Machine Translation - Seq2seq
Natural Language Processing assignment 3

Sunjidmaa SHAGDARSUREN
MSc Data Sciences and Business Analytics
b00770800@essec.edu

Vaibhav SABHAHIT
MSc Data Sciences and Business Analytics
b00759606@essec.edu

Osheen Mohan PANNIKOTE
MSc Data Sciences and Business Analytics
b00760786@essec.edu

Srijan GOYAL
MSc Data Sciences and Business Analytics
b00758962@essec.edu

ACM Reference Format:
Sunjidmaa SHAGDARSUREN, Vaibhav SABHAHIT, Osheen Mohan PANNIKOTE, and Srijan GOYAL. 2020. Machine Translation - Seq2seq Natural Language Processing assignment 3. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 PART 1

1.1 Analyze how attention is distributed along the tokens. Do you observe any special patterns for EOS/SOS tokens? What happens if you remove those tokens? Do you have any interpretation for such behaviour?

1.1.1 Analyze how attention is distributed along the tokens. :Attention in a seqtoseq model is used to prevent the bottle neck problem. The sentence is passed word by word to the encoder. Once the entire sentence is encoded, the encoding output is passed as one vector to the decoder. In this situation, it becomes difficult to pass the appropriate information for every word in the sentence to the decoder(as all the information is passed in one encoding tensor), especially if the sentence is long. Hence, we use attention. Attention basically is an interface between the encoder and decoder that provides the decoder with information from every encoder hidden state. This ensures that the decoder is informed of the most important information in the input provided by the encoder to determine the output, hence this prevents the bottle neck problem.
In figure 1, we have the attention distribution for the sentence "je suis ravie que tu aies souleve cela ." To predict the word "glad", we see that a majority weight is being given to the word encoding "que". So in this way the distribution of attention among the input encodings helps most accurately predict the output sequence.

1.1.2 Do you observe any special patterns for EOS/SOS tokens? In specific, we did not observe any specific patterns in the EOS and SOS token. These tokens are used to mark the start and the end of statement. Hence the occurrence of these tokens are proportional to the length of the statement. Another observation was that for longer statements we saw the attention probability distribution scatted more among the words as compared to shorter sentences.

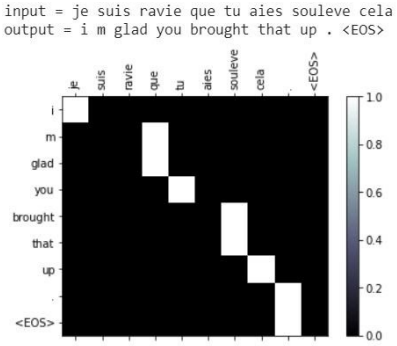

Figure 1: Attention plot for the translation

1.1.3 What happens if you remove those tokens? The SOS and EOS tokens are inplace to ensure the start and end of a statement is clearly known. These tokens are significantly more important in the decoder than the encoder.
Encoder- The removal of these tokens do not have a drastic effect on the encoder as the length of the input sequence is known. Hence we would not see any difference in the working of the encoder.
Decoder- The removal of the SOS and EOS tokens did have some impact on the functioning of the decoder. Below is the screenshot of the output before and after removing the EOS and SOS.

```
> nous sommes presque pretes .
= we re almost ready .
< we re almost ready . <EOS>
Score= 54.88116360940266

> vous etes une vraie loque .
= you re a total wreck .
< you re a total wreck . <EOS>
Score= 60.653065971263366
```

Figure 2: Before removing SOS and EOS

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

```
> les chats ne sont pas mon truc .
= i m not a cat person .
< i m not a bad person . . . . . .
Score= 7.632970657251339

> je suis tres contrarie .
= i m very upset .
< i m very upset . . . . . .
Score= 4.9787068367863965
```

Figure 3: After removing SOS and EOS tokens

We see that when the EOS and SOS tokens are removed the number of words predicted by the decoder is equal to the max-length(Max sentence length=20). So we see that when we remove the tokens the decoder gives a fixed number of outputs irrespective of the actual length of the input or expected outputs.

1.1.4 Do you have any interpretation for such behaviour? :Training period: we know that the length of the input and the target hence the effects of these tokens on the training is not prominent.
Testing and evaluation: The effects of the tokens can be seen at this stage. In this case we know only the length of the input sequence but the length of the target sequence is unknown. The functioning of the encoder is not affected as the encoder functioning depends on the input sequence size. The functioning of the decoder is affected as the size of the target is unknown, hence it does not know when the predicted sequence must end. Hence as seen in the above screenshot the decoder assumes the target to have the same length as max length and provides nonsensical values after the correct predicted words. This can be seen in our output as "." The EOS and SOS tokens help in determining the starting and ending of sequences, this could be with respect to the input sequence and the encoded sequence.

1.2 Why is attention so important in this model? What would happen if it gets removed?

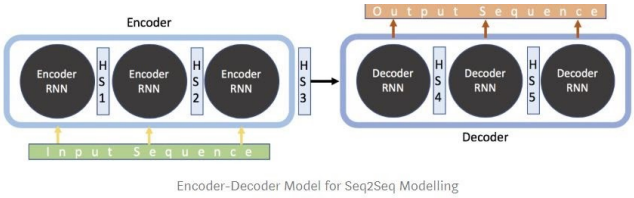

Figure 4: Model with no attention

The above is a diagram taken from a medium link to illustrate why attention is important in our model. As can be seen in the above diagram we pass the input sequence word by word to the encoder RNN and corresponding hidden layer HS1,HS2,HS3 are generated. In the end only the last hidden layer encoding generated is passed to the decoder. This is how the bottleneck problem comes into the picture. The last hidden layer passed to the hidden layer does not convey information of all the words in the sequence accurately. For this reason we have attention.

In Figure 5 we see how the attention for a specific sequence is calculated. The main goal of the attention mechanism is to generate

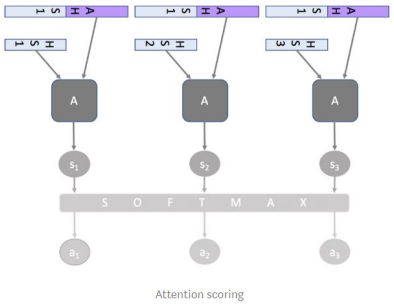

Figure 5: Attention generation

a probability distribution which is capable of determining which part of the input sequence is most important in determining the output sequence.

For this we make use of two hidden states. The attention hidden state which represents previous output and the hidden state represents the current input and the two are used to represent how well the current input matches the previous output. This is done for all parts of the input sequence. We later take softmax to get the attention score.

On each step of the decoder, attention uses direct connection to the decoder to focus on a particular part of the sentence. Attention scores are calculated from inputs coming from encoder and decoder.For calculating attention we make use of two types of hidden states. The attention hidden state which represents previous output and the hidden state represents the current input and the two are used to represent how well the current input matches the previous output. This is done for all parts of the input sequence. We later take softmax to get the attention score or distribution. This distribution gives an idea to the decoder on what words to focus on during a particular timestep. Then the Attention output uses the attention distribution to take weighted some of the encoder hidden states. The attention output mostly contains information from the hidden states that receive high attention(high probability in attention distribution).

So to conclude using attention has several advantages.

(1) Improves performance - by allowing decoder to focus on certain parts of the source

(1) Solves the bottleneck problem - information loss due to single vector representing the whole sentence

(1) Solves vanishing gradient problem

(1) Soft alignment - understands the context better

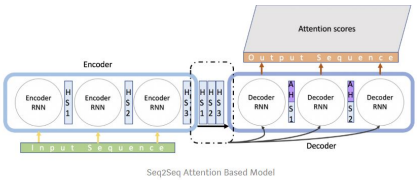

Figure 6: Model with Attention

Final representation of attention incorporated into our model can be seen in Figure 6 . In our model we observed a significant improvement in results when we added the attention mechanism as opposed to when it was not present. Below is an example of outputs with and without the attention mechanism.

```
> je suis desole mon pere est sorti .
= i m sorry my father is out .
< i m sorry my father is out . <EOS>

Score= 68.72892787909726
```

Figure 7: Prediction with attention

We observe the effect of attention only for long sentences(length>5). As seen above with attention even for longer sentences we have a fairly accurate prediction as compared to when we have no attention as can be seen below.

```
> je suis desole mon pere est sorti .
= i m sorry my father is out .
< i m sure my father is out . <EOS>
Score= 40.866465020165684
```

Figure 8: Prediction without attention

1.3 What is teacher forcing? why would you want to use teacher forcing?

1.3.1 What is teacher forcing? :When our decoder begins to predict parts of the output sequence, if the initial predictions are wrong it would affect the predictions from there on in the prediction sequence. Hence a mistake in prediction during the early phase of the sequence could affect the accuracy of the entire predicted sequence. This is what teacher forcing is used to rectify.
In teacher forcing we provide as input to the decoder the actual or the ground truth value of the current prediction, from the dataset as input for prediction of the next word. This ensures the decoder does not make mistakes early in the prediction sequence.
In our experience we observed two significant factors when it came to teacher forcing.

(1) Faster convergence
We observed that when we increase teacher forcing, that is increased the number of ground truth values being fed as input for the next step there is faster convergence. That is the computation time as we increase the teacher forcing decreases. This fits into our understanding as when we feed the ground truth for the decoder directly no time is wasted in looking for the optimal results in the sample space, hence there is faster convergence. This can also be seen from Figure 15 which shows the relation between the Time for teacher forcing with different values 0.3,32m 4s 0.5 (Optimal Value),31m 39s 0.8,20m 23s 1,19m 49s

Teacher Forcing Ratio	Time

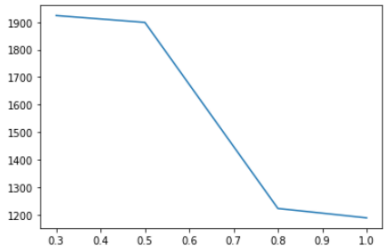

Figure 9: Teacher forcing Time vs value

(2) Instability in evaluation and testing
The ground truth is available to us only during the training stage but not during the evaluation stage. Hence there is a discrepancy in the training and evaluation process which leads to instability in prediction during evaluation. In our observation 0.5 was the optimal value for teacher forcing values below or above this had lower accuracy.

```
> elle est une personne egoiste .
= she is a selfish person .
< she is a selfish person . <EOS>

> c est une gentille fille .
= she is a very kind girl .
< she is a kind girl . <EOS>

> elle est tres occupee .
= she is as busy as a bee .
< she is very busy . <EOS>

> elle est aussi jeune que moi .
= she is as young as i am .
< she is as young as me . <EOS>

> elle est vetue de blanc .
= she is dressed in white .
< she is dressed in white . <EOS>
```

Figure 10: Teacher forcing value = 0.5

As seen from Figure 10 and Figure 11 we have more accurate values when the teacher forcing is set to the optimal parameter of 0.5.

```
> elle est une personne egoiste .
= she is a selfish person .
< she is a quiet person . <EOS>

> c est une gentille fille .
= she is a very kind girl .
< she s a kind girl . <EOS>

> elle est tres occupee .
= she is as busy as a bee .
< she is very busy . <EOS>

> elle est aussi jeune que moi .
= she is as young as i am .
< she is as young as i am . <EOS>

> elle est vetue de blanc .
= she is dressed in white .
< she is far from forty . <EOS>
```

Figure 11: Teacher forcing value = 0.8

$$ExhaustiveSearch = O(V^T)$$
$$BeamSearch = O(k^T)$$

Figure 12: Computational complexity

$$P(y|x) = P(y_1|x) * P(y_2|y_1, x) * P(y_3|y_2, y_1, x) \dots P(y_T|y_{T-1}, \dots y_1, x)$$

$$= \prod_{t=1}^{T} P(y_t|y_{t-1}, \dots y_1, x) = \sum_{t=1}^{T} \log P(y_t|y_{t-1}, \dots y_1, x)$$

Figure 13: Log probability calculation

2 PART 2

2.1 How do different beam sizes affect the result?

In the previous question we implemented greedy decoding. At each timestep during greedy decoding, the most probable output is calculated and this value is fed into the decoder to predict the next word. Even though, it gives the most probable output at a timestep, it might not give the best sentence as output. Hence, there is no way to undo decisions in case of greedy decoding. The optimal solution can be obtained only by doing exhaustive search, ie calculating all possible combinations of words to get the best sentence translation as output. However, this is highly computationally expensive. Hence, we go for beam search decoding.

Beam search decoding keeps track of k most probably partial translations(k specified by user). As k increases, the computational complexity increase, but accuracy also increases.

Working of beam search decoding :

(1) k different hypotheses considered at the same time. Each produce end tokens at different times.

(1) When a hypotheses produce end token, place it aside and continue exploring others

(1) We continue beam search till we completed T timesteps or till we have completed at least n hypotheses(not implemented in the code we used - future scope of improvement).

For a hypothesis (y1, y2, y3, ...yt), we want to find the argmax of P(y/x) which is same as maximising the sum of log probabilities as given in equation mentioned above in Figure 13.

In the given below figure, the french sentence 'il m'a entarte' which has the translation 'he hit me with a pie' in English language. The figure gives an idea of how probabilities are calculated in each step.

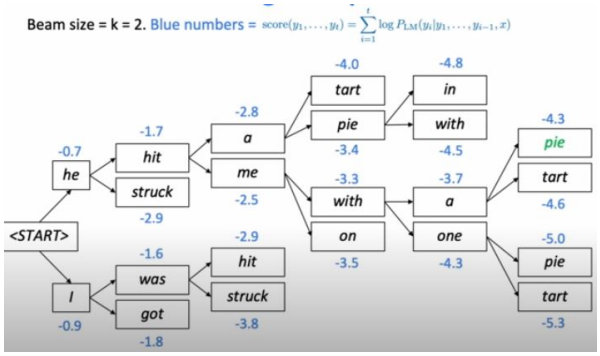

Figure 14: Beam Search, k=2

Length penalty : While performing beam search, we compute the log probability of these hypotheses and then choose the one with the highest probability. Problem with this approach is that it is biased towards selecting shorter sentences as they will have higher value of probability. To avoid this, we normalize the log probability score by dividing it with length of the hypotheses(number of timesteps taken to compute the hypotheses). The new probability score is calculated according to the equation below.

$$1/T * \sum_{t=1}^{T} \log P(y_t|y_{t-1}, \dots y_1, x)$$

Figure 15: Normalized beam search scoring

The table given below shows information about the execution time and sacreBLEU scores calculated for various beam sizes. Here, you can see that the translation predictions become better as beam size is increased. The accuracy doesn't improve much at k = 50 because of limited size of the data (the training and testing were only done for all sentences with MAXLENGTH < 20 - which only constitutes to 2000 sentences approx). Training on limited set of data resulting in bad predictions. The small execution times for all beam sizes is again due to the limited amounts of data. Testing for large datasets with beam size = 50 in low end systems would be almost impossible because of the computational complexity.

k	bleu.score	Time
1	0.90	0.52 min
3	1.43	0.76 min
5	1.44	1.2 min
50	1.45	5.6 min

Table 1: sacreBLEU score and time for beam sizes

BLEU score - It compares machine translations to one or more several human translations of the same sentence. It computes similarity score based on n-gram precision. In some of the BLEU implementations, we can specify the weights(importance) to be assigned

to 1-gram, 2-gram etc. If the system translation is significantly smaller than the human translation, the BLEU score will be very less. This is why we use brevity penalty. Preprocessing schemes have a large effect on scores. Importantly, BLEU scores computed after applying different preprocessing steps cannot be compared between references. Hence, we use sacreBELU.

2.2 what happens with a very large beam size (> 50)?

When beam size is greater than 50, the prediction accuracy approaches the accuracy of exhaustive search. It takes too much time to run. Beam size = 70 took 8.5 minutes to run. However, the bleu score did not improve for the above mentioned reasons.

3 PART 3

Part1- For this part we have used the data and code from the tutorial. The parameters taken are mentioned in the theory part of each question.

Part 2 - The code was executed using the data specified in the NLP Course page. The cleaned data for train and test has been added to the zipfile. We did not use teacher forcing for this part of the exercise.

4 PART 4

4.1 What is the impact on the vocabulary choice? Benchmark different strategies (eg: varying vocab size, char-based, BPE), again with BLEU and training/decoding effort.

Our tutorial's vocabulary is based on building own vocabulary from the training dataset. We tried 4 different scenarios to test if different vocabulary methods will have an impact on the BLEU scores for the test dataset. Here, we calculate the everage BLEU score rather than calculating on the entire corpus. When maximum number of words in sentence is 20, we have 1489 pairs of sentences eligible for the training.

- The first case is using all 1489 pairs and build vocabulary from this dataset. We have 3061 distinct French words and 2653 distinct English words in our vocabulary.
- The second case is, we are trimming the vocabulary and only choose the words when the frequency is more than 1. We are excluding rare words from our vocabulary.
- The third case is, we are choosing the words when the frequency is more than 2.
- The last case, we used character level vocabulary for the machine translation. In this case, instead of taking words and tokenize it, we took each characters from the sentence and indexed it. In French and English language, we have 32 items in our vocabulary, because we normalized all of our texts in the preprocessing step.

In conclusion, if we have enough words in our vocabulary, our BLEU score will increase. We have to use trimming in the case of large vocabulary size, but our training data-set has only 2500 pairs of sentences, which makes it too small for the effective training. Char level vocabulary takes long time to train the model, the reason is we are trying to predict the next character based on the previous characters, thus it will take longer than word level vocabularies.

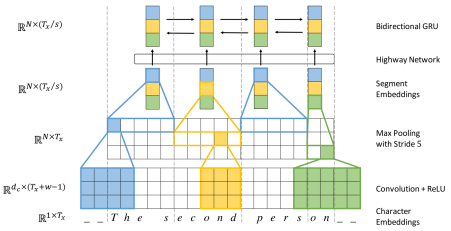

Figure 1: Encoder architecture schematics. Underscore denotes padding. A dotted vertical line delimits each segment.

Figure 16: Character level neural machine translation

	All voc	Min_req=2	Min_freq=3	Char_level
BLEU score	2.77	2.45	2.1	1.1
Number of pairs	1489	379	210	1489
French voc size	3061	1205	719	32
English voc size	2653	1087	687	32
Max-length	20	20	20	20
Number of iteration	40000	40000	40000	10000
Time	17 min	16 min	13.5 min	31 min

5 REFERENCES

(1) A Call for Clarity in Reporting BLEU Scores - Matt Post, Amazon Research
(1) https://towardsdatascience.com/word-sequence-decoding-in-seq2seq-architectures-d102000344ad
(1) https://medium.com/@dhartidhami/beam-search-in-seq2seq-model-7606d55b21a5
(1) https://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf
(1) Correcting Length Bias in Neural Machine Translation https://www.aclweb.org/anthology/W18-6322.pdf