g) The input sentences are padded to equal lengths with a pad token that contains no information. A mask is used to mark the location of the paddings. Masks prevent the decoder to output 'pad' padding tokens that are present in nearly every batch. The step() function helps set the padded locations in the attention vector et to -infinity to zero out those locations in the attention distribution αt after calculation.

i) **Dot product attention**

1. Simple, computationally easy to calculate as it doesn't contain any learnable parameters.
2. It's not very expressive and only measures the degree of alignment of the decoder and encoder hidden states. It therefore forces the decoder and encoder to have very similar embedding spaces. It can be viewed as a form of regularization that may hurt performance for certain language pairs that are very dissimilar.

Multiplicative attention

1. It is more expressive and allows the encoder and decoder to develop linearly dependent word vector representations. Since the weight matrix doesn't have to be square both embedding spaces can have different dimensions.
2. However there's additional computational cost compared to the dot product variant. Additive attention gives the decoder and encoder the most freedom to develop independent embedding spaces. The mapping has more degrees of freedom and allows for an affine non-linear mapping between the encoder and decoder space.

Additive attention

1. Similar computational complexity as Multiplicative attention. Better computational efficiency in higher dimensions.
2. Dimension is another hyperparameter.

2.) a) Since Cherokee is a polysynthetic language, subword level approach would be better as it extracts semantics from letter combination and morphemes that make up the word.

b) We use 1 D CNN for character embedding. The position invariance of the convolutional filters enables us to capture the meaning of a certain letter combination no matter where in the word such combination appears. Scalars are used. Since we extract information from smaller combinations of letters of a word, the embeddings are smaller.

c) Multilingual systems trained on a large amount of data of different languages improves over a baseline bilingual model and that it is also capable of performing zero-shot translation, assuming that the zero-shot source and target languages have been observed during training paired with some other languages. This makes model familiar with translations and thus can be trained on predicting languages with few resources.

d) (i) Reason: Model output two identical formulations. Not sufficient examples in the training set.
Suggestion: Need to include more words in a variety of sentences while training. Can increase attention over the translated sentence so the model can access what has been generated so far to avoid duplication.

(ii) Reason: Translation is too literal. Also it incorrectly assigns gender .
Suggestion: Need longer context from source to the target to retain certain features. Potential fix is subword modeling or simply training for a longer time.

(i) Reason: the word ‘yitsadawoesdi’ is new to the model and thus couldn’t translate it correctly.
Suggestion: Since the model could predict other words correctly, we need to include more variety of words. Can use reverse self training

e) (i)“Oh, Charlotte,” he said.
Not in the training set. There is not much semantics in this sentence. Model was able to retain certain features like gender, correct exclaim, etc.

(ii)correct translation: But Abraham saith, They have Moses and the prophets; let them hear them.
NMT translation: Then Abraham said unto him, Moses and the prophets are with thee: the same shall go away.
Increase training time.

f)
i) C1) p1 = 0.2(0+1+1+1+0) = 0.6
p2 = 0.25(0+1+1+0) = 0.5
r* = 4, c=5, BP=1

BLEU = 1(0.5*0.6 + 0.5*0.5) = 0.55

C2) p1 = 0.2(1+1+0+1+1) = 0.8
p2 = 0.25(1+0+0+1) = 0.5
r* = 4, c=5, BP=1

BLEU = 1(0.5*0.8 + 0.5*0.5) = 0.65

The result also makes intuitive sense. C2 looks like a better translation.

ii) C1) p1 = 0.2(0+1+1+1+0) = 0.6
p2 = 0.25(0+1+1+0) = 0.5
r* = 6, c=5, BP=0.82

BLEU = 0.82(0.5*0.6 + 0.5*0.5) = 0.45

C2) p1 = 0.2(1+1+0+0+0) = 0.4
p2 = 0.25(1+0+0+0) = 0.25
r* = 6, c=5, BP=0.82

BLEU = 0.82(0.5*0.4 + 0.5*0.25) = 0.27

The result is flipped and no longer matches the intuition.

iii.) A large amount of sentence translation has a lot of ambiguity about it and there's not a single correct answer. Since there are many possible translations, by providing multiple independent translations NMT system has a chance to see multiple answers to the same input sentence and learn them. BLEU score is also more accurate when there's more than one reference translation as was demonstrated in part (ii).

iv.) BLEU score is an algorithm and therefore is scalable and it's possible to do it automatically on a computer. It is also possible to calculate it relatively cheaply for large datasets.
Human translation can provide a lot of context when it comes to common sense and general world understanding that BLEU score lacks. Understanding the world and basic logic is very hard for an NLP system and it's not possible to infer how the world functions only from seeing individual sentences without broader context. That's where the human rating of translations can't be replaced.