Stanford CS224n Assignment 4

Aman Chadha

February 6, 2021

1 Neural Machine Translation with RNNs (45 points)

In Machine Translation, our goal is to convert a sentence from the source language (e.g. Cherokee) to the target language (e.g. English). In this assignment, we will implement a sequence-to-sequence (Seq2Seq) network with attention, to build a Neural Machine Translation (NMT) system. In this section, we describe the training procedure for the proposed NMT system, which uses a Bidirectional LSTM Encoder and a Unidirectional LSTM Decoder.

(g) (3 points) (written) The `generate_sent_masks()` function in `nmt_model.py` produces a tensor called `enc_masks`. It has shape (batch size, max source sentence length) and contains 1s in positions corresponding to 'pad' tokens in the input, and 0s for non-pad tokens. Look at how the masks are used during the attention computation in the `step()` function (lines 295-296).

First explain (in around three sentences) what effect the masks have on the entire attention computation. Then explain (in one or two sentences) why it is necessary to use the masks in this way.

Answer:

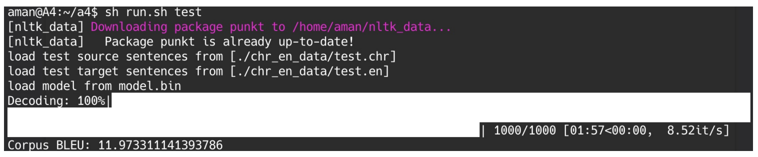
- Padding tokens are an artifact of processing data in batches which is done to speed up training. Using the `pad_sents()` function in `utils.py`, we pad the input sentences to equal lengths with a token that contains no information. Thus, a mask is used to mark the locations of the padding. The `step()` function sets the padded locations in the attention vector e_t to $-\infty$ (the smallest possible float) and thus apportions zero probability weight to these tokens in the attention distribution $\alpha_{t,i} = 0$ for all positions i that correspond to 'pad' tokens (since pre-softmax value of $-\infty$ sets their probability to zero since $e^{-\infty} = 0$). This means that the encoder hidden states h_i^{enc} that correspond to 'pad' tokens will fade away and have no effect on the attention output a_t. In other words, we're simply asking the model to ignore the entries marked as 'pad' tokens.
- It is necessary to apply the masks in this way because we do not want to put any attention on the encoder hidden states that correspond to 'pad' tokens. These 'padding' hidden states were computed because we have to pad the batch of source sentences up to maximum length m, but they are meaningless and thus should not affect our model. Applying the masks in this manner thus avoids skewing the attention distribution α_t due to padding tokens.

(h) (3 points) Once your model is done training (this should take under 1 hour on the VM), execute the following command to test the model:

```
sh run.sh test
(Windows) run.bat test
```

Please report the model's corpus BLEU Score. It should be larger than 10.

Answer:



```
aman@A4:~/a4$ sh run.sh test
[nltk_data] Downloading package punkt to /home/aman/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
load test source sentences from [./chr_en_data/test.chr]
load test target sentences from [./chr_en_data/test.en]
load model from model.bin
Decoding: 100%|
                                                                | 1000/1000 [01:57<00:00,  8.52it/s]
Corpus BLEU: 11.973311141393786
```

BLEU Score: `11.973311141393786`

(i) (4 points) (written) In class, we learned about dot product attention, multiplicative attention, and additive attention. As a reminder,

- Dot Product Attention: $e_{t,i} = s_t^T h_i$
- Multiplicative Attention: $e_{t,i} = s_t^T W h_i$
- Additive Attention: $e_{t,i} = v^T (W_1 h_i + W_2 s_t)$

where, $\{v, W, W_1, W_2\}$ are trainable parameters.

i. (2 points) Explain one advantage and one disadvantage of *dot product* attention compared to *multiplicative* attention.

ii. (2 points) Explain one advantage and one disadvantage of *additive* attention compared to *multiplicative* attention.

Answer:

Dot Product

- Advantages:
  - **Advantage vs. mult and add:** No learnable parameters/weights, thus the simplest, memory and computationally efficient variant.
  - **Advantage vs. mult and add:** Easy to interpret state similarities. Any source hidden state vector that is orthogonal-ish to the target hidden vector will be "washed away". This efficiency can be seen as advantage over the additive mechanism.
- Disadvantages:
  - **Disadvantage vs. mult and add:** Less expressive since it doesn't have any weights W. Can't transform the query or the value vectors; can't weigh the importance of different parts of the vectors.
  - **Disadvantage vs. mult and add:** Both the encoder and decoder are required to have the same embedding space dimensions, i.e., $dim(h_i) = dim(s_t)$ (note that this leads to a square weight matrix).
  - **Disadvantage vs. mult and add:** Returns a scalar, thus less information and flexibility compared to the other two.
- Differences:
  - **Difference vs. mult and add:** Requires that encoder and decoder hidden states with high alignment must also have high dot product similarity. This could be a disadvantage – for example, maybe it's important for the encoder and decoder hidden states to contain different information and thus be very different, even when they're aligned. This may hurt performance for certain language pairs that are very dissimilar. Or it could be an advantage, like a kind of regularization (the high dot product similarity requirement is an extra constraint that 'ties' the representations together).

Multiplicative

- Advantages:
  - **Advantage vs. dot:** Encoder and decoder hidden states, h_i and s_t respectively, can have different dimensions (note this is why we used multiplicative rather than dot product attention in the NMT system in part 1!).
  - **Advantage vs. dot:** More expressive since the learnable parameter matrix W allows us to linearly transform the source's hidden state vectors.

  - **Advantage vs. additive:** Generally more efficient to compute as there are fewer steps and no tanh.

- Disadvantages:
  - **Disadvantage vs. dot:** Additional computational cost owing to figuring out the optimal set of parameters W compared to the dot product variant, especially in high dimensional spaces.
  - **Difference vs. additive:** Not as flexible as additive attention which enables more degrees of freedom using two learned parameter matrices W_1 and W_2 (while additive attention involves only a single learned parameter matrix W).

- Differences:
  - **Difference vs. additive:** The (transformed) query and vector are combined in a multiplicative way rather than an additive way (i.e., if you expanded the equations, you would find terms containing a multiplication of elements from s_t and from h_i). It's hard to say if this is an advantage or a disadvantage, but it's a different way of combining the information - perhaps better suited to some applications than others.

Additive

- Advantages:
  - **Advantage vs. dot:** Encoder and decoder hidden states can have different dimensions.
  - **Advantage vs. dot:** More expressive (can learn to transform the vectors by learning the weight matrices W_1, W_2, and learn to combine them by learning v). Specifically, both the source and target hidden vector states get their own learned transformations W_1 and W_2 respectively. This mapping enables the decoder and encoder more degrees of freedom and thus flexibility in developing independent embedding spaces. This effectively allows for an affine non-linear mapping between the encoder and decoder space.
  - **Advantage vs. mult:** According to the paper "Massive Exploration of Neural Machine Translation Architectures", additive attention slightly but consistently outperforms multiplicative attention, at least within their particular NMT setting.
  - **Advantage vs. mult (and dot):** Additive attention can be viewed as a small feedforward neural network whose input is the concatenation $[h_i, s_t]$, followed by one linear layer with a $tanh$ nonlienarity, then another linear layer to get a scalar output. As such, it has the advantage of generality – that is, provided the internal attention dimension is large enough, it is a universal approximator.

- Disadvantages:
  - **Disadvantage vs. dot:** Computationally expensive since three sets of parameters $W1$, $W2$ and v need to be learnt. The number of attention operations grows approximately as $O(n^2)$, where n is the mean input and output sequence length.
  - **Disadvantage vs. mult:** Generally more computationally expensive due to tanh and multiple steps (note that additive attention does not necessarily contain more weights than multiplicative – this is dependent on the choice of hyperparameters).

- Differences:
  - **Difference vs. mult (and dot):** Additive attention requires that you choose the attention dimension, which is another hyperparameter of the network. This could be an advantage (something to tune to get better performance) or a disadvantage (more complex model).

2 Analyzing NMT Systems (30 points)

(a) (2 points) In part 1, we modeled our NMT problem at a subword-level. That is, given a sentence in the source language, we looked up subword components from an embeddings matrix. Alternatively, we could have modeled the NMT problem at the word-level, by looking up whole words from the embeddings matrix. Why might it be important to model our Cherokee-to-English NMT problem at the subword-level vs. the whole word-level? (Hint: Cherokee is a polysynthetic language.)

Answer:

- Cherokee is a polysynthetic language, which means its words are composed of many morphemes (subword components that may have independent meanings, but which can't stand alone).
- Polysynthetic words have a high morpheme-to-word ratio, meaning that a single word can be composed of a bunch of morphemes. (Sometimes entire sentences are composed of just a single word modified by a bunch of morphemes.)
- Given so much of the meaning in a single word is contained at the subword morpheme-level, we need to use a subword model to best capture that meaning – word-level models fail to capture that complexity.

(b) (2 points) Character-level and subword embeddings are often smaller than whole word embeddings. In 1-2 sentences, explain one reason why this might be the case.

Answer: The embedding sizes for characters and subwords are smaller than whole words because:

- A typical character/subword vocabulary is several orders of magnitude smaller than a word vocabulary (consider the English language for example). In other words, the number of unique characters is much smaller than the number of unique words (i.e. number of unique combinations of those characters).
- Characters or subwords do not usually encode as much information as an entire word (they usually contain more lexical and less semantic information).
- Additionally, since individual characters/subwords typically have much less meaning compared to individual words, character dependencies turn out to be not as long-ranged compared to words thus requiring smaller embeddings.

(c) (2 points) One challenge of training successful NMT models is lack of language data, particularly for resource-scarce languages like Cherokee. One way of addressing this challenge is with multilingual training, where we train our NMT on multiple languages (including Cherokee). You can read more about multilingual training here. How does multilingual training help in improving NMT performance with low-resource languages?

Answer:

- NMT systems are limited in translating low-resource languages owing to the challenge of having significant amounts of supervised training data to learn useful mappings between languages. Multilingual NMT tackles these challenges associated with low-resourced language translation.
- The underpinning principle of multilingual NMT is to force the creation of hidden representations of words in a shared semantic space for linguistically similar languages. This enables models to be better at generalization, and capable of capturing the representational similarity across a large body of languages.
- This is a classic example of transfer learning – the model takes insights gained by training on one language and applies it to the translation of other languages.

(d) (6 points) Here we present three examples of errors we found in the outputs of our NMT model (which is the same as the one you just trained). The errors are underlined in the NMT translation sentence. For each example of a source sentence, reference (i.e., 'gold') English translation, and NMT (i.e., 'model') English translation, please:

1. Provide possible Possible reason(s) why the model may have made the error (either due to a specific linguistic construct or a specific model limitation).
2. Describe one possible way we might alter the NMT system to fix the observed error. There are more than one possible fixes for an error. For example, it could be tweaking the size of the hidden layers or changing the attention mechanism.

Below are the translations that you should analyze as described above. Only analyze the underlined error in each sentence. Rest assured that you don't need to know Cherokee to answer these questions. You just need to know English! If, however, you would like additional color on the source sentences, feel free to use resources like https://www.cherokeedictionary.net to look up words.

1. (2 points) **Source Translation:** Yona utsesdo ustiyegv anitsilvsgi digvtanv uwoduisdei.
   Reference Translation: Fern had a crown of daisies in her hair.
   NMT Translation: Fern had her hair with her hair.

   Answer (two options):

   (a) Error: Unrealized concept (*crown of daisies*).

   Possible reason(s): The model does not understand the notion of a *crown of daisies.*

   Potential fix(es): The model needs to be trained on data that conveys the idea of a *crown of daises.*

   (b) Error: Repetition (*hair*).

   Possible reason(s): Repetition is a common problem in the output of RNN decoders. A possible reason is that the model attended to *hair* twice, thus producing it twice in the translation. Repetition can also be a problem with the decoding algorithm (e.g. greedy decoding/beam search).

   Potential fix(es): Look at the attention distribution to distinguish between possible cases above. Introduce an explicit penalty for word repetition (either during training or decoding), or add a coverage mechanism (which discourages repeated attention on the same word). Try a different decoding algorithm if greedy decoding or beam search is causing the problem.

2. (2 points) **Source Sentence:** Ulihelisdi nigalisda.
   Reference Translation: She is very excited.
   NMT Translation: It's joy.

   Answer:
   Error: Incomplete sentence/awkward phrasing.
   Possible reason(s): The model is unable to infer that the subject referred to in the sentence is female, which leads is to assign an incorrect gender token. Cherokee and English have different grammar; the Cherokee words in the source sentence roughly mean (in order): "happiness/gladness," and "it becomes". One reason for this error could be lack of context which NMT systems utilize during translation. If the model saw more of the preceding context, perhaps it would be more obvious that the pronoun should be *she* (and not *it*). The model is limited in that it is unable to attend over a wide range of dependencies.
   Potential fix(es): Adding world knowledge to NMT systems is an ongoing research area; there are efforts to integrate NMT systems with knowledge bases. We could try training and testing our NMT systems on whole documents rather than single sentences to solve the context issue. To enable the model to attend to a wide range of dependencies, increase the model's complexity by say, increasing the number of hidden units or the depth of the model.

3. (2 points) **Source Sentence:** Tsesdi hana yitsadawoesdi usdi atsadi!
   Reference Translation: Don't swim there, Littlefish!
   NMT Translation: Don't know how a small fish!

   Answer:
   Error: Mistranslation of a proper name.
   Possible reason(s): The name *Littlefish* was literally translated to *a small fish.* The model tracks subwords and whole words based on frequency of occurrence. If 'Littlefish' wasn't a word in its vocabulary, then it must not have been common enough for the model to add. The proper name *Littlefish* might not be present in the vocabulary of the training data (OOV, out-of-vocabulary error). As a result, it would've broken down the name into smaller pieces/subword components which were more common, and therefore were in its vocabulary (i.e. 'little' and 'fish').
   Potential fix(es): Either more training data where the proper name is used (to add the name to the model's vocabulary) or learn to "copy" from source text ("Pointing the unknown words" by Gulcehre et al., 2016). The source word can be identified using the `argmax` of the attention weights vector.

(e) (4 points) Now it is time to explore the outputs of the model that you have trained! The test-set translations your model produced in question `1-h` should be located in `outputs/testoutputs.txt`.

1. (2 points) Find a line where the predicted translation is correct for a long (4 or 5 word) sequence of words. Check the training target file (English); does the training file contain that string (almost) verbatim? If so or if not, what does this say about what the MT system learned to do?
2. (2 points) Find a line where the predicted translation starts off correct for a long (4 or 5 word) sequence of words, but then diverges (where the latter part of the sentence seems totally unrelated). What does this say about the model's decoding behavior?

Answer:

1. **NMT Translation**: Line 957: "I don't want to die.
   Reference: Line 957: "I don't want to die."
   Training Sample: Line 4653: "I don't want to die!
   Reason: The training set contains the string verbatim at line 4653 in the training sample and similar such strings at multiple locations (line 454, 1887). The fact that there are several instances in the training set that contain the same phrase indicates that the model has learned to memorize the training data.
2. **NMT Translation**: Line 77: Fern was sitting in the edge of Fern, and I'd going to another.
   Reference: Line 77: Fern was upstairs changing her sneakers.
   Reason(s):
   - The model struggles at longer range dependencies. The model is conditioned on its own predictions during testing, unlike in training it always sees the ground truth, so errors could accumulate to further distort longer predictions.
   - The model needs to be trained on more data for it to understand the concept of *sneakers* and the fact they can be *changed*. Line 6872 is the only line in the training set that mentions the concept of *sneakers*, and as such this is not enough for the model to build context.

(f) (14 points) BLEU score is the most commonly used automatic evaluation metric for NMT systems. It is usually calculated across the entire test set, but here we will consider BLEU defined for a single example. Suppose we have a source sentences, a set of k reference translations $r_1, ..., r_k$, and a candidate translation c. To compute the BLEU score of c, we first compute the *modified n-gram precision* p_n of c, for each of $n = 1, 2, 3, 4$, where n is the n in n-gram:

$$p_n = \frac{\sum_{ngram \in c} \min(\max_{i=1,...,k} Count_{r_i}(ngram), Count_c(ngram))}{\sum_{ngram \in c} Count_c(ngram)}$$

Here, for each of the n-grams that appear in the candidate translation c, we count the maximum number of times it appears in any one reference translation, capped by the number of times it appears in c (this is the numerator). We divide this by the number of n-grams in c (denominator).

Next, we compute the *brevity penalty* BP. Let *len(c)* be the length of c and let *len(r)* be the length of the reference translation that is closest to *len(c)* (in the case of two equally-close reference translation lengths, choose *len(r)* as the shorter one).

$$BP = \begin{cases} 1, & \text{if } len(c) \geq len(r) \\ \exp\left(1 - \frac{len(r)}{len(c)}\right), & \text{otherwise} \end{cases}$$

Lastly, the BLEU score for candidate c with respect to $r_1, ..., r_k$ is:

$$BLEU = BP \exp\left(\sum_{n=1}^{4} \lambda_n \log p_n\right)$$

where $\lambda_1, \lambda_2, \lambda_3, \lambda_4$ are weights that sum to 1. The log here is natural log.

1. (5 points) Please consider this example:

   Source Sentences: el amor todo lo puede
   Reference Translation r_1: love can always find a way
   Reference Translation r_2: love makes anything possible
   NMT Translation c_1: the love can always do
   NMT Translation c_2: love can make anything possible

   Please compute the BLEU scores for c_1 and c_2. Let $\lambda_i = 0.5$ for $i \in \{1, 2\}$ and $\lambda_i = 0$ for $i \in \{3, 4\}$ (this means we ignore 3-grams and 4-grams, i.e., don't compute p_3 or p_4). When computing BLEU scores, show your working (i.e., show your computed values for p_1, p_2, *len(c)*, *len(r)* and *BP*). Note that the BLEU scores can be expressed between 0 and 1 or between 0 and 100. The code is using the 0 to 100 scale while in this question we are using the 0 to 1 scale. Which of the two NMT translations is considered the better translation according to the BLEU score? Do you agree that it is the better translation?

   Answer:

   For c_1,

Unigram	$\max(Count_{r_i}, Count_c)$
the	0
love	1
can	1
always	1
do	0

Bigram	$\max(Count_{r_i}, Count_c)$
the love	0
love can	1
can always	1
always do	0

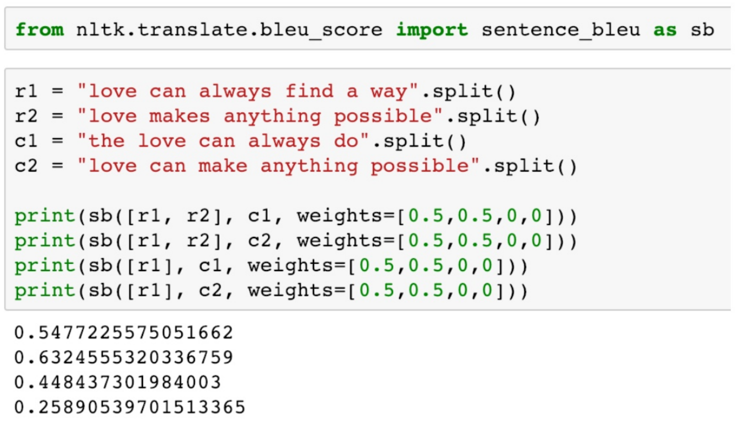
   $p_1 = \frac{0+1+1+1+0}{5} = \frac{3}{5} = 0.6.$
   $p_2 = \frac{0+1+1+0}{4} = \frac{1}{2} = 0.5.$
   $len(c) = 5$, $len(r) = 4$ (we chose the closest reference length as 4 because 4 and 6 are equally close to 5). Since $c \geq r$, $BP = 1$.
   BLEUc_1 = `1 * np.exp(0.5*np.log(0.6) + 0.5*np.log(0.5))` = **0.5477**

For c_2,

Unigram	$\max(Count_{r_i}, Count_c)$
love	1
can	1
make	0
anything	1
possible	1

Bigram	$\max(Count_{r_i}, Count_c)$
love can	1
can make	0
make anything	0
anything possible	1

$p_1 = \frac{1+1+0+1+1}{5} = \frac{4}{5} = 0.8.$
$p_2 = \frac{1+0+0+1}{4} = \frac{1}{2} = 0.5.$
$len(c) = 5$, $len(r) = 4$. Since $c \geq r$, $BP = 1$.
BLEUc_2 = `1 * np.exp(0.5*np.log(0.8) + 0.5*np.log(0.5))` = **0.6324**
Note that the numbers in both this part and the next can be validated using nltk like so:



```
from nltk.translate.bleu_score import sentence_bleu as sb
```

```
r1 = "love can always find a way".split()
r2 = "love makes anything possible".split()
c1 = "the love can always do".split()
c2 = "love can make anything possible".split()

print(sb([r1, r2], c1, weights=[0.5,0.5,0,0]))
print(sb([r1, r2], c2, weights=[0.5,0.5,0,0]))
print(sb([r1], c1, weights=[0.5,0.5,0,0]))
print(sb([r1], c2, weights=[0.5,0.5,0,0]))
```

```
0.5477225575051662
0.6324555320336759
0.448437301984003
0.25890539701513365
```

According to these BLEU scores, NMT translation c_2 is the better one. c$_2$ is indeed the better translation - it has the correct meaning and makes intuitive sense, whereas c$_1$ translates the Spanish phrase too literally, leading to unnatural and nonsensical English.

2. (5 points) Our hard drive was corrupted and we lost Reference Translation r_2. Please recompute BLEU scores for c_1 and c_2, this time with respect to r_1 only. Which of the two NMT translations now receives the higher BLEU score? Do you agree that it is the better translation?

Answer:

For c_1,

Unigram	$\max(Count_{r_i}, Count_c)$
the	0
love	1
can	1
always	1
do	0

Bigram	$\max(Count_{r_i}, Count_c)$
the love	0
love can	1
can always	1
always do	0

$p_1 = \frac{0+1+1+1+0}{5} = \frac{3}{5} = 0.6.$
$p_2 = \frac{0+1+1+0}{4} = \frac{1}{2} = 0.5.$
$len(c) = 5$, $len(r) = 6$. Because $c < r$, BP = `np.exp(1-6/5)` = 0.8187.
BLEUc_1 = `0.8187 * np.exp(0.5*np.log(0.6) + 0.5*np.log(0.5))` = **0.4484**

For c_2,

Unigram	$\max(Count_{r_i}, Count_c)$
love	1
can	1
make	0
anything	0
possible	0

Bigram	$\max(Count_{r_i}, Count_c)$
love can	1
can make	0
make anything	0
anything possible	0

$p_1 = \frac{1+1+0+0+0}{5} = \frac{2}{5} = 0.4$.
$p_2 = \frac{1+0+0+0}{4} = \frac{1}{4} = 0.25$.
$len(c) = 5$, $len(r) = 6$. Because $c < r$, $BP =$ `np.exp(1-6/5)` $= 0.8187$.

BLEUc_2 = `0.8187 * np.exp(0.5*np.log(0.4) + 0.5*np.log(0.25))` = **0.2589**

According to these BLEU scores, NMT translation c_1 is the better translation. I disagree because c_1 doesn't make intuitive sense.

3. (2 points) Due to data availability, NMT systems are often evaluated with respect to only a single reference translation. Please explain (in a few sentences) why this may be problematic.

   Answer:

   - The task of sentence translation is somewhat subjective especially in certain scenarios and usually doesn't have a single correct answer. In other words, often there are many valid ways to translate a source sentence. This is particularly true for idiomatic phrases such as the previous example. The BLEU metric is designed to accommodate this flexibility: an n-gram in c is rewarded if it appears in any one of the reference translations.
   - With only a single reference translation, the BLEU metric only recognizes similarity to that particular translation potentially penalizing other valid candidate translations. As a result, even good translations might receive a low BLEU score due to little n-gram overlaps. By providing multiple independent translations, the NMT system has a chance to learn patterns from multiple different answers for the same input sentence. With more reference sentences, the target space increases with more n-grams available for the model to generate and receive a good BLEU score, as was demonstrated in the above question. This leads to the BLEU metric rewarding similarity to any of the several valid translations.

4. (2 points) List two advantages and two disadvantages of BLEU, compared to human evaluation, as an evaluation metric for Machine Translation

   Answer:

   Advantages:

   - Single quantitative score that aims to remove subjectiveness from the evaluation process. Owing to its objectiveness (unlike human evaluation, which is hard to define and varies depending on the human judge), researchers can reproduce each others' BLEU results and use BLEU to compare different systems.
   - Simple and easy implementation, which implies that it is scalable and it's possible to calculate it relatively cheaply for large datasets.
   - Fully automated evaluation, faster than humans. Human evaluation would have to understand both the source and target languages.

   Disadvantages:

   - BLEU requires a reference translation (whereas human evaluation doesn't, assuming the human judges are bilingual), and optimally requires multiple reference translations.

- BLEU only measures n-gram overlaps, not the quality of the sentences produced by evaluating sentence/corpus semantics, grammar, etc. It doesn't reward synonyms, paraphrases, or different inflections of the same word (e.g., make and makes).
- Human translation can utilize commonsense and general world understanding to render a good translation. Since BLEU lacks this understanding, it can fail to fully capture the true notion of good translation in some cases. It's an extremely difficult task for an NLP system to infer how the world functions using only individual sentences without broader context, to know idioms, and recognizing what 'sounds good' and what doesn't, etc. In these scenarios, the human evaluation of translations is vital and can't be replaced with a BLEU score.