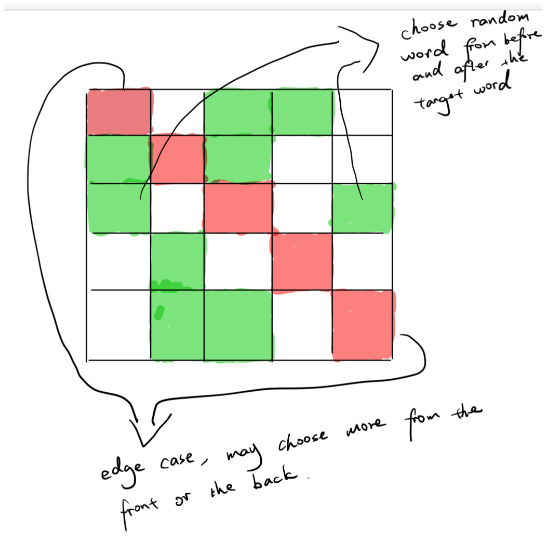
Question1
It is not going to perform very well on long sequences. When the input sequence is long, RNN encoder needs to encode more information but to a limit. However the encoder has a fix length vector, so information loss will happen after a threshold of sequence length. Therefore, the decoder of the RNN will not have all the information of the sequence to generate a good translation.

Question2

We can incorporate attention into the model to improve the performance of long sequence. Instead of only using the last hidden state of the encoder that only has limited information, attention enables decoder to have direct access to all the hidden states of the encoder. Precisely, it is achieved by calculating similarity scores between decoder and all encoders, generating attention weights by applying softmax to these similarity score, producing the context vector by the weighted sum of these encoder vector based on the attention weights, and then at last, concatenating the context vector to the decoder hidden vector to generate prediction.In conclude, this allows the decoder to pay more attention to the more related information in the encoder vectors instead of the more "nearby" encoder vector.

Question3
The parallelization in transformer makes training larger datasets possible.Instead of RNN unrolled sequentially, the transformer doesn't use recurrence but purely on attention. Precisely it uses a self-attention mechanism, which using each word as a query to access and incorporate information from a set of word value. In this way, transformer doesn't need to process the dataset sequentially, so it can use hardware resources in a parallel manner and therefore can be faster and more efficient when processing larger dataset than RNN.

Question4
Another way self-attention pattern, which I think of, is to sparsify the full self-attention by only attending to a fixed-number of random tokens before and after the token instead of fixed-size window surrounding each token. For example, as illustrated in the figure below, 5 word tokens in total and for a fixed size of attention of 2, the green block are the randomized index token for the target token(red) to calculate attention to.


choose random word from before and after the target word
edge case, may choose more from the front or the back.

One strength will be that it will be more computational efficient, rather than O(n^2) performance, it will be O(n). Also compares to fixed-size window surrounding each token that only pay attention to a radius of tokens around the target token, this randomized way can have a higher probability to incorporating important information in a long sequence, since the word it pays attention to can be any word in the sequence.

One limitation will be that for a shorter sequence that information are more compacted, this model may not work that well because it could pay attention to random word in the sequence that very far from the target token. Also the randomization needs to be carefully controlled or it can keep paying attention to the same token.

Question5
The Linformer architecture proposed by Wang et al, is an example of an approach that can make the Transformer architecture more efficient other than sparsifying self-attention.The key take-away the researchers in Linformer demonstrate, both theoretically and empirically, that the stochastic matrix formed by self-attention mechanism is low-rank. By leverage this observation, they propose a new, highly efficient self-attention mechanism that makes the calculation of the attention weights to be computed in O(N) time, which is a big improvement from O(N^2) time. The experiments have shown that the performance of Linformer is comparable to that of the standard Transformer, while a lot more efficient in computation.

Reference: Linformer: Self-Attention with Linear Complexity https://arxiv.org/pdf/2006.04768v3.pdf