Carolyn Kao (chkao831@stanford.edu)
CS224n Assignment 4
February 11, 2021

1. Neural Machine Translation with RNNs

(g) 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.
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.

```
A primary purpose of using the masks on the entire attention computation is to set
the attention scores to be minus infinity (the smallest possible float) for all positions
associated with pad tokens in the source sentence. This further ensures that the hidden state
of the encoder corresponding to the pad tokens shall not impact the attention output,
as these tokens got negligible weights with its hidden encoded states vanishing away.

We must use the masks in this way because we do not want to put any attention on the encoder
hidden states that correspond to the pad tokens; otherwise we would dampen the attention distribution,
further affecting the model while we don't want to.
```

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

```
epoch 7, iter 2800, cum. loss 55.93, cum. ppl 8.66 cum. examples 6375
begin validation ...
validation: iter 2800, dev. ppl 39.077485
hit patience 1
hit #5 trial
early stop!
chkao831@CS224nHW4:~/a4$ sh run.sh test
[nltk_data] Downloading package punkt to /home/chkao831/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 [01:54<00:00,  8.75it/s]
Corpus BLEU: 11.827701255120617
chkao831@CS224nHW4:~/a4$
[0] 0:bash*                                                                                                    "CS224nHW4" 10:22 10-Feb-21
```

```
The model's corpus BLEU Score is 11.827701255120617.
```

(i) In class, we learned about dot product attention, multiplicative attention, and additive attention. As a reminder, dot product attention is $e_{t,i} = S_t^T h_i$, multiplicative attention is $e_{t,i} = S_t^T W h_i$, and additive attention is $e_{t,i} = v^T \tanh(W_1 h_i + W_2 s_t)$.

Explain one advantage and one disadvantage of dot product attention compared to multiplicative attention.

```
The dot product attention is computationally cheap, as it costs only O(h) to do the dot product.
One disadvantage is that it requires the encoder and decoder to both have states of the same
dimension, so it lacks flexibility to some extent.
```

Explain one advantage and one disadvantage of additive attention compared to multiplicative attention.

The additive attention gives more flexibility than the multiplicative attention does, primarily because the target and source hidden vectors both get their own learned transformation in W1 and W2. In addition, the additive attention is potentially able to capture various relationships between words as it is basically a different operation from the former two attention methods.

However, these advantages also introduce its highest computational complexity, as it is very slow to compute than the former two.

2. Analyzing NMT Systems

(a) 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?

Since Cherokee is a polysynthetic language, the subword approach alone in modelling may not suffice. This is because such a language has very high number of hapax legomena (occurs only once within a context). Additionally, most words are composed on-the-fly, and so would not have been seen during training. This shows the need for appropriate morphological handling of words, without which it is impossible for a model to capture enough word statistics. Hence, with this in mind, we might need to model the Cherokee-to-English NMT problem beyond the pure subword level.

(Reference: Schwartz, L. et al. Neural Polysynthetic Language Modelling. ArXiv abs/2005.05477 (2020))

(b) 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.

The embedding vectors for character-level and subword embeddings are generally shorter than the whole word embeddings because at such lower level compositions, the discrete space is much smaller, as there are only about 97 English-language characters in common usage including punctuation marks, for example. On the other hand, storing the word embeddings require a lot of memory, further adding up many parameters to the model and resulting in a much higher computational cost.

(Reference: https://stats.stackexchange.com/questions/216000/advantage-of-character-based-language-models-over-word-based)

(c) 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). How does multilingual training help in improving NMT performance with low-resource languages?

```
The multilingual training helps in improving NMT performance with low-resource languages like
Cherokee primarily because such mass models learn shared representations for linguistically similar
languages without the need for external constraints. With strong enough positive transfers from
high-resource languages to low-resource languages, the translation quality for those at the tail of
the distribution is shown to be improved by significant BLEU points.
Hence, from this, we could see that such multilingual training would help capture the
representational similarity across a large body of languages.

(Reference: https://ai.googleblog.com/2019/10/exploring-massively-multilingual.html)
```

(d) 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 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.

(i) **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.*

```
A possible reason for this translation error might result from a word being polysemous. Perhaps in
Cherokee, a word within this sentence might hold dual meanings of 'hair' and 'crown of daisies'
at the same time. And because of the context right behind, the NMT system picks a wrong meaning
as it is biased towards picking the more possible meaning in the context.

A possible way of fixing the translation error within the context is to try training and testing
this NMT systems on whole documents rather than on single sentences, in order to increase the
probability of choosing a proper translation.
```

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

```
A possible reason for the model's incapability of capturing the subjective (she/it) is that at the
beginning of this sentence, the model fails to attend to the feminine objective from its context.

A possible way of fixing such an error is to increase the word embedding size beyond a single
sentence to capture the contextual relationship.
```

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

```
A possible reason for this error is that the term 'Littlefish' is an infrequent word of translation
which is outside of the feasible vocabulary; hence the model is only able to output similar
combination of words instead. It is due to a specific linguistic construct.

A possible way of fixing the error is to add a neural copy mechanism to copy specific words
with specific translation from the source sentence where we need special translation for certain
terms like 'Littlefish' in this case.
```

(e) 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-i` should be located in `outputs/test_outputs.txt`.

(e)i: 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?

`(test.en)` **Jesus answered and said unto her,**
`(text_outputs.txt)` **Jesus answered and said unto him,**

`(train.en)` **Jesus answered and said unto him,**

```
Yes, the training file contains the string 'Jesus answered and said unto' verbatim. From this, we
could tell that the MT system might not 'generate' these texts; instead, in this case, it simply
'send them back' from the training target file. This potentially implies some over-fitting might
have occurred in the system.
```

(e)ii: 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?

`(test.en)` **And if thy right hand causeth thee to stumble, cut it off, and cast it from thee: for it is profitable for thee that one of thy members should perish, and not thy whole body go into hell.**
`(text_outputs.txt)` **And if thy right hand to stumble, let him cast it, and cast it out: for it is good for thee, that we may be well: for it is good for thee.**

```
This probably implies that at the model's decoding stage in generating the next word, because of
the divergent nature of the probability for the next probable word, the decoder was unable to decode
the next word in the right direction. In this case, it might help to adopt the  top-k sampling,
in which we could randomly sample from Pt on each step t, restricting to just the top-k most
probable words at a time.
```

(f) 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.

(f)i: Please consider this example from Spanish:

Source Sentence **s**: **el amor todo lo puede**
Reference Translation $\mathbf{r}_1$: *love can always find a way*
Reference Translation $\mathbf{r}_2$: *love makes anything possible*
NMT Translation $\mathbf{c}_1$: *the love can always do*
NMT Translation $\mathbf{c}_2$: *love can make anything possible*

Please compute the BLEU scores for $\mathbf{c}_1$ and $\mathbf{c}_2$. 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?

```
###c1###
# the, love, can, always, do
```
$p_1 = \frac{0+1+1+1+0}{5} = \frac{3}{5}$ by the modified n-gram precision pn of c, n = 1, 2
```
# the love, love can, can always, always do
```
$p_2 = \frac{0+1+1+0}{4} = \frac{1}{2}$
Brevity Penalty: $c = len(c_1) = 5$, the closest reference length is 4, so the brevity penalty is 1.
Hence, BLEU = $exp(0.5 * log(3/5) + 0.5 * log(1/2)) = 0.548$

```
###c2###
# love, can, make, anything, possible
```
$p_1 = \frac{1+1+0+1+1}{5} = \frac{4}{5}$
```
# love can, can make, make anything, anything possible
```
$p_2 = \frac{1+0+0+1}{4} = \frac{1}{2}$
Brevity Penalty: $c = len(c_2) = 5$, the closest reference length is 4, so the brevity penalty is 1.
Hence, BLEU = $exp(0.5 * log(4/5) + 0.5 * log(1/2)) = 0.632$

```
###code###
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]) #output:  0.5477225575051662
print(sb([r1, r2], c2, weights=[0.5,0.5,0,0]) #output:  0.6324555320336759
```

According to these BLEU scores calculated above, the NMT translation c_2 is strictly better than c_1. This implies that c_2 is the better translation. This makes sense to me as c_1 translation is not as natural as c_2 does in English.

f(ii): 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?

```
###c1###
# the, love, can, always, do
```
$p_1 = \frac{0+1+1+1+0}{5} = \frac{3}{5}$

the love, love can, can always, always do
$p_2 = \frac{0+1+1+0}{4} = \frac{1}{2}$
Brevity Penalty: $c = len(c_1) = 5$, the closest reference length is now 6, so the brevity penalty is $exp(1 - 6/5)$. Hence, BLEU = $exp(1 - 6/5) * exp(0.5 * log(3/5) + 0.5 * log(1/2)) = 0.448$

###c2###
love, can, make, anything, possible
$p_1 = \frac{1+1+0+0+0}{5} = \frac{2}{5}$
love can, can make, make anything, anything possible
$p_2 = \frac{1+0+0+0}{4} = \frac{1}{4}$
Brevity Penalty: $c = len(c_2) = 5$, the closest reference length is now 6, so the brevity penalty is $exp(1 - 6/5)$. Hence, BLEU = $exp(1 - 6/5) * exp(0.5 * log(2/5) + 0.5 * log(1/4)) = 0.259$

###code###
```
print(sb([r1], c1, weights=[0.5,0.5,0,0]) #output:  0.448437301984003
print(sb([r1], c2, weights=[0.5,0.5,0,0]) #output:  0.25890539801513365
```

According to these BLEU scores calculated above, the NMT translation c_1 is strictly better than c_2. This implies that c_1 is the better translation. As explained above, it doesn't align with my belief.

f(iii) 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.

The references only serve as some subjective resources of translation. There's no 'correct version of translation'. Hence, we should have multiple reference translations for our BLEU metric to reward similarity to every valid translation, not a particular one. Otherwise, the metric might only recognize the similarity with respect to only a single reference translation and discredit other candidates.

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

Advantages:

(i) It is faster to compute, in comparison to human evaluation.

(ii) It substantially removes subjective components as it provides a reliable quantitative metric to assess the result.

Disadvantages:

(i) Unlike human evaluation, BLEU requires at least one reference translation upon its evaluation.

(ii) The lack of domain and world knowledge of BLUE, unlike human, hinders it from recognizing what words seem to be better choices in many cases. This is also the fundamental limitation that lies in a lot of NLP problems nowadays.