Q1.1

I think this architecture in figure 1 will not perform as good in a long sequence. This is because of the information bottleneck issue. When we have a long sequence in the encoder, a lot of information will be compressed into the last hidden state and the information might be lost and it will be very hard for the decoder to recover the information and to get the accurate translation. Also, with long sequences in the encoder, the vanishing gradient problem might arise when performing backpropagation.

Q1.2:

We can introduce an attention algorithm. 1) Calculate a similarity score between the decoder hidden vector and all the encoder hidden vectors. 2) Pass similarity score vector through a softmax to generate attention weights. 3) Use the attention weights to generate the context vector by taking a weighted sum of the encoder hidden vectors 4) concate the context vector to the decoder hidden vector 5) Proceed to the decoder as usual

By adding attention, the decoder has direct access to all of the hidden states of the encoder. Thus when predicting each word in the decoder, it aligns and searches through the encoder input where the most pertinent information focuses and then decodes based on this context vector. In this way, the information bottleneck is alleviated. In short, with attention, decoder has access to all the encoder input information and knows where to focus when predicting

Q2:

Transformers are solely attention-based models. Transformers are faster than RNN because RNN processes the inputs and predicts the outputs sequentially whereas transformers just need one attention layer which is just matrix multiplications and can be parallelly computed. Also, transformers don't suffer from vanish gradients that are associated with the length of the sequence as it's not sequential computing. Also, Positional embeddings that encode the position information still allows the transformers to learn the word position when it gets rid of sequential processing. With transformers coping with the above issues, it allows training on larger dataset than once possible.

Q3.1

The time complexity for sliding window attention is O(n*w) where w is the window size. While its time complexity is smaller than the full self-attention, it can be disadvantageous. For example, we have a long sentence with complicated syntax such as "The cat, …….., is… ". If we do the sliding window attention for word "is" and the window size isn't large enough to capture the information from "cat". The algorithm thus wouldn't know the reason why the word "is" singular is because the word "cat" is singular. In order to cope with this issue, sliding window attention will need to increase w size or more layers, which increase the memory and computation cost in this perspective. Thus self-attention mechanisms in such scenarios would outperform.

Q3.2

Another alternative attention variant is called global sliding window attention. It worked similar to sliding window attention except we added "global attention" to a few selected tokens.
The attention operation is symmetric: " a token with a global attention attends to all tokens across the sequence, and all tokens in the sequence attend to it" (Reference 1)

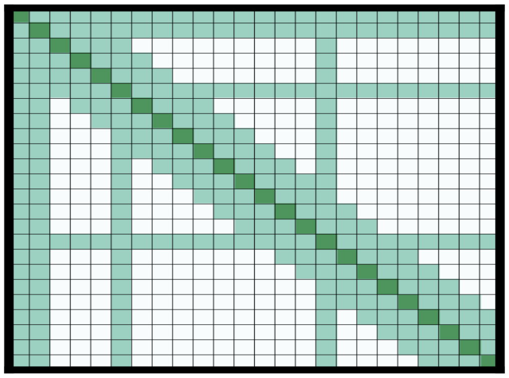
Figure 1(Global + Sliding Window Attention)

There are actually a few strength:

1) We can select which tokens to use global attention, thus we allow more specific representations to the task and model.
2) Since the number of tokens that uses global attention is limited, the complexity is still O(n)
3) Due to the addition of global attention, it performs better than just sliding window and faster than self-attention
4) “While specifying global attention is task specific, it is a easy way to add inductive bias to the model’s attention, and it is much simpler than existing task specific approaches that use complex architecture to combine information across smaller input chunks” (Reference 1)

Weakness:

1) As we have to select which tokens to use global attention, human effort is needed to select. It might require some preprocesses to find the tokens that need global attention. Sometimes preprocessing might need a lot of computational effort too. If we found the wrong tokens to use global attention, it wouldn't perform well.

Bonus Question:

Another alternative attention function is additive attention. One advantage of additive attention wrt to dot-product is that when queries and keys are vectors of different lengths, additive has to be used because for dot-product they have to be the same vector length in order to perform the operation.Also,we can use two different matrix to learn hi and Sj separately. In this way, additive attention is more flexible. Also, if we don’t scale dot-product attention, additive attention will perform better on larger dimensions. One disadvantage will be that dot-product attention is matrix multiplication thus can be parallel computing and thus more time and space efficient than additive attention.

$$f_{att}(\mathbf{h}_i, \mathbf{s}_j) = \mathbf{v}_a{}^{\top} \tanh(\mathbf{W}_1 \mathbf{h}_i + \mathbf{W}_2 \mathbf{s}_j)$$

(Equation for additive attention)

Reference:

Beltagy, Iz, et al. “Longformer: The Long-Document Transformer.” *ArXiv.org*, 2 Dec. 2020, https://arxiv.org/abs/2004.05150v2.