the batch matrix-matrix multiplication (dot product) of the attention weights (obtained by normalizing the attention scores) and encoder outputs.

Masking needs to be applied to handle attention scores of sentences of various lengths. Setting the padded values of the attention scores to -inf and applying softmax, causes the attention weights to focus only on the valid symbols in then sentence and assign around zero weights to pad symbols.

Q1.c.

The attention scores are computed by batch matrix-matrix multiplication (dot product) of the projected_encoder_output and the corresponding targets of the batch inputs. This dot product is done using the pytorch function, torch.bmm().

The projected_encoder_output is computed by applying src_projection to the encoder_ouputs in order to make the dimensions consistent for the dot product with the target inputs.

The result of the matrix multiplication enables the system, to obtain attention on the appropriate word.

Q1.d.

The Decoder state is initialized by testing for the cached_state. If there is a cached_state then the decoder state is assigned with cached_state. If there is no cached_state then the decoder_state creates zero vectors for the hidden_states, cell_states and the input_feed.

The value of cached_state is none when incremental, auto-regressive generation is not used or the system is on the first word in a sentence.

input_feed is the output of vector from the attention layer vectors. It passes within the LSTM, context information from the target hidden states.

Q1.e.

Attention is integrated into the decoder through self.attention which is activated if self.attention is set to True. The Attention step weights are updated through time steps and are stored in step_attn_weights.

The attention function is given the previous target state as one of its inputs because this enables information to pass from the previous alignment. This is then used to obtain the correct alignment for the current word.

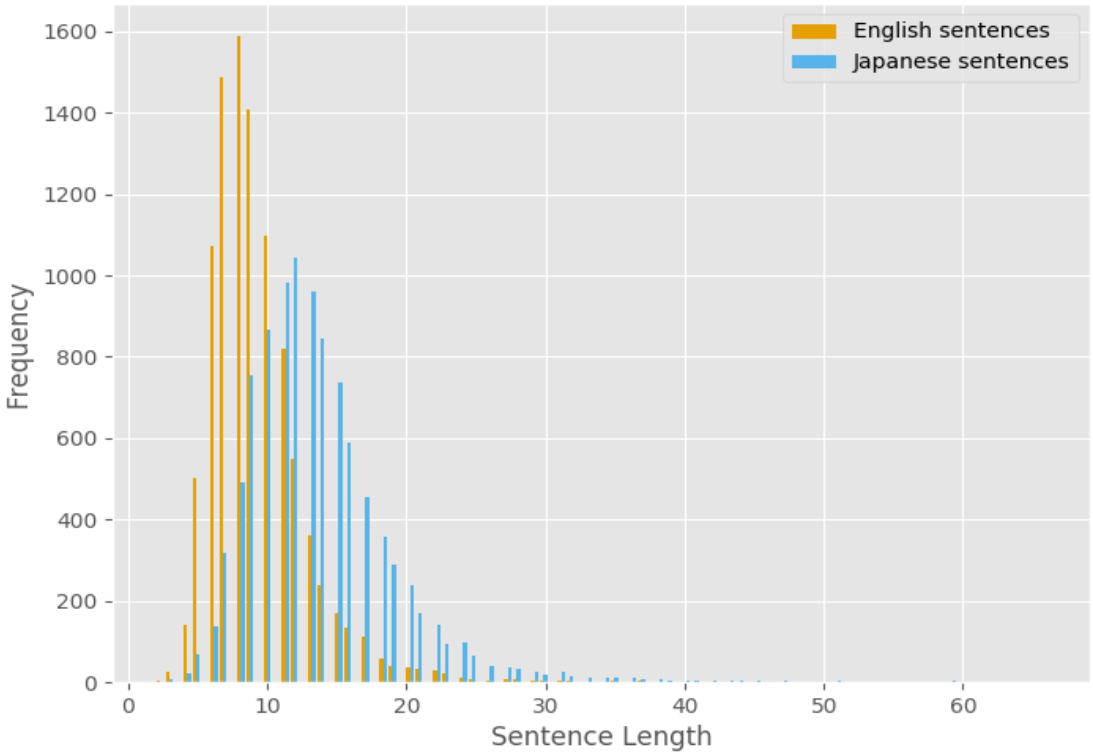
The dropout layer prevents overfitting and helps improve the performance of the model. This is because the inputs and recurrent connections to LSTM units are excluded through a probability, during the training process.

Q1.f.

The model is built and the CrossEntropyLoss is computed. Then, the loss is propagated backwards within the network. The norm is computed over all the gradients and the ADAM optimization technique is applied to update the network weights. The gradients of all of the model parameters are then set to zero.

Question 2

Q2.1.



Using the parallel training data, the sentence lengths of the English and Japanese sentences were plotted on the x axis and the corresponding frequency of the sentence lengths were plotted on the y axis.

From the graph it can be observed that there are a greater number of shorter sentences in English. It can be observed that Japanese sentences are longer in length when compared to the English sentences (as seen in the graph above). Longer sentences are relatively uncommon in English compared to Japanese. We could infer this from the table below:

	English words per sentence	Japanese words per sentence
Average	9.3	13.69
Minimum	2	2
Maximum	54	66
Number of sentences with length >20	155	843

Q2.2.
Word tokens in English data: 83234
Word tokens in Japanese data: 136899

Q2.3.
Word types in English data: 6713
Word types in Japanese data: 8058

Q2.4.
Number of tokens in English to be replaced by <UNK> are: 3208
Number of tokens in Japanese to be replaced by <UNK> are: 4113

Q2.5.
The number of words in Japanese sentences are more compared to the English ones. Around 8% (6713/83234) of the English words from the set of English tokens are unique as against, around 6% (8058/136899) in Japanese data. For the NMT translations, as the Japanese sentences are longer in length, one word from English might get translated to one or more number of words in Japanese. So this is not a one to one translation.
Unknown words constitute 3% (4113/136899) of the word tokens in Japanese and 3.8% (3208/83234) in English. When taken against word types, these constitute around 50% for both the languages. Hence this may put a serious constraint on the translation output even with our best model.

Question 3

Q3.1.
Greedy decoding could be problematic due to the following reasons:

a) When the emotions in a sentence are related to the context, greedy decoding may change the tone of the sentence. For example: From the training corpus - “my leaving early made them feel sorry.” With greedy decoding, we may be predicting the sentence as “my leaving early made them feel happy.” because the number of occurrences of happy are more than sorry. This changes the tone of the sentence and conveys a different meaning.

b) Greedy decoding also poses a significant challenge with high frequency words like ‘the’, ‘of’ and so on. We hypothesized that some of the translation could end up just having the high frequency words in the translation. An example is as follows:
Gold Standard: aren 't you going to get mr tate ?
Baseline Translation: can you put the mr mr mr mr mr mr mr mr correct ?

c) We also hypnotized that, this method of decoding also poses challenges with numeric estimation. For example in the sentence, “The Japanese economy recorded more than 60 months of continuous expansion.”, we have to correctly predict ‘months’. However, with greedy decoding it can predict it as years, given that years may co-occur more frequently with the time duration. Though, months being replaced by years may seem like a valid English sentence, it fails to convey the meaning intended by the original sentence. An example is as follows:
Gold Standard: she goes to the movies once a **week** .
Baseline Translation: she goes to see the movies once a **month**.

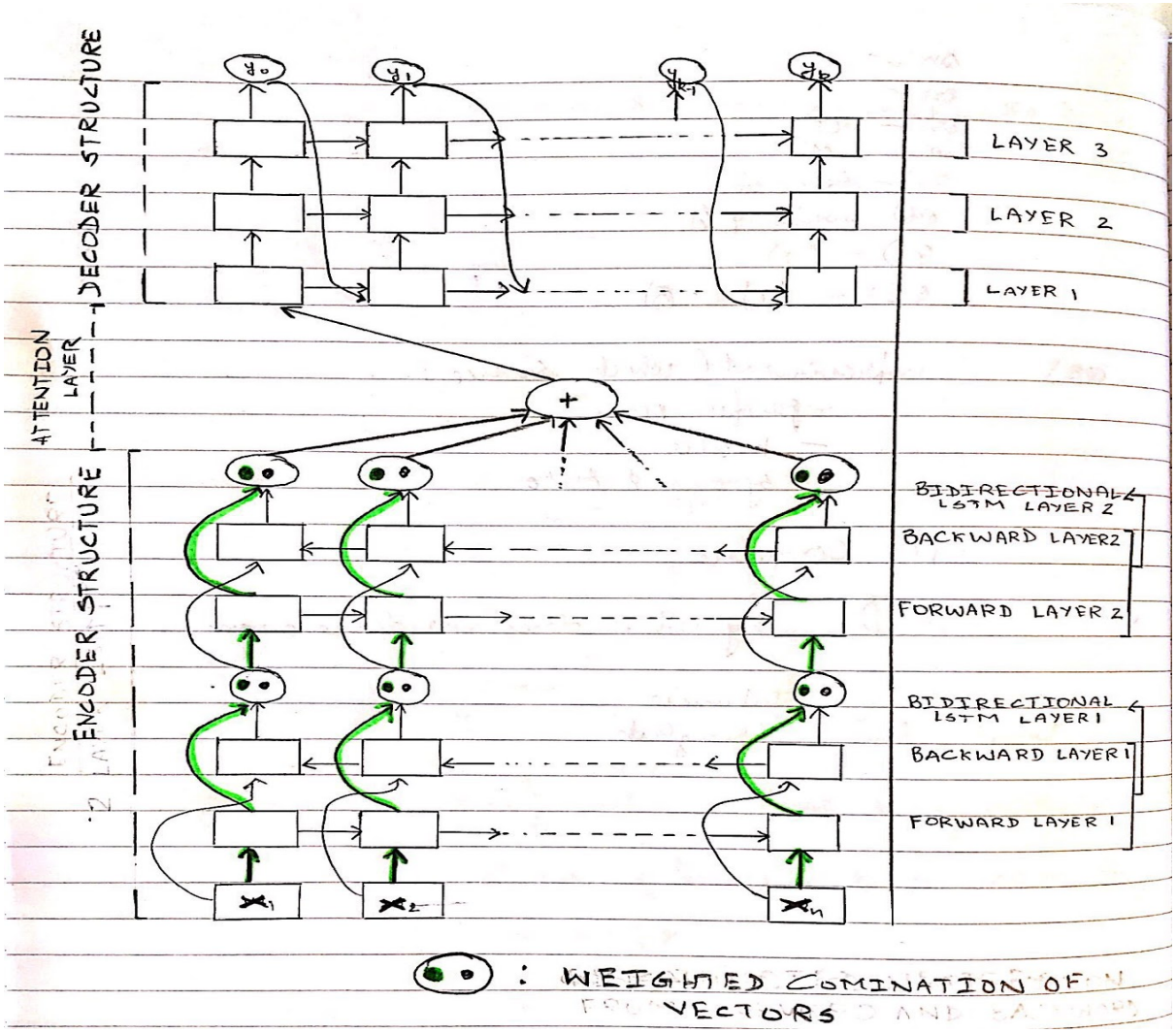
Q3.2.
Beam search outputs the K - best translations where k is the number of beams.
To implement the beam search we have to modify the translate.py file.
In torch.topk we have to provide the k=beam_width along with the decoder_outputs as input. This is to be followed by computing the decoder probabilities for the beam and putting them into a queue followed by choosing the k best paths (the k paths with maximum probability) as the generated sentence.

Q3.3.
The decoder favors shorter sentences because the probabilities of the words are multiplied during generation of a sentence. So when more number of words are added, the total probability of the sentence reduces.

Length normalisation does not provide a complete solution, possibly due to the fixed length representation of the encoder (embed_dim), it could encode input sentences of certain length. There is no complete representation when there is a large input sentence. Some context of the sentence is lost in the encoder. When this lossy encoder output is given to the decoder, we get a translation that may not be similar to the source and most of the times shorter than the source sentence. We may increase the length of the encoder but that needs larger memory and training of more complex models leads to additional cost. [1] also has a similar hypothesis.

Question 4
Q4.1.
python train.py --encoder-num-layers 2 --decoder-num-layers 3 --cuda True
Note: We initially tried training on CPU, however it was taking very long. Hence, we decided to make minor code changes, in order to make the code compatible for training on GPU (on Google Cloud Compute).

Q4.2.



Q4.3. [Snippet of epochs near convergence is provided in ANNEXE]

	Baseline	**Layered Model**
Test BLEU score	7.98	6.33
Dev-Set Perplexity for best epoch	22.4	24.6
Training Loss for epoch achieving lowest validation perplexity	2.334	2.376

From the above table it can be observed that the layered model when compared to the baseline model - the BLEU score on the test set is lower, the dev-set perplexity has increased and the training loss has also increased. Which implies that the layered model has performed worse on the test, dev and training set.

The performance degradation could be due to stacking of LSTM layers and providing output of one layer as input to the next. From our team discussion, we concluded that the output of layer-1 will have some loss which when passed on to the next layer as input might enhance the loss. As the number of layers increase, this loss also amplifies. This might be the reason for reduced performance.

Question 5

Code changes made. [See code]

Question 6 [Snippet of epochs near convergence is provided in ANNEXE]

	Baseline	**Lexical Model**
Test BLEU Score	7.98	9.36
Dev-Set Perplexity for best epoch	22.4	22.1
Training Loss for epoch achieving lowest validation perplexity	2.334	1.757

The lexical model has a better performance on the training, dev and test set. This could be because it uses the attention weights to obtain the weighted average of the embeddings. The lexical model uses attention layer which uses the local context for the translation. This introduction of attention instead of using hidden layers must have been the reason for better performance, as it gives rise to better predictive distribution over the output words.

Question 7 [Attention Heat-maps included in the annexe]
Yes, there is a difference between the alignments learnt by the extended model as compared to the baseline.

Example Number	**Japanese Sentence**	**Baseline Translation**	**Lexical Model Translation**
1	良 い 休暇 を 。	i have a good time .	i am good vacation .
2	無知 は 幸福 。	i 'm sorry .	the pot is happiness .

In example 1, from the heat map we inferred that the lexical model was trying to predict words based on the probabilistic localised context of the input sentence. This is evident from the fact that “良”: Good and “休暇”: Vacation - have a high confidence of attention score. The baseline model on the other hand is trying to learn the patterns from the sentence.
In example 2, The first Japanese word is (translates to “ignorance”) is replaced by unknown. The baseline model could not predict the proper word for the unknown translation and could not catch the meaning of the sentence. The lexical model, however replaced the unknown word with the one with high attention score and the important context of the sentence "幸福" is translated with confidence as happiness.

It appears from the empirical evidence that, the lexical model provides a better translation than the baseline model. This may be because, the lexical model uses attention layer that takes into account the localised context for translation.

From our discussions we were unable to conclude as to which model provided the best translation in every case. For certain cases the baseline model provided better translations whereas the lexical model provided better translation while detecting nouns like camera, vacation, so and so on. These are cited as evidence in the following tables. Both the models failed to capture exact numeric values in their translation, although the lexical model could detect 1000 as a large number.

Few of the examples where the baseline provided better translations

Example Number	**Gold standard**	**Baseline translation**	**Lexical translation**
1	he burst into tears .	he began to cry .	he went into laughter .
2	will you have some coffee ?	do you want a cup of coffee ?	can you drink me ?
3	she goes to the movies once a week .	she goes to see the movies once a month .	she goes to see a movies once a small movies .

Few of the examples where the lexical model provided better translations

Example Number	**Gold standard**	**Baseline translation**	**Lexical translation**
1	she seems to have been happy .	she seems to have been in happy .	she seems to have been happy .
2	miyuki has a **camera** , but she doesn 't like it .	but she is rich , but she is rich .	the jeans has no **camera** that she has no **camera** .
3	i read at least one book every month .	i have read books at the best of this time .	i read the book in least a month .
4	this hall holds a maximum of **1,000 people .**	this machine is by a man of being a man .	this hall is the room of the **largest** .

Question 8

8.1

Hypothesis:

We believe that the sentences representing emotions or sentiments, for example, happiness/sad/good/bad/nice - should benefit from lexical information. As the local context is taken into account by the attention layer. These kind of sentences should be translated appropriately, unlike the case of baseline model where hidden layers just place the words representing the emotions/sentiments irrespective of the context.
To provide an example:
If the source sentence is - "My leaving early made them feel sorry" it should be translated as something not so good or a sentence representing a negative emotion.

8.2

To support our hypothesis, we have gathered evidence from the translations of baseline model as well as Lexical model.

We could infer that:

1. Simple adjectives/adverbs like good, easy, happy were translated better by the lexical model.
2. Some words like fierce, honest, proud were not translated at all.
3. There were incorrect translations for words like “temper” - though it represents a negative emotion, it is translated as "good chance" by the lexical model.

Example #	Gold standard	Baseline translation	Lexical translation
	Similar emotion lexical translations		
1	i just meant it as a joke .	it 's a time that i 've been to the joke .	it 's only to be a joke of joke .
2	the room had a nice cozy feel .	the room was good at the room .	the room was good with a good place .
3	i really enjoyed myself tonight .	i had a good time today .	it was a chance today .
4	she seems to have been happy .	she seems to have been in happy .	she seems to have been happy .
5	he is a frank person and easy to talk to .	he is a good person of his mind .	he is a friendly person .
6	he is fine and gentle .	he is good at any good .	he is easy to come .
7	we are crying .	we are looking at all .	we are cry .
8	he bothered her with questions .	he was a questions in her .	he made up a good question at her .
	Incorrect and not so good translations		
9	his face and attitude showed the scorn he felt .	the first feelings was a big watch in the face .	his face were only to find the top of his face .
10	she may well be proud of her son .	she is a good son of son .	she is a good people of her son .
11	don 't press your opinion on me .	don 't make my idea .	don 't share the ideas of your ideas .
12	everyone is not honest .	nobody has no sense of mind .	nobody cannot have no drink with everyone .

13	father went red with anger when i behaved rudely towards him .	my father was put on him when i was a little angry .	father was sorry when i was sorry to him .
14	she is in a temper , because she missed her usual train in the subway and had to walk to work .	she is always up to the work , but i always get up to the work .	she 's a good chance , so she 's been to the same man to the foot of the party .
15	dan is often scolded by his mother .	the crowd will be more than my mother .	the jeans is more than mother mother .
16	he glared at me fiercely .	he made me a very good way .	he put me a great deal of great years .
17	i believe in you .	i 'm looking for you .	you look down .
18	he burst into tears .	he began to cry .	he went into laughter .
19	i saw a bird flying over a tree .	the concert is made to the number of order to put the economy is to the full of order to put the economy .	the jeans will be held as a great deal of economy .
20	he is a friendly person .	the crowd will be more than my mother .	the jeans is more than mother mother .

8.3

We believe that simple sentences (like "we are crying") are translated better than the ones that represent complex emotions (like "fierce").

However, we are unable to arrive at a definitive conclusion based on the empirical evidence.

May be this is due to:

1. There are large number of unknown words in the training data.
2. Japanese words when taken sequentially represent a particular sense of context and in combination have a different one.

For example -

‘過ぎ去’ represents "The past" and ‘っ た’ represents "it was" whereas these two together 過ぎ去+っ た represent "gone or passed"

3. May be if we are provided with larger training data, we might be able to support our hypothesis with better proof.

For the sentence in example #18, the baseline model translated better than the lexical one. Our hypothesis that the lexical model should handle emotions better was proved false by that example #18.

Also we ran into another problem with attention. There are few words in translations which are not present in the source. The baseline translation doesn’t have these words whereas they are present in lexical model translations.

Ex: jeans

Jeans is not present in Japanese test set corpus. While it is observed in lexical model translations. This might be due to an unknown word which was replaced with the one with highest attention score. [Examples 19,20]

REFERENCES

[1] Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, Yoshua Bengio. *On the Properties of Neural Machine Translation: Encoder–Decoder Approaches,* https://arxiv.org/pdf/1409.1259.pdf

ANNEXE

Q4.3

INFO: Epoch 050: loss 2.404 | lr 0.0003 | num_tokens 10.31 | batch_size 1 | grad_norm 47.79 | clip 1
INFO: Epoch 050: valid_loss 3.23 | num_tokens 10.1 | batch_size 500 | valid_perplexity 25.2
INFO: Epoch 051: loss 2.399 | lr 0.0003 | num_tokens 10.31 | batch_size 1 | grad_norm 47.75 | clip 1
INFO: Epoch 051: valid_loss 3.22 | num_tokens 10.1 | batch_size 500 | valid_perplexity 24.9
INFO: Epoch 052: loss 2.39 | lr 0.0003 | num_tokens 10.31 | batch_size 1 | grad_norm 47.63 | clip 0.9999
INFO: Epoch 052: valid_loss 3.21 | num_tokens 10.1 | batch_size 500 | valid_perplexity 24.8
INFO: Epoch 053: loss 2.376 | lr 0.0003 | num_tokens 10.31 | batch_size 1 | grad_norm 47.69 | clip 0.9999
INFO: Epoch 053: valid_loss 3.2 | num_tokens 10.1 | batch_size 500 | valid_perplexity 24.6
INFO: Epoch 054: loss 2.363 | lr 0.0003 | num_tokens 10.31 | batch_size 1 | grad_norm 47.98 | clip 0.9999
INFO: Epoch 054: valid_loss 3.2 | num_tokens 10.1 | batch_size 500 | valid_perplexity 24.6
INFO: Epoch 055: loss 2.356 | lr 0.0003 | num_tokens 10.31 | batch_size 1 | grad_norm 47.82 | clip 0.9999
INFO: Epoch 055: valid_loss 3.21 | num_tokens 10.1 | batch_size 500 | valid_perplexity 24.7
INFO: Epoch 056: loss 2.342 | lr 0.0003 | num_tokens 10.31 | batch_size 1 | grad_norm 47.61 | clip 1
INFO: Epoch 056: valid_loss 3.22 | num_tokens 10.1 | batch_size 500 | valid_perplexity 25
INFO: Epoch 057: loss 2.331 | lr 0.0003 | num_tokens 10.31 | batch_size 1 | grad_norm 48.1 | clip 1
INFO: Epoch 057: valid_loss 3.21 | num_tokens 10.1 | batch_size 500 | valid_perplexity 24.7
INFO: Epoch 058: loss 2.326 | lr 0.0003 | num_tokens 10.31 | batch_size 1 | grad_norm 48.06 | clip 1
INFO: Epoch 058: valid_loss 3.2 | num_tokens 10.1 | batch_size 500 | valid_perplexity 24.6
INFO: No validation set improvements observed for 5 epochs. Early stop!

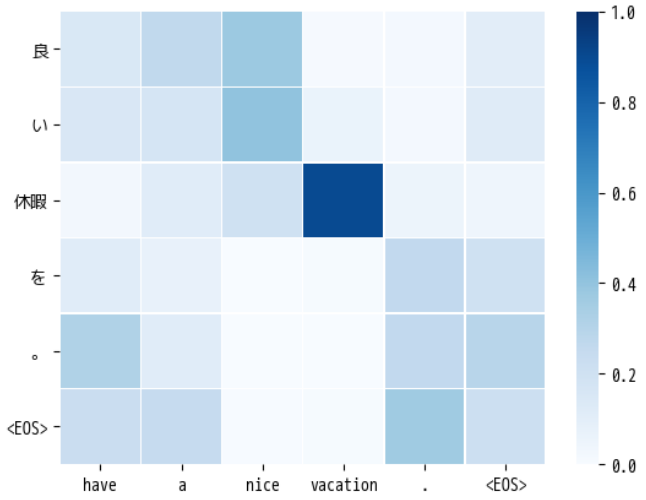
Question 6

INFO: Epoch 021: loss 1.921 | lr 0.0003 | num_tokens 10.31 | batch_size 1 | grad_norm 61.58 | clip 0.9996
INFO: Epoch 021: valid_loss 3.12 | num_tokens 10.1 | batch_size 500 | valid_perplexity 22.8
INFO: Epoch 022: loss 1.883 | lr 0.0003 | num_tokens 10.31 | batch_size 1 | grad_norm 61.56 | clip 0.9997
INFO: Epoch 022: valid_loss 3.11 | num_tokens 10.1 | batch_size 500 | valid_perplexity 22.3
INFO: Epoch 023: loss 1.85 | lr 0.0003 | num_tokens 10.31 | batch_size 1 | grad_norm 61.58 | clip 0.9995
INFO: Epoch 023: valid_loss 3.11 | num_tokens 10.1 | batch_size 500 | valid_perplexity 22.5
INFO: Epoch 024: loss 1.818 | lr 0.0003 | num_tokens 10.31 | batch_size 1 | grad_norm 61.61 | clip 0.9995
INFO: Epoch 024: valid_loss 3.11 | num_tokens 10.1 | batch_size 500 | valid_perplexity 22.5
INFO: Epoch 025: loss 1.786 | lr 0.0003 | num_tokens 10.31 | batch_size 1 | grad_norm 61.75 | clip 0.9997
INFO: Epoch 025: valid_loss 3.1 | num_tokens 10.1 | batch_size 500 | valid_perplexity 22.3

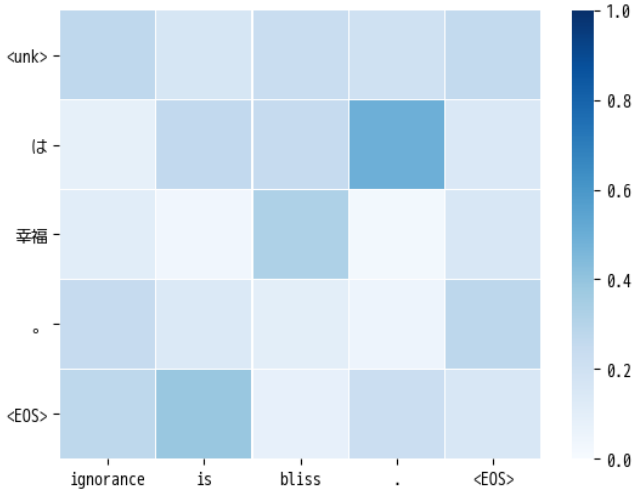
INFO: Epoch 026: loss 1.757 | lr 0.0003 | num_tokens 10.31 | batch_size 1 | grad_norm 61.74 | clip 0.9994
INFO: Epoch 026: valid_loss 3.1 | num_tokens 10.1 | batch_size 500 | valid_perplexity 22.1
INFO: Epoch 027: loss 1.728 | lr 0.0003 | num_tokens 10.31 | batch_size 1 | grad_norm 61.68 | clip 0.9989
INFO: Epoch 027: valid_loss 3.1 | num_tokens 10.1 | batch_size 500 | valid_perplexity 22.3
INFO: Epoch 028: loss 1.705 | lr 0.0003 | num_tokens 10.31 | batch_size 1 | grad_norm 61.77 | clip 0.9987
INFO: Epoch 028: valid_loss 3.11 | num_tokens 10.1 | batch_size 500 | valid_perplexity 22.5
INFO: Epoch 029: loss 1.676 | lr 0.0003 | num_tokens 10.31 | batch_size 1 | grad_norm 61.8 | clip 0.9988
INFO: Epoch 029: valid_loss 3.13 | num_tokens 10.1 | batch_size 500 | valid_perplexity 22.9
INFO: Epoch 030: loss 1.656 | lr 0.0003 | num_tokens 10.31 | batch_size 1 | grad_norm 61.79 | clip 0.9985
INFO: Epoch 030: valid_loss 3.11 | num_tokens 10.1 | batch_size 500 | valid_perplexity 22.3
INFO: Epoch 031: loss 1.629 | lr 0.0003 | num_tokens 10.31 | batch_size 1 | grad_norm 61.69 | clip 0.9984
INFO: Epoch 031: valid_loss 3.1 | num_tokens 10.1 | batch_size 500 | valid_perplexity 22.3
INFO: No validation set improvements observed for 5 epochs. Early stop!

Question 7

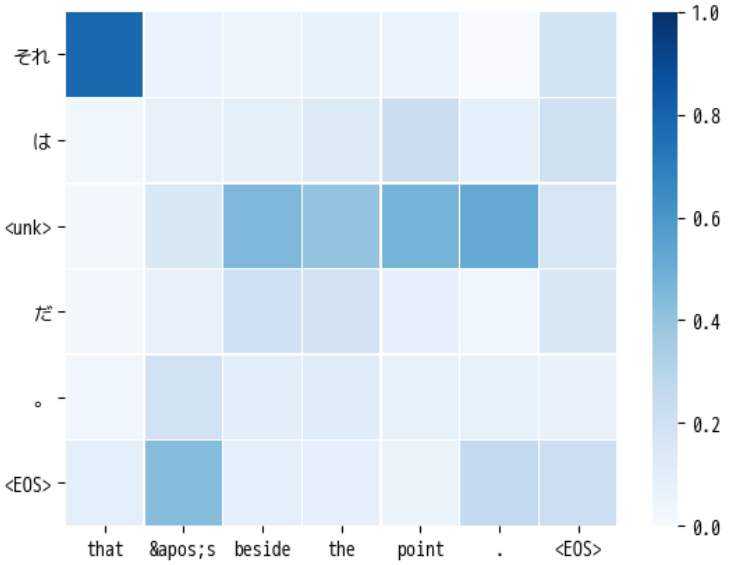
Visualizations Baseline



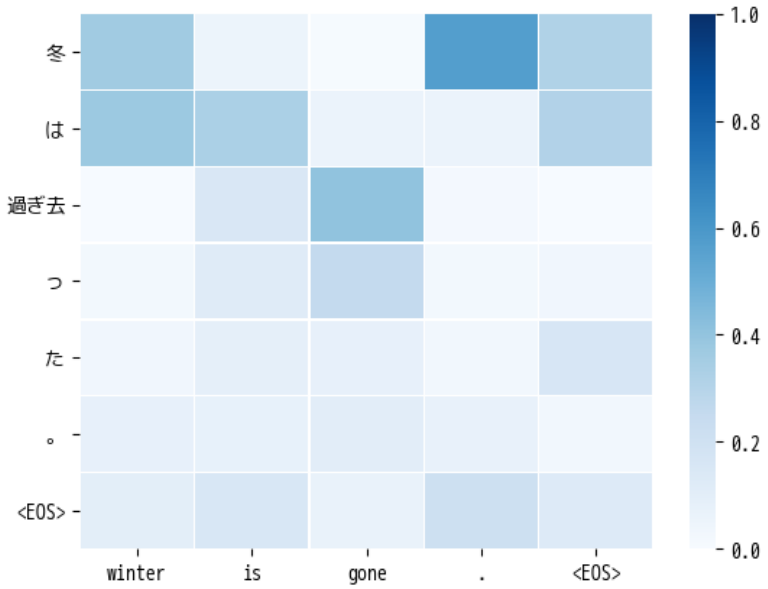
Sentence 1



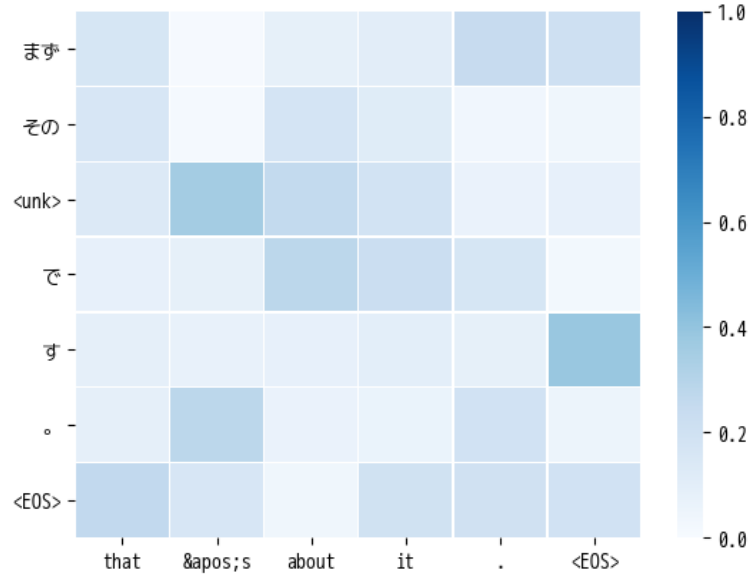
Sentence 2



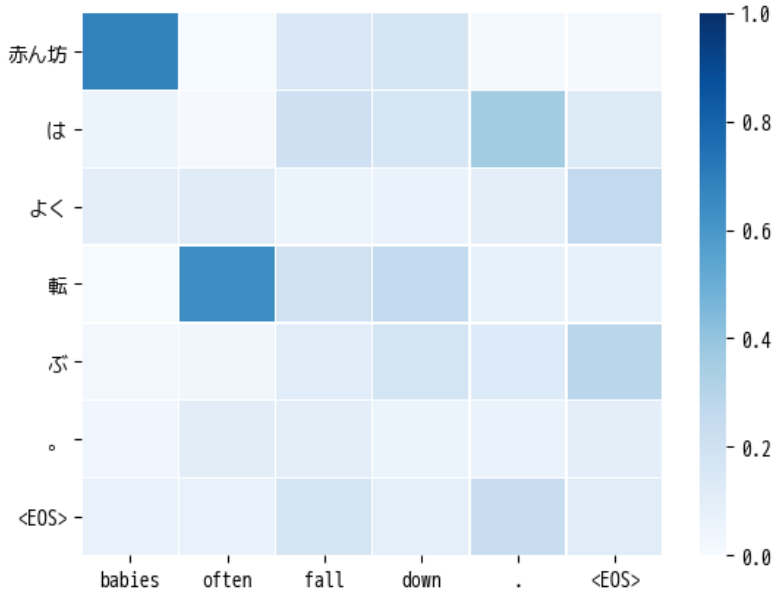
Sentence 3



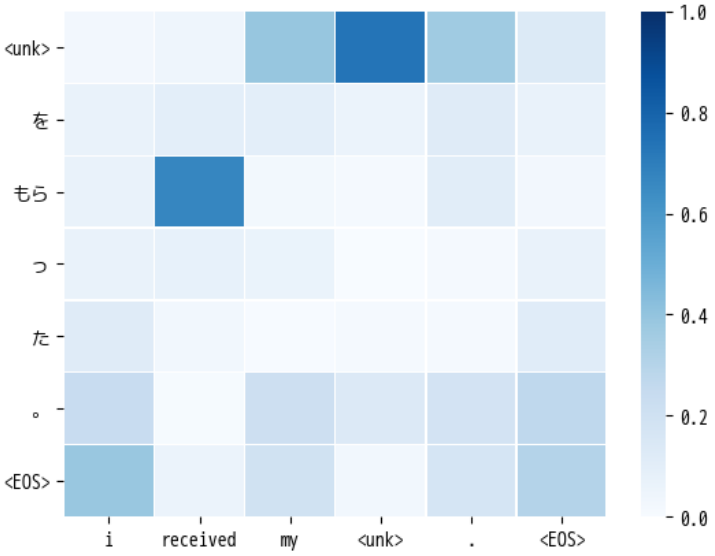
Sentence 4



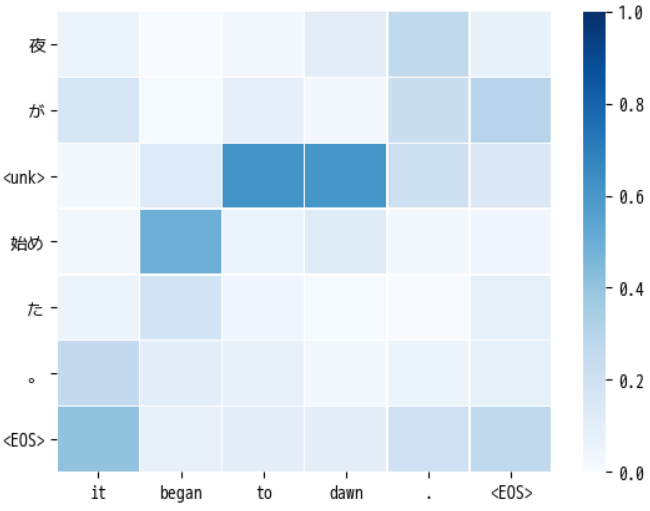
Sentence 5



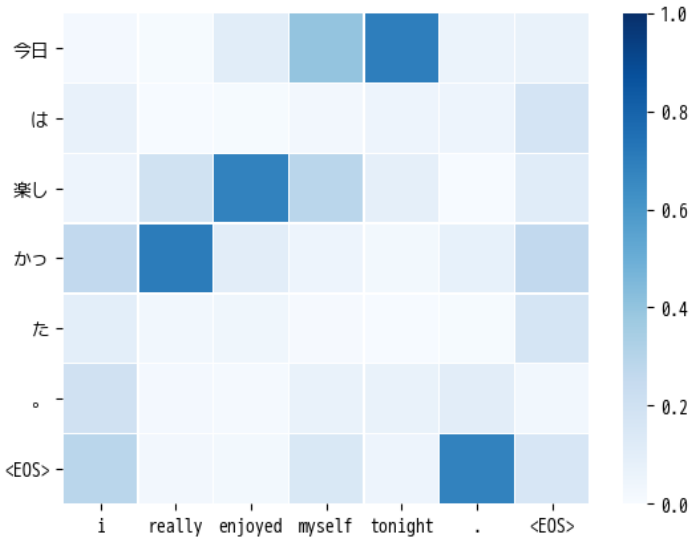
Sentence 6



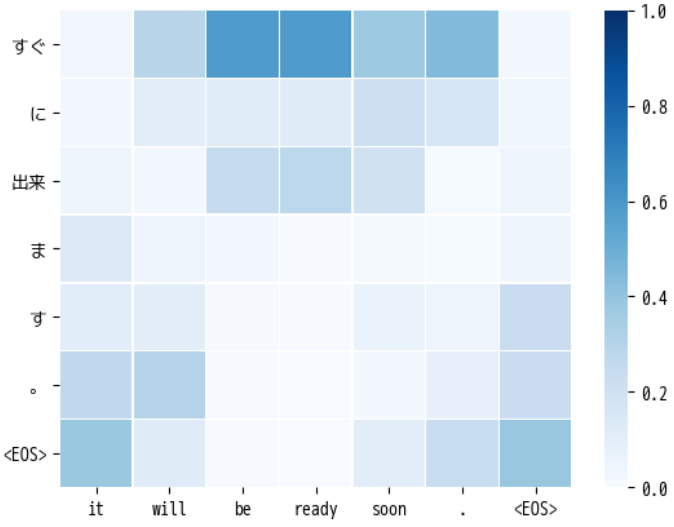
Sentence 7



Sentence 8

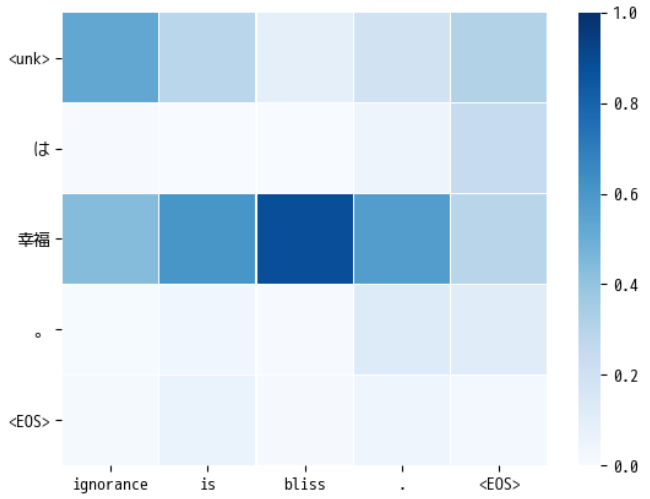


Sentence 9

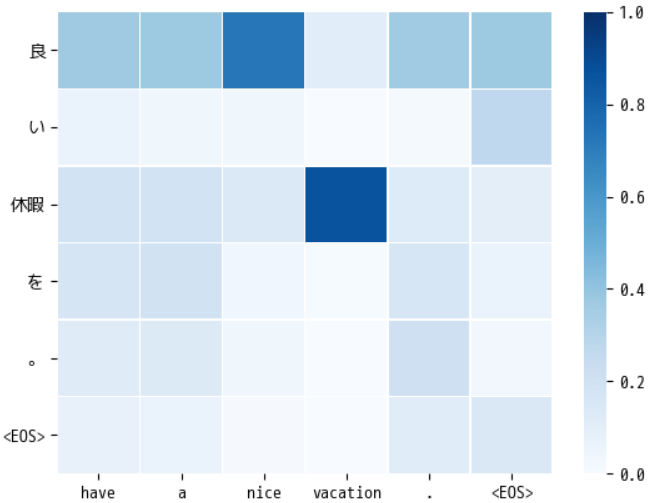


Sentence 10

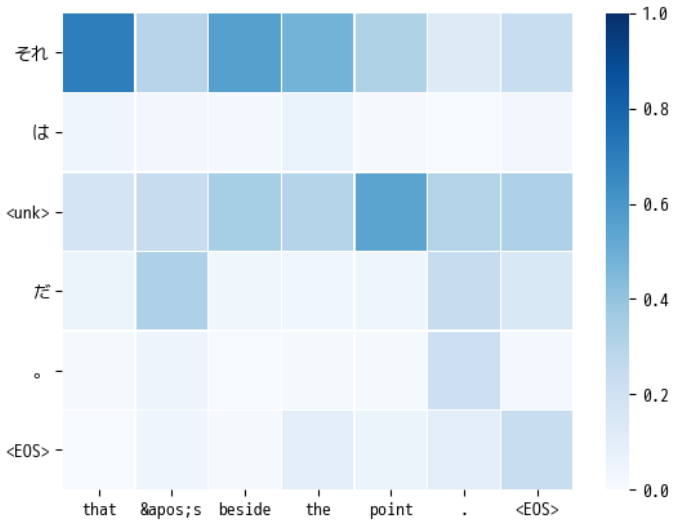
Visualizations Lexical Model



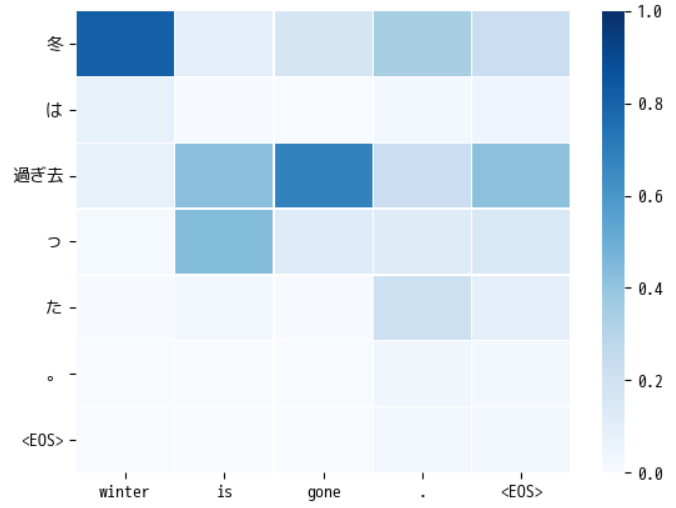
Sentence 1



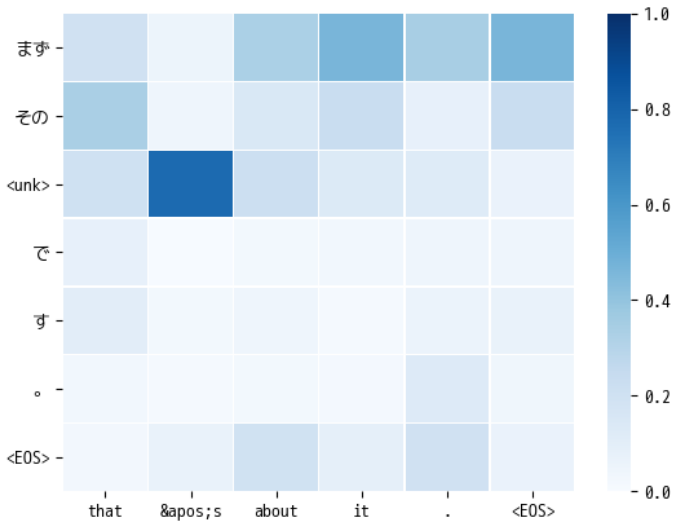
Sentence 2



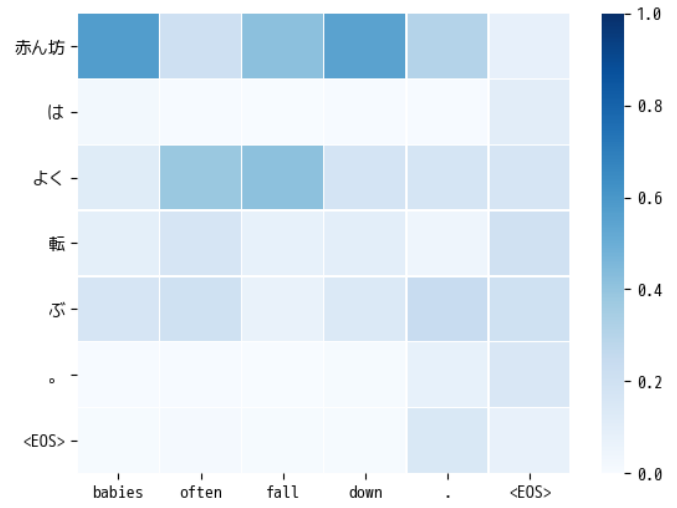
Sentence 3



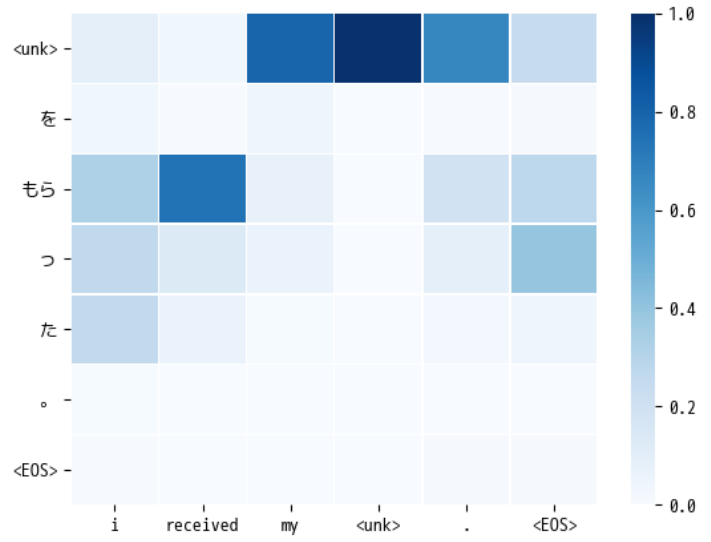
Sentence 4



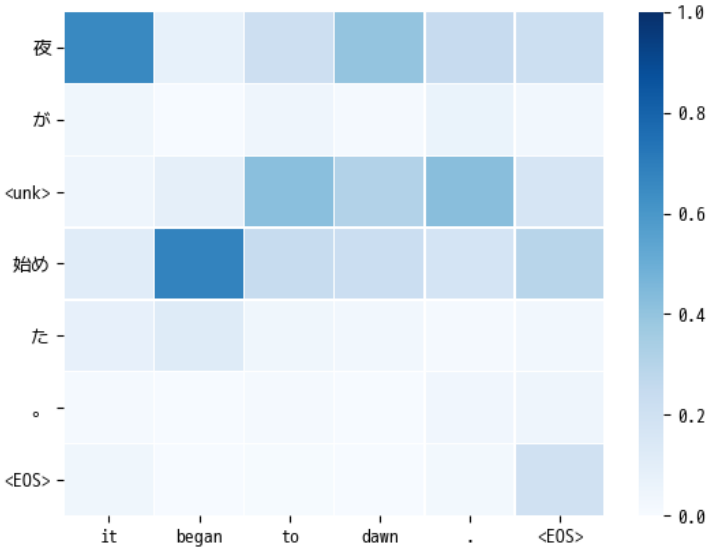
Sentence 5



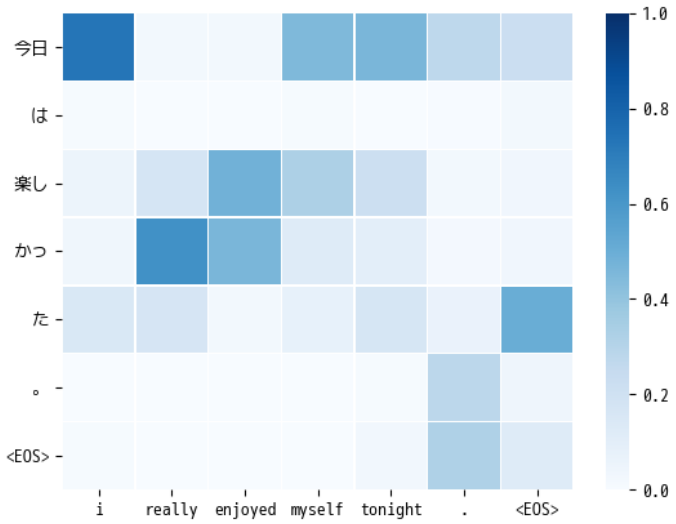
Sentence 6



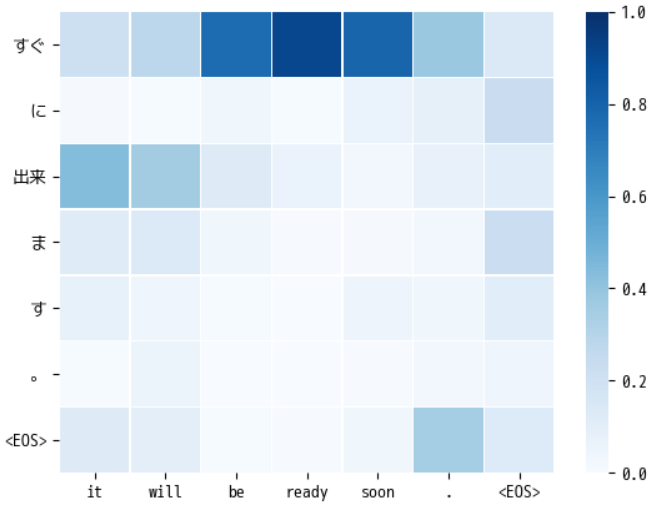
Sentence 7



Sentence 8



Sentence 9



Sentence 10