1 Question 1

The **greedy decoding** method is a simple yet effective method to decode sequences. It choose, at each time step, the most probable word given previous words:

$$\hat{y_t} = \underset{y_t \in V_{target}}{\text{argmax}} \log p(y_t|\mathbf{x}, \widehat{y_1}, \ldots, \widehat{y_{t-1}}).$$

This method is being used because:

- This is a very effective method in terms of computational and memory cost. It only stores one prediction and appends only one next word per time.
- It gives a fairly good target sequence with high probability as the word that maximizes the log probability at each step is chosen.

However, this method also has several drawbacks:

- As we store only one prediction sentence, as we produce translation, we cannot undo if we produce a wrong word.
- As a sequence of this, this method often does not give the best translation among possible translations.
- As the most probable word is chosen at each time, this decoding method often chooses highly probable words and therefore can generate repeated words.

In order to fix this problem, **beam search decoding** was proposed, where we store k-best translations B_t^k at each time t and choose among all the possibilities the k-best translations in the next generation:

$$B_t^k = \underset{\substack{(\widehat{y_1},\ldots,\widehat{y_{t-1}}) \in B_{t-1}^k \\ y_t \in V_{target}}}{k\text{-argbest}} \log p(y_t, \widehat{y_{t-1}}, \ldots, \widehat{y_1}|\mathbf{x}).$$

Although this method does not very efficient, it can produce very good translations with higher overall probabilities.

2 Question 2

Our translations suffer from a major problem: *lack of coverage*. This means that the decoder doesn't know if it has produced the corresponding translations of each word in the source sequence. This leads to the two following issues which are observed in our translations:

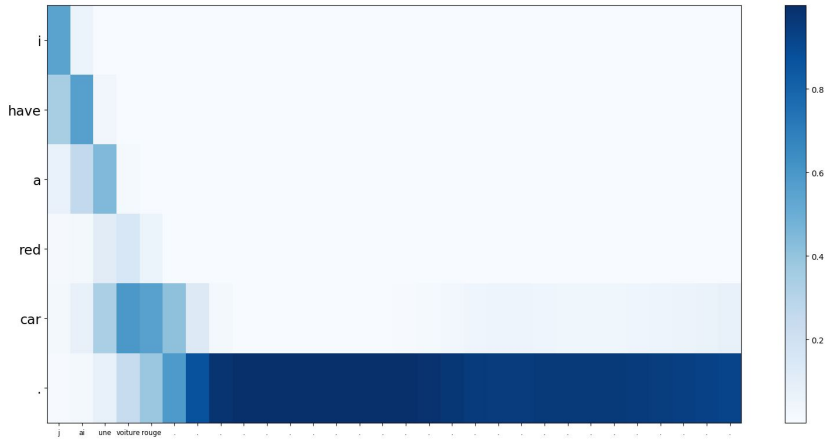
- Over-translation: Some words are repeated multiples times.
- Under-translation: Some words that appears in the source sentence are missing in the translation or wrongly translated.

In order to remediate this issue, *Luong et al. (2015)* ([1]) proposed an **Input-feeding** approach which concatenate the attentional hidden state $\tilde{\mathbf{h}}_t$ with the input vector of the next time step. This approach helps the network to be aware of previous alignments and therefore avoids to generate repeated words in translations, hence gives better translations.

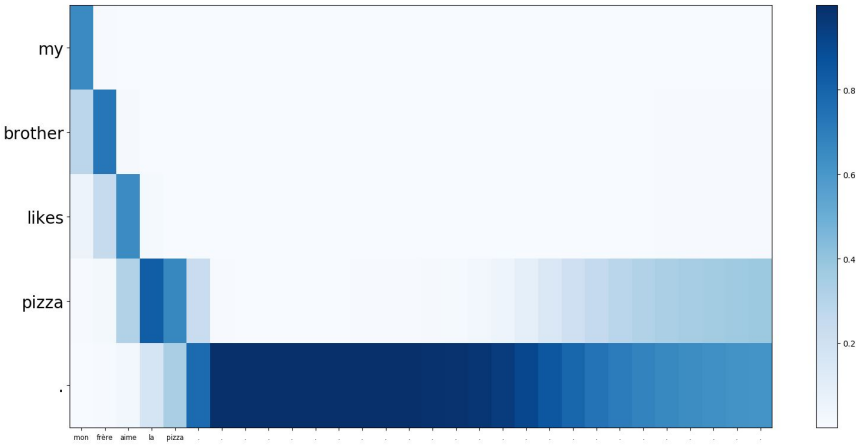
In addition to this approach, *Tu et al. (2016)* ([2]) proposed a coverage mechanism which consisting in using coverage vector $\mathcal{C}_{t,j}$ for the source word x_j that is updated at each time step t based on the coverage vector of the previous time step $\mathcal{C}_{t-1,j}$, the hidden state h_t and the attentional score $\alpha_{t,j}$. This approach has been shown to fix the coverage issues such as over or under-translation and gives a good alignment quality.

3 Question 3

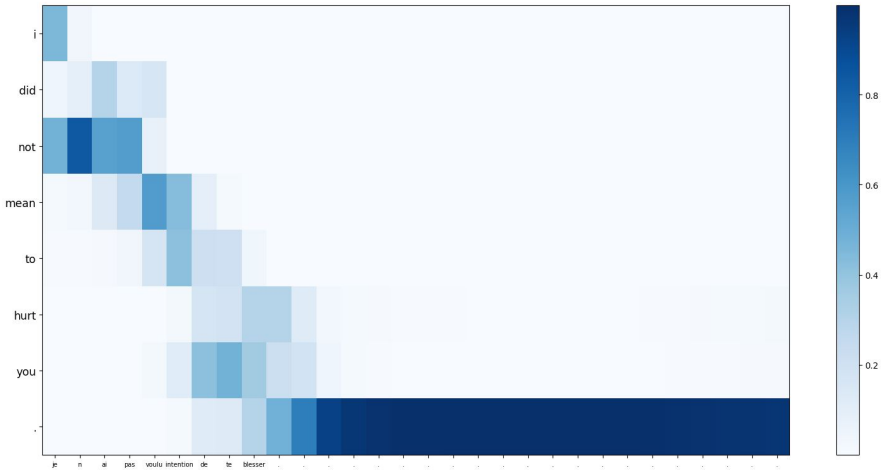
We obtain the following figure for the source sentence *I have a red car.* :



We see that the words are well aligned. We observe here an inversion effect for the noun phrase **red car** which is translated to **voiture rouge** (Noun-Adj inversion). We see in addition that the repeated **points** in the translation all focus on the **point** in the source sentence. Now, we'll observe the following figure for the translation of the sentence *my brother likes pizza.* :



We see that the word **pizza** is translated to **la pizza**. The word **la** also pays attention to the word **pizza** in order to put the correct gender. Finally we try the example *I did not mean to hurt you.* :



We observe here several interesting behaviors. The first one is the inversion **hurt you** and **te blesser** (Direct Object inversion). The next one is the alignment **not** with **n .. pas**, which is the negation structure. We also find an alignment in the past tense with **did** and **ai**.

4 Question 4

We obtain the following translations for the given input sentences:

1. I did not mean to hurt you . $\rightarrow$ je n ai pas voulu intention de te blesser .
2. She is so mean . $\rightarrow$ elle est tellement méchant .

We see that the model succeeds to translate the same word **mean** into 2 completely different words with different meanings (*homophone*): **avoir intention** et **méchant**, which is very hard to accomplish if we don't use an attention model. This difference comes from the different contextual representations of words, in which the vectors representing words also depend on the context of the surrounding sentence. The appearance of attentional models makes it able to compute the context-dependent words representations based on the output of pre-trained networks (BERT, ELMo) which are far better than traditional word vector models. Nowadays, this approach is extremely useful and can handle well most of NLP tasks.

References

[1] Thang Luong, Hieu Pham, and Christopher D. Manning. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal, September 2015. Association for Computational Linguistics.

[2] Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. Modeling coverage for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 76–85, Berlin, Germany, August 2016. Association for Computational Linguistics.