1. (10 points) Copy your code comments here.

A. 1. Add the encoder of a bidirectional LSTM layer and add link on each LSTM layer. 2. add_link() define the L.LSTM(n_units, n_units) as a child link. And it is named as "lstm_name". We could access Links added in Chainer by using their names("lstm_name"). 3. Because this is a bidirectional LSTM. So there is two lstm encoder layers. One is "self.lstm_enc" and another is "self.lstm_rev_enc".So we should add link respectively.

B. 1. L.EmbedID(vsize_dec, 2*n_units) Input=output of encoder, output=input of L.LSTM layer. L.LSTM(2*n_units, 2*n_units) Input size or output size is different from encoder. L.Linear(2*n_units, vsize_dec) Input=size of LSTM units. output=vocabulary size of target language. 2. Because we use this bidirectional LSTM, which can produce a translation of a source word according to the other words around it. So we have two lstm layer in encoder and each of them has n hidden units. And this is a two-way encoder. So we need to transfer the states of the last lstm layer of encoder to the first lstm layer of decoder. So the hidden units in decoder needs to be multiplied by 2.

C. 1. The set_decoder_state() concatenate c_state of both forward LSTM and backward LSTM. And concatenate h_state of both forward LSTM and backward LSTM. Then transfer the oucome frome encoder to decoder. It initialize decoder LSTM by using final encoder state. 2. c_state is the initial state of units in the first layer (self.lstm_dec[0]) of decoder. Cell states of LSTM units is affected by all the previous information or states. h_state is the initial the output of units in the first layer (self.lstm_dec[0]) of decoder in previous time step

D. Because it is a bidirectional LSTM, we need to do two encode operations for each way.

E. 1. Compute the softmax cross entropy between the predicted result and target words.
2. In every training process, the value represents the cost of the difference between the output of prediction and next word var(reference word).

F. Add a hook function to optimizer object. This optimizer will use this object to minimize the loss later. It setting the gradient, wich is bigger than and equals to 5 to prevent gradient exploration.

2. (10 points) Examine the parallel data and answer questions. (Plots may appear in the appendix.)

1. From the figure 1 we could find that generally the lengh of Japanese sentence is longer than English ones. And as for the same sentence of Japanese is longer than the same sentence of English. Therefore, the greater the difference in length may lead to a decline in translation performance.
2. The number of English word tokens: 97643, Japanese: 143580.
3. The number of English word types: 7211, Japanese: 8252.
4. The number of English word replaced by _UNK: 3384, Japanese: 4172.
5. type/token ratios of English: 0.074; Japanese:0.057
6. I think the differences in sentence length will lead the performance of NMT system decrease. Because there are more difficulties in matching words. And there is a small value of type/token ratio. It shows that there are more repetitive words in this data set. So it is easier to train the model. As for unknown word handing, it could help the NMT system to reduce the dimension of input by ignore them, so that model will be easier to train.

3. (10 points) What language phenomena might influence what you observed above?

*I*n Japanese, there are some words that are functional words, there is no practical meaning, their role is to make the sentence more smooth. So there is no corresponding word in English, which leads to the translation process in the deviation(Okita, 2012). In the following example, 'ね' helps smoothing of a sentence without carrying a meaning. This word is considered to be 'not-translated':
i 've never heard of this address around here
この住所はこの辺で聞いたことな いですね.

4. (10 points) Answers to questions about sampling, beam search, and dynamic programming.

1. From the result of translation below, we could find that, sampling method generally could not give a better result. It is because that sampling method chooses random hypothesis at each time step but un-sampling method always chooses the best hypothesis at each time step. So sampling method could choose a worse translation.

Src | ステーキ は 中位 で 焼い て ください。
Ref | i like my steak medium.
Hyp no sample | i like my steak medium . _EOS
Hyp with sample | please gently to medium . _EOS
However, in some special circumstances sampling method may have a better translation. For example:
Src | 彼 は 趣味 が 高尚 です。
Ref | he has elegant tastes .
Hyp no sample | he is a _UNK . _EOS
Hyp with sample | he has a nature _UNK _EOS
I think the system may choose unknown word cause by the best probability without sample words. But sample method may choose a word which is meaningful. And it is may be a better translation.

2. In order to implement beam search I think we could build a search space by storing first n possible hypothesis represented by the log probability at each time step. In the end of process we could choose the final translation by scan the every search space of each time step and use the best hypothesis which is the highest sum of the probability. For
3. I think it is hard to implement dynamic programming. Because the NMT is a sequence system, the current word is influenced by previous word. It is very different from the system we build in previous coursework. The search space of previous coursework is n * k where n is the length of the sentence and k is the width of candidates hypothesis space. In this system the search space is pretty larger than before. So it is hard to search in such search space.

5. (10 points) Experiment with changes to the model, and explain results.

I train the four different model which have different number of encoder and decoder layers on entire dataset and using the 10 epoch. Through observing the value of the pplx and BLEU, the performance of (1-1) is the best(As we known, pplx will decrease as the possibility of final translation increases. BLEU will increase With the increase of the number of n-gram phrases in the final translation.). As the result shown in Table below, we could see that the result becoming worse while the number of layers are increased. And we could find that the number of encoder may have more influence on final results than decoder. I think it is because the data set is not enough large and the hype parameters is not fit. So when we increase the neural network, it will be hard to converge. So the performance is not good.

encoder	decoder	BLEU	perplexity
1	1	19.487	53.7228
1	2	17.786	54.1298
2	1	19.121	55.7077
2	2	17.482	57.7341

And there is an example that shown the almost same conclusion above:
Src | 丘 の 上 に 1 軒 の 家 が あ り ま す。
Ref | there is a house on the hill .
Hyp1 | there is a the house the hill . _EOS
Hyp2 | there is a house on the hill .
Hyp3 | there is a of charge on . _EOS
Hyp4 | there is a dictionary on the table . _EOS
I find that except the last hypo, the other three is almost the same translation. I think the another reason why large depth layer model goes worse performance is that we train the model only using 10 epoch. It is not enough to train a such depth of layers model.

6. (10 points) Implement dropout, and explain the results.

*D*ropout is used for preventing overfitting. I add dropout method on the feed_lstm function with ration of 0.2. when we use dropout, the the value of pplx is 57.0590 and it is larger than baseline(53.7228) and BLEU is 18.911 and become smaller than baseline(19.487). So the performance is worse than baseline. But from the result of translation below we could find that the result of translation below sometimes become better with dropout sometimes worse.
Src | 私 は 早 く 着 き すぎ た。
Ref | i arrived there too early .

Hyp with out dropout | i got to early . . _EOS
Hyp with dropout | i felt to my . . _EOS
The next example below shows the dropout performs better than without dropout.
Src | 彼 は 、 その 川 を 泳 い で 渡 ろ う と し て 失敗 し た。
Ref | he failed in his attempt to swim across the river .
Hyp with out dropout | he tried to the the the the the door . _EOS
Hyp with dropout| he made the swim across the river river . . _EOS
I think dropout method drops some part of RNN and train the rest of it at each time. In other word, dropout method may drop some large weight but inaccurate translation, also give some chance to the week translation which may be better but has a lower weight. Actually, sometime dropout method may drop more accuracy one.

7. (20 points) Implement attention. This question will be evaluated on the basis of your code.

8. (10 points) Explain the results of implementing attention.

*W*e could find that the performance with attention(BLEU=24.378 , pplx=47.9372) become significantly better than baseline(BLEU=19.487 and pplx=53.7228). And the result of translation is better.For example:
Src | その 部屋 は 居心地 の 良 い 感じ が し た。
Ref | the room had a nice cozy feel .
Hyp with attention | the room like good in . _EOS
Hyp without attention | the stone is very moon . . . _EOS
I think it is because that attention method integrates the information in encoder hidden states. Because it take all the source information at a time as a context vector by computing the weighted average according to alpha. So the context vector know who has more information of previous state and decide the next prediction by using the weight information.

9. (10 points) Explain what you observe with attention. (Figures may appear in an appendix.)

*T*he alignment weights between the source word and target word is shown by the attention vector and their degree of alignment will shown on the plot, the deeper color shows a higher correlation. It means the source word is more correlated to a target word than others.
First of all, from Figure 1-5, I found that most of the upper left corner of plot's color is generally deeper. That means the alignment weight is high. I think this is because in the statement, the English word order is the subject - predicate - object, and Japanese is the subject - object - predicate. Their first part is generally the subject, so the subject is easier to correlate with each other and get a high weight of alignment. But the latter part is based on the length of the sentence (the number of intermediate modifier) and other factors make the gap between the predicate and the object increase. So it will become hard to correlate to a accuracy target word. For example, Figure 4, the subject part of the alignment weight is very high, the translation is accurate ("彼女" translate as "she"). But the Japanese object part "叔母" is reluctantly translated into "her", in fact, should be "aunt". At the same time predicate translation error. Figure 5 we see that the subject translation is very accurate. "彼" is translated into "he", but because the length of the sentence in Figure 5 is significantly longer than the length of the sentence in Figure 4, predicate and object translation is completely wrong, reference translation should be "he earns a good salary."

Reference:
Okita, T., 2012. Annotated Corpora for Word Alignment between Japanese and English and its Evaluation with MAP-based Word Aligner. In LREC (pp. 3241-3248).
Bahdanau, D., Cho, K. and Bengio, Y., 2014. Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473.
Luong, M.T., Pham, H. and Manning, C.D., 2015. Effective approaches to attention-based neural machine translation. arXiv preprint arXiv:1508.04025.
Zaremba, W., Sutskever, I. and Vinyals, O., 2014. Recurrent neural network regularization. arXiv preprint arXiv:1409.2329.

Optional: you may include an appendix after the line above. Everything above the line must appear in the first three pages of your submission.

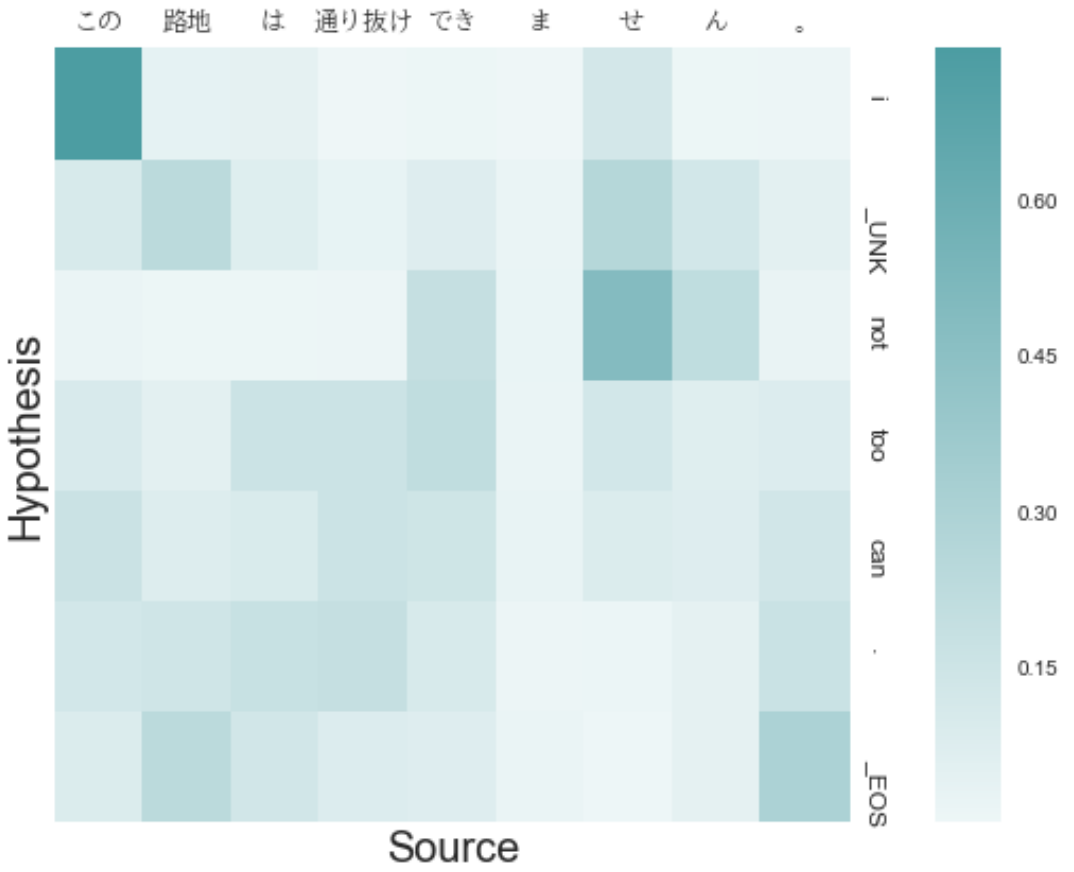
この 路地 は 通り抜け でき ま せ ん 。
Hypothesis
i
_UNK
not
too
can
.
_EOS
Source
0.60
0.45
0.30
0.15

Figure 1:

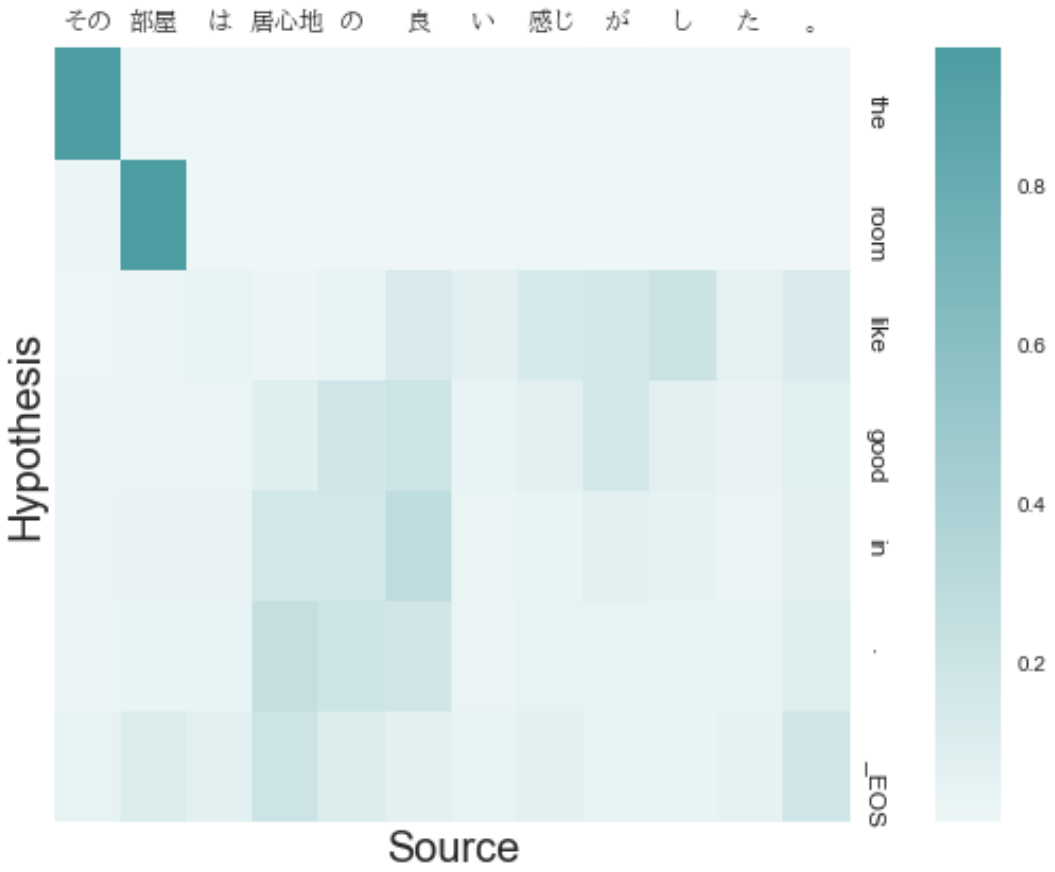
その 部屋 は 居心地 の 良 い 感じ が し た 。
the
room
like
good
in
.
_EOS
Hypothesis
Source
0.8
0.6
0.4
0.2

Figure 2:

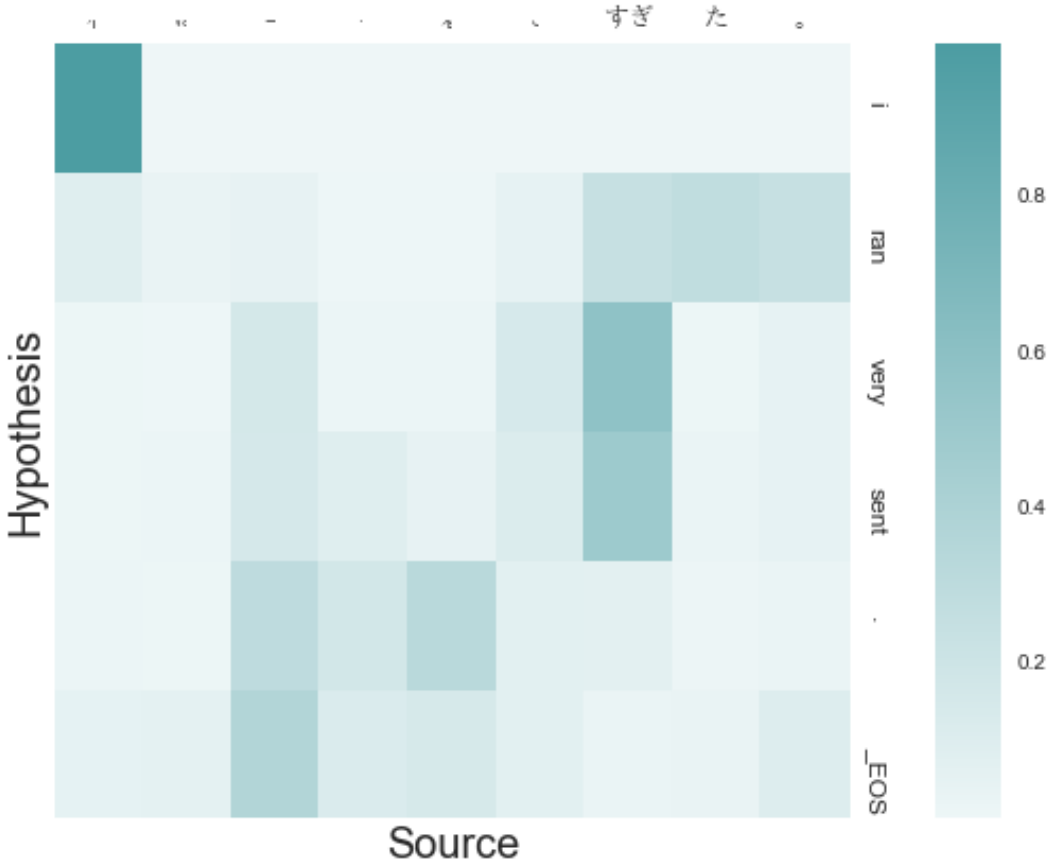

Figure 3:

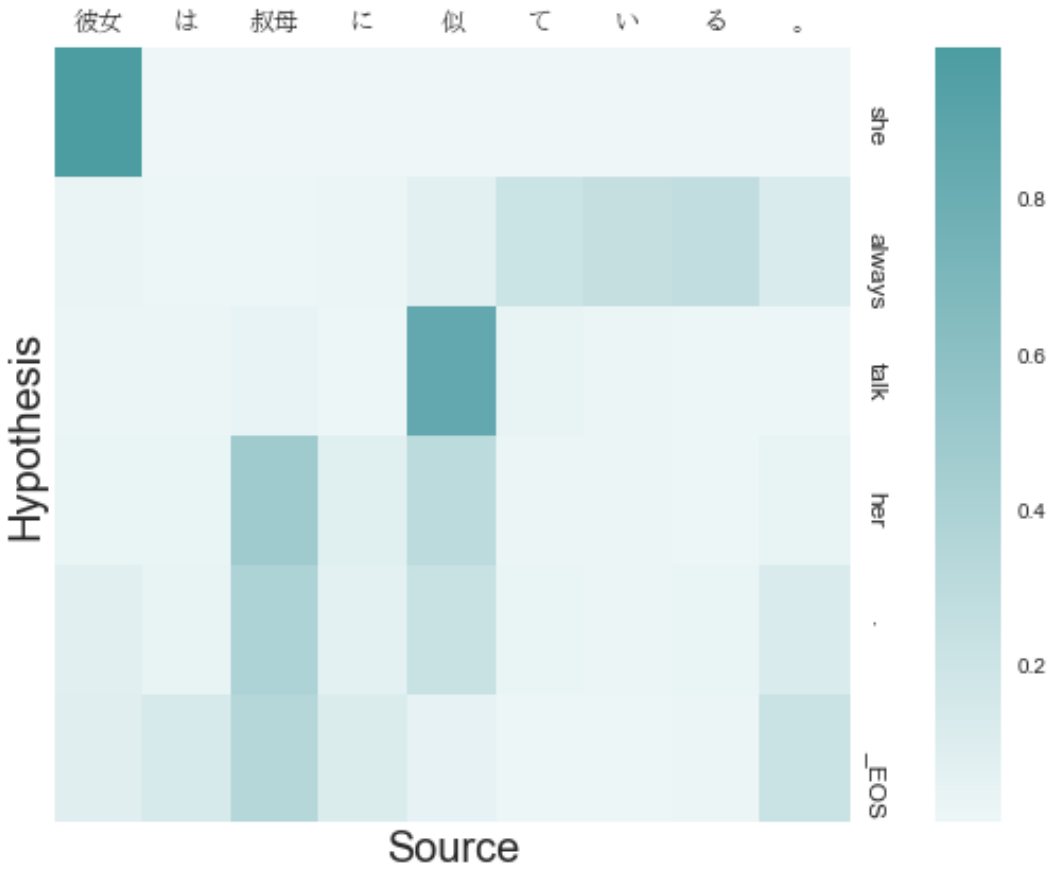
彼女 は 叔母 に 似 て い る 。
she
always
talk
her
.
_EOS
Hypothesis
Source
0.8
0.6
0.4
0.2

Figure 4:


。 " 、 、 給料 を もら っ て い る 。
he
has
_UNK
down
in
circumstances
_EOS
Hypothesis
Source
0.8
0.6
0.4
0.2

Figure 5: