How Reliable is MT Evaluation?

He HUANG 22107447

October 29, 2023

1. Implement the Bleu metric.

```python
from math import exp, log
import numpy as np
from collections import Counter

def ngrams(sequence, n):
  """Generate n-grams from a sequence."""

  sequence = sequence.split()
  if len(sequence) < n:
    return []

  sequence = iter(sequence)
  history = []

  while n > 1:
    history.append(next(sequence))
    n -=1
  for item in sequence:
    history.append(item)
    yield tuple(history)
    del history[0]

def brevity_penalty(ref, hyp):

  """Brevity penalty"""
  h = len(hyp.split())
  r = len(ref.split())

  if h > r:
    return 1.
  else:
    return exp(1 - (h / r))

def bleu_score(ref, hyp, n):
  """Implement BLEU score"""

  m_p = []
  weight = 1.0/n

  for i in range(n):
    # generate i-grams
    ref_ngrams = Counter(ngrams(ref, i+1))
    hyp_ngrams = Counter(ngrams(hyp, i+1))

```

```
    # modified precision
    common_ngrams = sum((min(hyp_ngrams[ngram], ref_ngrams.get(ngram, 0))
                         for ngram in hyp_ngrams))
    total = sum(hyp_ngrams.values())

    if total == 0:
      continue
    else:
      m_p.append(common_ngrams / total)

    bp = brevity_penalty(ref, hyp)
    # add epsilon to avoid log(0)
    log_p = sum(weight * log((p_n) + 1e-10) for p_n in m_p)
    bleu_compute = bp*np.exp(log_p)

  return bleu_compute

def corpus_bleu(refs, hyps, n):
  """Return the BLEU score in each translation and reference sample"""

  bleu = 0
  for _, (ref,hyp) in enumerate(zip(refs, hyps)):
    bleu += bleu_score(ref, hyp, n)
  return bleu / len(refs)
```

2. Using the WMT'15 test sets[1], evaluate the performance of mBart and MarianMT. These two models can be easily used with the HuggingFace API[2]

```
import re
import html
import torch

en_fr_src_path = "test/newsdiscusstest2015-enfr-src.en.sgm"
en_fr_ref_path = "test/newsdiscusstest2015-enfr-ref.fr.sgm"

def sgm_to_text(filename):

  """Read a SGM file and return the plain text."""

  plain_text = []

  with open(filename, "r") as sgm_file:
    for line in sgm_file.readlines():
      # convert html tags
      line = html.unescape(line)
      # for each line which starts with <seg>
      if re.search(r"<seg id=.+>.+</seg>", line):
        pattern = re.compile(r"<seg id=\"\d+\">|<\/seg>")
        line = re.sub(pattern, "", line)
        line = line.replace("\n", "")
        plain_text.append(line)
  return plain_text

src_en = sgm_to_text(en_fr_src_path)
ref_fr = sgm_to_text(en_fr_ref_path)
```

[1]https://statmt.org/wmt15/translation-task.html

[2]https://huggingface.co/docs/transformers/model_doc/mbart and https://huggingface.co/docs/transformers/model_doc/marian

```
28
29 !pip install transformers
30 !pip install sentencepiece
31 !pip install sacremoses
32
33 device = torch.device("cuda") if torch.cuda.is_available() else torch.device("
       cpu")
34
35 # mBART
36 # define the mBart model and tokenizer
37 from transformers import MBartForConditionalGeneration, MBart50TokenizerFast
38
39 mbart_name = "facebook/mbart-large-50-many-to-many-mmt"
40 mbart_model = MBartForConditionalGeneration.from_pretrained(mbart_name).to(
       device)
41 mbart_tokenizer = MBart50TokenizerFast.from_pretrained(mbart_name, src_lang="
       en_XX")
42
43 # MarianMT
44 # define the MarianMT model and tokenizer
45 from transformers import MarianMTModel, MarianTokenizer
46 mt_name = "Helsinki-NLP/opus-mt-en-fr"
47 mt_model = MarianMTModel.from_pretrained(mt_name).to(device)
48 mt_tokenizer = MarianTokenizer.from_pretrained(mt_name)
49
50 def translation(model, model_name, data, tokenizer):
51
52   """Translate the English sentences into French"""
53
54   translated = []
55
56   for sent in data:
57     inputs = tokenizer(sent, return_tensors="pt", truncation=True, max_length
       =100, padding=True).to(device)
58
59     if model_name == "mbart":
60       generated_tokens = model.generate(**inputs, forced_bos_token_id=tokenizer.
       lang_code_to_id["fr_XX"])
61       decode_fr = tokenizer.batch_decode(generated_tokens, skip_special_tokens=
       True)
62     else:
63       generated_tokens = model.generate(**inputs)
64       decode_fr = [tokenizer.decode(g, skip_special_tokens=True) for g in
       generated_tokens]
65
66     translated.append(decode_fr)
67   return translated
68
69 # Results : a list of sentences
70 mbart_en_fr = translation(mbart_model, "mbart", src_en, mbart_tokenizer)
71 mbart_en_fr = [''.join(sent) for sent in mbart_en_fr]
72
73 mt_en_fr = translation(mt_model, "mt", src_en, mt_tokenizer)
74 mt_en_fr = [''.join(sent) for sent in mt_en_fr]
75
76 # Compute BLEU score
77 mbart_en_fr_bleu = corpus_bleu(ref_fr, mbart_en_fr, 4)
78 mt_en_fr_bleu = corpus_bleu(ref_fr, mt_en_fr, 4)
79
80 print(f"The BLEU score of mBART is {mbart_en_fr_bleu}")
```

```
print(f"The BLEU score of MT is {mt_en_fr_bleu}")
```

	mBART	MarianMT
en->fr	0.2269	0.2918

MarianMT obtains a BLEU score of 0.2918 and mBART of 0.2269. MarianMT model is better at translating English sentences into French than mBART.

3. Explain on an example why such permutations will never decrease the Bleu score.
BLEU simply counts matching n-grams, it doesn't consider their order in the sentence. Let's take a reference sentence "She loves reading books." And the translation sentence could be "Reading she books loves" or "She books reading loves." All these sentences might have similar uigram (i.e."She","loves","reading","books"), since BlEU doesn't penalize word order, these two translations might have similar BLEU scores.

4. Given a sentence with n words and b bigram mismatches, how many sentences can you generate with this principle. Compute the number of sentences you will obtain on the WMT'15 test set.
There are (n − b)! sentences generated by this principle.

```
# compute permutation
import math

def permutation(references, hypothesis):
    total_nb = 0

    for i in range(len(references)):
        ref = references[i]
        ref_grams = ngrams(ref,2)

        hyp = hypothesis[i]
        hyp_len = len(hyp.split())
        hyp_grams = ngrams(hyp,2)

        common = set(hyp_grams).intersection(set(ref_grams))
        common_len = len(common)
        total_nb += math.factorial(hyp_len-common_len)
    return format(total_nb,'e')
```

	mBART	MarianMT
permutation	1.447390e+75	2.350574e+78

On the sentences from WMT'15 test set, there are 1.447390e+75 permutations for mBART and 2.350574e+78 for MarianMT.

5. Why this result question the use of Bleu as an evaluation metric.
In many languages, the word order and semantic meaning of a sentence is crucial for the evaluation of the quality of translation. A higher BLEU score does not necessarily mean a better translation. Because there are also scores for permutated gibberish.

6. sacreBleu[3] is an implementation of Bleu that aims to provide "hassle-free computation of shareable, comparable, and reproducible Bleu scores". Evaluate the two previous systems using sacreBleu. What can you conclude ?

```
!pip install sacrebleu

import sacrebleu

# en->fr
# mBART
sacrebleu_mbart_en_fr = sacrebleu.corpus_bleu(mbart_en_fr,[ref_fr])
# MarianMT
sacrebleu_mt_en_fr = sacrebleu.corpus_bleu(mt_en_fr,[ref_fr])
print (f"mBART sacrebleu : {sacrebleu_mbart_en_fr}")
print("\n")
print(f"MarianMT sacrebleu : {sacrebleu_mt_en_fr}")
```

```
Output:

mBART sacrebleu : BLEU = 32.59 60.8/38.5/26.3/18.3 (BP = 1.000 ratio = 1.016
    hyp_len = 28435 ref_len = 27975)


MarianMT sacrebleu : BLEU = 38.38 65.6/44.3/31.9/23.6 (BP = 0.998 ratio = 0.998
    hyp_len = 27929 ref_len = 27975)
```

This score has been multiplied by 100, so if I follow my own implementation, the score is actually 0.33 for mBART and 0.38 for MarianMT. Compared to my score (0.2269 for mBART and 0.2918 for MT), the sacreBleu score is a bit higher with the same translations. In my implementation BLEU, i separated the sentences on space, sacreBleu takes another way to tokenization. This may affect the calculation by these two metrics.

7. Using sacreBleu and your own implementation of Bleu compute the score achieved :
— when considering the "raw" translation hypotheses and references ;
— when the translation hypotheses and references have been tokenized in subword units [4] ;
— when the translation hypotheses and references have been tokenized in characters (this amounts to adding a space between each character of the references and of the translation hypotheses).

How can you explain these results ?

```
# Raw text
print(f"mBART BLEU score : {mbart_en_fr_bleu}")
print(f"MarianMT BLEU score : {mt_en_fr_bleu}")
print("\n")
print(f"mBART sacrebleu : {sacrebleu_mbart_en_fr}")
print(f"MarianMT sacrebleu : {sacrebleu_mt_en_fr}")

Output:
```

[3] https ://github.com/mjpost/sacrebleu

[4] You can use either one of the tokenizer provided by HuggingFace or train your own model, for instance on `Europarl` data.

```
mBART BLEU score : 0.22685186354146247
MarianMT BLEU score : 0.2918108984560062

mBART sacrebleu : BLEU = 32.59 60.8/38.5/26.3/18.3 (BP = 1.000 ratio = 1.016
    hyp_len = 28435 ref_len = 27975)
MarianMT sacrebleu : BLEU = 38.38 65.6/44.3/31.9/23.6 (BP = 0.998 ratio = 0.998
    hyp_len = 27929 ref_len = 27975)
```

```python
# subword units
from transformers import AutoTokenizer

def subword_tokenization(sentences):
    tokenizer =  AutoTokenizer.from_pretrained("bert-base-multilingual-cased")
    tokens = []

    for sent in sentences:
        tokenization = tokenizer.tokenize(sent)
        subword_sent = ' '.join(tokenization)
        tokens.append(subword_sent)
    return tokens
```

```python
# character level
def char_tokenization(sentences):
      return [' '.join(list(sent)) for sent in sentences]
```

For mBART:

model	score	raw	subword	character
mBART	BLEU	22.69	36.82	64.46
	sacreBleu	32.59	45.23	64.06

For MarianMT:

model	score	raw	subword	character
MarianMT	BLEU	29.18	44.38	70.48
	sacreBleu	38.38	50.82	68.85

It is obvious that when the tokenisation unit becomes smaller, the scores of both BLEU and sacreBleu will increase. When we split sentences into smaller units (words to subwords and characters), they will have more opportunities to match more n-grams. In this way, the score will become higher.