# BLEU Scores

This notebook demonstrates the code for BLEU scores followed by evaluation of a model on test data during training. For theory, refer to the Github README.

Our result will match that as shown on [Google's Page]() !!!

```
import numpy as np
from collections import Counter
import nltk
nltk.download("punkt")
from nltk.util import ngrams
np.seterr(divide = 'ignore') # to ignore errors if an n-gram sequence is missing
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
{'divide': 'ignore', 'invalid': 'warn', 'over': 'warn', 'under': 'ignore'}
```

```
# target means a good high-quality human-translation (in our case the original eng
# prediciton is what is generated from our model
def brevity_penalty(target, prediction):
    targ_length = len(target)
    pred_length = len(prediction)

    # Brevity Penalty
    if  pred_length > targ_length:
        BP = 1
    else:
        penalty = 1 - (targ_length / pred_length)
        BP = np.exp(penalty)

    return BP

def clipped_precision(target, prediction):
    """
    Clipped Precision function given a original and a machine translated sentences
    """

    clipped_precision_score = []

    for i in range(1, 5):
        prediction_n_gram = Counter(
            ngrams(prediction, i)
        )  # counts of n-gram n=1...4 tokens for the candidate
        target_n_gram = Counter(
            ngrams(target, i)
        )  # counts of n-gram n=1...4 tokens for the reference

        c = sum(
```

```
            prediction_n_gram.values()
        )  # sum of the values of the reference the denominator in the precision f

        for j in prediction_n_gram:  # for every n_gram token in the reference
            if j in target_n_gram:  # check if it is in the candidate n-gram

                if (
                    prediction_n_gram[j] > target_n_gram[j]
                ):  # if the count of the reference n-gram is bigger
                    # than the corresponding count in the candidate n-gram
                    prediction_n_gram[j] = target_n_gram[j]  # then set the count
                    # to the count of the candidate n-gram
            else:

                prediction_n_gram[j] = 0  # else reference n-gram = 0

        clipped_precision_score.append(sum(prediction_n_gram.values()) / c)

    weights = [0.25] * 4
    cl = np.array(clipped_precision_score)
    w = np.array(weights)

    s1 = w * np.log(cl)

    s = np.exp(np.sum(s1))
    return s

def bleu_score(target, prediction):
    BP = brevity_penalty(target, prediction)
    precision = clipped_precision(target, prediction)
    return BP * precision



reference = "The NASA Opportunity rover is battling a massive dust storm on Mars."
candidate_1 = "The Opportunity rover is combating a big sandstorm on Mars."
candidate_2 = "A NASA rover is fighting a massive storm on Mars."

tokenized_ref = nltk.word_tokenize(reference.lower())
tokenized_cand_1 = nltk.word_tokenize(candidate_1.lower())
tokenized_cand_2 = nltk.word_tokenize(candidate_2.lower())


print(
    "Results reference versus candidate 1 our own code BLEU: ",
    round(bleu_score(tokenized_ref, tokenized_cand_1) * 100, 1),
)
print(
    "Results reference versus candidate 2 our own code BLEU: ",
    round(bleu_score(tokenized_ref, tokenized_cand_2) * 100, 1),
)
```

```
Results reference versus candidate 1 our own code BLEU:  0.0
Results reference versus candidate 2 our own code BLEU:  27.2
```

As we can see, our results match the scores mentioned on [Google's Page]() (Screenshot below)

**Calculating the BLEU score**

**Reference:** `The NASA Opportunity rover is battling a massive dust storm on Mars .`
**Candidate 1:** `The Opportunity rover is combating a big sandstorm on Mars .`
**Candidate 2:** `A NASA rover is fighting a massive storm on Mars .`

The above example consists of a single reference and two candidate translations. The sentences are tokenized prior to computing the BLEU score as depicted above; for example, the final period is counted as a separate token.

To compute the BLEU score for each translation, we compute the following statistics.

- **N-Gram Precisions**
  The following table contains the n-gram precisions for both candidates.
- **Brevity-Penalty**
  The brevity-penalty is the same for candidate 1 and candidate 2 since both sentences consist of 11 tokens.
- **BLEU-Score**
  Note that at least one matching 4-gram is required to get a BLEU score > 0. Since candidate translation 1 has no matching 4-gram, it has a BLEU score of 0.

| Metric | Candidate 1 | Candidate 2 |
|---|---|---|
| $precision_1$ (1gram) | 8/11 | 9/11 |
| $precision_2$ (2gram) | 4/10 | 5/10 |
| $precision_3$ (3gram) | 2/9 | 2/9 |
| $precision_4$ (4gram) | 0/8 | 1/8 |
| Brevity-Penalty | 0.83 | 0.83 |
| **BLEU-Score** | **0.0** | **0.27** |

# BLEU Score on Week 8's Assignment

## Installing Modules

```
!pip install pytorch_lightning torchmetrics tableprint spacy==3
!python -m spacy download en_core_web_sm
!python -m spacy download de_core_news_sm
```

```
Requirement already satisfied: pydantic<1.8.0,>=1.7.1 in /usr/local/lib/pyt
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.7/
Requirement already satisfied: tqdm<5.0.0,>=4.38.0 in /usr/local/lib/python
Requirement already satisfied: wasabi<1.1.0,>=0.8.1 in /usr/local/lib/pytho
Requirement already satisfied: spacy-legacy<3.1.0,>=3.0.0 in /usr/local/lib
Requirement already satisfied: srsly<3.0.0,>=2.4.0 in /usr/local/lib/python
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dis
Requirement already satisfied: zipp>=0.5; python_version < "3.8" in /usr/lo
```

```
Requirement already satisfied: smart-open<6.0.0,>=5.0.0 in /usr/local/lib/py
Requirement already satisfied: click<7.2.0,>=7.1.1 in /usr/local/lib/python3
Requirement already satisfied: MarkupSafe>=0.23 in /usr/local/lib/python3.7/
Requirement already satisfied: pyparsing>=2.0.2 in /usr/local/lib/python3.7/
Installing collected packages: en-core-web-sm
  Found existing installation: en-core-web-sm 2.2.5
    Uninstalling en-core-web-sm-2.2.5:
      Successfully uninstalled en-core-web-sm-2.2.5
Successfully installed en-core-web-sm-3.0.0
✔ Download and installation successful
You can now load the package via spacy.load('en_core_web_sm')
2021-07-04 04:12:08.440927: I tensorflow/stream_executor/platform/default/d
Collecting de-core-news-sm==3.0.0
  Downloading https://github.com/explosion/spacy-models/releases/download/d
     |████████████████████████████████| 19.3MB 154kB/s
Requirement already satisfied: spacy<3.1.0,>=3.0.0 in /usr/local/lib/python3
Requirement already satisfied: typer<0.4.0,>=0.3.0 in /usr/local/lib/python3
Requirement already satisfied: typing-extensions>=3.7.4; python_version < "3
Requirement already satisfied: preshed<3.1.0,>=3.0.2 in /usr/local/lib/pytho
Requirement already satisfied: importlib-metadata>=0.20; python_version < "3
Requirement already satisfied: jinja2 in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: requests<3.0.0,>=2.13.0 in /usr/local/lib/py
Requirement already satisfied: setuptools in /usr/local/lib/python3.7/dist-p
Requirement already satisfied: blis<0.8.0,>=0.4.0 in /usr/local/lib/python3
Requirement already satisfied: cymem<2.1.0,>=2.0.2 in /usr/local/lib/python3
Requirement already satisfied: wasabi<1.1.0,>=0.8.1 in /usr/local/lib/pytho
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.7/d
Requirement already satisfied: catalogue<2.1.0,>=2.0.1 in /usr/local/lib/py
Requirement already satisfied: pathy in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: murmurhash<1.1.0,>=0.28.0 in /usr/local/lib/p
Requirement already satisfied: tqdm<5.0.0,>=4.38.0 in /usr/local/lib/python3
Requirement already satisfied: pydantic<1.8.0,>=1.7.1 in /usr/local/lib/pyth
Requirement already satisfied: thinc<8.1.0,>=8.0.0 in /usr/local/lib/python3
Requirement already satisfied: spacy-legacy<3.1.0,>=3.0.0 in /usr/local/lib/
Requirement already satisfied: srsly<3.0.0,>=2.4.0 in /usr/local/lib/python3
Requirement already satisfied: numpy>=1.15.0 in /usr/local/lib/python3.7/di
Requirement already satisfied: click<7.2.0,>=7.1.1 in /usr/local/lib/python3
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.7/dist-pa

Requirement already satisfied: MarkupSafe>=0.23 in /usr/local/lib/python3.7/
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /u
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dis
Requirement already satisfied: pyparsing>=2.0.2 in /usr/local/lib/python3.7/
Requirement already satisfied: smart-open<6.0.0,>=5.0.0 in /usr/local/lib/py
Installing collected packages: de-core-news-sm
Successfully installed de-core-news-sm-3.0.0
✔ Download and installation successful
```

## Imports

```
# Import Libraries
import random
from typing import Iterable, List, Tuple
import pandas as pd
import sys, os, pickle
import math
```

```
import matplotlib.pyplot  as plt
import spacy

# PyTorch related
import torch, torchtext
import torch.nn as nn
import torch.nn.functional as F
import torch.optim as optim
from torch import Tensor
from torchtext.data.utils import get_tokenizer
from torchtext.vocab import build_vocab_from_iterator
from torchtext.datasets import Multi30k
from torch.nn.utils.rnn import pad_sequence
from torch.utils.data import DataLoader

# My Custom Code
import pytorch_lightning as pl
import torchmetrics
from pytorch_lightning.loggers import CSVLogger
from pytorch_lightning.callbacks import ModelCheckpoint
import tableprint as tp
from torchtext.data.metrics import bleu_score
```

```
# Manual Seed
SEED = 1234

random.seed(SEED)
np.random.seed(SEED)
torch.manual_seed(SEED)
torch.cuda.manual_seed(SEED)
torch.backends.cudnn.deterministic = True
```

```
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
```

## Language Definitions

```
SRC_LANGUAGE = 'de'
TGT_LANGUAGE = 'en'

# Place-holders
token_transform = {}
vocab_transform = {}
```

## Tokenizers

```
token_transform[SRC_LANGUAGE]  = get_tokenizer('spacy', language='de_core_news_sm'
token_transform[TGT_LANGUAGE] = get_tokenizer('spacy', language='en_core_web_sm')
```

```
token_transform[TGT_LANGUAGE] = get_tokenizer('spacy', language='en_core_web_sm')
```

## Yield Function

This yields the tokens for the texts and will be used to build the vocab

```
def yield_tokens(data_iter: Iterable, language: str) -> List[str]:
    language_index = {SRC_LANGUAGE: 0, TGT_LANGUAGE: 1}

    for data_sample in data_iter:
        yield token_transform[language](data_sample[language_index[language]])
```

## Special Tokens

```
# Define special symbols and indices
UNK_IDX, PAD_IDX, BOS_IDX, EOS_IDX = 0, 1, 2, 3
# Make sure the tokens are in order of their indices to properly insert them in vo
special_symbols = ['<unk>', '<pad>', '<bos>', '<eos>']
```

Build the vocab here

```
for ln in [SRC_LANGUAGE, TGT_LANGUAGE]:
  # Training data Iterator
  train_iter = Multi30k(split='train', language_pair=(SRC_LANGUAGE, TGT_LANGUAGE))
  # Create torchtext's Vocab object
  vocab_transform[ln] = build_vocab_from_iterator(yield_tokens(train_iter, ln),
                                                  min_freq=1,
                                                  specials=special_symbols,
                                                  special_first=True)
```

```
training.tar.gz: 100%|██████████| 1.21M/1.21M [00:00<00:00, 1.63MB/s]
```

## Setting the default index as the token

```
# Set UNK_IDX as the default index. This index is returned when the token is not f
# If not set, it throws RuntimeError when the queried token is not found in the Vo
for ln in [SRC_LANGUAGE, TGT_LANGUAGE]:
  vocab_transform[ln].set_default_index(UNK_IDX)
```

```
len(vocab_transform['de'])
```

```
19215
```

```
len(vocab_transform['en'])
```

```
10838
```

## Collator

```
# helper function to club together sequential operations
def sequential_transforms(*transforms):
    def func(txt_input):
        for transform in transforms:
            txt_input = transform(txt_input)
        return txt_input
    return func

# function to add BOS/EOS and create tensor for input sequence indices
def tensor_transform(token_ids: List[int]):
    return torch.cat((torch.tensor([BOS_IDX]),
                      torch.tensor(token_ids),
                      torch.tensor([EOS_IDX])))

# src and tgt language text transforms to convert raw strings into tensors indices
text_transform = {}
for ln in [SRC_LANGUAGE, TGT_LANGUAGE]:
    text_transform[ln] = sequential_transforms(token_transform[ln], #Tokenization
                                               vocab_transform[ln], #Numericalizat
                                               tensor_transform) # Add BOS/EOS and


# function to collate data samples into batch tesors
def collate_fn(batch):
    src_batch, tgt_batch = [], []
    for src_sample, tgt_sample in batch:
        src_batch.append(text_transform[SRC_LANGUAGE](src_sample.rstrip("\n")))
        tgt_batch.append(text_transform[TGT_LANGUAGE](tgt_sample.rstrip("\n")))

    src_batch = pad_sequence(src_batch, padding_value=PAD_IDX)
    tgt_batch = pad_sequence(tgt_batch, padding_value=PAD_IDX)
    return src_batch, tgt_batch
```

## DataLoader

```
BATCH_SIZE = 32
train_iter = Multi30k(split='train', language_pair=(SRC_LANGUAGE, TGT_LANGUAGE))
train_loader = DataLoader(train_iter, batch_size=BATCH_SIZE, collate_fn=collate_fn

val_iter = Multi30k(split='valid', language_pair=(SRC_LANGUAGE, TGT_LANGUAGE))
val_loader = DataLoader(val_iter, batch_size=BATCH_SIZE, collate_fn=collate_fn, nu

test_iter = Multi30k(split='test', language_pair=(SRC_LANGUAGE, TGT_LANGUAGE))
test_loader = DataLoader(test_iter, batch_size=BATCH_SIZE, collate_fn=collate_fn,
```

```
validation.tar.gz: 100%|██████████| 46.3k/46.3k [00:00<00:00, 277kB/s]
mmt16_task1_test.tar.gz: 100%|██████████| 43.9k/43.9k [00:00<00:00, 261kB/s]
```

# Model

## Boilerplate Code for PyTorch Lightning

```
class TL(pl.LightningModule):
    def __init__(self):
        super(TL, self).__init__()

        self.train_acc =  torch.tensor(0.)
        self.avg_train_loss = torch.tensor(0.)
        self.table_context = None
        self.trgs = []
        self.preds = []



    def training_step(self, batch, batch_idx):
        src, trg = batch
        output = self(src, trg)
        output_dim = output.shape[-1]
        output = output[1:].view(-1, output_dim)
        trg = trg[1:].view(-1)
        loss_train = self.loss(output, trg)
        return loss_train

    def validation_step(self, batch, batch_idx):
        src, trg = batch
        output = self(src, trg, 0)


        out = output.argmax(2)

        o = torch.transpose(out,0,1)
        t = torch.transpose(trg,0,1)
        for o1, t1 in zip(o,t):
          stop_ind_trg = (t1==3).nonzero()[0].item() # stop when <eos> token is fo
          if any(o1==3) == False: # if <eos> token is not found
            stop_ind_pred = len(o1)   # use complete sentence
          else:
            stop_ind_pred = (o1==3).nonzero()[0].item() # stop when <eos> token is

          trg_sent_i = t1[:stop_ind_trg+1]
          pred_sent_i = o1[:stop_ind_pred+1]

          trg_sent_tok = [vocab_transform['en'].lookup_token(word_i) for word_i in
          pred_sent_tok = [vocab_transform['en'].lookup_token(word_i) for word_i i

          self.trgs.append([trg_sent_tok])
          self.preds.append(pred_sent_tok)
```

```
        output_dim = output.shape[-1]
        output = output[1:].view(-1, output_dim)
        trg = trg[1:].view(-1)
        loss_valid = self.loss(output, trg)


        return {"loss": loss_valid}

    def training_epoch_end(self, outputs):
        self.avg_train_loss = torch.stack([x['loss'] for x in outputs]).mean()

    def validation_epoch_end(self, outputs):
        if trainer.sanity_checking:
          print('sanity check')
          return
        bleu = bleu_score(self.preds, self.trgs) * 100
        bleur = round(bleu, 2)
        self.trgs = []
        self.preds = []

        avg_valid_loss = torch.stack([x['loss'] for x in outputs]).mean()
        metrics = {'epoch': self.current_epoch+1, 'Train PPL': math.exp(self.avg_t
        if self.table_context is None:
          self.table_context = tp.TableContext(headers=['epoch', 'Train PPL', 'Tra
          self.table_context.__enter__()
        self.table_context([self.current_epoch+1, math.exp(self.avg_train_loss.ite
        self.logger.log_metrics(metrics)
        if self.current_epoch == self.trainer.max_epochs - 1:
          self.validation_end(outputs)

    def validation_end(self, outputs):
        self.table_context.__exit__()
```

## Encoder

```
class Encoder(pl.LightningModule):
    def __init__(self, input_dim, emb_dim, hid_dim, dropout):
        super().__init__()

        self.hid_dim = hid_dim

        self.embedding = nn.Embedding(input_dim, emb_dim)
        self.rnn = nn.GRU(emb_dim, hid_dim)
        self.dropout = nn.Dropout(dropout)

    def forward(self, src):
        embedded = self.dropout(self.embedding(src))
        output, hidden = self.rnn(embedded)

        return hidden
```

# Decoder

```
class Decoder(pl.LightningModule):
    def __init__(self, output_dim, emb_dim, hid_dim, dropout):
        super().__init__()

        self.hid_dim = hid_dim
        self.output_dim = output_dim
        self.embedding = nn.Embedding(output_dim, emb_dim)
        self.rnn = nn.GRU(emb_dim + hid_dim, hid_dim)
        self.fc_out = nn.Linear(emb_dim + hid_dim * 2, output_dim)
        self.dropout = nn.Dropout(dropout)

    def forward(self, input, hidden, context):
        input = input.unsqueeze(0)
        embedded = self.dropout(self.embedding(input))
        emb_con = torch.cat((embedded, context), dim = 2)
        output, hidden = self.rnn(emb_con, hidden)
        output = torch.cat((embedded.squeeze(0), hidden.squeeze(0), context.squeez
        prediction = self.fc_out(output)
        return prediction, hidden
```

# Seq2Seq Model

```
# Define the model

class Seq2Seq(TL):
    def __init__(self, encoder, decoder, device):
        super(Seq2Seq, self).__init__()

        self.loss = nn.CrossEntropyLoss(ignore_index=PAD_IDX)
        self.lr = 1e-3

        self.encoder = encoder
        self.decoder = decoder
        # self.device = device # Doesn't work in PyTorchLightning since it is alre

        assert encoder.hid_dim == decoder.hid_dim, "Hidden Dimensions of Encoder a

    def forward(self, src, trg, teacher_forcing_ratio = 0.5):

        batch_size = trg.shape[1]
        trg_len = trg.shape[0]
        trg_vocab_size = self.decoder.output_dim

        outputs = torch.zeros(trg_len, batch_size, trg_vocab_size).to(self.device)

        context = self.encoder(src)
        hidden = context
```

```
        input = trg[0,:]

        for t in range(1, trg_len):

            output, hidden = self.decoder(input, hidden, context)

            outputs[t] = output

            teacher_force = random.random() < teacher_forcing_ratio

            top1 = output.argmax(1)

            input = trg[t] if teacher_force else top1

        return outputs

    def configure_optimizers(self):
        optim = torch.optim.Adam(self.parameters())
        return optim
```

## Model Initialization and Summary

```
INPUT_DIM = len(vocab_transform[SRC_LANGUAGE])
OUTPUT_DIM = len(vocab_transform[TGT_LANGUAGE])


ENC_EMB_DIM = 256
DEC_EMB_DIM = 256
HID_DIM = 512
ENC_DROPOUT = 0.5
DEC_DROPOUT = 0.5

enc = Encoder(INPUT_DIM, ENC_EMB_DIM, HID_DIM, ENC_DROPOUT)
dec = Decoder(OUTPUT_DIM, DEC_EMB_DIM, HID_DIM, DEC_DROPOUT)

model = Seq2Seq(enc, dec, device).to(device)
```

## Model Checkpoint

```
checkpoint_callback = ModelCheckpoint(
    monitor='val_loss',
    dirpath='/content',
    filename='sst-{epoch:02d}-{val_loss:.2f}',
    mode='min'
)
```

```
!rm -rf csv_logs
csvlogger = CSVLogger('csv_logs', name='END2_Assign_9', version=0)
trainer = pl.Trainer(max_epochs=10, num_sanity_val_steps=0, logger=csvlogger, gpus=
```

```
trainer.fit(model, train_dataloader=train_loader, val_dataloaders=val_loader)
checkpoint_callback.best_model_path
```

```
GPU available: True, used: True
TPU available: False, using: 0 TPU cores
LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]

  | Name    | Type             | Params
---------------------------------------------
0 | loss    | CrossEntropyLoss | 0
1 | encoder | Encoder          | 6.1 M
2 | decoder | Decoder          | 18.6 M
---------------------------------------------
24.7 M    Trainable params
0         Non-trainable params
24.7 M    Total params
98.916    Total estimated model params size (MB)
/usr/local/lib/python3.7/dist-packages/pytorch_lightning/utilities/data.py:42
  'Your `IterableDataset` has `__len__` defined.'
Epoch 9: 100%               939/939 [01:34<00:00, 9.96it/s, loss=2.08, v_num=0]
```

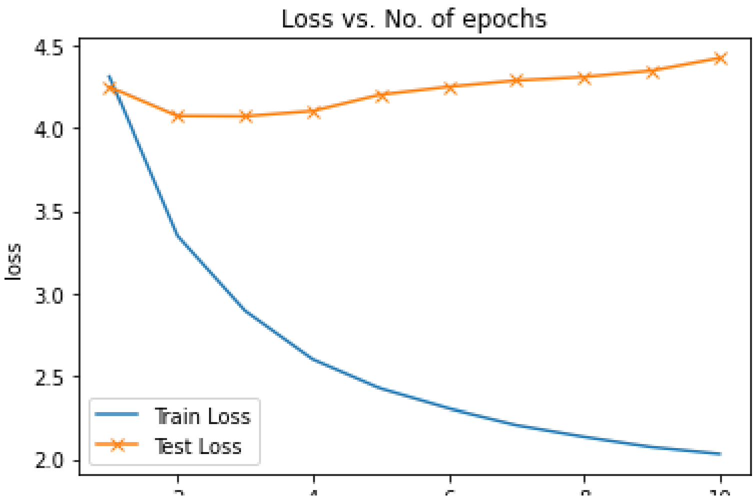
| epoch | Train PPL | Train Loss | Valid PPL | Valid Loss |
|---|---|---|---|---|
| 1 | 74.443 | 4.31 | 69.903 | 4.2471 |
| 2 | 28.479 | 3.3492 | 58.68 | 4.0721 |
| 3 | 18.081 | 2.8949 | 58.568 | 4.0702 |
| 4 | 13.478 | 2.6011 | 60.468 | 4.1021 |
| 5 | 11.296 | 2.4245 | 66.772 | 4.2013 |
| 6 | 10.024 | 2.3049 | 70.036 | 4.249 |
| 7 | 9.0696 | 2.2049 | 72.74 | 4.2869 |
| 8 | 8.4542 | 2.1347 | 74.306 | 4.3082 |
| 9 | 7.9432 | 2.0723 | 77.209 | 4.3465 |
| 10 | 7.624 | 2.0313 | 83.348 | 4.423 |

''

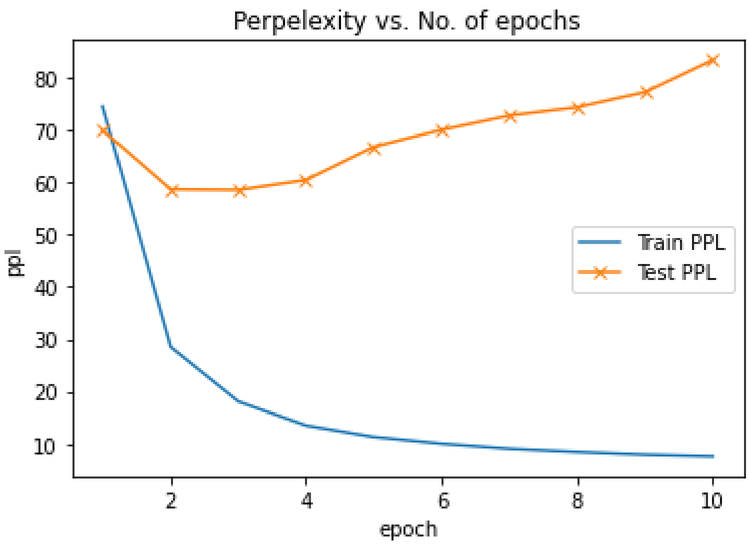
## Training Log

```
root='./csv_logs/' + 'END2_Assign_9' + '/'
dirlist = [ item for item in os.listdir(root) if os.path.isdir(os.path.join(root,
metricfile = root + dirlist[-1:][0] + '/metrics.csv'
metrics = pd.read_csv(metricfile)
```

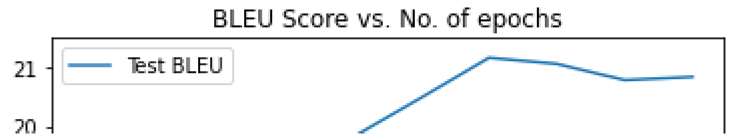
```
plt.plot(metrics['epoch'], metrics['Train Loss'], label="Train Loss")
plt.plot(metrics['epoch'], metrics['Valid Loss'], '-x', label="Test Loss")
plt.xlabel('epoch')
plt.ylabel('loss')
plt.legend()
plt.title('Loss vs. No. of epochs');
```




```
plt.plot(metrics['epoch'], metrics['Train PPL'], label="Train PPL")
plt.plot(metrics['epoch'], metrics['Valid PPL'], '-x', label="Test PPL")
plt.xlabel('epoch')
plt.ylabel('ppl')
plt.legend()
plt.title('Perpelexity vs. No. of epochs');
```

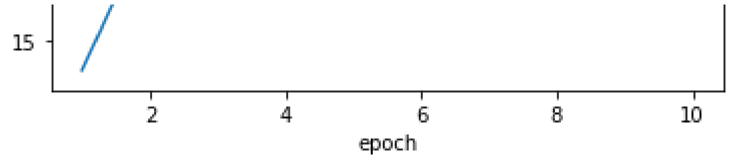



```
plt.plot(metrics['epoch'], metrics['Valid BLEU'], label="Test BLEU")
plt.xlabel('epoch')
plt.ylabel('BLEU')
plt.legend()
plt.title('BLEU Score vs. No. of epochs');
```




# Conclusion

We can see that in general, the BLEU score increases (except for slight dip near the end due to overfitting), which means that our model is learning and improving




✓ 0s completed at 9:44 AM