1 Question 1

From the ACL tutorial [5], this greedy approach is always preferable to brute force (impossible due to complexy). The greedy approach consists in choosing the "best" prediction for each iteration. They have the advantage of low computational and memory complexity. On the other hand, they are sub-optimal because they return a local optimum at each steps. The "best" at time t does not take into account the following words, which combined could give a better meaning. So, if the model makes a mistake at the first prediction (eg: with polysemy or in long, complex sentences), then the rest of the predictions will be off track and is going to get punished for every subsequent word it generates. This can completely distort the translation right from the start. Here is an example I generated with my model, the word **way** as least two meanings in french ('chemin' and 'la façon de'). If we have a sentence like :

"**The way to improve things is by trying**" translation $->$ "**la chemin améliorer meilleure meilleure est peut essayer d.**".

The model chooses the wrong meaning of the word at the beginning. The rest of the sentence is bad. On the other hand, by changing the sentence very slightly, we get the right meaning and the rest of the sentence is better translated. This shows that this decoding strategy is very sensitive.

"**The way to be smart is by learning.**" translation $->$ "**la façon doit être intelligent dans le apprendre**".

An intermediate approach is Beam Search, which considers the K best paths at each step and so it is always better in terms of prediction quality. This allows us to retain paths with potential. A non-optimal prediction at step 1 may become relevant as information is added during recursive predictions. But on the other hand, the computation and memory costs will be much higher, as we will have to store and track several paths. Other approaches such as **Top-K Sampling** or **Top-p (nucleus) sampling** [1] are methods that add temperature and avoid falling into a loop like greedy sampling. These methods consist in keeping the best predictions at each stage and redistributing the probability mass over the best predictions, then randomly drawing according to this density. In this way, it will not always return the most likely prediction.

2 Question 2

2.1 Problems observed

The major problem with our model is its inability to translate perfectly rather short, uncomplicated sequences (even if the meaning is present). The $< EOS >$ token very rarely appears at the end of a sequence. This can be seen at the end of sentences, where tokens are repeated. In addition, some words are not translated. This is particularly noticeable when there is no "." in the input sequence. In top of that, it sometimes miss the gender for articles (eg. "le" vs "la"). There is another drawback: for one prediction, the model has to compute attention to all inputs, which can be computationally intensive for this particular task.

2.2 Ideas to solve problems

Based on the above observations, we can start by ensuring that all sequences end with punctuation and use another decoding approach (such as Top-K sampling). We can also consider enlarging the model. For example, stacked RNNs can be used, providing both low- and high-level information on the sequence. The problem of **global attention**, which must compute attention to all inputs, is addressed in [4]. The **local attention**, which focuses on attention in a Gaussian neighborhood of the current word (centered: local-m or shifted and learned by the model: local-p) and is less intensive. The results obtained with local attention are better.

The problem of repetition and missing words is discussed in the article [7]. They name these concepts (Over-translation and Under-translation). To solve this problem, they define **coverage**, which consists in tracking whether a word is already translated in the decoder thanks to a coverage vector. Their NMT-COVERAGE method thus learns a vector C $\in \{0,1\}^{|input|}$ during model training, which is :

$$C_i = \begin{cases} 1 & \text{if we have already translated the word } i \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

So, the aim is to concentrate on $C_i = 0$ in order to not to duplicate translations and finally have $C = [1, ..., 1]$ and return the token $< EOS >$.

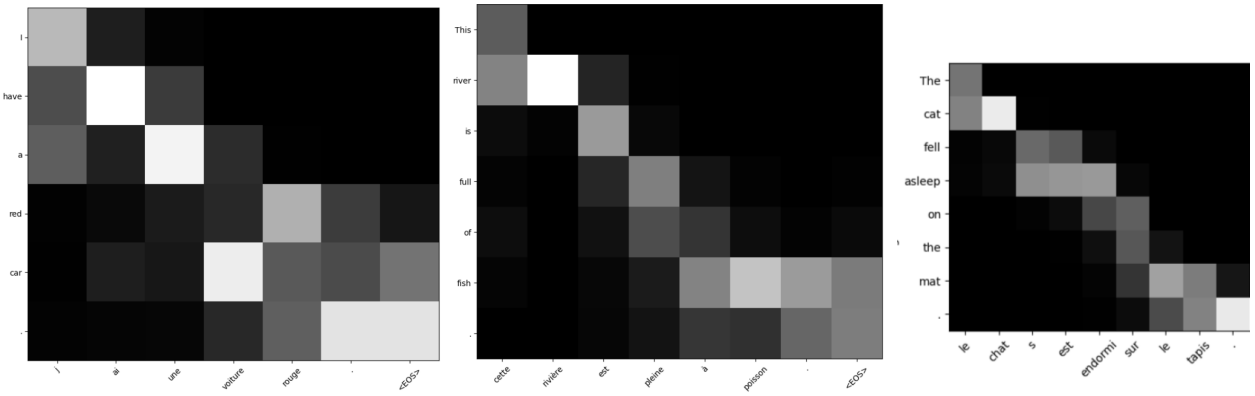
Another idea comes from [4] who presents the concept of Input-feeding inspired by coverage methods. The attention vector h_t is concatenated with the input at the next time step in the decoder. This allows the model to know the previous alignment choices and learn the coverage.

The teacher forcing method presented in (q.5 / Bonus) also helped me solve this problem.

3 Question 3

It is possible to visualize the alignment as in the articles [2] and [4] with the $\alpha_{t,i}$ attention matrix. We can spot syntactic structures that differ between languages, such as adjective-noun inversion between French and English (cf. figure 1a). We can see that "voiture" has given most attention to car and "rouge" to red. So there's an inversion on this heat-map. This example has been well chosen, and this is not always the case, as the model is very sensitive and not very efficient. We can also see a monotonic alignment for very good examples. The attention is strong on the diagonal, which shows that the prediction pays a lot of attention to its opposite. (see. figure 1b).

It is a well-known fact that many English words are translated into French with more words (an English text is often shorter than a French one). This can be seen in figure 1c where the 2 words fell asleep is translated by 3 words "s est endormi". Here we can see that the attention of "s est endormi" is focused on asleep and a bit on fell (with kind of flat area).



(a) Attention map showing adjective-noun inversion.

(b) Attention map showing quite perfect translation.

(c) Length disparity: 3 words in FR to translate 2 words in EN

Figure 1: The target on the x-axis and the input on the y-axis with th prediction of my training (white = 1 / black = 0)

4 Question 4

The sentences "**I did not mean to hurt you**" and "**She is so mean**" both contain the word **mean** but with a different meaning. These are polysemy (same spelling but context-dependent meanings). Thanks to the attention mechanism, our model is able to discern the meaning of a word according to the context of the sentence. This can be interpreted with the attention matrix (see. figure 2). In fact, in the first sentence, the model predicts "voulu" by paying attention to "mean", but also to "did" and "to", which are words that suggest the use of a verb in french. In the second sentence, however, the word "méchant" pay attention only to mean, because it is an adjective. This shows the importance of the attention mechanism.

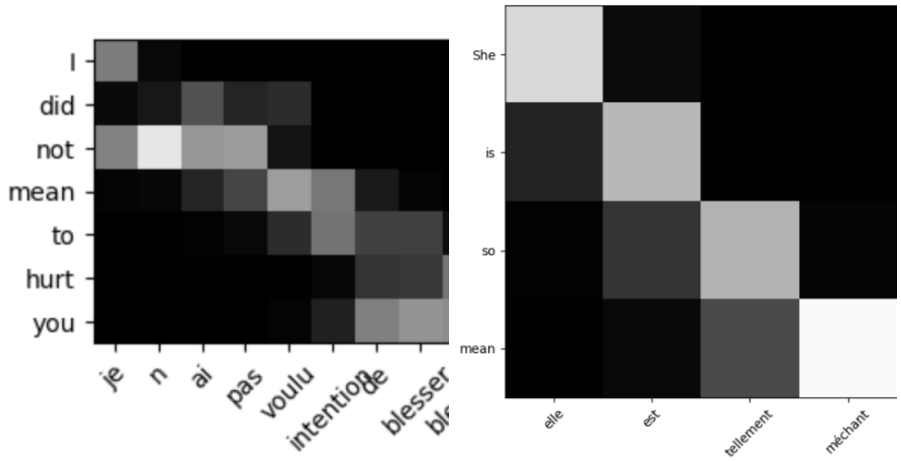


Figure 2: The target on the x-axis and the input on the y-axis (white = 1 / black = 0)

But it can sometimes make mistakes in long sentences (cf. question 1). The model we use is one-directional. In fact, hiddens states only encode information from the past. In papers, they are often noted as $\overrightarrow{h_k}$ for each step k. One-directional language models have limitations as they consider context forward, making them less effective at disambiguating word meanings, resolving polysemy, and understanding language structure.

In contrast, bidirectional models as discussed in [[3], [6]] capture contextual information form forward and also from backward. The backward give hidden states often denoted as $\overleftarrow{h_k}$. This enable better polysemy resolution, enhanced word order understanding, and improved contextual resolution because information from both directions is taken into account.

The [6] method uses biLMs (stacked LSTMs) to calculate $\overrightarrow{h_k}$ and $\overleftarrow{h_k}$. They then calculate an ELMo score that gives them better performance on semantics and context. In 2019, the authors of [3] proposed a similar approach, creating the BERT (Bidirectional Encoder Representations from Transformers) model, which uses the transform structure and bidirectional encoding. This state-of-the-art model also uses the attention mechanism. In addition, it is parallelizable and more robust to long sequences.

5 Question 5 / BONUS

Teacher forcing is a training technique used in recurrent neural networks (RNN) for supervised sequence learning. During training, the model receives the true expected outputs at each step of sequence generation instead of predictions (see. figure 3). This guides the model to learn faster. In the inference phase, the model uses its own previous predictions to generate the sequence. This is a useful approach for improving learning convergence, but can lead to sensitivity to cumulative errors.

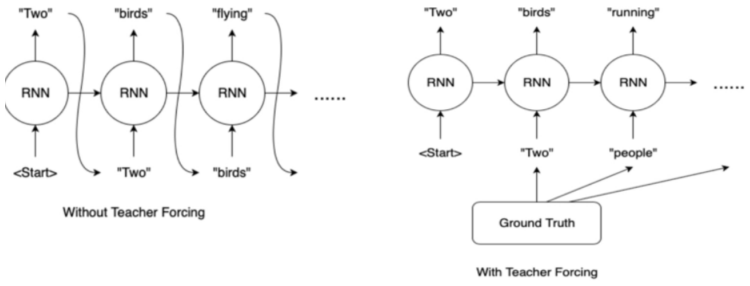


Figure 3: Illustration of teacher forcing illustration from the internet.

We can see that losses are also decreasing. This training solved the problem of question 2 with tokens that are repeated at the end of a sentence (it predicts $< EOS >$).

```
= = = = =
 I am a student. -> je suis étudiant . <EOS>
= = = = =
 I have a red car. -> j ai une voiture rouge . <EOS>
= = = = =
 I love playing video games. -> j adore jouer aux jeux vidéo . <EOS>
= = = = =
 This river is full of fish. -> cette rivière est pleine à poisson . <EOS>
= = = = =
 The fridge is full of food. -> le frigo est la nourriture . <EOS>
= = = = =
 The cat fell asleep on the mat. -> le chat est tombée en train de dormir dans le tapis . <EOS>
= = = = =
 my brother likes pizza. -> mon frère aime la pizza . <EOS>
= = = = =
```

Figure 4: Prediction with teacher forcing training.

References

[1] How to generate text: using different decoding methods for language generation with transformers.

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. Number arXiv:1409.0473. arXiv.

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. Number arXiv:1810.04805. arXiv.

[4] Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. Effective approaches to attention-based neural machine translation. Number arXiv:1508.04025. arXiv.

[5] Thang Luong, Kyunghyun Cho, and Christopher Manning. Neural machine translation.

[6] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. Number arXiv:1802.05365. arXiv.

[7] Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. Modeling coverage for neural machine translation. Number arXiv:1601.04811. arXiv.