Basile Terver
basile.terver@polytechnique.edu

Lab session # 2
ALTEGRAD 2023

10/17/23

1 Question 1

There exist 3 main decoding strategies, according to the slides 87-95 of this ACL tutorial `https://sites.google.com/site/acl16nmt/home`, ancestral sampling, greedy search and beam search. **Beam search** is more computationally expensive than the greedy search and not easy to parallelize but gives a decoding and translation of much better quality. Indeed, it is quite hard to code the pruning of probability paths described in slide 92 of the ACl tutorial mentioned above `https://sites.google.com/site/acl16nmt/home`.
We can use simpler methods like **random sampling with temperature**. Temperature increases the probability of probable tokens and reduces the one that is not. The temperature parameter t is in $[0, 1]$. When $t = 1$, we do random sampling according to the softmax provided by the prediction of the decoder. When $t = 0$, we end up taking the argmax on the probability distribution, which is exactly greedy search.
Another approach is **top-K sampling**, where only the top K probable tokens are considered for decoding. This puts to zero the probability of the other tokens to be chosen. This technique can be combined with temperature among the top-K probable tokens.
All these decoding approaches are described in [1].

As it is not executed in the notebook I handout, I include in Figure 1 the checking that the loss indeed decreases during training. I did not have time to include a plot of the testing and training loss through the training epochs, which would have been useful to check for potential overfitting of our model. Also, my model gives translations quite similar to the ones of the pretrained model, with the same issues mentioned in Question 2.

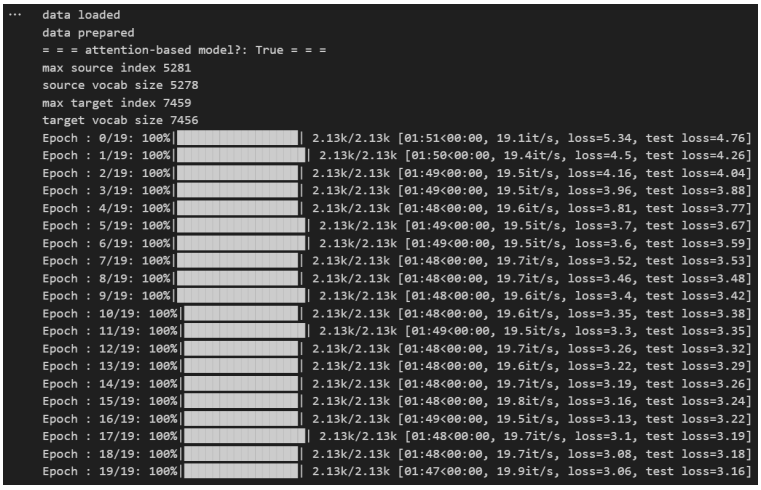

Figure 1: The loss goes down to 3.16 at the end of training for my model".

2 Question 2

If we judge our model using the pretrained model, here are the main problems we encounter with our translations:

- We have a problem of repeated words or repeated dots at the end of the translated sentences. This is due to our greedy search. This can be called **over-translation**. Indeed, we have repeated end of sentence dots in each translated sentence, except in "She is so mean $->$ elle est tellement méchant méchant . $< EOS >$". Our seq2seq model is that it produces, for most sentences, a long sequence of dots at the end of the translation, like in "I am a student. $->$ je suis étudiant . . . . . .". Thus, our function "sourceNl_to_ints" of the "seq2seq" model might not tokenize well the dots into $< eos >$ tokens. We would like to just have an $< eos >$ token at the end of each translation. One could simply remediate to this simply coding something like "if a token is the same as the previous two, stop the sentence".

- Conversely, some words are not translated or missed, this can be called **under-translation**, as in "I can't help but smoking weed -¿ je ne peux pas empêcher de de fumer fumer fumer [...]" where the contracted pronoun "m' " is absent of the translation.
- Our model struggles to capture **long range dependencies**. This can be seen in the "She is so mean $->$ elle est tellement méchant méchant . $< EOS >$", where the model does not capture that "mean" refers to "she" and therefore does not agree the adjective in terms of gender. This issue, just like the two previous ones, is addressed in [6]. Basically, this paper's approach is to keep track of the attention history in a "**coverage vector**", which is fed to the attention model to adjust future attention.

Another way to address all these would be to implement beam search or other **decoding approaches** mentioned in Question 1.
In our encoder, The GRU cannot look backwards as it is uni-directional. Using a **bidirectional GRU** for our encoder might also help to manage long range dependencies, and maybe over and under-translation.
Obviously, as stated in the handout, we use a one-layer RNN for the encoder. Stacking **multiple layers** might increase the precision of our model. Yet, if we add more parameters to the model, one must increase the **size of the training dataset** and the **number of training epochs**.

3 Question 3

Adjective-noun pairs: As is described in Section 5.2.1 of [2] and Appendix A of [4], we want our attention weights to capture the translation of grammatical structures such as adjective-noun pairs. In the source language, English, the adjective is before the noun, whereas in the target language, French, it is placed after. Thus, to show that the attention network is working well, we need, on examples with adjective-noun pairs, to see pairs of "whiter" boxes that are orthogonal to the "main diagonal". In our figures below, the main diagonal goes from the upper left corner to the lower-right corner.
We see an example of such inversions in Figures 2 and 3.

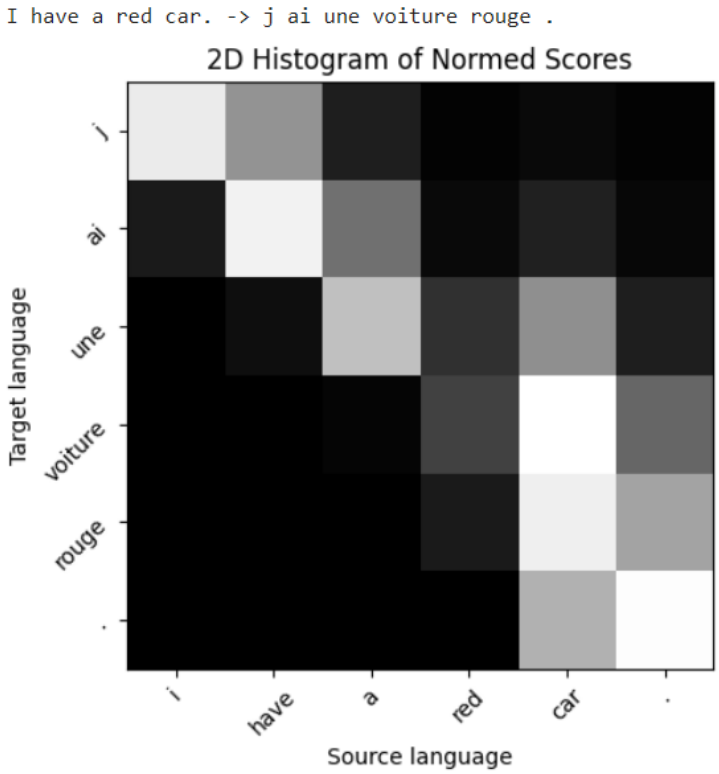

Figure 2: We can see an (at least partial) adjective-noun inversion here. Indeed, "car" is mostly linked to its corresponding noun, "voiture". Although we would like to see "red" being mostly linked to "rouge" and it is mostly linked to "voiture", it still has some link with "rouge".

Presence or absence of article: In other cases without adjective-noun pairs, we highlight another interesting grammatical structure that is captured by the attention network, the fact that articles, such as "a" (indefinite article) in English and "la"(definite article) in French, can be present in a sentence of one language but should not grammatically be present in the other. This can be seen in Figures 4 and 5.

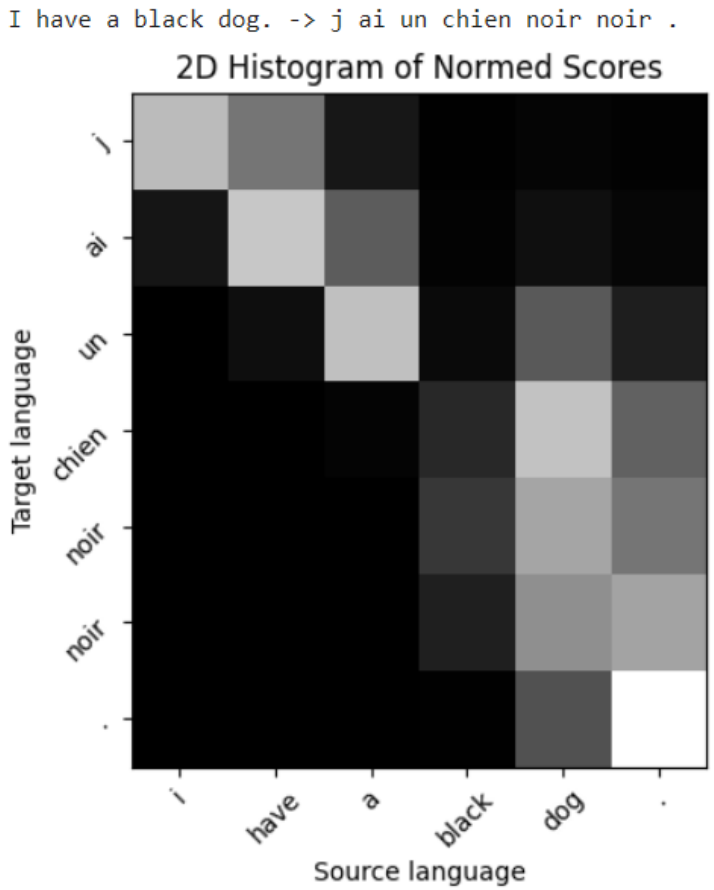

Figure 3: We can see an (at least partial) adjective-noun inversion here. Indeed, ”dog” is mostly linked to its corresponding noun, ”chien”, which gives rise to a whiter box out of the main diagonal. Moreover, the source word ”black” is mostly linked to ”noir”, its corresponding adjective. Thus, if we forget about the second ”noir” row, we get a beautiful ”adjective-noun inversion” figure here.

4 Question 4

One interesting thing to remark is that our model successfully detects the **polysemy**. Indeed, the English word ”mean” is a verb in the first case, whereas it is an adjective in the second case. They are written the same, therefore they have the same embedding vector in the encoder. This illustrates the ability of language models to get a **contextualized word representation**.
ELMo [5] propose a new type of ”deep contextualized word representations” using a bidirectional RNN for the encoder, as we proposed in Question 2.
Eventually, it seems that now, state-of-the art methods to capture complex contextualized representations are transformers, like it is done in BERT [3].

References

[1] Decoding strategies that you need to know for response generation. `https://towardsdatascience.com/decoding-strategies-that-you-need-to-know-for-response-generation-ba95ee0faadc`. Accessed on 2023-10-16.

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate, 2016.

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[4] Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. Effective approaches to attention-based neural machine translation, 2015.

[5] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations, 2018.

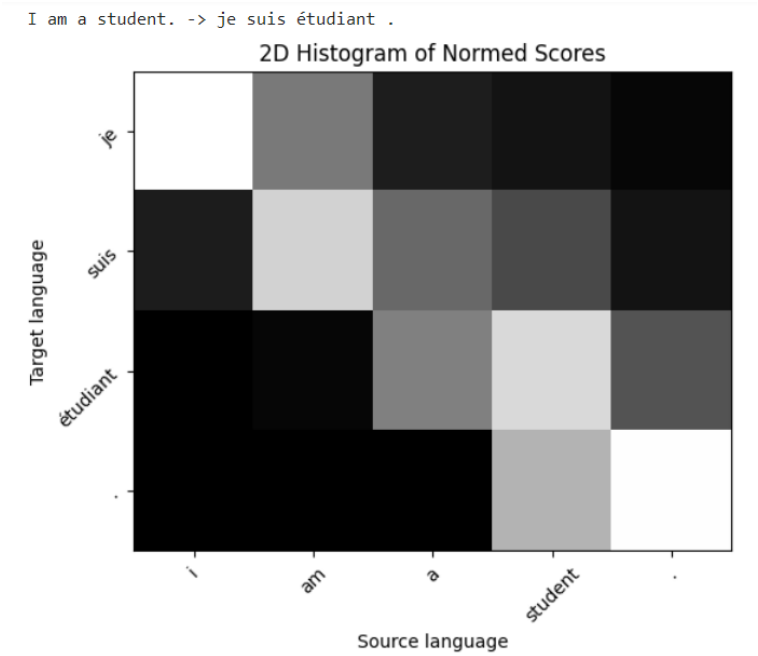

Figure 4: Here, ”the student” translates into ”étudiant”, so two words translate in only one word. We see that the attention of both ”a” and ”student” is mostly on ”étudiant”, which explains why the network made the correct translation.

[6] Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. Modeling coverage for neural machine translation, 2016.

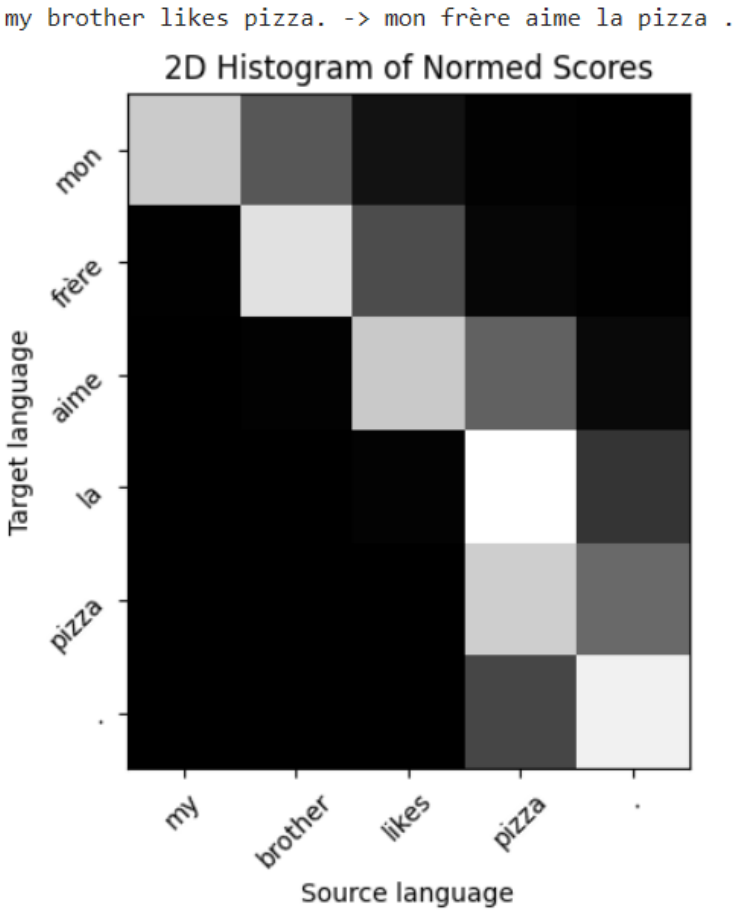

Figure 5: Here, ”pizza” translates into ”la pizza”, so one word translates in two words. We see that the attention of the English word ”pizza” is mostly on ”la” and ”pizza”, which explains why the network made the correct translation.