Solutions to CS 224n Assignment #4

Ayaka

1. Neural Machine Translation with RNNs

(g) This is a common technique when using `softmax`. For the `softmax` function, if we want to let some output positions to be 0, we need to set the corresponding input positions to `-inf`. I have also used this technique when implementing Transformer, BERT and BART from scratch.

In this example, we set the attention scores of the `PAD` tokens to be 0, so that the model will not pay attention to these `PAD` tokens in the following calculation.

(h) The training takes about 6.2 minutes to run for 7 epochs on a single NVIDIA A100-PCIE-40GB GPU. I have also attempted to modify the script to make it work on my TPU VM, but it does not work due to the relatively poorer support for PyTorch XLA. I am able to further migrate the PyTorch code to JAX, so that the code will run much faster on TPUs.

The BLEU score with the default hyperparameter setting is 12.9131.

(i) Compared to multiplicative attention, dot product attention is computationally cheaper, but it requires the two vectors to be the same in dimensionality.

Compared to multiplicative attention, additive attention is more flexible since it can map the vectors to arbitrary hidden dimensions. However, additive attention is also more computationally expensive and slower to compute.

2. Analysing NMT Systems

(a) For polysynthetic languages, we must model the NMT problem at the subword-level instead of the whole word-level. This is because words in polysynthetic languages are often compositions of stems, prefixes and suffixes. If we model the problem at the whole word-level, it will lead to a huge vocabulary file and frequent OOVs.

Take Greenlandic, another polysynthetic language that I am also familiar with, as an example. In Greenlandic, *sila* means 'weather', *silagippoq* means 'the weather is good' and *silagissimagaluarpat* means 'if the weather had been good'. If we model the NMT problem at the subword-level, *sila* would be a subword common to all the three words, which implies something about the weather. If we model the problem at the whole word-level, the three words would be totally unrelated, and the third word would possibly be OOV if the training corpus is not large enough.

(b) In Cherokee, *ts-* is a common prefix, but it is separated to Ꮳ *tsa*, Ꮴ *tse*, Ꮹ *tsi*, Ꮻ *tso*, Ꮼ *tsu* and Ꮽ *tse* in the written form, so the model will not get the information that the six characters have *ts-* in common.

Japanese is another language that uses syllabic writing. In Japanese, *kak-* is the stem for 'to write', but it is written as かか *kaka*, かき *kaki*, かく *kaku* and かけ *kake* in different kinds of conjugations. However, it is not a common practice in Japanese NLP to train the model on transliterated texts. Perhaps the impact of this problem would be small if the training corpus is large enough.

Another possible solution is to train the model directly on raw bytes, as proposed in ByT5 [1]. This approach preserves the greatest amount of information. And since the code points of either Ꮳ, Ꮴ, Ꮵ, Ꮶ, Ꮷ, Ꮸ or か, き, く, け are adjacent in Unicode, hopefully the model will be able to figure out that they form the same morphemes in different kinds of conjugations. This method, on the other hand, requires more computational resources.

(c) The intuition of multilingual training is that the knowledge gained through the training on high-resource languages may be possible to be transferred to low-resource languages.

Most of the previous works on multilingual training is English-centric, which means that the models are only trained on the data that are translated from or to English. M2M-100 [2] proposes a novel method that leverages similar language groups. This method, however, is hard to be applied to the Cherokee language, since Cherokee belongs to the Iroquoian language family, while all languages under this language family are short of resources. Generally speaking, multilingual training is beneficial to low-resource languages that are similar to a certain high-resource language (e.g. Norwegian and Danish to English), but it would be less beneficial to those low-resource languages that are not similar to a certain high-resource language at all (e.g. Greenlandic and Northwest Caucasian languages).

Another difficulty in utilizing multilingual training to improve NMT of Cherokee lies in the training data. For low-resource languages, the training corpus are often religious texts, which are quite different from other domains such as daily conversations. Even for high-resource languages, the situation is not ideal either. Take mT5 [3] as an example. mT5 is a multilingual language model trained on mC4 corpus, which covers 101 languages. However, scrutinising the Chinese part of the corpus reveals that it is replete with meaningless texts and advertisements such as '成都一体机租赁_打印机租赁_成都复印机租赁_成都合纵租赁公司' (*Chengdu All-in-One Machine Leasing_Printer Leasing_Chengdu Copier Leasing_Chengdu Hezong Leasing Company*). Kreutzer *et al.* [4] make a more detailed assessment of web-crawled multilingual datasets. Moreover, mC4 utilises CLD3 [5] to determine the language of a given piece of text when aggregating the corpus, but CLD3 does not support low-resource languages like Cherokee, and gives strange misclassifications for high-resource languages such as detecting Chinese word 污水 *'u sywijX* 'sewage' as Haitian Creole.

(d) i. Cherokee does not distinguish between *he*, *she* and *it*, so it is reasonable for the NMT model to use 'it' instead of 'she' and 'he' instead of 'her'. However, there is an inconsistency that the model uses 'it' at first but then changed to 'she'. This problem can be resolved by improving the attention mechanism. Moreover, the size of the training data should be increased, or there should be a post-processing module to fix the inconsistency.

ii. In this example, the word 'little' is replicated for five times. This is a common problem of the decoder part of text generation models called text degeneration [6]. And according to Fu *et al.* [7], although are many conjectures for the reason of the repetition problem, the real cause is still unknown. Methods such as temperature sampling, top-k sampling and nucleus sampling have been proposed to solve the problem.

iii. In this example, the two words ᎤᏓᎸᏉᏗ ᏂᎨᏒᎾ *udalvquodi nigesvna* are expected to be translated to 'humble', but are translated to 'it's not a lot'. In Cherokee, ᎤᏓᎸᏉᏗ *udalvquodi* means 'arrogant', and ᏂᎨᏒᎾ *nigesvna* is a word that negates the original meaning. Since latter implies negation, it is somewhat reasonable for the model to translate the two words to a negative sentences, but the former does not make sense. A thorough inspection of the model reveals that this word, together with the quotation mark before it, is regarded as a single word, and split into four subwords ▁", Ꭴ, Ꮣ and ᎸᏉᏗ by the tokenizer. Therefore, the solution is to modify the tokenizer so that words can be separated from quotation marks. And we should further enlarge the training corpus to get more accurate results.

(e) i. **Source sentence:** ᎬᏩᎦᏗᏍ ᎤᏓᏙᎵᏍᏗ ᎨᏒ ᎠᎴ ᏅᏩᏙᎯᏍᏛ ᎢᏣᏤᎵ ᎨᏎᏍᏗ ᏅᏓᏳᏫᎵᏍ ᎠᏚᎳᏅᎯ ᎤᏁᎳᏅᎯ ᎢᎩᏙᏓ, ᎠᎴ ᎤᎬᏫᏳᎯ ᏥᏌ ᎦᎶᏁᏛ.
Reference translation: Grace to you and peace from God our Father and the Lord Jesus Christ.
NMT translation: Grace to you and peace from God our Father and the Lord Jesus Christ.

It turns out that several similar sentences have occurred in the training corpus. Those sentences are all translated to the same English sentences. This means that the model can perform better if the input sentence is similar to some sentences in the training set.

ii. **Source sentence:** “ᎰᏚᏭ,” ᎤᏛᏁ ᏌᎳᏓ, “ᎨᎵᎠ ᎠᏇᏅᏒ ᏱᎨᏙᎰ ᏯᏆᏁᏚᎳᏫ ᏗᎩᎷᏫᏍᏗᎳᏁ Ꮧ ᎦᏂᏟ.”
Reference translation: “No,” said Charlotte, “I believe I’d better stay home and see if I can’t get some work done.”
NMT translation: “No,” said Charlotte, “I think you have a very little.”

It turns out that the beginning of the sentence is similar to some sentences in the training corpus, but the end of the sentence is not. We may improve the decoder to get better performance, but in this case, the most direct resolution is to enlarge the training corpus.

(f) i. Manual calculation is trivial, so I implement it in Python.

```python
from collections import Counter
from itertools import islice, tee
from math import exp, log

def make_n_grams(words: str, n: int) -> Counter:
    return Counter(zip(*(islice(words_, i, None) for i, words_ in enumerate(tee(words, n)))))

def get_precision_n(refs: list[list[str]], cand: list[str], n: int):
    n_grams_refs = [make_n_grams(ref, n) for ref in refs]
    n_grams_cand = make_n_grams(cand, n)
    count_refs = sum(min(max(n_grams_ref.get(n_gram, 0) \
        for n_grams_ref in n_grams_refs), count_cand) \
        for n_gram, count_cand in n_grams_cand.items())
    count_cand = len(cand) - n + 1
    return count_refs / count_cand

def get_bp(refs: list[list[str]], cand: list[str]) -> float:
    len_refs = map(len, refs)
    len_cand = len(cand)
    min_len_ref = min(len_refs, \
        key=lambda len_ref: (abs(len_ref - len_cand), len_ref >= len_cand))
    print(f'len(c): {len_cand}')
    print(f'len(r): {min_len_ref}')
    return 1. if len_cand >= min_len_ref else exp(1. - min_len_ref / len_cand)

def get_weighted_precision(refs: list[list[str]], cand: list[str], n: int, lambda_: float):
    precision_n = get_precision_n(refs, cand, n)
    print(f'p_{n}: {precision_n:.4f}')
    weighted_precision = lambda_ * log(precision_n)
    return weighted_precision

def get_bleu(refs: list[list[str]], cand: list[str], lambdas: list[float]) -> float:
    bp = get_bp(refs, cand)
    print(f'BP: {bp}')
    bleu = bp * exp(sum(get_weighted_precision(refs, cand, i, lambda_) \
        for i, lambda_ in enumerate(lambdas, 1)))
    print(f'BLEU: {bleu:.4f}')
    return bleu
```

Computation:

```
>>> refs = [
...     'the light shines in the darkness and the darkness has not overcome it'.split(),
...     'and the light shines in the darkness and the darkness did not comprehend it'.split(),
... ]
>>> cand = 'and the light shines in the darkness and the darkness can not comprehend'.split()
>>> get_bleu(refs, cand, [0.5, 0.5]) and None
len(c): 13
len(r): 13
BP: 1.0
p_1: 0.9231
p_2: 0.8333
BLEU: 0.8771
>>> cand = 'the light shines the darkness has not in the darkness and the trials'.split()
>>> get_bleu(refs, cand, [0.5, 0.5]) and None
len(c): 13
len(r): 13
BP: 1.0
p_1: 0.8462
p_2: 0.7500
BLEU: 0.7966
```

According to the BLEU score, the first sentence is better. The first sentence is indeed better because the second sentence does not make sense.

ii. When considering $\mathbf{r}_1$ only, the BLEU scores of the first and the second sentences are 0.7161 and 0.7966 respectively. Now the second sentence gets a higher score, but according to our previous discussion, the first sentence should be better.

iii. As shown in the previous example, NMT systems evaluated with respect to only a single reference translation is problematic because the score may be biased. A better translation may receive a lower BLEU score. When the number of reference translations increases, the BLEU score is less likely to be biased.

iv. Compared to human evaluation, BLEU is a fully automated metric, so it is faster and it is easier to obtain the result. Besides, its calculation process is accurately described by algorithm, so it is reproducible and not influenced by subjective factors of human evaluators. However, BLEU scores may be biased if the number of reference translations is limited. Another disadvantage is that BLEU is a literal metric, which means that it does not take the syntax and semantics of the sentences into account.

In addition, several metrics have been proposed to improve BLEU. Hossain *et al.* [8] give a powerful demonstration that chrF++ and METEOR are better metrics than BLEU when negation is present. However, in order to compare with previous studies, it is still difficult to adopt metrics other than BLEU in new research.

References

[1] L. Xue *et al.*, *ByT5: Towards a token-free future with pre-trained byte-to-byte models*, 2021. arXiv: 2105.13626 [cs.CL].

[2] A. Fan *et al.*, "Beyond English-centric multilingual machine translation," *Journal of Machine Learning Research*, vol. 22, no. 107, pp. 1–48, 2021.

[3] L. Xue *et al.*, "mT5: A massively multilingual pre-trained text-to-text transformer," in *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2021, pp. 483–498.

[4] J. Kreutzer *et al.*, "Quality at a glance: An audit of web-crawled multilingual datasets," *Transactions of the Association for Computational Linguistics*, vol. 10, pp. 50–72, 2022.

[5] A. Salcianu *et al.*, *Compact language detector v3 (CLD3)*, 2021. [Online]. Available: https://github.com/google/cld3.

[6] A. Holtzman, J. Buys, L. Du, M. Forbes, and Y. Choi, "The curious case of neural text degeneration," in *International Conference on Learning Representations*, 2019.

[7] Z. Fu, W. Lam, A. M.-C. So, and B. Shi, "A theoretical analysis of the repetition problem in text generation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, 2021, pp. 12 848–12 856.

[8] M. M. Hossain, A. Anastasopoulos, E. Blanco, and A. Palmer, "It's not a non-issue: Negation as a source of error in machine translation," in *Findings of the Association for Computational Linguistics, ACL 2020: EMNLP 2020*, Association for Computational Linguistics (ACL), 2020, pp. 3869–3885.