RENNARD VIRGILE
Virgile@rennard.org

Lab session # 4
ALTEGRAD 2019

11/24/19

1 Question 1

There are several methods for decoding our encoding that are presented in the ACL tutorial, which are the Exhaustive Search, the Ancestral Sampling, the Greedy search, which we use, and the Beam search. Obviously, the computational power needed to work with the exhaustive search makes it impossible as of yet, leaving us with the three other methods. Ancestral sampling works by sampling variable values one at a time, working on more and more context, so that at a time t, we sample a word from a distribution $P(x_t|x_{t-1}, \cdots, x_1)$. This is inefficient. What interests us is the Greedy Search along with the Beam Search. Greedy search is just a subcase of beam search with a beam-width of 1. The greedy search will simply output the translation that the model think is best, by selecting the word that has the highest probability at each time step. This is the same as ancestral sampling except that instead of sampling the word randomly from the distribution, we choose the argmax. Contrary to ancestral sampling, we are not guaranteed to find the translation with the highest probability. One way to solve this problem is through the beam search. Instead of working on one path of the argmax, we will work on "B" path at the same time, and thus works on comparing hypotheses over the entire sentence. As it is equivalent to multiple simultaneous greedy search, it will obviously be both more expensive, and more accurate.

2 Question 2

The main problems are both over and under translation. Over-translation is the act of translating the same word multiple times, and under translation is to just skip the translation of one word. As is explained in [2], one way to fix this is to work with statistical machine translation instead of Neural machine translation. But this would be counter productive as the statistical approach leads to worse results, however, one of its main idea should be transferred to Neural machine translation : Coverage. In NMT, we do not keep track of words that have been translated and only wait for the end of sentence token to be predicted. One way to solve this is to work with a NMT-Coverage architecture, which appends a coverage vector to the representations of an NMT model, updated after each read during the decoding process. A classical attention based model does not take advantage of the alignment information useful to avoid over/under-translation problems. If a word has already been translated, its odds of getting translated again should be lower. The main difference implementation wise is that, while we use a GRU unit, which, given an input sentence $x = x_1, \cdots, x_J$ and previously generated words $y_1, \cdots, y_{i-1}$ the probability of generating the next word y_i is a softmax :

$$P(y_i|y_{<i}, x) = softmax(g(y_{i-1}, t_i, s_i))$$

such that g is the non-linearity, t_i is a decoding state for i, computed by

$$t_i = f(t_{i-1}, y_{i-1}, s_i)$$

where $f(.)$ is a GRU and s_i is a source representation for time i, calculated as

$$s_i = \sum_{j=1}^{J} \alpha_{i,j} h_j$$

where h_j is the annotation of x_j in an BRNN, and α is computed by

$$\alpha_{i,j} = \frac{exp(e_{i,j})}{\sum_{k=1}^{J} exp(e_{i,k}}$$

and $e_{i,j} = v_a^T tanh(W_a t_{i-1} + U_a h_j)$
The main change in order to keep track of the coverage is to create a coverage vector $\mathcal{C}$ created with an RNN, that we plug into the $e_{i,j} = v_a^T tanh(W_a t_{i-1} + U_a h_j + V_a C_{i-1,j})$ where $C_{i-1,j}$ is the coverage of the source word x_j before time i and V_a is the weight matrix for coverage.

3 Question 3

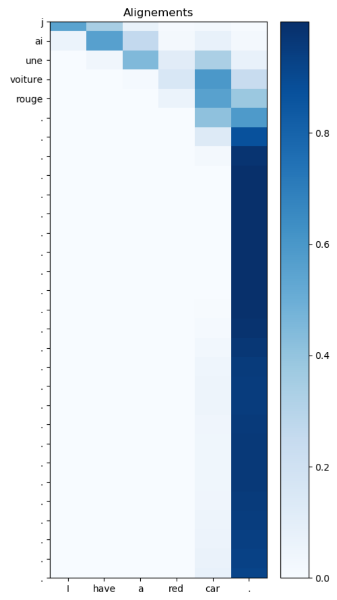

Figure 1: Plot showing the weights assigned to each input word for each output.

This is an important example. In English we say ”Red Car” while in french we say ”Voiture Rouge”, inversing the order between the noun and the adjective. We can see from this example that the model was able to understand the relationship between the inputs and the outputs, and understood this inversion between the two languages, as the weight given for car is heavier for voiture and the same can be said for red and rouge. Thus, we do not get a diagonal line as there is this inversion. An attention based neural network can work out this kind of language differences.

4 Question 4

We can see that the sentence ”I did not mean to hurt you” has been wrongly translated, as mean has multiple significations, and can only be translated correctly if we have the whole context. Here, we are not using a bidirectionnal RNN, which means that the decoder works on the translation of ”mean” given only the information ”I did not”, which would of course have a different output if we were to have given it the information carried by the word ”hurt”. This problem of bidirectionality is at the origin of the BERT architecture presented in [1]. This is a method that is more closely related to how a human would read and understand a sentence, as the information in a phrase isnt unidirectionnal.

References

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.

[2] Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. Coverage-based neural machine translation. *CoRR*, abs/1601.04811, 2016.