Natural Language Understanding, Generation and Machine Translation
Coursework 2: Neural Machine Translation

The University of Edinburgh, School of Informatics
s1886258, s1789342

March 14, 2019

Question 1: Baseline Model

Comments:

ANSWER A:
When self.bidirectional=True, the final hidden states and the final cell states in both left-to-right and right-to-left directions are concatenated to form a single final_hidden_state and a single final_cell_state of the encoder input sequence.
The final_cell_states are the internal states of each cell or the memory of the cell.
The final_hidden_states are the scaled and transformed versions of the cell_states depending on the output gate.

ANSWER B:
To calculate the attention context:
1. We calculate the attention weights by normalizing the calculated attention scores using softmax.
2. Then calculate the attention context vector by performing a matrix multiplication of the attention weights and the encoder outputs.
Mask:
When using the batch mode, the input sequence is padded with zeros based on the length of the longest sequence in the batch. We use the mask to fill -inf in the attention score tensor at all the positions that were padded in the shorter sequences in the input.

ANSWER C:
Attention scores are calculated by multiplying the linear projections of the encoder output with the target input. Matrix Multiplication (torch.bmm) takes as input the decoder target input representation with dims after unsqueeze [batch_size,1,input_dims], and the encoder projected outputs with dims before transpose [batch_size,src_time_steps, output_dims], to give attention scores with dims [batch_size,1,src_time_steps] thus, aligning the encoder and decoder representations.

ANSWER D:
The decoder is incrementally initialized, meaning that it receives a previous output and produces the next output. The decoder states, i.e. the target hidden states, the target cell states and the input feed are initialized to zeros when no cache exists. If the cache exits, the target cell and hidden states are initialized to the previous hidden and cell states. The input feed is initialized to the input feed from the previous step. The previous step input feed corresponds to the dropped out version final target hidden state, if not using attention or the dropped out version of the attention outputs, if attention is applied (Assuming dropout is used). The model caches any long-term history needed for the translation of the sequence. However, cached_state is activated after the first word of the sequence, as only then the model is able to start storing representations, when there is an available previous hidden state. Input feed refers to the approach of feeding the current token embedding to the next time step in order to inform the model about alignments of the past.

ANSWER E:
When attention is enabled, the decoder uses the attentional hidden state computed by the attention layer to predict the next word in the translation sequence. When there is no attention, the decoder considers only the last hidden state of the encoder.
The current target hidden state is one of the inputs to the attention function, as it is used to compute the attention scores by comparing it to the source hidden state. Once the context vector is generated, the current target hidden state is also concatenated with the context vector to produce the attentional hidden state.
Dropout:
Large neural networks, especially the ones using a small data-set for training, suffer from the problem of over-fitting thus, increasing the generalization error. Dropout layer is added as a form of regularization in order to avoid over-fitting.

ANSWER F:
The first line invokes the model to run the encoder-decoder on the sampled inputs and reference outputs as per the batch size, to generate the output predictions. The next line invokes the criterion to calculate the cross entropy loss on the training data. The loss.backward() accumulates the gradients over time for every parameter of the network. As these gradients are added over numerous time steps, they tend to explode and hence, we use grad norm to clip these gradients to the threshold (here 4). The optim.step() function applies these gradients to the network parameters to obtain the updated parameters. The optim.zero_grad_() clears the gradients (sets to zero) before the next batch starts.

Question 2: Understanding the Data

2.1: Sentence Distributions and their correlations

The distribution of sentence lengths in English and Japanese is shown in Figure 1 below.

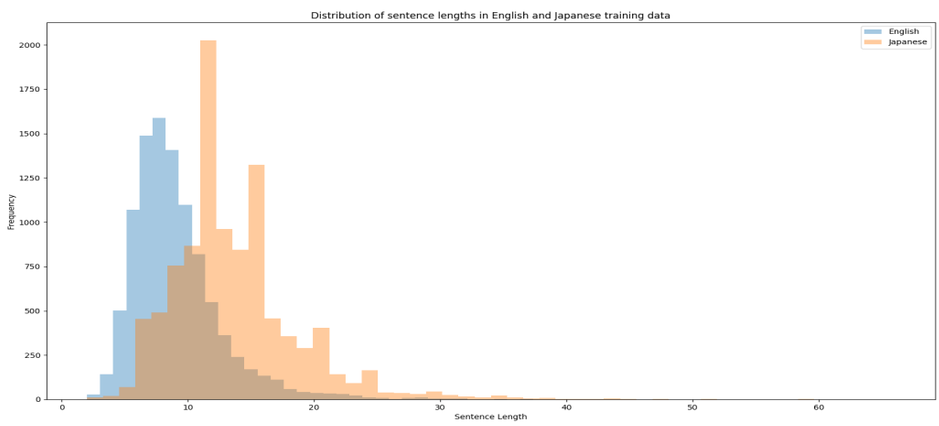


Figure 1: Distributions of lengths of sentences in the training set for source language Japanese and target language English

As seen from the distributions, English sentences are on average much shorter than the Japanese ones. The correlations between the sentence lengths, shown in Figure 2, indicate that the sentence pairs do not exactly follow strong correlation patterns, for example 10 word token sentences in Japanese can have English translation lengths ranging from anything between 2 and 16 in the training corpus. This is expected given that in general, languages, and especially in the case of translating from Japanese to English, i.e. from a morphologically richer language to a morphologically poorer one, usually do not have an one-to-one correspondence. Moreover, this can be explained from to the fact that Japanese is an agglutinative language, a synthetic type language based on aspects of derivational morphology for the construction of its meanings, while English is fusional, a synthetic language based on a single inflectional morpheme to denote various linguistic features [8]. Sentences in Japanese would thus be tokenised to individual morphemes that represent the modifiers, like the verbal tense, along with the base forms [2], resulting in longer sentences. Each individual morpheme,

when slammed with other morphemes to form a sentence, could mean a completely different thing. The morpheme [頼] alone means 'request', the morpheme [り] defines the article 'the', but when they are combined, as in [頼り], the generated morpheme is translated into 'reliance' (generated using Google translate). This would cause potential problems when translating between these two languages.

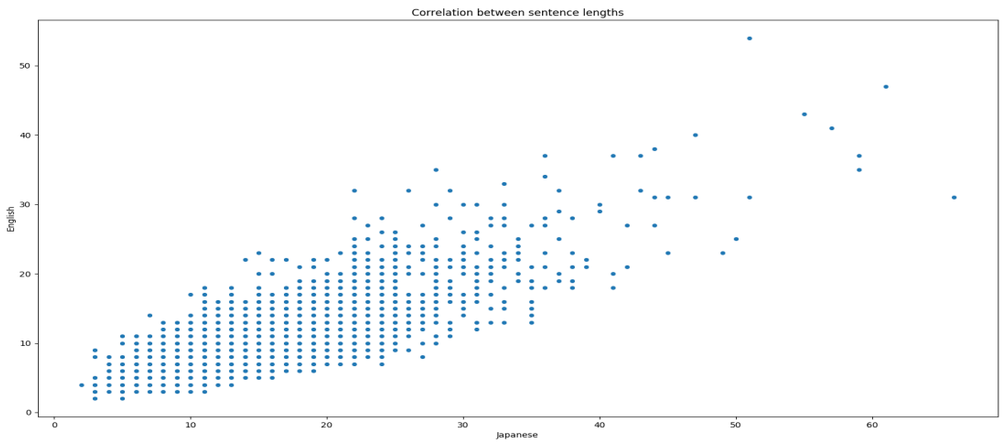


Figure 2: Correlations between lengths of sentences in the training set for source language Japanese and target language English

2.2: Word Tokens

Number of word tokens in English are: 93086
Number of word tokens in Japanese are: 136899

2.3: Word Types

Number of word types in English are: 7040
Number of word types in Japanese are: 8058

2.4: Unknowns

Number of words that will be replaced by <UNK> in English: 3331
Number of words that will be replaced by <UNK> in Japanese: 4113

2.5: Discussion on observation of data

Difference in Sentence Lengths and UNKs
The differences in sentence lengths between the source and target language indicate many-to-one as well as one-to-many relations between the source and the target words which can complicate the translation procedure. Moreover, the words that occur only once are replaced by UNK. These words could be the open class words in the vocabulary and replacing them with UNK, would further increase the ambiguity of the sentences.

Type-Token Ratios
Type-to-Token ratios (TTR) are used as a measure of lexical diversity. [3] The higher the ratio, the more diverse the vocabulary and vice versa. Based on the data, the TTR for Japanese is 0.05 whereas for English is 0.07 in the original text. While the ratio for English is slightly better than Japanese, it itself, is not a very good ratio. This indicates that the diversity in the given data is low and can possibly introduce repetitions in the translations.

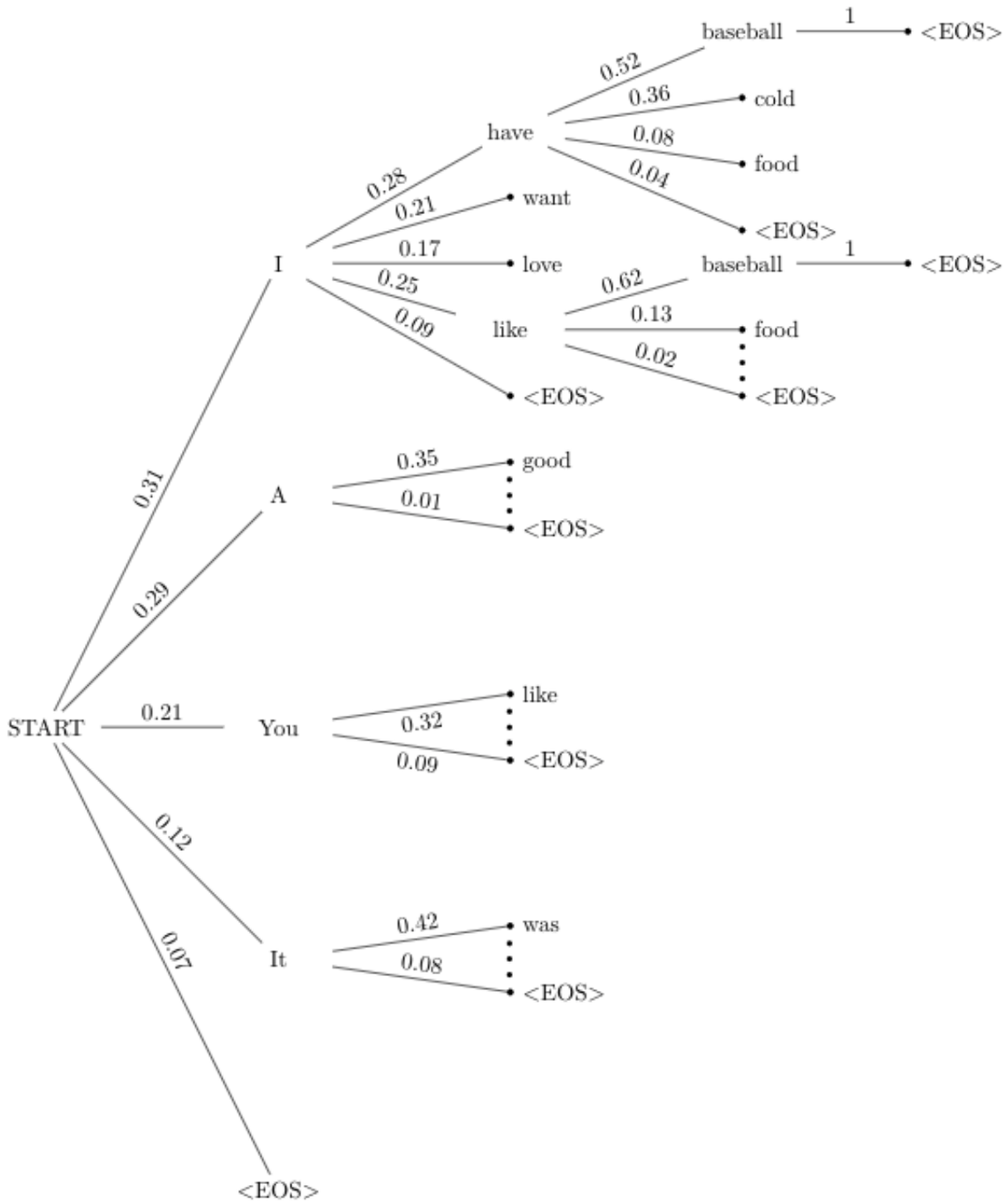
Unknown word handling
The given model employs a back-off mechanism to handle unknown words. Particularly, it replaces single frequency words with UNK. As these words are likely to contain nouns and inflected forms,

with the back-off mechanism the model would find it hard to correctly translate these inflected words and the translation quality would thus, deteriorate.

Question 3: Improved Decoding

3.1: Greedy Decoding

Greedy decoding is a heuristic search algorithm which, with respect to machine translation, is used to find the optimal translation from a source language to a target language. The aim of this algorithm is to generate a 1-best result by calculating the probability p_t at each time step in order to select the most likely word to be used as the next word in the output sequence [9].
However, the results of greedy search can be far from optimal. As it selects only the most probable path at every time step, the possibility of selecting other sentences more likely to output a better translation are reduced. This happens because the distributions at the next steps depend on the decisions made at the previous steps, and since there is no way of back tracking, greedy decoding cannot recover from its mistakes. [1]. Consider the example shown below. The source sentence '私は野球が好きだ ' has the correct translation 'i like baseball' but greedy search chooses 'have' instead of 'like' at the second step and ends up producing 'i have baseball', instead of 'i like baseball', which has an overall higher probability.



3.2: Beam Search

A solution to this problem is to use beam search. This is another heuristic approach, similar to greedy search, but instead of one, it considers B likely hypotheses at every time step. B stands for the 'width' of the beam, meaning the number of active hypotheses taken into consideration at each time step.
Beam search can be implemented by calculating the conditional probabilities of all the words in the vocabulary V starting from time step t = 1. The B most likely words are chosen and B hypotheses are initialized. The conditional probabilities of all the words in the vocabulary are now computed in terms of each of these hypotheses, resulting in B x $|V|$ candidates. The procedure is repeated until the final B most likely candidates are chosen.
Experiments have shown that the size of the beam plays an important role in improving the quality of the translation. However, while the translation outputs tend to be better when the beam size is narrow, for larger beam sizes the translation quality decreases. The reason for this is that wider beam bands tend to favor shorter sentences. [5].

3.3: Length Normalization

In machine translation models, the translation output is the product of the conditional probabilities of each translation at each time step given the previous time step and the source sentence:

$$P(E|F) = \prod_{t}^{|E|} P(e_t|F, e_1^{t-1})$$

Such models tend to favor shorter sentences due to the fact that, the partial translations stored alongside with their probabilities grow exponentially with every new word output, reducing the probability of the whole sentence. The larger the beam size, the easier it is for the model to generate better translation outputs for shorter sentences. However, this causes the model with a larger beam size to have a length bias towards the shorter sentences [9].
One approach to solve this problem is the normalization of the log probability by the length of the target sentence aiming at finding the sentence with the highest log probability per word on average [1].

$$\hat{E} = \arg\max_{E} log P(E|F)/|E|$$

However, length normalization can introduce over- and/or under-translation problems [7]. Over-translation refers to the problem of the target language containing more information than the source language while under-translation refers to the exact opposite problem; the target language contains less information than the source language. Normalizing the translation of variable length sentences means that there is a chance of repeating or omitting words in order for the model to cover the amount of words in the source language to the reference translation. Articles or auxiliary verbs could be an example of such a behavior as, such word types vary linguistically across languages and handling these cases might be problematic.

Question 4: Deeper Architecture

4.1: Training the deeper model

Command used :

```
python train.py --encoder-num-layers=2 --decoder-num-layers=3 \
    --save-dir ./q4_checkpts/
```

4.2: The Deeper Architecture

Figure 3 below shows the deeper architecture with 2 bidirectional encoder layers and 3 unidirectional decoder layers. The figure shows the state at decoding time step t; $h_t^l(0)$ and $c_t^1(0)$(not shown in the figure as it is the internal memory of the LSTM cell) are 0 if t=0, and previous target hidden and target cell states respectively, if t is not 0. The projected output from each time step goes to the next time step (not shown in the figure for timesteps >t). Dropout is indicated wherever applied

but not included as a separate layer in the figure.

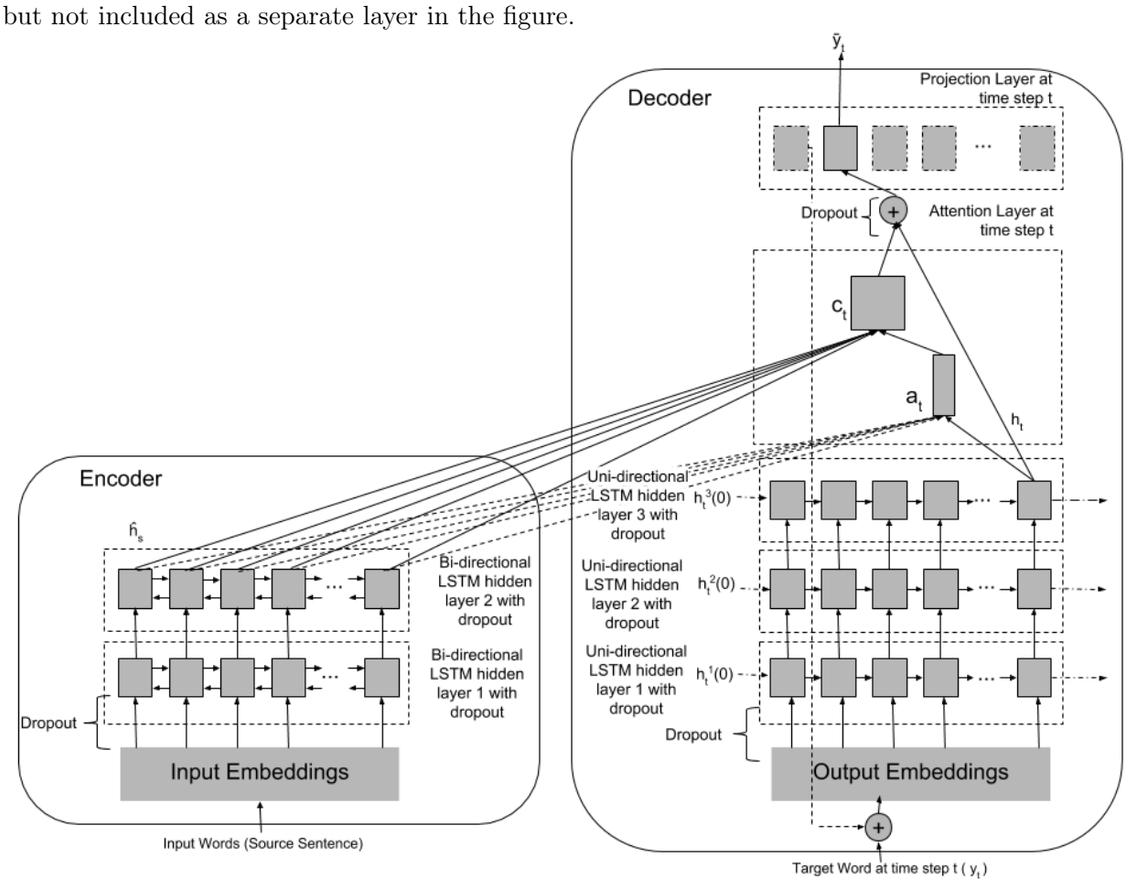


Figure 3: The Deeper Architecture

4.3: Effects of deeper architecture

The addition of more layers to the baseline model has affected its performance slightly. The results of this deeper architecture are:

- Training loss: 2.508
- Test set BLEU: 5.81
- Dev set perplexity during last epoch: 26.8

The results indicate a deterioration in the model's performance compared to the baseline one. Particularly, the training loss and the perplexity of the model have increased and the BLEU score has decreased.
In general, as the hypothesis space of the neural network gets bigger, the algorithm will be able to learn richer structures. A network with reasonably deep layers is considered better at generalization than a shallow one. As the size of the network starts to grow however, beyond a point, deeper architectures start overfitting to training data. Overfitting is possible when the architecture is not extremely deep as well, but the amount of training data used is small. For our training, we use a data set with 10000 sentences, which is small considering the task at hand. During training, we observe that while the loss on the dev set keeps decreasing after the 47th epoch, the validation loss, as well as the perplexity, starts to increase.
To further investigate the hypothesis, we carried out a quick test on the tiny data set using 2 encoder and 2 decoder hidden layers, as well as 3 encoder and 2 decoder hidden layers. We found similar patterns in both these tests, the dev loss keeps decreasing but the validation loss and the perplexity start increasing after the 8th and the 5th epochs for the two tests, respectively. Thus, there is reasonable evidence that the deeper architecture with 2 encoder hidden layers and 3 decoder hidden layers, with the small training set, has overfit to training data. The BLEU score in addition, is lower than the one for the baseline, as expected.

Question 6: Effects on Model Quality

The results on the lexical model are:

- Training loss: 2.079
- Test set BLEU: 10.09
- Dev set perplexity during last epoch: 20.2

As compared to the baseline model, the training loss and the dev perplexity have decreased and the BLEU score has increased by +2, indicating an improvement over the baseline model. This is due to the fact that the model can now translate words based on attention on the source words and not solely on the context, and will therefore, generate relatively more accurate translations for the low frequency words as compared to the baseline. This is also evident from the individual n-gram scores used for calculation of BLEU, as seen in Table 1.

	1-gram	2-gram	3-gram	4-gram
Lexical Model	44.0	15.2	6.5	3.0
Baseline Model	40.1	12.2	4.9	2.1

Table 1: n-gram precision scores

The model still uses a small data set for training and more importantly a back-off UNK replacement method and hence, does not guarantee proper nouns to be translated properly as in [10], However, it manages to translate the words present in the corpus, properly. The Table 2 shows a translation for the same source sentence from the baseline and lexical model.

	Translation
Source	**カナダ** へ行ったことがありますか。
Reference	have you ever been to **canada** ?
Baseline Model	have you ever been to **the movies** ?

Table 2: Sample translations from the models

Question 7: Effects on Attention

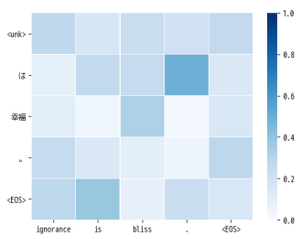
In order to evaluate the performance of the lexical model, we need to visualize the attention weights that are learned by the two models. A comparison of the heatmaps shows that the lexical model attends the words better, indicating its capability of learning better alignments between the source and the target language. The lexical model is better at attending the correct source word as well as capturing the morphological information and hence, produce sentences that are closer to the source language in terms of grammaticality and meaning.
The results can, firstly, be explained by the architecture of the two models. Both models implement a recurrent encoder-decoder network, which encodes a source sentence $x = (x_1, ..., x_l)$, where l is the length of the sequence, into word embeddings and then, into a sequence of $h = (h_1, ..., h_l)$ hidden states. The decoder outputs the words of the target sentence with the use of a global attention mechanism. Particularly, at every time step t, the decoder generates a set of attention weights $a_t(s)$ which form a weighted average of the encoder's hidden states in order to create a context vector c_t. Then, the decoder output are created from the context vector and the hidden states of the decoder. The probability of each generated word is normalized using softmax. Finally, the output word distribution is parameterized with $||\widetilde{h}||$, which indicates the contextual confidence of the model, the $\cos\theta_{W_{e,\widetilde{h}}}$, which shows how much each word fits into a context and b_e and $||W_e||$, which control the amount of each word e generation.

$$p(e) \propto \exp(||W_e||||\widetilde{h}||\cos\theta_{W_{e,\widetilde{h}}} + b_e)$$

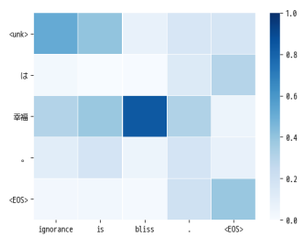
The baseline implements the above model which results in it producing translations which are based on the context. The outcome is the production of translations that can be fluent at times but not accurate, neither close at being accurate. Greedy search will pick the language pairs with the highest probability without considering any morphological or semasiological information. The lexical model implements an improved architecture. While also using gredy decoding, it takes advantage of the

information contained in the source embeddings, which include the information the baseline model was not taking into consideration; information about the words themselves in order to produce more accurate results.[6].

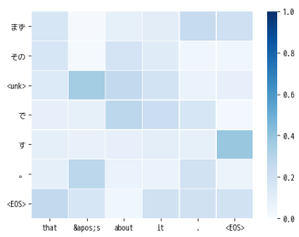
Below, there are three pairs of baseline-lexical model translations heatmaps. The attention weights in both models attend almost the same words but with a different correlation strength among the other words. The lexical model attends the words with stronger attention weights, indicating its ability for both a per-word translation and syntactical correlation between the words. In the first translation model pair, the word *bliss* is attended more in the lexical model while the baseline one is more confused. Indeed the word being attended in Japanese means 'happy' and hence, the attention is correct in the lexical model. In the second pair, the attention of the lexical model indicates its ability to learn syntactic information, as this sentence contains more structural than lexical complexity. In the last pair, the lexical model, again manages to learn good alignments. It attends to the fullstop when generating apos which can be justified as they are both punctuations. The baseline on the other hand, attends to EOS. The lexical model also, strongly attends to <unk> when producing 'point' and it indicates that it will translate to the right word, if we use a better <UNK> replacement method. The baseline model fails to strongly align with the source word, it will likely use the context to translate the word. The performance of the lexical model is expected due to its improved architecture and proves that, considering the source words directly along with the context information, helps the model learn better word alignments between languages. More visualizations of the attentional weights can be found in the Appendix.



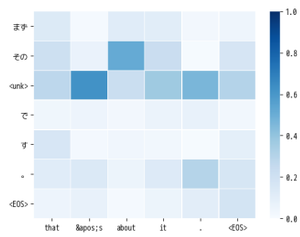
(a) Baseline Sentence Visualization



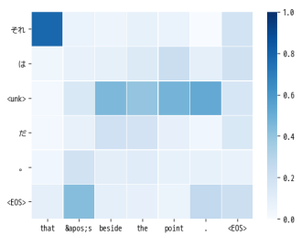
(b) Lexical Model Sentence Visualization



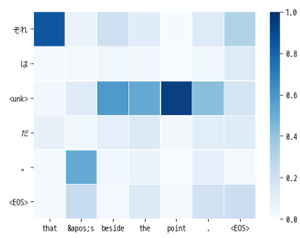
(a) Baseline Sentence Visualization



(b) Lexical Model Sentence Visualization



(a) Baseline Sentence Visualization



(b) Lexical Model Sentence Visualization

Question 8: Effects on Translation Quality

Motivation and Hypothesis

Translating between Japanese and English , in either direction, is a difficult task and translations cannot always be perfect or close to the source language both from the human and the machine translation perspective. The reasons are due to the morphology and syntactic structures of the Japanese language.
The writing system itself consists of three scripts, Hiragana and Katakana, which are two different versions of the same set of sounds, and Kanji, from which most of the words are generated. Kanji differs from English in the way that meaning and structure convention comes from various strokes on complex character sets, like stroke placement, placement in between characters, as well as different ways of interpretation. This leads to the English language not being able to capture the syntax and semasiology, as there is no true English equivalent, neither with the structure nor the meaning. The lack of English equivalents leads to the generation of non literal translations, especially in the cases of complex sentences and sayings. The lack of translational equivalence also has its basis on the grammatical differences between the two languages [4]. Major differences include - Japanese language does not contain definite and indefinite articles or plural forms; it uses different morphology to measure singularity and plurality. In the case of verbs, their meaning comes from different structural particles. Particles are considered very important linguistic features in Japanese. Although they have no English equivalent, they cannot be omitted since they add nuance to the meaning through different syntactical structures. This makes the layout of the language more complex to decode and transfer into another language with more simple morphology, like English. They are also hard to keep track of since there are almost 188 particles included in Japanese. Last but not least, due to this syntactical and morphological complexity, it is almost impossible to maintain an equivalent number of words during translation. The meaning is conveyed in different ways and many aspects can be completely alien to speakers of other languages.
Particles are Hiragana characters attached to the end of a word in order to define its grammatical function. A change in particle can lead to a change on the entire meaning of a sentence so, it is very important to identify them correctly.
Considering these facts about Japanese, we claim that the lexical model would be better at translating structurally simple sentences, with lesser particles changing the meaning, and will struggle with longer sequences containing a more complex structure and meaning. Complexity in our discussion refers to the amount of interfering particles that change the meaning of the words. The greater the number of particles, the more complex and difficult the translation is.

Tests and Evidence

We investigated how particles affect the meaning and infer information on how the lexical model deals with such cases based on the output translations. Particularly, we are examining how the lexical model deals with long sentences, which contain a more complex structure and meaning. We also, examined the translation of different particles, like the [は] topic particle, the [も] inclusive topic particle, which indicates additive meaning as for 'also' and 'either', the [が] identifier particle, which indicates the subject, the [に] locative particle, the [の] possessive particle and the [へ] particle of direction, in combination with each other as well as other particles. Due to their wide variety, we aimed at mainly inferring their meaning from the sentences based on the reference translation.

Table 3 below shows some sentences from the translations generated by the lexical model and the baseline as against the reference sentences. Particles have been highlighted in source sentences.

Source Sentence	Reference Translation	Lexical Model	Baseline
ピクニックは雨のために中止になった。	the picnic was called off because of rain .	the picnic was called off because of rain .	the picnic has gone to rain .
私たちは泣いているの。	we are crying .	we are crying .	we are looking at all .
カナダへ行ったことがありますか。	have you ever been to canada ?	have you ever been to canada ?	have you ever been to the movies ?
彼女は教師のように見える。	she looks like a teacher .	she looks like a teacher .	she looks her her mind .
ボブは読みきれないほどたくさんの本を持っている。	bob has too many books to read .	bob has a lot of books to read .	bob has a lot of books to read a lot of books .
彼は私の提案に同意した。	he acceded to my proposal .	he reluctantly agreed to my proposal .	he was on my proposal .
彼女は叔母に似ている。	she resembles her aunt .	she takes after her aunt .	she is as a aunt .
昨日通りで友人に会った。	i met my friend on the street .	i met friend in the street yesterday .	i met my friend yesterday .

The lexical model clearly manages to output accurate or close to the original meaning sentences while the baseline, because of the monotonic use of attention, produces sentences based on proba-

bility distributions through the greedy search; the word *canada* has been translated to *movies*, as the phrase 'have you ever been to' is more often used with reference to *movies* rather than, a country.

However, it is worth noting that, regarding the Lexical model's translations, besides the sentences which are correctly translated, there are others whose meaning is conveyed using synonymous words or periphrastic sentences. This leads to meaning conservation but linguistic loss, since the words which are carrying the basic meaning or the structure are not correctly output.

The translations provided in Table 4 below, are an indication of how the lexical model handles the complexity of particles during translation. The more complex the sentences, the more confused the generated translation. The first five sentences do not contain many linguistic phenomena, yet the model confuses the use of certain particles, the quantity in the third sentence *maximum* with *largest*, the verbal particle in the fourth sentence, *is gone* with *was* or the word order, as seen in the first sentence from the shift in the position of *medium*. However, it manages to convey some meaning which, compared to the baseline model's performance. is an improvement.

Source Sentence	Reference Translation	Lexical model	Baseline
ステーキは中位で焼いてください。	i like my steak medium.	i am medium with my steak.	i am sorry to the same mistake.
彼女とはよく会いますか。	do you see her often?	do you see her?	do you see her for?
このホールは最大で１０００人収容できる。	this hall holds a maximum of 1,000 people.	this city can be largest by largest.	this machine is by a man of being a man.
冬は過ぎ去った。	winter is gone.	the winter \textcolor{red}{was} at the same winter.	we had a lot of finished in the winter.
この町は気に入りましたか。	how do you like this town?	what did this town on your notice?	did this town up?
サッカーチーム全体が優勝して浮き浮きしてた。	the whole soccer team was on cloud nine after winning the championship.	the soccer of the soccer of team was win.	the team was made up of the team of the team.
せっかく弁護士の資格があるのにもったいない。	what a waste of your lawyer qualifications!	there is no sense of the lawyer.	there is no one of the examination in the lawyer.
あすの朝までには一面の銀世界になっているだろう。	by tomorrow morning all the world will have been covered with snow.	the tomorrow will be world world by the morning.	it will be been to the world of the next day.
彼女は機嫌が悪いというのも、いつも地下鉄に乗り遅れ仕事場まで歩く羽目になったからだ。	she is in a temper , because she missed her usual train in the subway and had to walk to work.	she is always because she was young , but she has a walk to the party , she is late to the party.	she is always up to the work, but i always get up to the work.
父は私が彼に対して失礼な言動をとったとき、怒りで顔が真っ赤になった。	father went red with anger when i behaved rudely towards him.	father had already been in the middle of the sun , and i got a teacher.	my father was put on him when i was a little angry.
その名の人は学籍簿に載ってません。	no person by that name is listed in the register of the school.	the name of the name is carrying to the same.	the name is too small to be in the old man.
バスの前のほうはすいていた。	the front of the bus was not crowded.	the bus was bus in the bus.	the bus was full of the bus.
歌を歌っている少年は私の弟です。	the boy singing a song is my brother.	the boy sing to sing , the boy is my brother.	the boy is my brother.
減税はしばしば経済を刺激する重要な財政政策手段として使われる。	tax cuts are often used as a major fiscal tool to stimulate the economy.	the old thing is used to the means of the economy.	the concert is made to the number of order to put the economy is to the full of order to put the economy.
２日間ベッドにいなくてはならなかった。	i have had to stay in bed for two days.	i had no choice to the more.	i had to have been to bed with two.
車が衝突したときシートベルトをしていたら、彼は今でも生きているだろう。	he would still be alive if he had been wearing his seat belt when the car crashed.	when i was the car , he would be alive now.	when he was a car , he will live now.
彼にどんな欠点があるにしても、卑劣なところはない。	whatever faults he may have , meanness is not one of them.	it is not a good idea of the faults that he is a faults.	i don't know the less of him.
ええ、届いたのをお知らせするのを忘れてしまってすみません。	yes , sorry , i forgot to acknowledge it.	i don't forget to have made a reached , but i don't have to put up with me.	we have to forget to forget to know the truth. i have to forget to know your own for a moment.

Discussion

The implemented lexical model is one suggestion for solving the syntactic and semantic constraints across languages as it makes use of source words for attention along with the attentional hidden states, generated by the decoder, to produce more contextually accurate words that align better with the source words. The improved results of the model's performance prove that it is able to output better translations than the baseline model.
However, due to the linguistic differences and limited data between Japanese and English, this extra layer has improved but not managed to generate completely accurate sentences. As can be seen from Table 3, there are target sentences that are almost or even completely accurate, compared to the reference translations, while there are others which, although may capture some meaning, the output translation is not accurate.

The model, however, fails to reproduce logic and coherent translations when the source sentences include more grammatical information; more syntactically complex sentences include more information hence, the model needs to learn to identify and combine words with particles with meaning. The model does not succeed in the reproduction of this Japanese language complexity. The sentences provided in Table 4 provide an insight of this model's inability. It must be noted that, although the translations might differ at a great degree from the original sentences, the model is still able to capture some linguistic and grammatical information, on the contrary to the baseline which generates sentences out of the context. We thus, conclude that lexical information gathered from the model, favors sentences with plain syntactical structure and contextual meaning.
A recent approach on the Japanese-English NMT has proposed to use predicate conjugation to deal with the multiple verb inflections and hence, reduce the vocabulary size and deal with unknown and

OOV words. In particular, they treat verb predicates at first, as conjugation tokens and then, the concatenated tokens, as conjugation features alongside with POS information. This approach improved the performance of NMT models and yielded better BLEU scores in the translation outputs [6].

References

[1] K. Cho. Natural language understanding with distributed representation. *CoRR*, abs/1511.07916, 2015.

[2] A. Farghaly and K. F. Shaalan. Arabic natural language processing: Challenges and solutions. *ACM Trans. Asian Lang. Inf. Process.*, 8:14:1–14:22, 2009.

[3] K. Kettunen. Can type-token ratio be used to show morphological complexity of languages? *Journal of Quantitative Linguistics*, 21(3):223–245, 2014.

[4] T. Kim. *A Guide to Japanese Grammar: A Japanese Approach to Learning Japanese Grammar*. Kim, 2014.

[5] P. Koehn and R. Knowles. Six challenges for neural machine translation. In *NMT@ACL*, 2017.

[6] M. Kurosawa, Y. Matsumura, H. Yamagishi, and M. Komachi. Japanese predicate conjugation for neural machine translation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Student Research Workshop*, pages 100–105. Association for Computational Linguistics, 2018.

[7] Y. Li, T. Xiao, Y. Li, Q. Wang, C. Xu, and J. Zhu. A simple and effective approach to coverage-aware neural machine translation. In *ACL*, 2018.

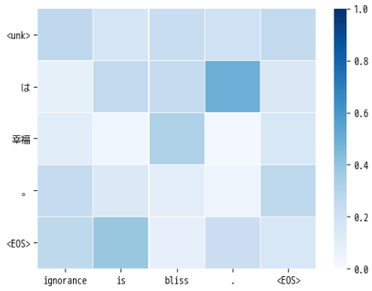
[8] M. Mager, E. Mager, A. Medina-Urrea, I. V. Meza Ruiz, and K. Kann. Lost in translation: Analysis of information loss during machine translation between polysynthetic and fusional languages. In *Proceedings of the Workshop on Computational Modeling of Polysynthetic Languages*, pages 73–83. Association for Computational Linguistics, 2018.

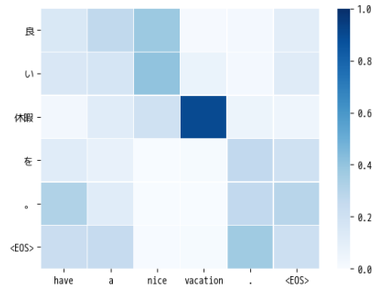
[9] G. Neubig. Neural machine translation and sequence-to-sequence models: A tutorial. *CoRR*, abs/1703.01619, 2017.

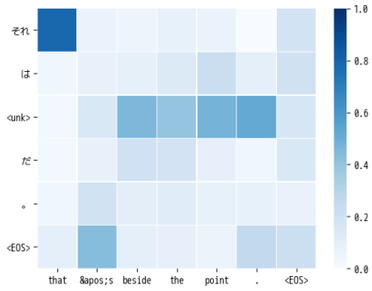
[10] T. Q. Nguyen and D. Chiang. Improving lexical choice in neural machine translation. 2017.

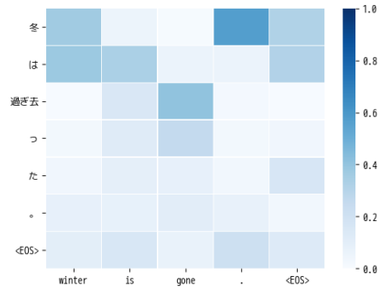
Appendix: Visualizations of attention weights of the translations produced by the baseline and the lexical model

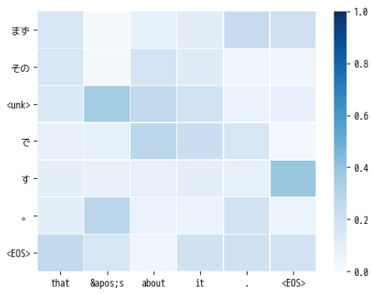
Attention of Baseline Translations

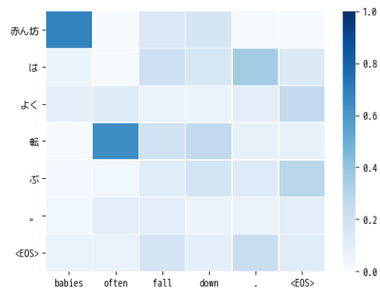


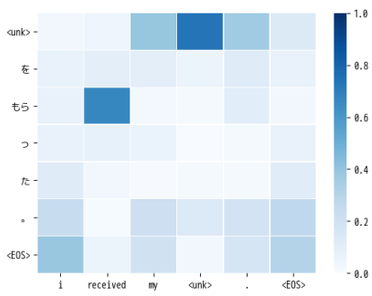


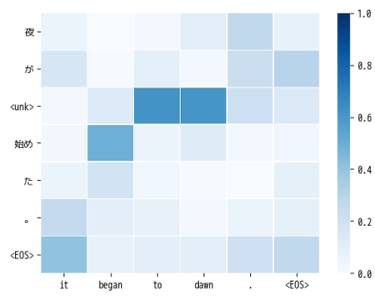


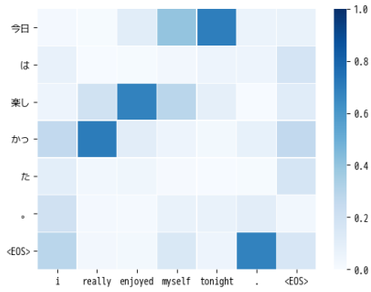


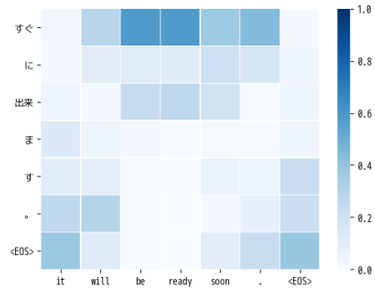




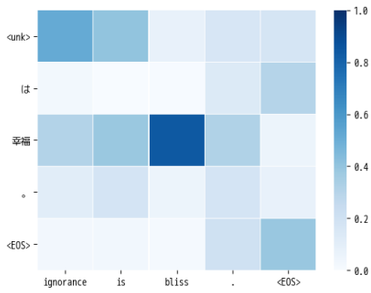


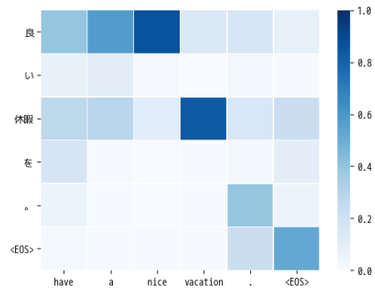


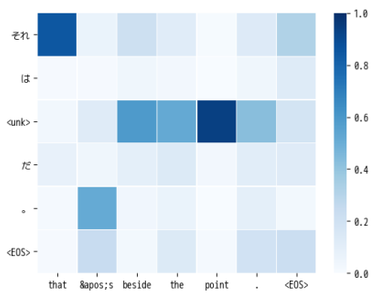


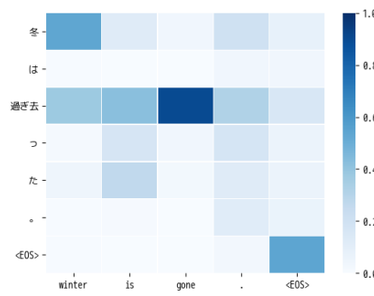


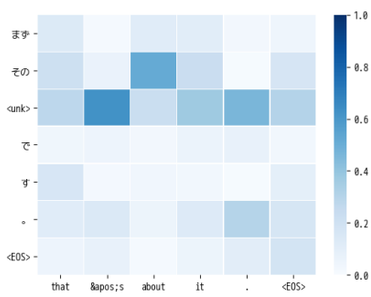
Attention of Lexical Model Translations

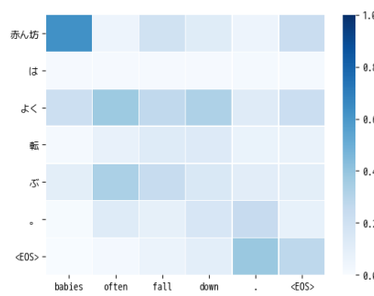


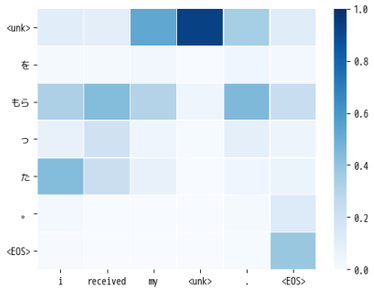


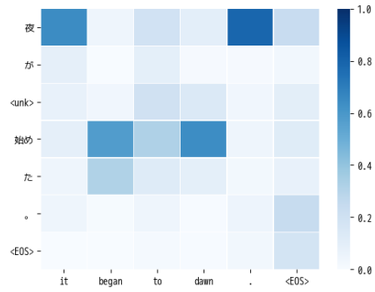


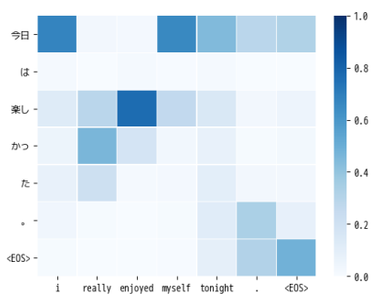


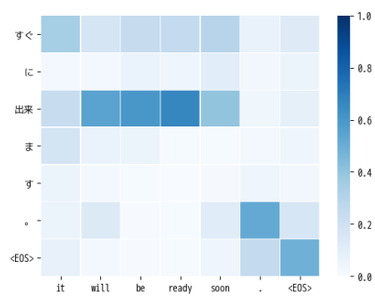





<unk>
を
もら
っ
た
。
<EOS>
i
received
my
<unk>
.
<EOS>
1.0
0.8
0.6
0.4
0.2
0.0


夜
が
<unk>
始め
た
。
<EOS>
it
began
to
dawn
.
<EOS>
1.0
0.8
0.6
0.4
0.2
0.0


今日
は
楽し
かっ
た
。
<EOS>
i
really
enjoyed
myself
tonight
.
<EOS>
1.0
0.8
0.6
0.4
0.2
0.0


すぐ
に
出来
ま
す
。
<EOS>
it
will
be
ready
soon
.
<EOS>
1.0
0.8
0.6
0.4
0.2
0.0