Multilingual NLP — Lab 3
How Reliable is MT Evaluation?

Lina Conti

January 2023

1 Re-evaluating the role of Bleu in machine translation research

```python
from math import exp, log
from collections import Counter
from nltk.tokenize import word_tokenize
import nltk
nltk.download('punkt')

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


def ngrams(sequence, n):
    if len(sequence) < n: # no ngrams
      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 ngram_matches(ref, hyp, n):
    bag_ref = Counter(ngrams(ref, n))
    bag_hyp = Counter(ngrams(hyp, n))

    common = sum((min(bag_hyp[ngram], bag_ref[ngram]) for ngram in bag_hyp))
    total = sum(bag_hyp.values())
    return common, total


def my_BLEU(references, hypotheses, max_n):
    r = 0 # number of tokens in the reference corpus
    h = 0 # number of tokens in the hypotheses corpus
    matched_ngrams = [0] * max_n  # number of matched ngrams for each n
    total_ngrams = [0] * max_n  # number of ngrams in hypotheses corpus for each n

    for ref, hyp in zip(references, hypotheses):
        hyp = word_tokenize(hyp)
        ref = word_tokenize(ref)

        r += len(ref)
        h += len(hyp)

        for n in range(max_n):
            common, total = ngram_matches(ref, hyp, n+1)
            matched_ngrams[n] += common
            total_ngrams[n] += total

    precisions = []
    for n in range(max_n):
        precisions.append(matched_ngrams[n]/total_ngrams[n])
    log_precisions = [log(pn) for pn in precisions]

    avg_precision = sum(log_precisions)/len(log_precisions)
    bp = brevity_penalty(r, h)
    bleu = bp * exp(avg_precision)
    return {'bleu': round(bleu,4),
            'precisions': [round(pn, 3) for pn in precisions],
            'bp': round(bp,3),
            'ratio': round(r/h,3),
            'hyp_len': h,
            'ref_len': r}
```

2.

```python
import re

def sgm_to_txt(sgm_file, txt_file):
    with open(sgm_file, 'r') as f_in:
        with open(txt_file, 'w') as f_out:
            for line in f_in:
                if line.startswith("<seg id"):
                    f_out.write(re.sub('(<seg id="\d*">)|(<\/seg>)', "", line))

sgm_to_txt("/content/test/newsdiscusstest2015-enfr-src.en.sgm", "newsdiscusstest2015-enfr-src.en")
sgm_to_txt("/content/test/newsdiscusstest2015-enfr-ref.fr.sgm", "newsdiscusstest2015-enfr-ref.fr")
```

```python
from tqdm.notebook import tqdm

def generate_batches(src_sentences, tokenizer, batch_size, device):
    bstart = 0
    N = len(src_sentences)
    while bstart < N:
        bend = min(bstart+batch_size, N)
        b_src_sentences = src_sentences[bstart:bend]
        b_encoded_src = tokenizer(b_src_sentences, return_tensors="pt", padding=True, truncation=True, max_length=512)
        yield b_encoded_src.to(device)
        bstart += batch_size


def translate(src_file, out_file, tokenizer, model, batch_size, device):
    with open(src_file, 'r') as f:
        src_sentences = f.readlines()

    nb_batches = (len(src_sentences)/batch_size)
    for b_encoded_src in tqdm(generate_batches(src_sentences, tokenizer, batch_size, device), total=nb_batches):

        print("torch.cuda.memory_reserved: %fGB"%(torch.cuda.memory_reserved(0)/1024/1024/1024))

        try:  # MBart
            generated_tokens = model.generate(**b_encoded_src,  max_new_tokens=512, forced_bos_token_id=tokenizer.lang_code_to_id["fr_XX"])
        except AttributeError:  # MarianMT
            generated_tokens = model.generate(**b_encoded_src,  max_new_tokens=512)

        b_decoded_trg = tokenizer.batch_decode(generated_tokens, skip_special_tokens=True)

        with open(out_file, 'a') as f:
            for line in b_decoded_trg:
                f.write(f"{line}\n")
```

```python
from transformers import MarianTokenizer, MarianMTModel

model_name = "Helsinki-NLP/opus-mt-en-fr"
marian_model = MarianMTModel.from_pretrained(model_name).to(DEVICE)
marian_tokenizer = MarianTokenizer.from_pretrained(model_name)

translate('newsdiscusstest2015-enfr-src.en',
          'marian-newsdiscusstest2015-enfr-pred.fr',
          marian_tokenizer,
          marian_model,
          batch_size=128,
          device=DEVICE)
```

```python
from transformers import MBartForConditionalGeneration, MBart50TokenizerFast

mbart_model = MBartForConditionalGeneration.from_pretrained("facebook/mbart-large
    -50-many-to-many-mmt").to(DEVICE)
mbart_tokenizer = MBart50TokenizerFast.from_pretrained("facebook/mbart-large-50-
    many-to-many-mmt")
mbart_tokenizer.src_lang = "en_XX"

translate('newsdiscusstest2015-enfr-src.en',
          'mbart-newsdiscusstest2015-enfr-pred.fr',
          mbart_tokenizer,
          mbart_model,
          batch_size=16,
          device=DEVICE)
```

```python
with open('newsdiscusstest2015-enfr-ref.fr', 'r') as f:
    references = f.readlines()

with open('marian-newsdiscusstest2015-enfr-pred.fr', 'r') as f:
    marian_hypotheses = f.readlines()

with open('mbart-newsdiscusstest2015-enfr-pred.fr', 'r') as f:
    mbart_hypotheses = f.readlines()

print('MarianMT Bleu:')
print(my_BLEU(references, marian_hypotheses, 4))

print('\nMBart Bleu:')
print(my_BLEU(references, mbart_hypotheses, 4))
```

With my implementation, MarianMT obtains a Bleu score of 0.3747 and MBart of 0.3231. It looks like MarianMT's model is better at translating from English to French then MBart's.

3. Let's take the sentence 'I think it 's called democracy .' The reference translation is 'Je crois que c'est ce qu'on appelle la démocratie .' and MBart's hypothesis is 'Je pense qu'il s'agit de la démocratie .'

   Bigram mismatches are indicated with pipe symbols: 'Je | pense | qu'il | s'agit | de | la démocratie .' An example of permutation that does not lower the BLEU score is 'Je | la démocratie . | qu'il | s'agit | pense | de'.

   This happens because the number of matching unigrams does not change with permutation, the number of matching ngrams obviously does not change when permutations happen on bigram mismatches and neither does the number of higher order ngrams, since the latter are composed of matching bigrams. For instance, in the example above, the trigram 'la démocratie .' will never be separated by such permutations. The length of the hypotheses will not change either, so all the building blocks of the BLEU score stay the same with these permutations.

4. Given a sentence with n words and b bigram mismatches, there are $b' = n - b + 1$ bigram matches, so $(n - b')! = (b + 1)!$ sentences can be generated with this principle (cf. Callison-Burch et al. 2006).

```python
from math import factorial

def possible_permutations(references, hypotheses):

    permutations = 0
    for ref, hyp in zip(references, hypotheses):
        hyp = word_tokenize(hyp)
        ref = word_tokenize(ref)

        n = len(hyp)
        b, _ = ngram_matches(ref, hyp, 2)
        permutations += factorial(n-b)

    return f"{permutations:.2e}"

possible_permutations(references, mbart_hypotheses)
```

On the full WMT'15 test set, there are 9.90e+163 permutations for MarianMT's translations and 3.01e+213 for MBart's.

5. This questions the use of BLEU as an evaluation metric because it is unlikely that that many variations of each translation hypothesis are all really of equal quality. This would mean that BLEU cannot correlate well with human judgement for all hypotheses and that a higher BLEU score does not necessarily mean better translations (since a high score can also be achieve by permutated gibberish).

6.
```python
from sacrebleu.metrics import BLEU
bleu = BLEU()

print('MarianMT sacrebleu:')
print(bleu.corpus_score(marian_hypotheses, [references]))

print('\nMBart sacrebleu:')
print(bleu.corpus_score(mbart_hypotheses, [references]))
```

With sacrebleu, MarianMT gets a score of 38.36 and MBart of 32.58. With my implementation the scores were respectively 37.47 and 32.31. The scores are not too different, however, they are not the same. For MarianMT especially, the difference is of almost one BLEU point. This means that one should be careful when comparing BLEU scores for different models that were computed by different teams for different papers, for example. If the implementation of BLEU is not the same the results may not be comparable.

7. Raw text:

```python
print('MarianMT Bleu:')
print(my_BLEU(references, marian_hypotheses, 4))

print('\nMBart Bleu:')
print(my_BLEU(references, mbart_hypotheses, 4))

print('\nMarianMT sacrebleu:')
print(bleu.corpus_score(marian_hypotheses, [references]))

print('\nMBart sacrebleu:')
print(bleu.corpus_score(mbart_hypotheses, [references]))
```

Output:

```
MarianMT Bleu:
{'bleu': 0.3747, 'precisions': [0.655, 0.44, 0.316, 0.234], 'bp': 0.981, 'ratio':
    1.019, 'hyp_len': 28047, 'ref_len': 28589}

MBart Bleu:
{'bleu': 0.3231, 'precisions': [0.606, 0.381, 0.261, 0.182], 'bp': 0.999, 'ratio':
    1.001, 'hyp_len': 28550, 'ref_len': 28589}

MarianMT sacrebleu:
BLEU = 38.36 65.6/44.2/31.8/23.5 (BP = 1.000 ratio = 1.005 hyp_len = 28115 ref_len
    = 27975)

MBart sacrebleu:
BLEU = 32.58 60.7/38.4/26.3/18.4 (BP = 1.000 ratio = 1.021 hyp_len = 28575 ref_len
    = 27975)
```

Subword tokenization:

```python
def subword_tokenization(tokenizer, in_file, out_file):
    with open(in_file, 'r') as f_in:
        with open(out_file, 'w') as f_out:
            for line in f_in:
                f_out.write(' '.join(tokenizer.tokenize(line.strip())) + '\n')

subword_tokenization(mbart_tokenizer, 'marian-newsdiscusstest2015-enfr-pred.fr', '
    subtok-marian-newsdiscusstest2015-enfr-pred.fr')
subword_tokenization(mbart_tokenizer, 'mbart-newsdiscusstest2015-enfr-pred.fr', '
    subtok-mbart-newsdiscusstest2015-enfr-pred.fr')

with open('subtok-marian-newsdiscusstest2015-enfr-pred.fr', 'r') as f:
    marian_hypotheses = f.readlines()

with open('subtok-mbart-newsdiscusstest2015-enfr-pred.fr', 'r') as f:
    mbart_hypotheses = f.readlines()

print('MarianMT subword Bleu:')
print(my_BLEU(references, marian_hypotheses, 4))

print('\nMBart subword Bleu:')
print(my_BLEU(references, mbart_hypotheses, 4))

print('\nMarianMT subword sacrebleu:')
print(bleu.corpus_score(marian_hypotheses, [references]))

print('\nMBart subword sacrebleu:')
print(bleu.corpus_score(mbart_hypotheses, [references]))
```

Output:

```
MarianMT subword Bleu:
{'bleu': 0.0029, 'precisions': [0.089, 0.006, 0.001, 0.0], 'bp': 1.0, 'ratio':
    0.727, 'hyp_len': 39315, 'ref_len': 28589}

MBart subword Bleu:
{'bleu': 0.003, 'precisions': [0.087, 0.006, 0.001, 0.0], 'bp': 1.0, 'ratio':
    0.712, 'hyp_len': 40143, 'ref_len': 28589}

MarianMT subword sacrebleu:
BLEU = 0.73 8.6/0.8/0.3/0.1 (BP = 1.000 ratio = 1.410 hyp_len = 39447 ref_len =
    27975)

MBart subword sacrebleu:
BLEU = 0.74 8.3/0.8/0.3/0.1 (BP = 1.000 ratio = 1.437 hyp_len = 40212 ref_len =
    27975)
```

Character tokenization:

```python
def char_tokenization(in_file, out_file):
    with open(in_file, 'r') as f_in:
        with open(out_file, 'w') as f_out:
            for line in f_in:
                f_out.write(' '.join(line))

char_tokenization('marian-newsdiscusstest2015-enfr-pred.fr', 'chartok-marian-
    newsdiscusstest2015-enfr-pred.fr')
char_tokenization('mbart-newsdiscusstest2015-enfr-pred.fr', 'chartok-mbart-
    newsdiscusstest2015-enfr-pred.fr')

with open('chartok-marian-newsdiscusstest2015-enfr-pred.fr', 'r') as f:
    marian_hypotheses = f.readlines()

with open('chartok-mbart-newsdiscusstest2015-enfr-pred.fr', 'r') as f:
    mbart_hypotheses = f.readlines()

print('MarianMT character Bleu:')
print(my_BLEU(references, marian_hypotheses, 4))

print('\nMBart character Bleu:')
print(my_BLEU(references, mbart_hypotheses, 4))

print('\nMarianMT character sacrebleu:')
print(bleu.corpus_score(marian_hypotheses, [references]))

print('\nMBart character sacrebleu:')
print(bleu.corpus_score(mbart_hypotheses, [references]))
```

Output:

```
MarianMT character Bleu:
{'bleu': 0.0016, 'precisions': [0.03, 0.002, 0.001, 0.0], 'bp': 1.0, 'ratio':
    0.228, 'hyp_len': 125250, 'ref_len': 28589}

MBart character Bleu:
{'bleu': 0.0016, 'precisions': [0.03, 0.002, 0.001, 0.0], 'bp': 1.0, 'ratio':
    0.223, 'hyp_len': 128008, 'ref_len': 28589}

MarianMT character sacrebleu:
BLEU = 0.24 2.8/0.3/0.1/0.0 (BP = 1.000 ratio = 4.477 hyp_len = 125250 ref_len =
    27975)

MBart character sacrebleu:
BLEU = 0.24 2.8/0.3/0.1/0.0 (BP = 1.000 ratio = 4.576 hyp_len = 128008 ref_len =
    27975)
```

The scores for tokenized text are oddly low (lower than 1.) I find this hard to explain, but given that it is the same both with sacrebleu and my implementation, I don't think it is a problem with how I am using either of those. The tokenized text files also look normal. Maybe it is due to the fact that there are a lot more tokens this way, so mistakes in the translations are amplified, but I am not sure.

Anyway, the moral is that BLEU scores change according to the implementation of BLEU **and** according to the tokenization of the texts, so one should be wary of any difference in those regards when comparing scores.