BY:

Aymen_Zeghaida_1926415

Simeon_Sanmar_1938126

Machine translation

The goal of this TP is to build a machine translation model. You will be comparing the performance of three different architectures:

- A vanilla RNN
- A GRU-RNN
- A transformer

You are provided with the code to load and build the pytorch dataset, and the code for the training loop. You "only" have to code the architectures. Of course, the use of built-in torch layers such as `nn.GRU`, `nn.RNN` or `nn.Transformer` is forbidden, as there would be no exercise otherwise.

The source sentences are in english and the target language is french.

This is also for you the occasion to see what a basic machine learning pipeline looks like. Take a look at the given code, you might learn a lot!

Do not forget to **select the runtime type as GPU!**

Sources

- Dataset: Tab-delimited Bilingual Sentence Pairs
- The code is inspired by this pytorch tutorial.

This notebook is quite big, use the table of contents to easily navigate through it.

Imports and data initializations

We first download and parse the dataset. From the parsed sentences we can build the vocabularies and the torch datasets. The end goal of this section is to have an iterator that can yield the pairs of translated datasets, and where each sentences is made of a sequence of tokens.

✓ 0s completed at 6:40 AM

```
!python3 -m spacy download en
!python3 -m spacy download fr
!pip install torchinfo
!pip install einops
!pip install wandb


from itertools import takewhile
from collections import Counter, defaultdict

import numpy as np
from sklearn.model_selection import train_test_split
import pandas as pd

import torch
import torch.nn as nn
import torch.optim as optim
from torch.utils.data.dataset import Dataset
from torch.utils.data import DataLoader
from torch.nn.utils.rnn import pad_sequence

import torchtext
from torchtext.data.utils import get_tokenizer
from torchtext.vocab import build_vocab_from_iterator, Vocab
from torchtext.datasets import IWSLT2016

import einops
import wandb
from torchinfo import summary
```

```
2023-04-10 09:05:59.413434: I tensorflow/core/util/port.cc:110] oneDNN custom
2023-04-10 09:05:59.483335: I tensorflow/core/platform/cpu_feature_guard.cc:1
To enable the following instructions: AVX2 AVX512F AVX512_VNNI FMA, in other
2023-04-10 09:06:00.502070: W tensorflow/compiler/tf2tensorrt/utils/py_utils.
⚠ As of spaCy v3.0, shortcuts like 'en' are deprecated. Please use the
full pipeline package name 'en_core_web_sm' instead.
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-
Collecting en-core-web-sm==3.5.0
  Downloading https://github.com/explosion/spacy-models/releases/download/en_
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 12.8/12.8 MB 86.0 MB/s eta 0:00
Requirement already satisfied: spacy<3.6.0,>=3.5.0 in /usr/local/lib/python3.
Requirement already satisfied: srsly<3.0.0,>=2.4.3 in /usr/local/lib/python3.
Requirement already satisfied: pathy>=0.10.0 in /usr/local/lib/python3.9/dist
Requirement already satisfied: spacy-loggers<2.0.0,>=1.0.0 in /usr/local/lib/
Requirement already satisfied: wasabi<1.2.0,>=0.9.1 in /usr/local/lib/python3
Requirement already satisfied: pydantic!=1.8,!=1.8.1,<1.11.0,>=1.7.4 in /usr/
Requirement already satisfied: requests<3.0.0,>=2.13.0 in /usr/local/lib/pyth
Requirement already satisfied: smart-open<7.0.0,>=5.2.1 in /usr/local/lib/pyt
Requirement already satisfied: spacy-legacy<3.1.0,>=3.0.11 in /usr/local/lib/
Requirement already satisfied: catalogue<2.1.0,>=2.0.6 in /usr/local/lib/pyth
Requirement already satisfied: setuptools in /usr/local/lib/python3.9/dist-pa
Requirement already satisfied: jinja2 in /usr/local/lib/python3.9/dist-packag
Requirement already satisfied: langcodes<4.0.0,>=3.2.0 in /usr/local/lib/pyth
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.9/di
Requirement already satisfied: numpy>=1.15.0 in /usr/local/lib/python3.9/dist
Requirement already satisfied: cymem<2.1.0,>=2.0.2 in /usr/local/lib/python3.
Requirement already satisfied: thinc<8.2.0,>=8.1.8 in /usr/local/lib/python3.
Requirement already satisfied: typer<0.8.0,>=0.3.0 in /usr/local/lib/python3.
Requirement already satisfied: tqdm<5.0.0,>=4.38.0 in /usr/local/lib/python3.
Requirement already satisfied: murmurhash<1.1.0,>=0.28.0 in /usr/local/lib/py
Requirement already satisfied: preshed<3.1.0,>=3.0.2 in /usr/local/lib/python
Requirement already satisfied: typing-extensions>=4.2.0 in /usr/local/lib/pyt
Requirement already satisfied: charset-normalizer~=2.0.0 in /usr/local/lib/py
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.9/dist-
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /usr/local/lib/python
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.9
Requirement already satisfied: blis<0.8.0,>=0.7.8 in /usr/local/lib/python3.9
Requirement already satisfied: confection<1.0.0,>=0.0.1 in /usr/local/lib/pyt
Requirement already satisfied: click<9.0.0,>=7.1.1 in /usr/local/lib/python3.
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.9/di
✔ Download and installation successful
You can now load the package via spacy.load('en_core_web_sm')
2023-04-10 09:06:12.309865: I tensorflow/core/util/port.cc:110] oneDNN custom
2023-04-10 09:06:12.364914: I tensorflow/core/platform/cpu_feature_guard.cc:1
To enable the following instructions: AVX2 AVX512F AVX512_VNNI FMA, in other
2023-04-10 09:06:13.359388: W tensorflow/compiler/tf2tensorrt/utils/py_utils.
⚠ As of spaCy v3.0, shortcuts like 'fr' are deprecated. Please use the
full pipeline package name 'fr_core_news_sm' instead.
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-
Collecting fr-core-news-sm==3.5.0
  Downloading https://github.com/explosion/spacy-models/releases/download/fr_
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 16.3/16.3 MB 42.6 MB/s eta 0:00
Requirement already satisfied: spacy<3.6.0,>=3.5.0 in /usr/local/lib/python3.
Requirement already satisfied: tqdm<5.0.0,>=4.38.0 in /usr/local/lib/python3.
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.9/di
Requirement already satisfied: pydantic!=1.8,!=1.8.1,<1.11.0,>=1.7.4 in /usr/
Requirement already satisfied: jinja2 in /usr/local/lib/python3.9/dist-packag
Requirement already satisfied: srsly<3.0.0,>=2.4.3 in /usr/local/lib/python3.
```

The tokenizers are objects that are able to divide a python string into a list of tokens (words, punctuations, special tokens...) as a list of strings.

The special tokens are used for a particular reasons:

- *<unk>*: Replace an unknown word in the vocabulary by this default token
- *<pad>*: Virtual token used to as padding token so a batch of sentences can have a unique length
- *<bos>*: Token indicating the beggining of a sentence in the target sequence
- *<eos>*: Token indicating the end of a sentence in the target sequence

```
# Original dataset, but there's a bug on Colab with it
# Original dataset, but there's a bug on Colab with it
# train, valid, _ = IWSLT2016(language_pair=('fr', 'en'))
# train, valid = list(train), list(valid)

# Another dataset, but it is too huge
# !wget https://www.statmt.org/wmt14/training-monolingual-europarl-v7/europarl-v7.
# !wget https://www.statmt.org/wmt14/training-monolingual-europarl-v7/europarl-v7.
# !gunzip europarl-v7.en.gz
# !gunzip europarl-v7.fr.gz

# with open('europarl-v7.en', 'r') as my_file:
#     english = my_file.readlines()

# with open('europarl-v7.fr', 'r') as my_file:
#     french = my_file.readlines()

# dataset = [
#     (en, fr)
#     for en, fr in zip(english, french)
# ]
# print(f'\n{len(dataset):,} sentences.')

# dataset, _ = train_test_split(dataset, test_size=0.8, random_state=0)  # Remove
# train, valid = train_test_split(dataset, test_size=0.2, random_state=0)  # Split

# Our current dataset
!wget http://www.manythings.org/anki/fra-eng.zip
!unzip fra-eng.zip


df = pd.read_csv('fra.txt', sep='\t', names=['english', 'french', 'attribution'])
train = [
    (en, fr) for en, fr in zip(df['english'], df['french'])
]
train, valid = train_test_split(train, test_size=0.1, random_state=0)
print(len(train))

en_tokenizer, fr_tokenizer = get_tokenizer('spacy', language='en'), get_tokenizer(

SPECIALS = ['<unk>', '<pad>', '<bos>', '<eos>']
```

```
--2023-04-10 09:06:36--  http://www.manythings.org/anki/fra-eng.zip
Resolving www.manythings.org (www.manythings.org)... 173.254.30.110
Connecting to www.manythings.org (www.manythings.org)|173.254.30.110|:80... c
HTTP request sent, awaiting response... 200 OK
Length: 7420323 (7.1M) [application/zip]
Saving to: ‘fra-eng.zip.2’

fra-eng.zip.2       100%[===================>]   7.08M  6.00MB/s    in 1.2s

2023-04-10 09:06:38 (6.00 MB/s) - ‘fra-eng.zip.2’ saved [7420323/7420323]
```

```
Archive:  fra-eng.zip
replace _about.txt? [y]es, [n]o, [A]ll, [N]one, [r]ename: n
replace fra.txt? [y]es, [n]o, [A]ll, [N]one, [r]ename: n
196177
/usr/local/lib/python3.9/dist-packages/torchtext/data/utils.py:105: UserWarni
  warnings.warn(
/usr/local/lib/python3.9/dist-packages/torchtext/data/utils.py:105: UserWarni
  warnings.warn(
```

Datasets

Functions and classes to build the vocabularies and the torch datasets. The vocabulary is an object able to transform a string token into the id (an int) of that token in the vocabulary.

```
class TranslationDataset(Dataset):
    def __init__(
            self,
            dataset: list,
            en_vocab: Vocab,
            fr_vocab: Vocab,
            en_tokenizer,
            fr_tokenizer,
        ):
        super().__init__()

        self.dataset = dataset
        self.en_vocab = en_vocab
        self.fr_vocab = fr_vocab
        self.en_tokenizer = en_tokenizer
        self.fr_tokenizer = fr_tokenizer

    def __len__(self):
        """Return the number of examples in the dataset.
        """
        return len(self.dataset)

    def __getitem__(self, index: int) -> tuple:
        """Return a sample.

        Args
        ----
            index: Index of the sample.

        Output
        ------
            en_tokens: English tokens of the sample, as a LongTensor.
            fr_tokens: French tokens of the sample, as a LongTensor.
        """
        # Get the strings
        en_sentence, fr_sentence = self.dataset[index]

        # To list of words
        # We also add the beggining-of-sentence and end-of-sentence tokens
        en_tokens = ['<bos>'] + self.en_tokenizer(en_sentence) + ['<eos>']
        fr_tokens = ['<bos>'] + self.fr_tokenizer(fr_sentence) + ['<eos>']

        # To list of tokens
        en_tokens = self.en_vocab(en_tokens)  # list[int]
        fr_tokens = self.fr_vocab(fr_tokens)

        return torch.LongTensor(en_tokens), torch.LongTensor(fr_tokens)


def yield_tokens(dataset, tokenizer, lang):
    """Tokenize the whole dataset and yield the tokens.
    """
    assert lang in ('en', 'fr')
    sentence_idx = 0 if lang == 'en' else 1

    for sentences in dataset:
        sentence = sentences[sentence_idx]
        tokens = tokenizer(sentence)
        yield tokens


def build_vocab(dataset: list, en_tokenizer, fr_tokenizer, min_freq: int):
    """Return two vocabularies, one for each language.
    """
    en_vocab = build_vocab_from_iterator(
        yield_tokens(dataset, en_tokenizer, 'en'),
        min_freq=min_freq,
        specials=SPECIALS,
    )
    en_vocab.set_default_index(en_vocab['<unk>'])  # Default token for unknown wor

    fr_vocab = build_vocab_from_iterator(
        yield_tokens(dataset, fr_tokenizer, 'fr'),
        min_freq=min_freq,
        specials=SPECIALS,
    )
    fr_vocab.set_default_index(fr_vocab['<unk>'])

    return en_vocab, fr_vocab


def preprocess(
        dataset: list,
        en_tokenizer,
        fr_tokenizer,
        max_words: int,
    ) -> list:
    """Preprocess the dataset.
    Remove samples where at least one of the sentences are too long.
    Those samples takes too much memory.
    Also remove the pending '\n' at the end of sentences.
    """
    filtered = []

    for en_s, fr_s in dataset:
        if len(en_tokenizer(en_s)) >= max_words or len(fr_tokenizer(fr_s)) >= max_
            continue

        en_s = en_s.replace('\n', '')
        fr_s = fr_s.replace('\n', '')

        filtered.append((en_s, fr_s))

    return filtered


def build_datasets(
        max_sequence_length: int,
        min_token_freq: int,
        en_tokenizer,
        fr_tokenizer,
        train: list,
        val: list,
    ) -> tuple:
    """Build the training, validation and testing datasets.
    It takes care of the vocabulary creation.

    Args
    ----
        - max_sequence_length: Maximum number of tokens in each sequences.
            Having big sequences increases dramatically the VRAM taken during trai
        - min_token_freq: Minimum number of occurences each token must have
            to be saved in the vocabulary. Reducing this number increases
            the vocabularies's size.
        - en_tokenizer: Tokenizer for the english sentences.
        - fr_tokenizer: Tokenizer for the french sentences.
        - train and val: List containing the pairs (english, french) sentences.


    Output
    ------
        - (train_dataset, val_dataset): Tuple of the two TranslationDataset object
    """
    datasets = [
        preprocess(samples, en_tokenizer, fr_tokenizer, max_sequence_length)
        for samples in [train, val]
        for samples in [train, val]
    ]

    en_vocab, fr_vocab = build_vocab(datasets[0], en_tokenizer, fr_tokenizer, min_

    datasets = [
        TranslationDataset(samples, en_vocab, fr_vocab, en_tokenizer, fr_tokenizer
        for samples in datasets
    ]

    return datasets


def generate_batch(data_batch: list, src_pad_idx: int, tgt_pad_idx: int) -> tuple:
    """Add padding to the given batch so that all
    the samples are of the same size.

    Args
    ----
        data_batch: List of samples.
            Each sample is a tuple of LongTensors of varying size.
        src_pad_idx: Source padding index value.
        tgt_pad_idx: Target padding index value.

    Output
    ------
        en_batch: Batch of tokens for the padded english sentences.
            Shape of [batch_size, max_en_len].
        fr_batch: Batch of tokens for the padded french sentences.
            Shape of [batch_size, max_fr_len].
    """
    en_batch, fr_batch = [], []
    for en_tokens, fr_tokens in data_batch:
        en_batch.append(en_tokens)
        fr_batch.append(fr_tokens)

    en_batch = pad_sequence(en_batch, padding_value=src_pad_idx, batch_first=True)
    fr_batch = pad_sequence(fr_batch, padding_value=tgt_pad_idx, batch_first=True)
    return en_batch, fr_batch
```

Models architecture

This is where you have to code the architectures.

In a machine translation task, the model takes as input the whole source sentence along with the current known tokens of the target, and predict the next token in the target sequence. This means that the target tokens are predicted in an autoregressive manner, starting from the first token (right after the *<bos>* token) and producing tokens one by one until the last *<eos>* token.

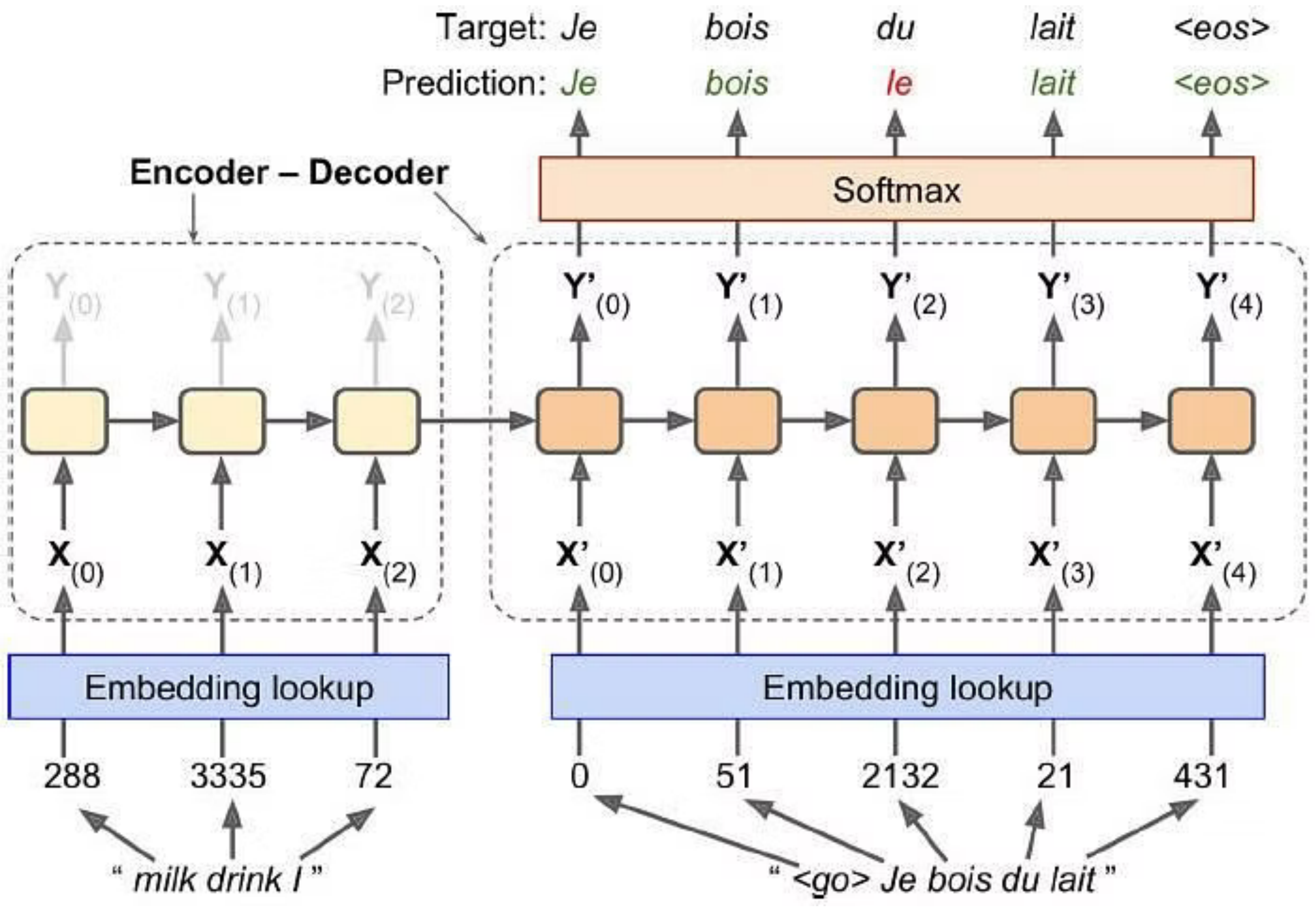
Formally, we define $s = [s_1, \ldots, s_{N_s}]$ as the source sequence made of N_s tokens. We also define $t^i = [t_1, \ldots, t_i]$ as the target sequence at the beginning of the step i.

The output of the model parameterized by θ is:

$$T_{i+1} = p(t_{i+1} | s, t^i; \theta)$$

Where T_{i+1} is the distribution of the next token t_{i+1}.

The loss is simply a *cross entropy loss* over the whole steps, where each class is a token of the vocabulary.



Note that in this image the english sentence is provided in reverse.

In pytorch, there is no dinstinction between an intermediate layer or a whole model having multiple layers in itself. Every layers or models inherit from the `torch.nn.Module`. This module needs to define the `__init__` method where you instanciate the layers, and the `forward` method where you decide how the inputs and the layers of the module interact between them. Thanks to the autograd computations of pytorch, you do not have to implement any backward method!

A really important advice is to **always look at the shape of your input and your output.** From that, you can often guess how the layers should interact with the inputs to produce the right output. You can also easily detect if there's something wrong going on.

You are more than advised to use the `einops` library and the `torch.einsum` function. This will require less operations than 'classical' code, but note that it's a bit trickier to use. This is a way of describing tensors manipulation with strings, bypassing the multiple tensor methods executed in the background. You can find a nice presentation of `einops` here. A paper has just been released about einops here.

A great tutorial on pytorch can be found here. Spending 3 hours on this tutorial is *no* waste of time.

RNN models

RNN

Here you have to implement a recurrent neural network. You will need to create a single RNN Layer, and a module allowing to stack these layers. Look up the pytorch documentation to figure out this module's operations and what is communicated from one layer to another.

The `RNNCell` layer produce one hidden state vector for each sentence in the batch (useful for the output of the encoder), and also produce one embedding for each token in each sentence (useful for the output of the decoder).

The `RNN` module is composed of a stack of `RNNCell`. Each token embeddings coming out from a previous `RNNCell` is used as an input for the next `RNNCell` layer.

Be careful ! Our `RNNCell` implementation is not exactly the same thing as the PyTorch's `nn.RNNCell`. PyTorch implements only the operations for one token (so you would need to loop through each tokens inside the `RNN` instead). You are free to implement `RNN` and `RNNCell` the way you want, as long as it has the expected behaviour of a RNN.

The same thing apply for the `GRU` and `GRUCell`.

```
class RNNCell(nn.Module):
    """A single RNN layer.

    Parameters
    ----------
        input_size: Size of each input token.
        hidden_size: Size of each RNN hidden state.
        dropout: Dropout rate.

    Important note: This layer does not exactly the same thing as nn.RNNCell does.
    PyTorch implementation is only doing one simple pass over one token for each b
    This implementation is taking the whole sequence of each batch and provide the
    final hidden state along with the embeddings of each token in each sequence.
    """
    def __init__(
            self,
            input_size: int,
            hidden_size: int,
            dropout: float,
        ):
        super().__init__()

        self.Wih = nn.Linear(input_size, hidden_size)
        # TODO
        self.Whh = nn.Linear(hidden_size, hidden_size)
        self.tanh = nn.Tanh()
        self.dropout = nn.Dropout(p=dropout)

        # Linear transformation from hidden state to hidden state
        self.hidden_size = hidden_size
        self.device = config["device"]


    def forward(self, x: torch.FloatTensor, h: torch.FloatTensor) -> tuple:
        """Go through all the sequence in x, iteratively updating
        the hidden state h.

        Args
        ----
            x: Input sequence.
                Shape of [batch_size, seq_len, input_size].
            h: Initial hidden state.
                Shape of [batch_size, hidden_size].

        Output
        ------
            y: Token embeddings.
                Shape of [batch_size, seq_len, hidden_size].
            h: Last hidden state.
                Shape of [batch_size, hidden_size].
        """
        batch_size, seq_len, _ = x.size()

        # Create an empty tensor for the token embeddings y
        y = torch.empty(batch_size, seq_len, self.hidden_size).to(self.device)

        # Iterate over the sequence in x
        for sequence in range(seq_len):
            # Linear transformation of the current input token
            # dims : batch_size x hidden_size
            # Wih * x + bih
            ih = self.Wih(x[:, sequence])

            # Linear transformation of the previous hidden state
            # Whh*h + bhh
            h = self.Whh(h)

            # Apply the activation function to the sum of the two linear transform
            h = self.tanh(h + ih) # [batch_size, hidden_size]

            # Store the current hidden state as the token embedding for the curren
            # dims batch_size x seq_len x hidden_size]
            y[:, sequence] = h

        h = self.dropout(h)
        y = self.dropout(y)
        return y, h



class RNN(nn.Module):
    """Implementation of an RNN based
    on https://pytorch.org/docs/stable/generated/torch.nn.RNN.html.

    Parameters
    ----------
        input_size: Size of each input token.
        hidden_size: Size of each RNN hidden state.
        num_layers: Number of layers (RNNCell or GRUCell).
        dropout: Dropout rate.
        model_type: Either 'RNN' or 'GRU', to select which model we want.
            This parameter can be removed if you decide to use the module `GRU`.
            Indeed, `GRU` should have exactly the same code as this module,
            but with `GRUCell` instead of `RNNCell`. We let the freedom for you
            to decide at which level you want to specialise the modules (either
            in `TranslationRNN` by creating a `GRU` or a `RNN`, or in `RNN`
            by creating a `GRUCell` or a `RNNCell`).
    """
    def __init__(
            self,
            input_size: int,
            hidden_size: int,
            num_layers: int,
            dropout: float,
            model_type: str,
        ):
        super().__init__()

        self.num_layers = num_layers

        # Linear transformation from hidden state to hidden state
        self.hidden_size = hidden_size
        self.device = config["device"]

        # Create the RNN layers
        self.model = nn.ModuleList([
            RNNCell(
                input_size if layer == 0 else hidden_size,
                hidden_size,
                dropout if layer != num_layers - 1 else 0
            ) if model_type == 'RNN' else
            GRUCell(
                input_size if layer == 0 else hidden_size,
                hidden_size,
                dropout if layer != num_layers - 1 else 0
            ) for layer in range(num_layers)
        ])

    def forward(self, x: torch.FloatTensor, h: torch.FloatTensor=None) -> tuple:
        """Pass the input sequence through all the RNN cells.
        Returns the output and the final hidden state of each RNN layer

        Args
        ----
            x: Input sequence.
                Shape of [batch_size, seq_len, input_size].
            h: Hidden state for each RNN layer.
                Can be None, in which case an initial hidden state is created.
                Shape of [batch_size, n_layers, hidden_size].

        Output
        ------
            y: Output embeddings for each token after the RNN layers.
                Shape of [batch_size, seq_len, hidden_size].
            h: Final hidden state.
                Shape of [batch_size, n_layers, hidden_size].
        """
        batch_size,_,_ = x.size()

········# Output embeddings for each token after the RNN layers
        output_emb = x

        # Final hidden state for each RNN layer
        final_hidden_state = torch.empty(batch_size, self.num_layers, self.hidden_

        if h == None:
            h = torch.zeros(batch_size, self.num_layers, self.hidden_size).to(self


        # Iterate over all the RNN layers
        # l is the layer
        for l, cell in enumerate(self.model):
            # Select the current hidden state for this RNN layer
            hh = h[:, l]
            # Apply the RNN cell to the input and the hidden state
            output_emb, hh = cell(output_emb, hh) # in/out ([batch_size, seq_len,
            # Save the hidden state for this RNN layer
            final_hidden_state[:, l] = hh
        return output_emb, final_hidden_state
```

GRU

Here you have to implement a GRU-RNN. This architecture is close to the Vanilla RNN but perform different operations. Look up the pytorch documentation to figure out the differences.

```
class GRU(nn.Module):
    """Implementation of a GRU based on https://pytorch.org/docs/stable/generated/

    Parameters
    ----------
        input_size: Size of each input token.
        hidden_size: Size of each RNN hidden state.
        num_layers: Number of layers.
        dropout: Dropout rate.
    """
    def __init__(
            self,
            input_size: int,
            hidden_size: int,
            num_layers: int,
            dropout: float,
        ):
        super().__init__()

        self.hidden_size = hidden_size
        self.num_layers = num_layers
        self.device = config["device"]

        self.model = nn.ModuleList([GRUCell(
            input_size if n == 0 else hidden_size,
            hidden_size,
            dropout if n != (num_layers-1) else 0) for n in range(num_layers)])


    def forward(self, x: torch.FloatTensor, h: torch.FloatTensor=None) -> tuple:
        """
        Args
        ----
            x: Input sequence
                Shape of [batch_size, seq_len, input_size].
            h: Initial hidden state for each layer.
                If 'None', then an initial hidden state (a zero filled tensor)
                is created.
                Shape of [batch_size, n_layers, hidden_size].

        Output
        ------
            output:
                Shape of [batch_size, seq_len, hidden_size].
            h_n: Final hidden state.
                Shape of [batch_size, n_layers, hidden size].
        """
        # TODO
        # pass
        batch_size,_,_ = x.size()

        # Output embeddings for each token after the RNN layers
        output_emb = x

        # Final hidden state for each RNN layer
        final_hidden_state = torch.empty(batch_size, self.num_layers, self.hidden_

        if h == None:
            h = torch.zeros(batch_size, self.num_layers, self.hidden_size).to(self

        # Iterate over all the RNN layers
        # l is the layer
        for l, cell in enumerate(self.model):
            hh = h[:, l]
            output_emb, hh = cell(output_emb, hh)
            final_hidden_state[:, l] = hh
        return output_emb, final_hidden_state


class GRUCell(nn.Module):
    """A single GRU layer.

    Parameters
    ----------
        input_size: Size of each input token.
        hidden_size: Size of each RNN hidden state.
        dropout: Dropout rate.
    """
    def __init__(
            self,
            input_size: int,
            hidden_size: int,
            dropout: float,
        ):
        super().__init__()
        # TODO
        self.gate_size = 3

        self.Wih = nn.Linear(input_size, hidden_size * self.gate_size)
        self.Whh = nn.Linear(hidden_size, hidden_size * self.gate_size)

        self.device = config["device"]
        self.hidden_size = hidden_size
        self.dropout = nn.Dropout(p=dropout)


        std = 1.0 / np.sqrt(self.hidden_size)

        for weight in self.parameters():
            weight.data.uniform_(-std, std)

    def forward(self, x: torch.FloatTensor, h: torch.FloatTensor) -> tuple:
        """
        Args
        ----
            x: Input sequence.
                Shape of [batch_size, seq_len, input_size].
            h: Initial hidden state.
                Shape of [batch_size, hidden_size].

        Output
        ------
            y: Token embeddings.
                Shape of [batch_size, seq_len, hidden_size].
            h: Last hidden state.
                Shape of [batch_size, hidden_size].
        """

        batch_size, seq_len, _ = x.size()

        # Output embeddings for each token after the GRU layer
        output_emb = torch.empty(batch_size, seq_len, self.hidden_size).to(self.de

        # Last hidden state
        final_hidden_state = h

        # Iterate over all the tokens in the input sequence
        for token in range(seq_len):
            # Apply the GRU cell to the current input token and hidden state
            # Wih * input + bih
            x_t = self.Wih(x[:, token])
            # Whh * hidden_state_t-1 + bhh
            hidden_state_t = self.Whh(final_hidden_state) # [batch_size, hidden_si

            # Split the input and hidden state into the reset, update, and new gat
            # and compute the gate activations
            i_reset, i_update, i_new = x_t.chunk(self.gate_size, 1)
            h_reset, h_update, h_new = hidden_state_t.chunk(self.gate_size, 1)

            reset_gate = torch.sigmoid(i_reset + h_reset)
            update_gate = torch.sigmoid(i_update + h_update)
            new_candidate_state = torch.tanh(i_new + (reset_gate * h_new))

            final_hidden_state = update_gate * final_hidden_state + (1 - update_ga

            output_emb[:, token] = final_hidden_state # [batch_size, seq_len, hidd

        output_emb = self.dropout(output_emb)
        final_hidden_state = self.dropout(final_hidden_state)

        return output_emb, final_hidden_state
```

Translation RNN

This module instanciates a vanilla RNN or a GRU-RNN and performs the translation task. You have to:

- Encode the source and target sequence
- Pass the final hidden state of the encoder to the decoder (one for each layer)
- Decode the hidden state into the target sequence

We use teacher forcing for training, meaning that when the next token is predicted, that prediction is based on the previous true target tokens.

```
class TranslationRNN(nn.Module):
    """Basic RNN encoder and decoder for a translation task.
    It can run as a vanilla RNN or a GRU-RNN.

    Parameters
    ----------
        n_tokens_src: Number of tokens in the source vocabulary.
        n_tokens_tgt: Number of tokens in the target vocabulary.
        dim_embedding: Dimension size of the word embeddings (for both language).
        dim_hidden: Dimension size of the hidden layers in the RNNs
            (for both the encoder and the decoder).
        n_layers: Number of layers in the RNNs.
        dropout: Dropout rate.
        src_pad_idx: Source padding index value.
        tgt_pad_idx: Target padding index value.
        model_type: Either 'RNN' or 'GRU', to select which model we want.
    """

    def __init__(
            self,
            n_tokens_src: int,
            n_tokens_tgt: int,
            dim_embedding: int,
            dim_hidden: int,
            n_layers: int,
            dropout: float,
            src_pad_idx: int,
            tgt_pad_idx: int,
            model_type: str,
        ):
        super().__init__()

        # TODO
        self.device = config['device']
        self.dim_hidden = dim_hidden
        self.n_layers = n_layers

        self.embedding_src = nn.Embedding(num_embeddings=n_tokens_src, embedding_d
        self.embedding_tgt = nn.Embedding(num_embeddings=n_tokens_tgt, embedding_d

        self.encoder = None
        self.decoder = None

        if model_type == 'RNN':
            self.encoder = RNN(input_size=dim_embedding,
                                hidden_size=dim_hidden,
                                num_layers=n_layers,
                                dropout=dropout,
                                model_type=model_type)
            self.decoder = RNN(input_size=dim_embedding,
                                hidden_size=dim_hidden,
                                num_layers=n_layers,
                                dropout=dropout,
                                model_type=model_type)

        elif model_type == 'GRU':
            self.encoder = GRU(input_size=dim_embedding,
                                hidden_size=dim_hidden,
                                num_layers=n_layers,
                                dropout=dropout)
            self.decoder = GRU(input_size=dim_embedding,
                                hidden_size=dim_hidden,
                                num_layers=n_layers,
                                dropout=dropout)

        self.layer_norm = nn.LayerNorm(dim_hidden)

        self.mlp = nn.Sequential(
            nn.Linear(dim_hidden, 128),
            nn.LeakyReLU(),
            nn.LayerNorm(128),
            nn.Linear(128, 128),
            nn.LeakyReLU(),
            nn.LayerNorm(128),
            nn.Linear(128, 128),
            nn.LeakyReLU(),
            nn.LayerNorm(128),
            nn.Linear(128, n_tokens_tgt)
        )


    def forward(
        self,
        source: torch.LongTensor,
        target: torch.LongTensor
    ) -> torch.FloatTensor:
        """Predict the target tokens logites based on the source tokens.

        Args
        ----
            source: Batch of source sentences.
                Shape of [batch_size, src_seq_len].
            target: Batch of target sentences.
                Shape of [batch_size, tgt_seq_len].

        Output
        ------
            y: Distributions over the next token for all tokens in each sentences.
                Those need to be the logits only, do not apply a softmax because
                it will be done in the loss computation for numerical stability.
                See https://pytorch.org/docs/stable/generated/torch.nn.CrossEntrop
                Shape of [batch_size, tgt_seq_len, n_tokens_tgt].
        """

        # Embed the source sentence
        x_source = self.embedding_src(source)

        # Encode the source sentence
        _, h_enc = self.encoder(x_source)

        # Embed the target sentence
        x_target = self.embedding_tgt(target)

        # Normalize the encoded hidden state
        h_enc = self.layer_norm(h_enc)

        # Decode the target sentence
        x_dec, _ = self.decoder(x_target, h_enc)

        # Pass the decoded target sentence through the MLP to get logits
        y = self.mlp(x_dec)
        return y
```

Transformer model

Here you have to code the Transformer architecture. It is divided in three parts:

- Attention layers
- Encoder and decoder layers
- Main layers (gather the encoder and decoder layers)

The illustrated transformer blog can help you understanding how the architecture works. Once this is done, you can use the annontated transformer to have an idea of how to code this architecture. We encourage you to use `torch.einsum` and the `einops` library as much as you can. It will make your code simpler.

Implementation order

To help you with the implementation, we advise you following this order:

- Implement `TranslationTransformer` and use `nn.Transformer` instead of `Transformer`
- Implement `Transformer` and use `nn.TransformerDecoder` and `nn.TransformerEnocder`
- Implement the `TransformerDecoder` and `TransformerEncoder` and use `nn.MultiHeadAttention`
- Implement `MultiHeadAttention`

Do not forget to add `batch_first=True` when necessary in the `nn` modules.

Attention layers

We use a `MultiHeadAttention` module, that is able to perform self-attention aswell as cross-attention (depending on what you give as queries, keys and values).

Attention

It takes the multiheaded queries, keys and values as input. It computes the attention between the queries and the keys and return the attended values.

The implementation of this function can greatly be improved with *einsums*.

MultiheadAttention

Computes the multihead queries, keys and values and feed them to the `attention` function. You also need to merge the key padding mask and the attention mask into one mask.

The implementation of this module can greatly be improved with *einops.rearrange*.

```
from einops.layers.torch import Rearrange
from einops import rearrange


def attention(
        q: torch.FloatTensor,
        k: torch.FloatTensor,
        v: torch.FloatTensor,
        mask: torch.BoolTensor=None,
        dropout: nn.Dropout=None,
    ) -> tuple:
    """Computes multihead scaled dot-product attention from the
    projected queries, keys and values.

    Args
    ----
        q: Batch of queries.
            Shape of [batch_size, seq_len_1, n_heads, dim_model].
        k: Batch of keys.
            Shape of [batch_size, seq_len_2, n_heads, dim_model].
        v: Batch of values.
            Shape of [batch_size, seq_len_2, n_heads, dim_model].
        mask: Prevent tokens to attend to some other tokens (for padding or autore
            Attention is prevented where the mask is `True`.
            Shape of [batch_size, n_heads, seq_len_1, seq_len_2],
            or broadcastable to that shape.
        dropout: Dropout layer to use.

    Output
    ------
        y: Multihead scaled dot-attention between the queries, keys and values.
            Shape of [batch_size, seq_len_1, n_heads, dim_model].
        attn: Computed attention between the keys and the queries.
            Shape of [batch_size, n_heads, seq_len_1, seq_len_2].
    """
    # Compute the scaling factor for the dot product
    scaling_factor = torch.tensor(q.size(-1))

    # Compute the dot product between queries and keys, and scale it down by the s
    attn = torch.einsum('bshd,bihd->bhsi', q, k) / torch.sqrt(scaling_factor)

    # Apply the mask to prevent attention to certain tokens (e.g. for padding or a
    if mask is not None:
        attn = attn.masked_fill(mask, float("-1e20"))
    #  for numerical stability
```

```
        # for numerical stability

        # Compute the softmax of the dot product to get the attention weights
        attn_out = torch.softmax(attn, dim = -1)
        if dropout is not None:
            attn_out = dropout(attn_out)

        # Compute the weighted sum of values using the attention weights to get the fi
        y = torch.einsum('bhst,bthd->bshd', attn_out, v)

        return y, attn_out

    class MultiheadAttention(nn.Module):
        """Multihead attention module.
        Can be used as a self-attention and cross-attention layer.
        The queries, keys and values are projected into multiple heads
        before computing the attention between those tensors.

        Parameters
        ----------
            dim: Dimension of the input tokens.
            n_heads: Number of heads. `dim` must be divisible by `n_heads`.
            dropout: Dropout rate.
        """
        def __init__(
                self,
                dim: int,
                n_heads: int,
                dropout: float,
            ):
            super().__init__()

            assert dim % n_heads == 0

            # TODO
            self.n_heads = n_heads
            self.dim = dim
            self.attention = attention
            self.dropout = nn.Dropout(dropout)

            self.q_lin  = nn.Linear(dim, dim)
            self.v_lin  = nn.Linear(dim, dim)
            self.k_lin  = nn.Linear(dim, dim)
            self.linear = nn.Linear(dim, dim)

        def forward(
                self,
                q: torch.FloatTensor,
                k: torch.FloatTensor,
                v: torch.FloatTensor,
                key_padding_mask: torch.BoolTensor = None,
                attn_mask: torch.BoolTensor = None,
            ) -> torch.FloatTensor:
            """Computes the scaled multi-head attention form the input queries,
            keys and values.

            Project those queries, keys and values before feeding them
            to the `attention` function.

            The masks are boolean masks. Tokens are prevented to attends to
            positions where the mask is `True`.

            Args
            ----
                q: Batch of queries.
                    Shape of [batch_size, seq_len_1, dim_model].
                k: Batch of keys.
                    Shape of [batch_size, seq_len_2, dim_model].
                v: Batch of values.
                    Shape of [batch_size, seq_len_2, dim_model].
                key_padding_mask: Prevent attending to padding tokens.
                    Shape of [batch_size, seq_len_2].
                attn_mask: Prevent attending to subsequent tokens.
                    Shape of [seq_len_1, seq_len_2].

            Output
            ------
                y: Computed multihead attention.
                    Shape of [batch_size, seq_len_1, dim_model].
            """
            batch,seq_len_q,_ = q.size() # query sequence length
            seq_len_k         = k.size(1) # key sequence length
            seq_len_v         = v.size(1) # value sequence length

            q = self.q_lin(q)
            k = self.k_lin(k)
            v = self.v_lin(v)

            # Project queries, keys, and values into multiple heads
            q = q.reshape(batch, seq_len_q, self.n_heads, self.dim // self.n_heads)

            k = k.reshape(batch, seq_len_k, self.n_heads, self.dim // self.n_heads)

            v = v.reshape(batch, seq_len_v, self.n_heads, self.dim // self.n_heads)

            # Merge key_padding mask with attn_mask
            key_padding_mask = key_padding_mask.view(batch, 1, 1, seq_len_k).expand(-1

            if attn_mask is not None:
                attn_mask = attn_mask.logical_or(key_padding_mask)
                attn_mask = attn_mask.reshape(batch, self.n_heads, seq_len_q, seq_len_
            # Compute attention
            attn, _ = self.attention(q, k, v, attn_mask, self.dropout)
            # Merge heads
            attn = rearrange(attn, 'b s h e -> b s (h e)')
            # Linear projection for output
            attn_out = self.linear(attn)

            return attn_out
```

Encoder and decoder layers

TranformerEncoder

Apply self-attention layers onto the source tokens. It only needs the source key padding mask.

TranformerDecoder

Apply masked self-attention layers to the target tokens and cross-attention layers between the source and the target tokens. It needs the source and target key padding masks, and the target attention mask.

```
class TransformerDecoderLayer(nn.Module):
    """Single decoder layer.

    Parameters
    ----------
        d_model: The dimension of decoders inputs/outputs.
        dim_feedforward: Hidden dimension of the feedforward networks.
        nheads: Number of heads for each multi-head attention.
        dropout: Dropout rate.
    """

    def __init__(
            self,
            d_model: int,
            d_ff: int,
            nhead: int,
            dropout: float
        ):
        super().__init__()

        # TODO
        self.attention = MultiheadAttention(
            dim=d_model,
            n_heads=nhead,
            dropout=dropout
        )

        self.multihead_attention = MultiheadAttention(
            dim=d_model,
            n_heads=nhead,
            dropout=dropout
        )

        self.dropout1 = nn.Dropout(dropout)
        self.dropout2 = nn.Dropout(dropout)
        self.dropout3 = nn.Dropout(dropout)

        self.layer_norm1 = nn.LayerNorm(d_model)
        self.layer_norm2 = nn.LayerNorm(d_model)
        self.layer_norm3 = nn.LayerNorm(d_model)

        self.feed_forward = nn.Sequential(
            nn.Linear(d_model, d_ff),
            nn.ReLU(),
            nn.Dropout(dropout),
            nn.Linear(d_ff, d_model)
        )


    def forward(
            self,
            src: torch.FloatTensor,
            tgt: torch.FloatTensor,
            tgt_mask_attn: torch.BoolTensor,
            src_key_padding_mask: torch.BoolTensor,
            tgt_key_padding_mask: torch.BoolTensor,
        ) -> torch.FloatTensor:
        """Decode the next target tokens based on the previous tokens.

        Args
        ----
            src: Batch of source sentences.
                Shape of [batch_size, src_seq_len, dim_model].
            tgt: Batch of target sentences.
                Shape of [batch_size, tgt_seq_len, dim_model].
            tgt_mask_attn: Mask to prevent attention to subsequent tokens.
                Shape of [tgt_seq_len, tgt_seq_len].
            src_key_padding_mask: Mask to prevent attention to padding in src sequ
                Shape of [batch_size, src_seq_len].
            tgt_key_padding_mask: Mask to prevent attention to padding in tgt sequ
                Shape of [batch_size, tgt_seq_len].

        Output
        ------
            y:  Batch of sequence of embeddings representing the predicted target
                Shape of [batch_size, tgt_seq_len, dim_model].
        [illegible]
        """

        # compute self-attention
        attn_out, _ = self.attention(tgt, tgt, tgt,
                                attn_mask=tgt_mask_attn, key_padding_mask=tgt_key_
                                )

        attn_out = self.dropout1(attn_out)
        attn_out = self.layer_norm1(tgt + attn_out)

        masked_attn_out, _ = self.multihead_attention(attn_out, src, src,
                                            attn_mask=None, key_padding_mask=src_k
                                            )

        # compute multi-head attention between source and target
        masked_attn_out = self.dropout2(masked_attn_out)
        masked_attn_out = self.layer_norm2(attn_out + masked_attn_out)

        # compute feedforward layer
        feedforward_out = self.dropout3(self.feed_forward(masked_attn_out))
        y = self.layer_norm3(masked_attn_out + feedforward_out)

        return y




class TransformerDecoder(nn.Module):
    """Implementation of the transformer decoder stack.

    Parameters
    ----------
        d_model: The dimension of decoders inputs/outputs.
        dim_feedforward: Hidden dimension of the feedforward networks.
        num_decoder_layers: Number of stacked decoders.
        nheads: Number of heads for each multi-head attention.
        dropout: Dropout rate.
    """

    def __init__(
            self,
            d_model: int,
            d_ff: int,
            num_decoder_layer:int ,
            nhead: int,
            dropout: float
        ):
        super().__init__()

        self.num_decoder_layer = num_decoder_layer
        self.decoder_layer = TransformerDecoderLayer(
            d_model=d_model,
            d_ff=d_ff,
            nhead=nhead,
            dropout=dropout)

        self.decoder_layers = nn.ModuleList([self.decoder_layer for _ in range(num



    def forward(
            self,
            src: torch.FloatTensor,
            tgt: torch.FloatTensor,
            tgt_mask_attn: torch.BoolTensor,
            src_key_padding_mask: torch.BoolTensor,
            tgt_key_padding_mask: torch.BoolTensor,
        ) -> torch.FloatTensor:
        """Decodes the source sequence by sequentially passing.
        the encoded source sequence and the target sequence through the decoder st

        Args
        ----
            src: Batch of encoded source sentences.
                Shape of [batch_size, src_seq_len, dim_model].
            tgt: Batch of taget sentences.
                Shape of [batch_size, tgt_seq_len, dim_model].
            tgt_mask_attn: Mask to prevent attention to subsequent tokens.
                Shape of [tgt_seq_len, tgt_seq_len].
            src_key_padding_mask: Mask to prevent attention to padding in src sequ
                Shape of [batch_size, src_seq_len].
            tgt_key_padding_mask: Mask to prevent attention to padding in tgt sequ
                Shape of [batch_size, tgt_seq_len].

        Output
        ------
            y:  Batch of sequence of embeddings representing the predicted target
                Shape of [batch_size, tgt_seq_len, dim_model].
        """
        y = tgt
        for decoder_layer in self.decoder_layers:
            y = decoder_layer(
                        tgt=y,
                        tgt_mask_attn=tgt_mask_attn,
                        src=src,
                        src_key_padding_mask=src_key_padding_mask,
                        tgt_key_padding_mask=tgt_key_padding_mask)

        return y
```

```
class TransformerEncoderLayer(nn.Module):
    """Single encoder layer.

    Parameters
    ----------
        d_model: The dimension of input tokens.
        dim_feedforward: Hidden dimension of the feedforward networks.
        nheads: Number of heads for each multi-head attention.
        dropout: Dropout rate.
    """

    def __init__(
            self,
            d_model: int,
            d_ff: int,
            nhead: int,
            dropout: float,
        ):
        super().__init__()

        self.attention = MultiheadAttention(
            dim=d_model,
            n_heads=nhead,
            dropout=dropout)

        self.dropout1 = nn.Dropout(dropout)
        self.dropout2 = nn.Dropout(dropout)

        self.layer_norm1 = nn.LayerNorm(d_model)
        self.layer_norm2 = nn.LayerNorm(d_model)

        self.feed_forward = nn.Sequential(
            nn.Linear(d_model, d_ff),
            nn.ReLU(),
            nn.Dropout(dropout),
            nn.Linear(d_ff, d_model)
        )

    def forward(
        self,
        src: torch.FloatTensor,
        key_padding_mask: torch.BoolTensor
        ) -> torch.FloatTensor:
        """Encodes the input. Does not attend to masked inputs.

        Args
        ----
            src: Batch of embedded source tokens.
```

```
                Shape of [batch_size, src_seq_len, dim_model].
            key_padding_mask: Mask preventing attention to padding tokens.
                Shape of [batch_size, src_seq_len].

        Output
        ------
            y: Batch of encoded source tokens.
                Shape of [batch_size, src_seq_len, dim_model].
        """
        attn_out, _ = self.attention(src , src , src ,
                                     attn_mask=None,
                                     key_padding_mask=key_padding_mask
                                     )
        attn_out = self.dropout1(attn_out)

        attn_out = self.layer_norm1(src + attn_out)

        feedforward_out = self.dropout2(self.feed_forward(attn_out))
        y = self.layer_norm2(attn_out + feedforward_out)

        return y



class TransformerEncoder(nn.Module):
    """Implementation of the transformer encoder stack.

    Parameters
    ----------
        d_model: The dimension of encoders inputs.
        dim_feedforward: Hidden dimension of the feedforward networks.
        num_encoder_layers: Number of stacked encoders.
        nheads: Number of heads for each multi-head attention.
        dropout: Dropout rate.
    """

    def __init__(
            self,
            d_model: int,
            dim_feedforward: int,
            num_encoder_layers: int,
            nheads: int,
            dropout: float
        ):
        super().__init__()

        self.num_decoder_layer = num_encoder_layers

        self.encoder_layer = TransformerEncoderLayer(
            d_model=d_model,
            d_ff=dim_feedforward,
            nhead = nheads,
            dropout=dropout)
        # Puts encoder layers in list
        self.encoder_layers = nn.ModuleList([self.encoder_layer for _ in range(num



    def forward(
            self,
            src: torch.FloatTensor,
            key_padding_mask: torch.BoolTensor
        ) -> torch.FloatTensor:
        """Encodes the source sequence by sequentially passing.
        the source sequence through the encoder stack.

        Args
        ----
            src: Batch of embedded source sentences.
                Shape of [batch_size, src_seq_len, dim_model].
            key_padding_mask: Mask preventing attention to padding tokens.
                Shape of [batch_size, src_seq_len].

        Output
        ------
            y: Batch of encoded source sequence.
                Shape of [batch_size, src_seq_len, dim_model].
        """
        y = src
        for layer in self.encoder_layers:
          # Calling the forward of each of each encoder layer
            y = layer(y, key_padding_mask=key_padding_mask)

        return y
```

Main layers

This section gather the `Transformer` and the `TranslationTransformer` modules.

Transformer

The classical transformer architecture. It takes the source and target tokens embeddings and do the forward pass through the encoder and decoder.

Translation Transformer

Compute the source and target tokens embeddings, and apply a final head to produce next token logits. The output must not be the softmax but just the logits, because we use the `nn.CrossEntropyLoss`.

It also creates the *src_key_padding_mask*, the *tgt_key_padding_mask* and the *tgt_mask_attn*.

```
class Transformer(nn.Module):
    """Implementation of a Transformer based on the paper: https://arxiv.org/pdf/1

    Parameters
    ----------
        d_model: The dimension of encoders/decoders inputs/ouputs.
        nhead: Number of heads for each multi-head attention.
        num_encoder_layers: Number of stacked encoders.
        num_decoder_layers: Number of stacked encoders.
        dim_feedforward: Hidden dimension of the feedforward networks.
        dropout: Dropout rate.
    """

    def __init__(
            self,
            d_model: int,
            nhead: int,
            num_encoder_layers: int,
            num_decoder_layers: int,
            dim_feedforward: int,
            dropout: float,
        ):
        super().__init__()

        # Encoder
        self.encoder = TransformerEncoder(
            d_model=d_model,
            dim_feedforward=dim_feedforward,
            num_encoder_layers=num_encoder_layers,
            nheads=nhead,
            dropout=dropout)
        # Decoder
        self.decoder = TransformerDecoder(
            d_model=d_model,
            d_ff=dim_feedforward,
            num_decoder_layer=num_decoder_layers,
            nhead=nhead,
            dropout=dropout)


    def forward(
            self,
            src: torch.FloatTensor,
            tgt: torch.FloatTensor,
            tgt_mask_attn: torch.BoolTensor,
            src_key_padding_mask: torch.BoolTensor,
            tgt_key_padding_mask: torch.BoolTensor
        ) -> torch.FloatTensor:
```

```
        """Compute next token embeddings.

        Args
        ----
            src: Batch of source sequences.
                Shape of [batch_size, src_seq_len, dim_model].
            tgt: Batch of target sequences.
                Shape of [batch_size, tgt_seq_len, dim_model].
            tgt_mask_attn: Mask to prevent attention to subsequent tokens.
                Shape of [tgt_seq_len, tgt_seq_len].
            src_key_padding_mask: Mask to prevent attention to padding in src sequ
                Shape of [batch_size, src_seq_len].
            tgt_key_padding_mask: Mask to prevent attention to padding in tgt sequ
                Shape of [batch_size, tgt_seq_len].

        Output
        ------
            y: Next token embeddings, given the previous target tokens and the sou
                Shape of [batch_size, tgt_seq_len, dim_model].
        """
        encoder_out = self.encoder(src=src, key_padding_mask=src_key_padding_mask)

        decoder_out = self.decoder(
                                tgt=tgt,
                                src=encoder_out,
                                tgt_mask_attn=tgt_mask_attn,
                                src_key_padding_mask=src_key_padding_mask,
                                tgt_key_padding_mask=tgt_key_padding_mask)
        return decoder_out



class TranslationTransformer(nn.Module):
    """Basic Transformer encoder and decoder for a translation task.
    Manage the masks creation, and the token embeddings.
    Position embeddings can be learnt with a standard `nn.Embedding` layer.

    Parameters
    ----------
        n_tokens_src: Number of tokens in the source vocabulary.
        n_tokens_tgt: Number of tokens in the target vocabulary.
        n_heads: Number of heads for each multi-head attention.
        dim_embedding: Dimension size of the word embeddings (for both language).
        dim_hidden: Dimension size of the feedforward layers
            (for both the encoder and the decoder).
        n_layers: Number of layers in the encoder and decoder.
        dropout: Dropout rate.
        src_pad_idx: Source padding index value.
        tgt_pad_idx: Target padding index value.
    """
    def __init__(
    def __init__(
            self,
            n_tokens_src: int,
            n_tokens_tgt: int,
            n_heads: int,
            dim_embedding: int,
            dim_hidden: int,
            n_layers: int,
            dropout: float,
            src_pad_idx: int,
            tgt_pad_idx: int,
        ):
        super().__init__()

        self.device = config["device"]
        self.max_seq_len = config["max_sequence_length"]
        self.src_pad_idx = src_pad_idx
        self.tgt_pad_idx = tgt_pad_idx

        self.embedding_src = nn.Embedding(num_embeddings=n_tokens_src, embedding_d
        self.embedding_tgt = nn.Embedding(num_embeddings=n_tokens_tgt, embedding_d
        self.position_embedding = nn.Embedding(num_embeddings=self.max_seq_len, em

        self.transformer = Transformer(
            d_model=dim_embedding,
            nhead=n_heads,
            num_encoder_layers=n_layers,
            num_decoder_layers=n_layers,
            dim_feedforward=dim_hidden,
            dropout=dropout)

        # (N, T, dim_embedding) -> (N, T, n_tokens_tgt)
        self.linear = nn.Linear(dim_embedding, n_tokens_tgt) # test
        self.mlp = nn.Sequential(
            nn.Linear(dim_embedding, 128),
            nn.LeakyReLU(),
            nn.LayerNorm(128),
            nn.Linear(128, 128),
            nn.LeakyReLU(),
            nn.LayerNorm(128),
            nn.Linear(128, 128),
            nn.LeakyReLU(),
            nn.LayerNorm(128),
            nn.Linear(128, n_tokens_tgt)
        )


    def make_key_padding_mask(self, x, pad_idx) -> torch.BoolTensor:
        # Out : src_key_padding_mask (seq_len_src, seq_len_src)
        return (x == pad_idx).to(self.device)
    def forward(
            self,
            source: torch.LongTensor,
            target: torch.LongTensor
        ) -> torch.FloatTensor:
        """Predict the target tokens logites based on the source tokens.

        Args
        ----
            source: Batch of source sentences.
                Shape of [batch_size, seq_len_src].
            target: Batch of target sentences.
                Shape of [batch_size, seq_len_tgt].

        Output
        ------
            y: Distributions over the next token for all tokens in each sentences.
                Those need to be the logits only, do not apply a softmax because
                it will be done in the loss computation for numerical stability.
                See https://pytorch.org/docs/stable/generated/torch.nn.CrossEntrop
                Shape of [batch_size, seq_len_tgt, n_tokens_tgt].
        """
        batch = source.size(0)
        seq_len_src = source.size(1)
        seq_len_tgt = target.size(1)

        src_positions = (torch.arange(0, seq_len_src).unsqueeze(0).expand(batch, s
        tgt_positions = (torch.arange(0, seq_len_tgt).unsqueeze(0).expand(batch, s
        src_emb = self.embedding_src(source) + self.position_embedding(src_positio
        tgt_emb = self.embedding_tgt(target) + self.position_embedding(tgt_positio

        square_subsequent_mask = torch.triu(torch.full((seq_len_tgt, seq_len_tgt),
        tgt_mask = square_subsequent_mask.to(self.device) # (seq_len_tgt, seq_len_

        src_key_padding_mask = self.make_key_padding_mask(source, self.src_pad_idx
        tgt_key_padding_mask = self.make_key_padding_mask(target, self.tgt_pad_idx



        output = self.transformer(src=src_emb, tgt=tgt_emb, tgt_mask_attn=tgt_mask

        # output = self.linear(output)
        output = self.mlp(output)

        return output
```

Greedy search

Here you have to implement a geedy search to generate a target translation from a trained model and an input source string. The next token will simply be the most probable one.

```
def greedy_search(
        model: nn.Module,
        source: str,
        src_vocab: Vocab,
        tgt_vocab: Vocab,
        src_tokenizer,
        device: str,
        max_sentence_length: int,
    ) -> str:
    """Do a beam search to produce probable translations.

    Args
    ----
        model: The translation model. Assumes it produces logits score (before sof
        source: The sentence to translate.
        src_vocab: The source vocabulary.
        tgt_vocab: The target vocabulary.
        device: Device to which we make the inference.
        max_target: Maximum number of target sentences we keep at the end of each
        max_sentence_length: Maximum number of tokens for the translated sentence.

    Output
    ------
        sentence: The translated source sentence.
    """

    tgt_tokens = ['<bos>']
    EOS_TOKEN = '<eos>'

    src_tokens = ['<bos>'] + src_tokenizer(source) + ['<eos>']
    src_tokens = src_vocab(src_tokens)

    tgt_tokens = tgt_vocab(tgt_tokens)

    EOS_IDX = tgt_vocab['<eos>']
    model.to(device)

    with torch.no_grad():

      src_tokens = torch.LongTensor(src_tokens).unsqueeze(dim=0).to(device)
      tgt_tokens = torch.LongTensor(tgt_tokens).unsqueeze(dim=0).to(device)
      s = 0
      predicted = model.forward(src_tokens, tgt_tokens)

      predicted = torch.softmax(predicted, dim=-1)
      token_max = torch.argmax(predicted)
      predicted_token = tgt_vocab.lookup_token(token_max)
      predicted_sentence = [predicted_token]

      while predicted_token is not EOS_TOKEN or len(predicted_sentence) < max_sent
        predicted_token = torch.LongTensor(tgt_vocab(predicted_token)).unsqueeze(0
        predicted_token = model.forward(src_tokens, predicted_token)
        predicted = torch.softmax(predicted, dim=-1)
        token_max = torch.argmax(predicted)
        predicted_token = tgt_vocab.lookup_token(token_max)

        predicted_sentence.append(predicted_token)
        s += 1
```

Beam search

Beam search is a smarter way of producing a sequence of tokens from an autoregressive model than just using a greedy search.

The greedy search always choose the most probable token as the unique and only next target token, and repeat this processus until the *<eos>* token is predicted.

Instead, the beam search selects the k-most probable tokens at each step. From those k tokens, the current sequence is duplicated k times and the k tokens are appended to the k sequences to produce new k sequences.

You don't have to understand this code, but understanding this code once the TP is over could improve your torch tensors skills.

More explanations

Since it is done at each step, the number of sequences grows exponentially (k sequences after the first step, k² sequences after the second...). In order to keep the number of sequences low, we remove sequences except the top-s most likely sequences. To do that, we keep track of the likelihood of each sequence.

Formally, we define $s = [s_1, \ldots, s_{N_s}]$ as the source sequence made of N_s tokens. We also define $t^i = [t_1, \ldots, t_i]$ as the target sequence at the beginning of the step i.

The output of the model parameterized by θ is:

$$T_{i+1} = p(t_{i+1} | s, t^i; \theta)$$

Where T_{i+1} is the distribution of the next token t_{i+1}.

Then, we define the likelihood of a target sentence $t = [t_1, \ldots, t_{N_t}]$ as:

$$N_t - 1$$

$$L(t) = \prod_{i=1} p(t_{i+1}|s, t_i; \theta)$$

Pseudocode of the beam search:

```
source: [N_s source tokens]  # Shape of [total_source_tokens]
target: [1, <bos> token]  # Shape of [n_sentences, current_target_tokens]
target_prob: [1]  # Shape of [n_sentences]
# We use `n_sentences` as the batch_size dimension

while current_target_tokens <= max_target_length:
    source = repeat(source, n_sentences)  # Shape of [n_sentences, total_source_tokens]
    predicted = model(source, target)[:, -1]  # Predict the next token distributions of a
    tokens_idx, tokens_prob = topk(predicted, k)

    # Append the `n_sentences * k` tokens to the `n_sentences` sentences
    target = repeat(target, k)  # Shape of [n_sentences * k, current_target_tokens]
    target = append_tokens(target, tokens_idx)  # Shape of [n_sentences * k, current_targ

    # Update the sentences probabilities
    target_prob = repeat(target_prob, k)  # Shape of [n_sentences * k]
    target_prob *= tokens_prob

    if n_sentences * k >= max_sentences:
        target, target_prob = topk_prob(target, target_prob, k=max_sentences)
    else:
        n_sentences *= k

    current_target_tokens += 1
```

```
def beautify(sentence: str) -> str:
    """Removes useless spaces.
    """
    punc = {'.', ',', ';'}
    for p in punc:
        sentence = sentence.replace(f' {p}', p)

    links = {'-', "'"}
    for l in links:
        sentence = sentence.replace(f'{l} ', l)
        sentence = sentence.replace(f' {l}', l)

    return sentence
def indices_terminated(
        target: torch.FloatTensor,
        eos_token: int
    ) -> tuple:
    """Split the target sentences between the terminated and the non-terminated
    sentence. Return the indices of those two groups.

    Args
    ----
        target: The sentences.
            Shape of [batch_size, n_tokens].
        eos_token: Value of the End-of-Sentence token.

    Output
    ------
        terminated: Indices of the terminated sentences (who's got the eos_token).
            Shape of [n_terminated, ].
        non-terminated: Indices of the unfinished sentences.
            Shape of [batch_size-n_terminated, ].
    """
    terminated = [i for i, t in enumerate(target) if eos_token in t]
    non_terminated = [i for i, t in enumerate(target) if eos_token not in t]
    return torch.LongTensor(terminated), torch.LongTensor(non_terminated)


def append_beams(
        target: torch.FloatTensor,
        beams: torch.FloatTensor
    ) -> torch.FloatTensor:
    """Add the beam tokens to the current sentences.
    Duplicate the sentences so one token is added per beam per batch.

    Args
    ----
        target: Batch of unfinished sentences.
            Shape of [batch_size, n_tokens].
        beams: Batch of beams for each sentences.
            Shape of [batch_size, n_beams].

    Output
    ------
        target: Batch of sentences with one beam per sentence.
            Shape of [batch_size * n_beams, n_tokens+1].
    """
    batch_size, n_beams = beams.shape
    n_tokens = target.shape[1]

    target = einops.repeat(target, 'b t -> b c t', c=n_beams)
    beams = beams.unsqueeze(dim=2)
    target = torch.cat((target, beams), dim=2)            # [batch_size, n_beams,
    target = target.view(batch_size*n_beams, n_tokens+1)  #·[batch_size·*·n_beams,
    return target


def beam_search(
        model: nn.Module,
        source: str,
        src_vocab: Vocab,
        tgt_vocab: Vocab,
        src_tokenizer,
        device: str,
        beam_width: int,
        max_target: int,
        max_sentence_length: int,
    ) -> list:
    """Do a beam search to produce probable translations.

    Args
    ----
        model: The translation model. Assumes it produces linear score (before sof
        source: The sentence to translate.
        src_vocab: The source vocabulary.
        tgt_vocab: The target vocabulary.
        device: Device to which we make the inference.
        beam_width: Number of top-k tokens we keep at each stage.
        max_target: Maximum number of target sentences we keep at the end of each
        max_sentence_length: Maximum number of tokens for the translated sentence.

    Output
    ------
        sentences: List of sentences orderer by their likelihood.
    """
    src_tokens = ['<bos>'] + src_tokenizer(source) + ['<eos>']
    src_tokens = src_vocab(src_tokens)

    tgt_tokens = ['<bos>']
    tgt_tokens = tgt_vocab(tgt_tokens)

    # To tensor and add unitary batch dimension
    src_tokens = torch.LongTensor(src_tokens).to(device)
    tgt_tokens = torch.LongTensor(tgt_tokens).unsqueeze(dim=0).to(device)
    target_probs = torch.FloatTensor([1]).to(device)
    model.to(device)

    EOS_IDX = tgt_vocab['<eos>']
    with torch.no_grad():
        while tgt_tokens.shape[1] < max_sentence_length:
            batch_size, n_tokens = tgt_tokens.shape

            # Get next beams
            # Get next beams
            src = einops.repeat(src_tokens, 't -> b t', b=tgt_tokens.shape[0])
            predicted = model.forward(src, tgt_tokens)
            predicted = torch.softmax(predicted, dim=-1)
            probs, predicted = predicted[:, -1].topk(k=beam_width, dim=-1)

            # Separe between terminated sentences and the others
            idx_terminated, idx_not_terminated = indices_terminated(tgt_tokens, EO
            idx_terminated, idx_not_terminated = idx_terminated.to(device), idx_no

            tgt_terminated = torch.index_select(tgt_tokens, dim=0, index=idx_termi
            tgt_probs_terminated = torch.index_select(target_probs, dim=0, index=i

            filter_t = lambda t: torch.index_select(t, dim=0, index=idx_not_termin
            tgt_others = filter_t(tgt_tokens)
            tgt_probs_others = filter_t(target_probs)
            predicted = filter_t(predicted)
            probs = filter_t(probs)

            # Add the top tokens to the previous target sentences
            tgt_others = append_beams(tgt_others, predicted)

            # Add padding to terminated target
            padd = torch.zeros((len(tgt_terminated), 1), dtype=torch.long, device=
            tgt_terminated = torch.cat(
                (tgt_terminated, padd),
                dim=1
            )

            # Update each target sentence probabilities
            tgt_probs_others = torch.repeat_interleave(tgt_probs_others, beam_widt
            tgt_probs_others *= probs.flatten()
            tgt_probs_terminated *= 0.999  # Penalize short sequences overtime

            # Group up the terminated and the others
            target_probs = torch.cat(
                (tgt_probs_others, tgt_probs_terminated),
                dim=0
            )
            tgt_tokens = torch.cat(
                (tgt_others, tgt_terminated),
                dim=0
            )

            # Keep only the top `max_target` target sentences
            if target_probs.shape[0] <= max_target:
                continue

            target_probs, indices = target_probs.topk(k=max_target, dim=0)
            tgt_tokens = torch.index_select(tgt_tokens, dim=0, index=indices)
```

```
    sentences = []
    for tgt_sentence in tgt_tokens:
        tgt_sentence = list(tgt_sentence)[1:]  # Remove <bos> token
        tgt_sentence = list(takewhile(lambda t: t != EOS_IDX, tgt_sentence))
        tgt_sentence = ' '.join(tgt_vocab.lookup_tokens(tgt_sentence))
        sentences.append(tgt_sentence)

    sentences = [beautify(s) for s in sentences]

    # Join the sentences with their likelihood
    sentences = [(s, p.item()) for s, p in zip(sentences, target_probs)]
    # Sort the sentences by their likelihood
    sentences = [(s, p) for s, p in sorted(sentences, key=lambda k: k[1], reverse=

    return sentences
```

Training loop

This is a basic training loop code. It takes a big configuration dictionnary to avoid never ending arguments in the functions. We use Weights and Biases to log the trainings. It logs every training informations and model performances in the cloud. You have to create an account to use it. Every accounts are free for individuals or research teams.

```
def print_logs(dataset_type: str, logs: dict):
    """Print the logs.

    Args
    ----
        dataset_type: Either "Train", "Eval", "Test" type.
        logs: Containing the metric's name and value.
    """
    desc = [
        f'{name}: {value:.2f}'
        for name, value in logs.items()
    ]
    desc = '\t'.join(desc)
    desc = f'{dataset_type} -\t' + desc
    desc = desc.expandtabs(5)
    print(desc)


def topk_accuracy(
        real_tokens: torch.FloatTensor,
        probs_tokens: torch.FloatTensor,
        k: int,
        tgt_pad_idx: int,
    ) -> torch.FloatTensor:
```

```
    """Compute the top-k accuracy.
    We ignore the PAD tokens.

    Args
    ----
        real_tokens: Real tokens of the target sentence.
            Shape of [batch_size * n_tokens].
        probs_tokens: Tokens probability predicted by the model.
            Shape of [batch_size * n_tokens, n_target_vocabulary].
        k: Top-k accuracy threshold.
        src_pad_idx: Source padding index value.

    Output
    ------
        acc: Scalar top-k accuracy value.
    """
    total = (real_tokens != tgt_pad_idx).sum()

    _, pred_tokens = probs_tokens.topk(k=k, dim=-1)            # [batch_size * n_t
    real_tokens = einops.repeat(real_tokens, 'b -> b k', k=k)  # [batch_size * n_t

    good = (pred_tokens == real_tokens) & (real_tokens != tgt_pad_idx)
    acc = good.sum() / total
    return acc


def loss_batch(
        model: nn.Module,
        source: torch.LongTensor,
        target: torch.LongTensor,
        config: dict,
    )-> dict:
    """Compute the metrics associated with this batch.
    The metrics are:
        - loss
        - top-1 accuracy
        - top-5 accuracy
        - top-10 accuracy

    Args
    ----
        model: The model to train.
        source: Batch of source tokens.
            Shape of [batch_size, n_src_tokens].
        target: Batch of target tokens.
            Shape of [batch_size, n_tgt_tokens].
        config: Additional parameters.

    Output
    ------
```

```
            metrics: Dictionnary containing evaluated metrics on this batch.
        """
        device = config['device']
        loss_fn = config['loss'].to(device)
        metrics = dict()

        source, target = source.to(device), target.to(device)
        target_in, target_out = target[:, :-1], target[:, 1:]

        # Loss
        pred = model(source, target_in)      # [batch_size, n_tgt_tokens-1, n_vocab]
        pred = pred.view(-1, pred.shape[2])  # [batch_size * (n_tgt_tokens - 1), n_voc
        target_out = target_out.flatten()    # [batch_size * (n_tgt_tokens - 1),]
        metrics['loss'] = loss_fn(pred, target_out)

        # Accuracy - we ignore the padding predictions
        for k in [1, 5, 10]:
            metrics[f'top-{k}'] = topk_accuracy(target_out, pred, k, config['tgt_pad_i

        return metrics


    def eval_model(model: nn.Module, dataloader: DataLoader, config: dict) -> dict:
        """Evaluate the model on the given dataloader.
        """
        device = config['device']
        logs = defaultdict(list)

        model.to(device)
        model.eval()

        with torch.no_grad():
            for source, target in dataloader:
                metrics = loss_batch(model, source, target, config)
                for name, value in metrics.items():
                    logs[name].append(value.cpu().item())

        for name, values in logs.items():
            logs[name] = np.mean(values)
        return logs


    def train_model(model: nn.Module, config: dict):
        """Train the model in a teacher forcing manner.
        """
        train_loader, val_loader = config['train_loader'], config['val_loader']
        train_dataset, val_dataset = train_loader.dataset.dataset, val_loader.dataset.
        optimizer = config['optimizer']
        clip = config['clip']
        device = config['device']
        columns = ['epoch']
        for mode in ['train', 'validation']:
            columns += [
                f'{mode} - {colname}'
                for colname in ['source', 'target', 'predicted', 'likelihood']
            ]
        log_table = wandb.Table(columns=columns)


        print(f'Starting training for {config["epochs"]} epochs, using {device}.')
        for e in range(config['epochs']):
            print(f'\nEpoch {e+1}')

            model.to(device)
            model.train()
            logs = defaultdict(list)

            for batch_id, (source, target) in enumerate(train_loader):
                optimizer.zero_grad()

                metrics = loss_batch(model, source, target, config)
                loss = metrics['loss']

                loss.backward()
                torch.nn.utils.clip_grad_norm_(model.parameters(), clip)
                optimizer.step()

                for name, value in metrics.items():
                    logs[name].append(value.cpu().item())  # Don't forget the '.item'

                if batch_id % config['log_every'] == 0:
                    for name, value in logs.items():
                        logs[name] = np.mean(value)

                    train_logs = {
                        f'Train - {m}': v
                        for m, v in logs.items()
                    }
                    wandb.log(train_logs)
                    logs = defaultdict(list)

            # Logs
            if len(logs) != 0:
                for name, value in logs.items():
                    logs[name] = np.mean(value)
                train_logs = {
                    f'Train - {m}': v
                    for m, v in logs.items()
                }
            else:
                logs = {
                    m.split(' - ')[1]: v
                    for m, v in train_logs.items()
                }

            print_logs('Train', logs)

            logs = eval_model(model, val_loader, config)
            print_logs('Eval', logs)
            val_logs = {
                f'Validation - {m}': v
                for m, v in logs.items()
            }

            val_source, val_target = val_dataset[ torch.randint(len(val_dataset), (1,)
            val_pred, val_prob = beam_search(
                model,
                val_source,
                config['src_vocab'],
                config['tgt_vocab'],
                config['src_tokenizer'],
                device,  # It can take a lot of VRAM
                beam_width=10,
                max_target=100,
                max_sentence_length=config['max_sequence_length'],
            )[0]
            print(val_source)
            print(val_pred)

            logs = {**train_logs, **val_logs}  # Merge dictionnaries
            wandb.log(logs)  # Upload to the WandB cloud

            # Table logs
            train_source, train_target = train_dataset[ torch.randint(len(train_datase
            train_pred, train_prob = beam_search(
                model,
                train_source,
                config['src_vocab'],
                config['tgt_vocab'],
                config['src_tokenizer'],
                device,  # It can take a lot of VRAM
                beam_width=10,
                max_target=100,
                max_sentence_length=config['max_sequence_length'],
            )[0]

            data = [
                e + 1,
                train_source, train_target, train_pred, train_prob,
                val_source, val_target, val_pred, val_prob,
            ]
```

```
        log_table.add_data(*data)

    # Log the table at the end of the training
    wandb.log({'Model predictions': log_table})
```

Training the models

We can now finally train the models. Choose the right hyperparameters, play with them and try to find ones that lead to good models and good training curves. Try to reach a loss under 1.0.

So you know, it is possible to get descent results with approximately 20 epochs. With CUDA enabled, one epoch, even on a big model with a big dataset, shouldn't last more than 10 minutes. A normal epoch is between 1 to 5 minutes.

This is considering Colab Pro, we should try using free Colab to get better estimations.

To test your implementations, it is easier to try your models in a CPU instance. Indeed, Colab reduces your GPU instances priority with the time you recently past using GPU instances. It would be sad to consume all your GPU time on implementation testing. Moreover, you should try your models on small datasets and with a small number of parameters. For exemple, you could set:

```
MAX_SEQ_LEN = 10
MIN_TOK_FREQ = 20
dim_embedding = 40
dim_hidden = 60
n_layers = 1
```

You usually don't want to log anything onto WandB when testing your implementation. To deactivate WandB without having to change any line of code, you can type `!wandb offline` in a cell.

Once you have rightly implemented the models, you can train bigger models on bigger datasets. When you do this, do not forget to change the runtime as GPU (and use `!wandb online`)!

```
# Checking GPU and logging to wandb
import locale
locale.getpreferredencoding = lambda: "UTF-8"

!wandb login
# 4635e7b3acf7ef8066e44259dbb7c61cd48a3dda

!nvidia-smi
```

```
wandb: Currently logged in as: ayzeg (8225_team_). Use `wandb login --relogin
Mon Apr 10 09:08:13 2023
+-----------------------------------------------------------------------------
| NVIDIA-SMI 525.85.12    Driver Version: 525.85.12    CUDA Version: 12.0
|-------------------------------+----------------------+----------------------
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M.
|                               |                      |               MIG M.
|===============================+======================+======================
|   0  NVIDIA A100-SXM...  Off  | 00000000:00:04.0 Off |                    0
| N/A   28C    P0    50W / 400W |   1281MiB / 40960MiB |      0%      Default
|                               |                      |             Disabled
+-------------------------------+----------------------+----------------------

+-----------------------------------------------------------------------------
| Processes:
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory
|        ID   ID                                                   Usage
|=============================================================================
+-----------------------------------------------------------------------------
```

```
# Instanciate the datasets
# MAX_SEQ_LEN = 10
# MIN_TOK_FREQ = 20

MAX_SEQ_LEN = 60
MIN_TOK_FREQ = 2
train_dataset, val_dataset = build_datasets(
    MAX_SEQ_LEN,
    MIN_TOK_FREQ,
    en_tokenizer,
    fr_tokenizer,
    train,
    valid,
)

print(f'English vocabulary size: {len(train_dataset.en_vocab):,}')
print(f'French vocabulary size: {len(train_dataset.fr_vocab):,}')

print(f'\nTraining examples: {len(train_dataset):,}')
print(f'Validation examples: {len(val_dataset):,}')
```

```
English vocabulary size: 11,753
French vocabulary size: 17,820

Training examples: 196,176
Validation examples: 21,797
```

```
# Build the model, the dataloaders, optimizer and the loss function
# Log every hyperparameters and arguments into the config dictionnary
```

```
config = {
    # General parameters
    'epochs': 12,
    'batch_size': 128,
    'lr': 1e-3,
    'betas': (0.9, 0.99),
    'clip': 5,
    'device': 'cuda' if torch.cuda.is_available() else 'cpu',
    # Model parameters
    'n_tokens_src': len(train_dataset.en_vocab),
    'n_tokens_tgt': len(train_dataset.fr_vocab),
    'n_heads': 4,
    'dim_embedding': 196,
    'dim_hidden': 256,
    'n_layers': 3,
    'dim_embedding': 40,
    'dim_hidden': 60,
    'n_layers': 1,
    'dropout': 0.1,
    'model_type': 'GRU',

    # Others
    'max_sequence_length': MAX_SEQ_LEN,
    'min_token_freq': MIN_TOK_FREQ,
    'src_vocab': train_dataset.en_vocab,
    'tgt_vocab': train_dataset.fr_vocab,
    'src_tokenizer': en_tokenizer,
    'tgt_tokenizer': fr_tokenizer,
    'src_pad_idx': train_dataset.en_vocab['<pad>'],
    'tgt_pad_idx': train_dataset.fr_vocab['<pad>'],
    'seed': 0,
    'log_every': 50,  # Number of batches between each wandb logs
}

torch.manual_seed(config['seed'])

config['train_loader'] = DataLoader(
    train_dataset,
    batch_size=config['batch_size'],
    shuffle=True,
    collate_fn=lambda batch: generate_batch(batch, config['src_pad_idx'], config['
)

config['val_loader'] = DataLoader(
    val_dataset,
    batch_size=config['batch_size'],
    shuffle=True,
    collate_fn=lambda batch: generate_batch(batch, config['src_pad_idx'], config['
)
```

```
# model = TranslationRNN(
#     config['n_tokens_src'],
#     config['n_tokens_tgt'],
#     config['dim_embedding'],
#     config['dim_hidden'],
#     config['n_layers'],
#     config['dropout'],
#     config['src_pad_idx'],
#     config['tgt_pad_idx'],
#     config['model_type'],
# )

model = TranslationTransformer(
    config['n_tokens_src'],
    config['n_tokens_tgt'],
    config['n_heads'],
    config['dim_embedding'],
    config['dim_hidden'],
    config['n_layers'],
    config['dropout'],
    config['src_pad_idx'],
    config['tgt_pad_idx'],
)



config['optimizer'] = optim.Adam(
    model.parameters(),
    lr=config['lr'],
    betas=config['betas'],
)

weight_classes = torch.ones(config['n_tokens_tgt'], dtype=torch.float)
weight_classes[config['tgt_vocab']['<unk>']] = 0.1  # Lower the importance of that
config['loss'] = nn.CrossEntropyLoss(
    weight=weight_classes,
    ignore_index=config['tgt_pad_idx'],  # We do not have to learn those
)

summary(
    model,
    input_size=[
        (config['batch_size'], config['max_sequence_length']),
        (config['batch_size'], config['max_sequence_length'])
    ],
    dtypes=[torch.long, torch.long],
    depth=3,
)
```

```
  /usr/local/lib/python3.9/dist-packages/torchinfo/torchinfo.py:477: UserWarnin
    action_fn=lambda data: sys.getsizeof(data.storage()),
  /usr/local/lib/python3.9/dist-packages/torch/storage.py:665: UserWarning: Typ
```

```
/usr/local/lib/python3.9/dist-packages/torch/storage.py:665: UserWarning: Typ
  return super().__sizeof__() + self.nbytes()
================================================================================
Layer (type:depth-idx)                                    Output Shape
Param #
================================================================================
TranslationTransformer                                    [128, 60, 17820]
730,620
├─Embedding: 1-1                                          [128, 60, 40]
470,120
├─Embedding: 1-2                                          [128, 60, 40]
2,400
├─Embedding: 1-3                                          [128, 60, 40]
712,800
├─Embedding: 1-4                                          [128, 60, 40]
(recursive)
├─Transformer: 1-5                                        [128, 60, 40]
--
│    └─TransformerEncoder: 2-1                            [128, 60, 40]
--
│    │    └─ModuleList: 3-1                               --
11,620
│    └─TransformerDecoder: 2-2                            [128, 60, 40]
--
│    │    └─ModuleList: 3-2                               --
18,260
├─Sequential: 1-6                                         [128, 60, 17820]
--
│    └─Linear: 2-3                                        [128, 60, 128]
5,248
│    └─LeakyReLU: 2-4                                     [128, 60, 128]
--
│    └─LayerNorm: 2-5                                     [128, 60, 128]
256
│    └─Linear: 2-6                                        [128, 60, 128]
16,512
│    └─LeakyReLU: 2-7                                     [128, 60, 128]
--
│    └─LayerNorm: 2-8                                     [128, 60, 128]
256
│    └─Linear: 2-9                                        [128, 60, 128]
16,512
│    └─LeakyReLU: 2-10                                    [128, 60, 128]
--
│    └─LayerNorm: 2-11                                    [128, 60, 128]
256
│    └─Linear: 2-12                                       [128, 60, 17820]
2,298,780
================================================================================
Total params: 4,283,640
Trainable params: 4,283,640
Non-trainable params: 0
Total mult-adds (M): 455.09
================================================================================
Input size (MB): 0.12
Forward/backward pass size (MB): 1205.94
```

```
    Forward/backward pass size (MB): 1203.94
```

```
#!wandb online
!wandb login --relogin
!WANDB_MODE=online
```

```
    wandb: Logging into wandb.ai. (Learn how to deploy a W&B server locally: http
    wandb: You can find your API key in your browser here: https://wandb.ai/autho
    wandb: Paste an API key from your profile and hit enter, or press ctrl+c to q
    wandb: Appending key for api.wandb.ai to your netrc file: /root/.netrc
```

```
!wandb online  # online / offline to activate or deactivate WandB logging

# print("Config ", config)

def train(config=config): # Needed for sweeps only
  with wandb.init(
          config=config,
          project='INF8225 - TP3',  # Title of your project
          group='TranslationTransformer run group',  # In what group of runs do yo
          save_code=True,
      ) as run:
      # config = wandb.config
      # run.name =
      train_model(model, config)
```

```
    W&B online. Running your script from this directory will now sync to the clou
```

```
sentence = "It is possible to try your work here."

preds = beam_search(
    model,
    sentence,
    config['src_vocab'],
    config['tgt_vocab'],
    config['src_tokenizer'],
    config['device'],
    beam_width=10,
    max_target=100,
    max_sentence_length=config['max_sequence_length']
)[:5]

for i, (translation, likelihood) in enumerate(preds):
    print(f'{i}. ({likelihood*100:.5f}%) \t {translation}')
```

```
    0. (5.81071%)    C'est possible d'essayer ton travail ici.
    1. (4.09919%)    C'est possible de tenter votre travail ici.
    2. (2.32683%)    Il est possible d'essayer ton travail ici.
    3. (2.06878%)    C'est possible de tenter ton travail ici.
    4. (1.95494%)    Il est possible de tenter votre travail ici.
```

Questions

1. Explain the differences between Vanilla RNN, GRU-RNN, and Transformers.

Vanilla RNN implements a basic cell design, applying a tanh function between the hidden and source inputs. In contrast, a GRU cell incorporates additional connections, known as gates, including a reset and an update gate. These gates possess their own weights and enable the passage of extra information, which enhances the GRU cell's ability to preserve long-term information.

GRU-RNN employs two gates to tackle the vanishing gradient challenge:

```
An update gate : which assesses how much the prior hidden state should be influenced by t
A reset gate : which determines the amount of the previous hidden state that should be di
```

RNNs exhibit faster training and require less computational power compared to GRU-RNNs. Nevertheless, GRU-RNNs surpass RNNs when handling extensive sequences.

With GRU-RNN, words at a sentence's conclusion exert a more substantial effect on upcoming predictions, a problem stemming from recursion.

On the other hand, the transformer architecture deviates from RNN/GRU-RNN due to the absence of recurrent connections. All tokens within a sentence are processed simultaneously, eliminating the notion of timesteps. An attention mechanism is utilized to maintain information about the sequence's history.

2. Why is positionnal encoding necessary in Transformers and not in RNNs?

RNNs feature recurrent connections, allowing the current block to receive the last input as supplementary input, thus providing an understanding of word placement within the sequence input. Conversely, in Transformers, each input within a sequence is independent, rendering Transformers incapable of distinguishing between sequences with jumbled words. To address this, positional encoding is employed.

3. Describe the preprocessing process. Detail how the initial dataset is processed before being fed to the translation models.

The process begins with a text file containing two columns, one with the English source sequence and the other with the corresponding French translation, each line representing a distinct sentence pair. Sentences are then separated into lists of tokens (For example, 'Nice to meet you !' becomes ['Nice','to','meet','you','!']).

Next, the Vocab() class is employed to convert these token lists into one-hot encodings with additional Special tokens (<bos>) and (<eos>) as delimiters for each sequence. (For example ['<bos>', 'my', 'pleasure', '<eos>'] -> [0, 4, 15, 1])

When a PyTorch DataLoader is created from the tokenized dataset, a function is triggered to append a special padding token (<pad>) to ensure that all token batches are of the same size. A special padding (<pad>) is finally appended to all token when a PyTorch DataLoader is created from the tokenized data, to ensure token batches are the same size.

Small report - experiments

Once everything is working fine, you can explore aspects of these models and do some research of your own into how they behave.

For exemple, you can experiment with the hyperparameters. What are the effect of the differents hyperparameters with the final model performance? What about training time?

What are some other metrics you could have for machine translation? Can you compute them and add them to your WandB report?

Those are only examples, you can do whatever you think will be interesting. This part account for many points, *feel free to go wild!*

Make a concise report about your experiments here.

Hyperparameters search (sweep)

3 sweeps have been defined (RNN, GRU , transformer)

For the RNN and GRU, good results seem to come from the models with 6~7 epochs, so we will fix 6 epochs for our following hyperparameter search. For the transformer, the number of epochs will be 12.

```
wandb.agent("INF8225 - TP3/ ", train, count=10) # RNN
```

```
wandb: Agent Starting Run: ooz8sege with config:
wandb:  batch_size: 256
wandb:  clip: 4
wandb:  dim_embedding: 370
wandb:  dim_hidden: 634
wandb:  dropout: 0.2782527633094745
```

```
wandb:   dropout: 0.27825276330947457
wandb:   lr: 0.0009833031838853794
wandb:   n_heads: 6
wandb:   n_layers: 8
wandb: Currently logged in as: ayzeg (8225_team_). Use `wandb login --relogin
wandb: WARNING Ignored wandb.init() arg project when running a sweep.
```

Tracking run with wandb version 0.14.2
Run data is saved locally in `/content/wandb/run-20230410_062824-ooz8sege`
Syncing run **vocal-sweep-4** to Weights & Biases (docs)
Sweep page: https://wandb.ai/8225_team_/INF8225%20-%20TP3/sweeps/1t5pcxwf
View project at https://wandb.ai/8225_team_/INF8225%20-%20TP3
View sweep at https://wandb.ai/8225_team_/INF8225%20-%20TP3/sweeps/1t5pcxwf
View run at https://wandb.ai/8225_team_/INF8225%20-%20TP3/runs/ooz8sege

```
Starting training for 6 epochs, using cuda.

Epoch 1
Train -   loss: 3.37     top-1: 0.38    top-5: 0.58    top-10: 0.66
Eval -    loss: 3.35     top-1: 0.38    top-5: 0.57    top-10: 0.65
I'm beginning to lose patience with Tom.
Je ne sais rien.

Epoch 2
Train -   loss: 3.19     top-1: 0.41    top-5: 0.60    top-10: 0.67
Eval -    loss: 3.15     top-1: 0.41    top-5: 0.60    top-10: 0.68
We want to fix that problem.
Nous sommes prêtes.

Epoch 3
Train -   loss: 3.07     top-1: 0.42    top-5: 0.61    top-10: 0.69
Eval -    loss: 3.07     top-1: 0.41    top-5: 0.61    top-10: 0.69
Nothing lasts forever.
Quand j'étais enfant.

Epoch 4
Train -   loss: 3.01     top-1: 0.43    top-5: 0.62    top-10: 0.70
Eval -    loss: 3.03     top-1: 0.42    top-5: 0.62    top-10: 0.69
He was playing the piano.
Il y a beaucoup d'amis.

Epoch 5
Train -   loss: 2.91     top-1: 0.43    top-5: 0.63    top-10: 0.71
Eval -    loss: 2.95     top-1: 0.43    top-5: 0.63    top-10: 0.70
Get ready.
« ?

Epoch 6
Train -   loss: 2.90     top-1: 0.44    top-5: 0.64    top-10: 0.71
Eval -    loss: 2.94     top-1: 0.44    top-5: 0.63    top-10: 0.71
Too many sweets make you fat.
Selon moi !
```

Waiting for W&B process to finish... **(success).**

Run history:

Train - loss

Train - top-1	
Train - top-10	
Train - top-5	
Validation - loss	
Validation - top-1	
Validation - top-10	
Validation - top-5	

Run summary:

Train - loss	2.89893
Train - top-1	0.43852
Train - top-10	0.71247
Train - top-5	0.63624
Validation - loss	2.94027
Validation - top-1	0.43506
Validation - top-10	0.70537
Validation - top-5	0.63057

View run **vocal-sweep-4** at: https://wandb.ai/8225_team_/INF8225%20-%20TP3/runs/ooz8sege
Synced 5 W&B file(s), 1 media file(s), 3 artifact file(s) and 1 other file(s)
Find logs at: `./wandb/run-20230410_062824-ooz8sege/logs`

```
wandb: Agent Starting Run: xz282xbu with config:
wandb:   batch_size: 128
wandb:   clip: 11
wandb:   dim_embedding: 457
wandb:   dim_hidden: 73
wandb:   dropout: 0.22992617160098855
wandb:   lr: 0.0006994117711121344
wandb:   n_heads: 10
wandb:   n_layers: 7
wandb: WARNING Ignored wandb.init() arg project when running a sweep.
```

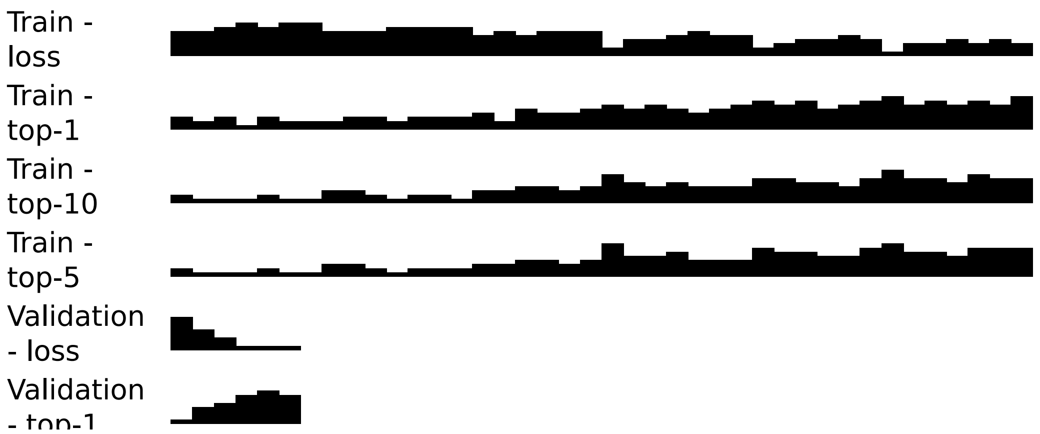
Tracking run with wandb version 0.14.2
Run data is saved locally in `/content/wandb/run-20230410_063423-xz282xbu`
Syncing run **rare-sweep-5** to Weights & Biases (docs)
Sweep page: https://wandb.ai/8225_team_/INF8225%20-%20TP3/sweeps/1t5pcxwf
View project at https://wandb.ai/8225_team_/INF8225%20-%20TP3

View sweep at https://wandb.ai/8225_team_/INF8225%20-%20TP3/sweeps/1t5pcxwf
View run at https://wandb.ai/8225_team_/INF8225%20-%20TP3/runs/xz282xbu

```
Starting training for 6 epochs, using cuda.

Epoch 1
Train -    loss: 2.83     top-1: 0.45    top-5: 0.65    top-10: 0.72
Eval -     loss: 2.90     top-1: 0.44    top-5: 0.64    top-10: 0.71
Tom made a list of things he wanted to do before he died.
Tom n'a pas le choix.

Epoch 2
Train -    loss: 2.81     top-1: 0.45    top-5: 0.65    top-10: 0.72
Eval -     loss: 2.87     top-1: 0.44    top-5: 0.64    top-10: 0.72
I'm going back to my office.
Je suis désolé ?

Epoch 3
Train -    loss: 2.73     top-1: 0.45    top-5: 0.66    top-10: 0.74
Eval -     loss: 2.85     top-1: 0.45    top-5: 0.64    top-10: 0.72
I assure you Tom will be perfectly safe.
Je suis désolé.

Epoch 4
Train -    loss: 2.74     top-1: 0.46    top-5: 0.66    top-10: 0.73
Eval -     loss: 2.84     top-1: 0.45    top-5: 0.65    top-10: 0.72
Please keep me informed.
Quand j'étais enfant !

Epoch 5
Train -    loss: 2.71     top-1: 0.46    top-5: 0.66    top-10: 0.73
Eval -     loss: 2.83     top-1: 0.45    top-5: 0.65    top-10: 0.72
She is too young to know the truth.
Elle me rend nerveux.

Epoch 6
Train -    loss: 2.69     top-1: 0.46    top-5: 0.66    top-10: 0.74
Eval -     loss: 2.83     top-1: 0.45    top-5: 0.65    top-10: 0.72
Who's that woman over there?
Ça te dérange ?
```

Waiting for W&B process to finish... **(success).**

Run history:



Validation - top-10	
Validation - top-5	

Run summary:

Train - loss	2.6862
Train - top-1	0.46161
Train - top-10	0.73852
Train - top-5	0.66399
Validation - loss	2.83212
Validation - top-1	0.45059
Validation - top-10	0.72233
Validation - top-5	0.64873

View run **rare-sweep-5** at: https://wandb.ai/8225_team_/INF8225%20-%20TP3/runs/xz282xbu
Synced 5 W&B file(s), 1 media file(s), 3 artifact file(s) and 1 other file(s)
Find logs at: `./wandb/run-20230410_063423-xz282xbu/logs`

```
wandb: Agent Starting Run: 8bpyw80j with config:
wandb:   batch_size: 128
wandb:   clip: 7
wandb:   dim_embedding: 700
wandb:   dim_hidden: 289
wandb:   dropout: 0.13130109068684753
wandb:   lr: 0.0009352356160543272
wandb:   n_heads: 3
wandb:   n_layers: 6
wandb: WARNING Ignored wandb.init() arg project when running a sweep.
```

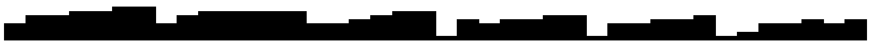
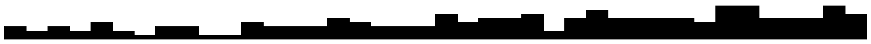
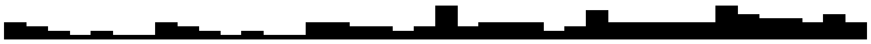
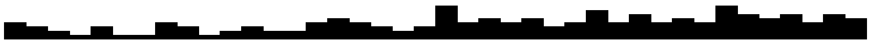
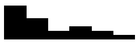
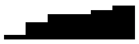
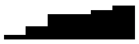
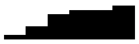
Tracking run with wandb version 0.14.2
Run data is saved locally in `/content/wandb/run-20230410_064023-8bpyw80j`
Syncing run **effortless-sweep-6** to Weights & Biases (docs)
Sweep page: https://wandb.ai/8225_team_/INF8225%20-%20TP3/sweeps/1t5pcxwf
View project at https://wandb.ai/8225_team_/INF8225%20-%20TP3
View sweep at https://wandb.ai/8225_team_/INF8225%20-%20TP3/sweeps/1t5pcxwf
View run at https://wandb.ai/8225_team_/INF8225%20-%20TP3/runs/8bpyw80j

```
Starting training for 6 epochs, using cuda.

Epoch 1
Train -   loss: 2.67     top-1: 0.46    top-5: 0.67    top-10: 0.74
Eval -    loss: 2.82     top-1: 0.45    top-5: 0.65    top-10: 0.72
I let the cat in.
J'ai fait ça.
```

```
Epoch 2
Train -   loss: 2.65     top-1: 0.47    top-5: 0.67    top-10: 0.74
Eval -    loss: 2.81     top-1: 0.46    top-5: 0.65    top-10: 0.73
I would like you to trust me.
Je l'aime.

Epoch 3
Train -   loss: 2.63     top-1: 0.47    top-5: 0.67    top-10: 0.74
Eval -    loss: 2.79     top-1: 0.46    top-5: 0.66    top-10: 0.73
Hone your skills.
Le temps est écoulé.

Epoch 4
Train -   loss: 2.62     top-1: 0.47    top-5: 0.67    top-10: 0.75
Eval -    loss: 2.79     top-1: 0.46    top-5: 0.66    top-10: 0.73
What this club is today is largely due to the effort of these people.
Ce n'est pas vrai.

Epoch 5
Train -   loss: 2.63     top-1: 0.47    top-5: 0.67    top-10: 0.74
Eval -    loss: 2.79     top-1: 0.46    top-5: 0.66    top-10: 0.73
Give me a call later, OK?
Je peux nager ?

Epoch 6
Train -   loss: 2.61     top-1: 0.47    top-5: 0.67    top-10: 0.75
Eval -    loss: 2.78     top-1: 0.46    top-5: 0.66    top-10: 0.73
Clothes don't make the man.
Les choses changent.
```

Waiting for W&B process to finish... **(success).**

Run history:

Train - loss	
Train - top-1	
Train - top-10	
Train - top-5	
Validation - loss	
Validation - top-1	
Validation - top-10	
Validation - top-5	

Run summary:

Train - loss	2.61123

Train - top-1	0.47132
Train - top-10	0.74525
Train - top-5	0.67314
Validation - loss	2.783
Validation - top-1	0.46099
Validation - top-10	0.72984
Validation - top-5	0.65733

View run **effortless-sweep-6** at: https://wandb.ai/8225_team_/INF8225%20-%20TP3/runs/8bpyw80j
Synced 5 W&B file(s), 1 media file(s), 3 artifact file(s) and 1 other file(s)
Find logs at: `./wandb/run-20230410_064023-8bpyw80j/logs`

```
wandb: Agent Starting Run: tyz03tut with config:
wandb:   batch_size: 128
wandb:   clip: 12
wandb:   dim_embedding: 651
wandb:   dim_hidden: 481
wandb:   dropout: 0.07011145463813385
wandb:   lr: 0.0006600273023362729
wandb:   n_heads: 7
wandb:   n_layers: 9
wandb: WARNING Ignored wandb.init() arg project when running a sweep.
```

Tracking run with wandb version 0.14.2
Run data is saved locally in `/content/wandb/run-20230410_064623-tyz03tut`
Syncing run **leafy-sweep-7** to Weights & Biases (docs)
Sweep page: https://wandb.ai/8225_team_/INF8225%20-%20TP3/sweeps/1t5pcxwf
View project at https://wandb.ai/8225_team_/INF8225%20-%20TP3
View sweep at https://wandb.ai/8225_team_/INF8225%20-%20TP3/sweeps/1t5pcxwf
View run at https://wandb.ai/8225_team_/INF8225%20-%20TP3/runs/tyz03tut

```
Starting training for 6 epochs, using cuda.

Epoch 1
Train -    loss: 2.58     top-1: 0.48    top-5: 0.68    top-10: 0.75
Eval -     loss: 2.77     top-1: 0.46    top-5: 0.66    top-10: 0.73
Who's your favorite super hero?
Que dirais-tu ?

Epoch 2
Train -    loss: 2.58     top-1: 0.48    top-5: 0.68    top-10: 0.75
Eval -     loss: 2.77     top-1: 0.46    top-5: 0.66    top-10: 0.73
While he was sick, he lost a lot of weight.
Même s'il vous plaît.

Epoch 3
Train -    loss: 2.60     top-1: 0.47    top-5: 0.68    top-10: 0.75
Eval -     loss: 2.77     top-1: 0.46    top-5: 0.66    top-10: 0.73
I have eaten a lot this morning.
J'ai fait ça.
```

```
Epoch 4
Train -   loss: 2.54     top-1: 0.48    top-5: 0.68    top-10: 0.76
Eval -    loss: 2.76     top-1: 0.46    top-5: 0.66    top-10: 0.73
I didn't know you were expecting anyone.
Je ne peux pas faire ça.

Epoch 5
Train -   loss: 2.52     top-1: 0.48    top-5: 0.69    top-10: 0.76
Eval -    loss: 2.76     top-1: 0.46    top-5: 0.66    top-10: 0.73
It was a revelation to me.
Ce fut une excuse bidon.

Epoch 6
Train -   loss: 2.51     top-1: 0.49    top-5: 0.69    top-10: 0.76
Eval -    loss: 2.77     top-1: 0.46    top-5: 0.66    top-10: 0.73
I want to go and see the cherry trees in blossom.
Je veux t'aider.
```

Waiting for W&B process to finish... **(success).**

Run history:

Train - loss	
Train - top-1	
Train - top-10	
Train - top-5	
Validation - loss	
Validation - top-1	
Validation - top-10	
Validation - top-5	

Run summary:

Train - loss	2.51018
Train - top-1	0.48612
Train - top-10	0.7624
Train - top-5	0.68778
Validation - loss	2.77283
Validation - top-1	0.46288

Validation - top-10	0.73263
Validation - top-5	0.66092

View run **leafy-sweep-7** at: https://wandb.ai/8225_team_/INF8225%20-%20TP3/runs/tyz03tut
Synced 5 W&B file(s), 1 media file(s), 3 artifact file(s) and 1 other file(s)
Find logs at: `./wandb/run-20230410_064623-tyz03tut/logs`

```
wandb: Agent Starting Run: zepyrp0f with config:
wandb:   batch_size: 128
wandb:   clip: 8
wandb:   dim_embedding: 577
wandb:   dim_hidden: 575
wandb:   dropout: 0.253616715822834
wandb:   lr: 0.000770141803485162
wandb:   n_heads: 11
wandb:   n_layers: 6
wandb: WARNING Ignored wandb.init() arg project when running a sweep.
```

Tracking run with wandb version 0.14.2
Run data is saved locally in `/content/wandb/run-20230410_065222-zepyrp0f`
Syncing run **twilight-sweep-8** to Weights & Biases (docs)
Sweep page: https://wandb.ai/8225_team_/INF8225%20-%20TP3/sweeps/1t5pcxwf
View project at https://wandb.ai/8225_team_/INF8225%20-%20TP3
View sweep at https://wandb.ai/8225_team_/INF8225%20-%20TP3/sweeps/1t5pcxwf
View run at https://wandb.ai/8225_team_/INF8225%20-%20TP3/runs/zepyrp0f

```
Starting training for 6 epochs, using cuda.

Epoch 1
Train -   loss: 2.53    top-1: 0.48   top-5: 0.68   top-10: 0.76
Eval -    loss: 2.75    top-1: 0.47   top-5: 0.67   top-10: 0.74
Tom died at a very old age.
Elle lui a conseillé d'arrêter de fumer.

Epoch 2
Train -   loss: 2.53    top-1: 0.48   top-5: 0.69   top-10: 0.76
Eval -    loss: 2.75    top-1: 0.47   top-5: 0.66   top-10: 0.74
I've finished all except the last page.
J'ai fait une erreur.

Epoch 3
Train -   loss: 2.55    top-1: 0.48   top-5: 0.68   top-10: 0.76
Eval -    loss: 2.75    top-1: 0.47   top-5: 0.67   top-10: 0.74
I'm too short to reach the top shelf.
Je suis né.

Epoch 4
Train -   loss: 2.51    top-1: 0.49   top-5: 0.69   top-10: 0.76
Eval -    loss: 2.74    top-1: 0.47   top-5: 0.67   top-10: 0.74
You sing as well as you dance.
Tu parles.

Epoch 5
Train -   loss: 2.50    top-1: 0.49   top-5: 0.69   top-10: 0.76
Eval -    loss: 2.75    top-1: 0.47   top-5: 0.66   top-10: 0.74
It was horrendous
It was horrendous.
Ce fut une erreur.

Epoch 6
Train -    loss: 2.48     top-1: 0.49    top-5: 0.69    top-10: 0.76
Eval -     loss: 2.75     top-1: 0.47    top-5: 0.67    top-10: 0.74
Of course!
Du calme.
```

Waiting for W&B process to finish... **(success).**

Run history:

Train - loss	
Train - top-1	
Train - top-10	
Train - top-5	
Validation - loss	
Validation - top-1	
Validation - top-10	
Validation - top-5	

Run summary:

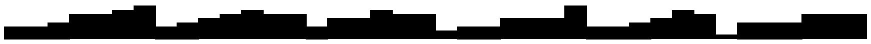
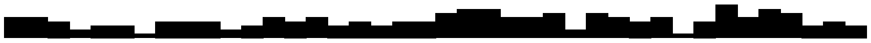
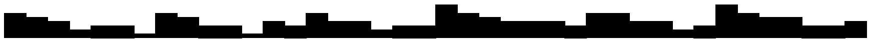
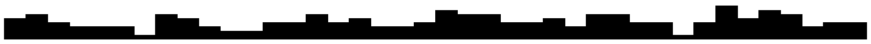
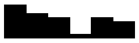
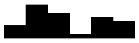
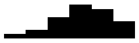
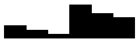
Train - loss	2.47857
Train - top-1	0.48715
Train - top-10	0.76385
Train - top-5	0.69236
Validation - loss	2.74602
Validation - top-1	0.47018
Validation - top-10	0.73715
Validation - top-5	0.66607

View run **twilight-sweep-8** at: https://wandb.ai/8225_team_/INF8225%20-%20TP3/runs/zepyrp0f
Synced 5 W&B file(s), 1 media file(s), 3 artifact file(s) and 1 other file(s)
Find logs at: `./wandb/run-20230410_065222-zepyrp0f/logs`

```
wandb: Agent Starting Run: xaj0a64f with config:
wandb:   batch_size: 256
```

```
wandb:   clip: 7
wandb:   dim_embedding: 493
wandb:   dim_hidden: 422
wandb:   dropout: 0.19391275354122212
wandb:   lr: 0.000645164751201379
wandb:   n_heads: 3
wandb:   n_layers: 3
wandb: WARNING Ignored wandb.init() arg project when running a sweep.
Tracking run with wandb version 0.14.2
Run data is saved locally in /content/wandb/run-20230410_065827-xaj0a64f
Syncing run treasured-sweep-9 to Weights & Biases (docs)
Sweep page: https://wandb.ai/8225_team_/INF8225%20-%20TP3/sweeps/1t5pcxwf
View project at https://wandb.ai/8225_team_/INF8225%20-%20TP3
View sweep at https://wandb.ai/8225_team_/INF8225%20-%20TP3/sweeps/1t5pcxwf
View run at https://wandb.ai/8225_team_/INF8225%20-%20TP3/runs/xaj0a64f
Starting training for 6 epochs, using cuda.

Epoch 1
Train -    loss: 2.49     top-1: 0.49    top-5: 0.69    top-10: 0.76
Eval -     loss: 2.74     top-1: 0.47    top-5: 0.67    top-10: 0.74
I'm skinny.
Je suis désolée.

Epoch 2
Train -    loss: 2.46     top-1: 0.49    top-5: 0.70    top-10: 0.77
Eval -     loss: 2.74     top-1: 0.47    top-5: 0.67    top-10: 0.74
That's a lame excuse.
Il nous faut partir.

Epoch 3
Train -    loss: 2.49     top-1: 0.49    top-5: 0.69    top-10: 0.76
Eval -     loss: 2.74     top-1: 0.47    top-5: 0.67    top-10: 0.74
I know now what we have to do.
Je sais que Tom est parti.

Epoch 4
Train -    loss: 2.48     top-1: 0.49    top-5: 0.69    top-10: 0.77
Eval -     loss: 2.74     top-1: 0.47    top-5: 0.67    top-10: 0.74
Life without books is unimaginable.
Une minute.

Epoch 5
Train -    loss: 2.49     top-1: 0.49    top-5: 0.69    top-10: 0.76
Eval -     loss: 2.74     top-1: 0.47    top-5: 0.67    top-10: 0.74
I told you the truth.
Je m'appelle Tom.

Epoch 6
Train -    loss: 2.44     top-1: 0.50    top-5: 0.70    top-10: 0.77
Eval -     loss: 2.74     top-1: 0.47    top-5: 0.67    top-10: 0.74
Who knows?
Tu rentres ?
Waiting for W&B process to finish... (success).
```

Run history:

Train - loss	
Train - top-1	
Train - top-10	
Train - top-5	
Validation - loss	
Validation - top-1	
Validation - top-10	
Validation - top-5	

Run summary:

Train - loss	2.4434
Train - top-1	0.49629
Train - top-10	0.76742
Train - top-5	0.69725
Validation - loss	2.7391
Validation - top-1	0.47063
Validation - top-10	0.73844
Validation - top-5	0.66792

```
View run treasured-sweep-9 at: https://wandb.ai/8225_team_/INF8225%20-%20TP3/runs/xaj0a64f
Synced 5 W&B file(s), 1 media file(s), 3 artifact file(s) and 1 other file(s)
Find logs at: ./wandb/run-20230410_065827-xaj0a64f/logs
wandb: Agent Starting Run: s5ilk9hp with config:
wandb:   batch_size: 256
wandb:   clip: 9
wandb:   dim_embedding: 402
wandb:   dim_hidden: 144
wandb:   dropout: 0.2394009804916092
wandb:   lr: 0.0009383851854113872
wandb:   n_heads: 11
wandb:   n_layers: 8
wandb: WARNING Ignored wandb.init() arg project when running a sweep.
Tracking run with wandb version 0.14.2
Run data is saved locally in /content/wandb/run-20230410_070431-s5ilk9hp
```

Syncing run **ethereal-sweep-10** to Weights & Biases (docs)
Sweep page: https://wandb.ai/8225_team_/INF8225%20-%20TP3/sweeps/1t5pcxwf
View project at https://wandb.ai/8225_team_/INF8225%20-%20TP3
View sweep at https://wandb.ai/8225_team_/INF8225%20-%20TP3/sweeps/1t5pcxwf
View run at https://wandb.ai/8225_team_/INF8225%20-%20TP3/runs/s5ilk9hp

```
Starting training for 6 epochs, using cuda.

Epoch 1
Train -   loss: 2.47    top-1: 0.49   top-5: 0.69   top-10: 0.76
Eval -    loss: 2.74    top-1: 0.47   top-5: 0.67   top-10: 0.74
I rewrote it.
J'ai fait ça.

Epoch 2
Train -   loss: 2.43    top-1: 0.50   top-5: 0.70   top-10: 0.77
Eval -    loss: 2.74    top-1: 0.47   top-5: 0.67   top-10: 0.74
Tom lived in Boston for a long time.
Tom a l'air surpris.

Epoch 3
Train -   loss: 2.43    top-1: 0.50   top-5: 0.70   top-10: 0.77
Eval -    loss: 2.73    top-1: 0.47   top-5: 0.67   top-10: 0.74
I can hear you, but I can't see you.
Je suis désolé.

Epoch 4
Train -   loss: 2.43    top-1: 0.50   top-5: 0.70   top-10: 0.77
Eval -    loss: 2.74    top-1: 0.47   top-5: 0.67   top-10: 0.74
I kind of liked them.
Je jouais au football.

Epoch 5
Train -   loss: 2.41    top-1: 0.50   top-5: 0.70   top-10: 0.77
Eval -    loss: 2.73    top-1: 0.47   top-5: 0.67   top-10: 0.74
If it's possible, I want to go home now.
Si tu te trompes.

Epoch 6
Train -   loss: 2.43    top-1: 0.49   top-5: 0.70   top-10: 0.77
Eval -    loss: 2.74    top-1: 0.47   top-5: 0.67   top-10: 0.74
Tickets are $30, parking is free and children under ten receive free admissio
Une tasse.
```

Waiting for W&B process to finish... **(success).**

Run history:

Train - loss
Train - top-1
Train - top-10
Train - top-5
Validation

- loss	
Validation - top-1	
Validation - top-10	
Validation - top-5	

Run summary:

Train - loss	2.4344
Train - top-1	0.49492
Train - top-10	0.77142
Train - top-5	0.69922
Validation - loss	2.73602
Validation - top-1	0.47396
Validation - top-10	0.7407
Validation - top-5	0.67019

View run **ethereal-sweep-10** at: https://wandb.ai/8225_team_/INF8225%20-%20TP3/runs/s5ilk9hp
Synced 5 W&B file(s), 1 media file(s), 3 artifact file(s) and 1 other file(s)
Find logs at: `./wandb/run-20230410_070431-s5ilk9hp/logs`

```
wandb: Agent Starting Run: ue80qxhc with config:
wandb:   batch_size: 512
wandb:   clip: 12
wandb:   dim_embedding: 451
wandb:   dim_hidden: 428
wandb:   dropout: 0.2263208499636016
wandb:   lr: 0.0006378170606560637
wandb:   n_heads: 10
wandb:   n_layers: 8
wandb: WARNING Ignored wandb.init() arg project when running a sweep.
```

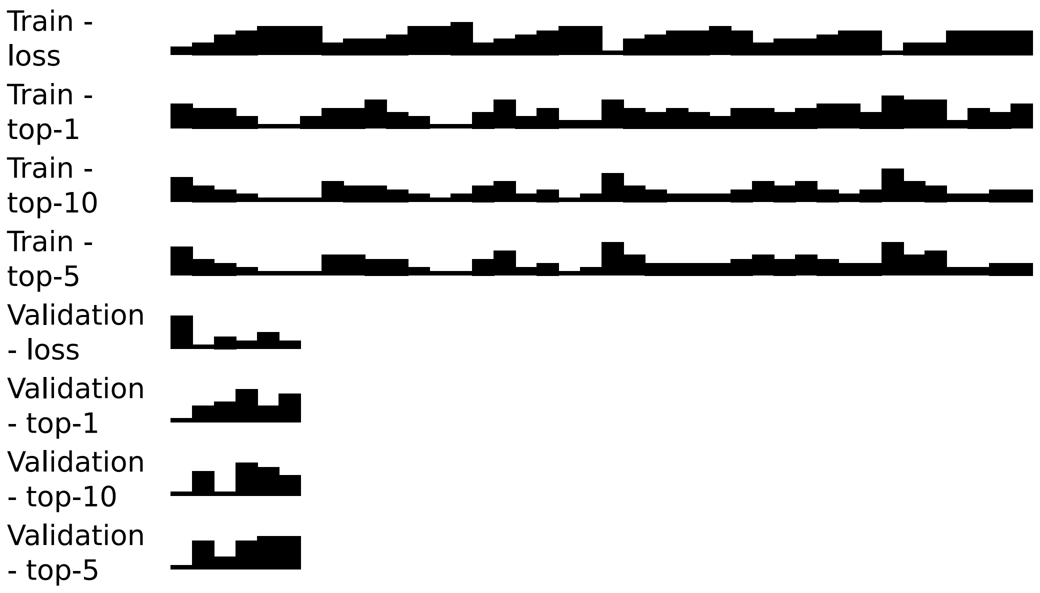
Tracking run with wandb version 0.14.2
Run data is saved locally in `/content/wandb/run-20230410_071036-ue80qxhc`
Syncing run **good-sweep-11** to Weights & Biases (docs)
Sweep page: https://wandb.ai/8225_team_/INF8225%20-%20TP3/sweeps/1t5pcxwf
View project at https://wandb.ai/8225_team_/INF8225%20-%20TP3
View sweep at https://wandb.ai/8225_team_/INF8225%20-%20TP3/sweeps/1t5pcxwf
View run at https://wandb.ai/8225_team_/INF8225%20-%20TP3/runs/ue80qxhc

```
Starting training for 6 epochs, using cuda.

Epoch 1
Train -    loss: 2.44     top-1: 0.50    top-5: 0.70    top-10: 0.77
Eval -     loss: 2.74     top-1: 0.47    top-5: 0.67    top-10: 0.74
I don't want to talk about that right now
```

```
I don't want to talk about that right now.
Je ne veux pas faire ça.

Epoch 2
Train -   loss: 2.44     top-1: 0.50    top-5: 0.70    top-10: 0.77
Eval -    loss: 2.73     top-1: 0.47    top-5: 0.67    top-10: 0.74
I know that Japanese songs are very difficult for us.
Je sais tout.

Epoch 3
Train -   loss: 2.39     top-1: 0.50    top-5: 0.71    top-10: 0.78
Eval -    loss: 2.73     top-1: 0.48    top-5: 0.67    top-10: 0.74
Do you have a pet?
Vous êtes-vous ?

Epoch 4
Train -   loss: 2.39     top-1: 0.50    top-5: 0.71    top-10: 0.77
Eval -    loss: 2.73     top-1: 0.48    top-5: 0.67    top-10: 0.74
We don't need you anymore.
Vos cheveux sont beaux.

Epoch 5
Train -   loss: 2.41     top-1: 0.50    top-5: 0.70    top-10: 0.78
Eval -    loss: 2.73     top-1: 0.47    top-5: 0.67    top-10: 0.74
I ran away from home when I was thirteen years old.
J'étudie le français.

Epoch 6
Train -   loss: 2.40     top-1: 0.50    top-5: 0.70    top-10: 0.77
Eval -    loss: 2.73     top-1: 0.48    top-5: 0.67    top-10: 0.74
I made him go.
Je vous verrai demain.
```

Waiting for W&B process to finish... **(success).**

Run history:



Run summary:

Train - loss	2.40289
Train - top-1	0.50154
Train - top-10	0.77374

```
wandb.agent("INF8225 - TP3/kob3h22b", train, count=10) # GRU
```

```
wandb: Agent Starting Run: touyswa2 with config:
wandb:  batch_size: 512
wandb:  clip: 10
wandb:  dim_embedding: 360
wandb:  dim_hidden: 303
wandb:  dropout: 0.17781868034215215
wandb:  lr: 0.0007024670687252072
wandb:  n_heads: 9
wandb:  n_layers: 4
wandb: WARNING Ignored wandb.init() arg project when running a sweep.
Tracking run with wandb version 0.14.2
Run data is saved locally in /content/wandb/run-20230410_072838-touyswa2
Syncing run efficient-sweep-1 to Weights & Biases (docs)
Sweep page: https://wandb.ai/8225_team_/INF8225%20-%20TP3/sweeps/kob3h22b
View project at https://wandb.ai/8225_team_/INF8225%20-%20TP3
View sweep at https://wandb.ai/8225_team_/INF8225%20-%20TP3/sweeps/kob3h22b
View run at https://wandb.ai/8225_team_/INF8225%20-%20TP3/runs/touyswa2
Starting training for 6 epochs, using cuda.

Epoch 1
Train -   loss: 3.09    top-1: 0.44    top-5: 0.62    top-10: 0.69
Eval -    loss: 3.05    top-1: 0.44    top-5: 0.62    top-10: 0.69
She asked him to give her some money.
Elle m'a demandé à la maison.

Epoch 2
Train -   loss: 2.79    top-1: 0.48    top-5: 0.66    top-10: 0.72
Eval -    loss: 2.76    top-1: 0.47    top-5: 0.66    top-10: 0.72
Whether we succeed or not, we have to do our best.
Les gens ne m'ont pas dit à Tom de faire ça.

Epoch 3
Train -   loss: 2.59    top-1: 0.50    top-5: 0.69    top-10: 0.75
Eval -    loss: 2.61    top-1: 0.49    top-5: 0.68    top-10: 0.74
I could've done better, I think.
Je lui ai demandé à Tom.

Epoch 4
Train -   loss: 2.48    top-1: 0.51    top-5: 0.70    top-10: 0.76
Eval -    loss: 2.51    top-1: 0.51    top-5: 0.70    top-10: 0.76
Neither of my brothers will be there.
Mes parents sont à Tom.

Epoch 5
```

```
Train -    loss: 2.36     top-1: 0.53    top-5: 0.72    top-10: 0.78
Eval -     loss: 2.43     top-1: 0.52    top-5: 0.71    top-10: 0.77
She began to sing.
Elle est malade.

Epoch 6
Train -    loss: 2.29     top-1: 0.54    top-5: 0.73    top-10: 0.79
Eval -     loss: 2.37     top-1: 0.53    top-5: 0.72    top-10: 0.78
How many letters are there in the English alphabet?
Combien de temps es-tu à Boston ?
```

Waiting for W&B process to finish... **(success).**

Run history:

Train - loss	
Train - top-1	
Train - top-10	
Train - top-5	
Validation - loss	
Validation - top-1	
Validation - top-10	
Validation - top-5	

Run summary:

Train - loss	2.28725
Train - top-1	0.53545
Train - top-10	0.7887
Train - top-5	0.73003
Validation - loss	2.36853
Validation - top-1	0.52694
Validation - top-10	0.77958
Validation - top-5	0.71928

View run **efficient-sweep-1** at: https://wandb.ai/8225_team_/INF8225%20-%20TP3/runs/touyswa2
Synced 5 W&B file(s), 1 media file(s), 3 artifact file(s) and 1 other file(s)
Find logs at: ./wandb/run-20230410_072838-touyswa2/logs

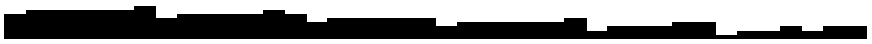
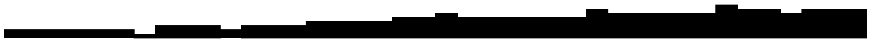
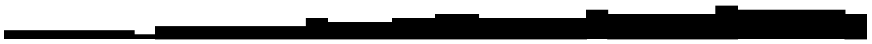
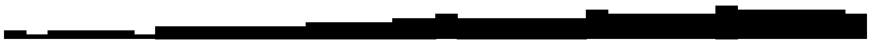
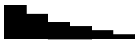
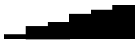
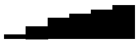
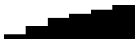
```
wandb: Agent Starting Run: lfbtmukp with config:
wandb:   batch_size: 256
wandb:   clip: 8
wandb:   dim_embedding: 731
wandb:   dim_hidden: 534
wandb:   dropout: 0.12045946109992596
wandb:   lr: 0.0008779021020786119
wandb:   n_heads: 8
wandb:   n_layers: 4
wandb: WARNING Ignored wandb.init() arg project when running a sweep.
```

Tracking run with wandb version 0.14.2
Run data is saved locally in /content/wandb/run-20230410_073712-lfbtmukp
Syncing run **valiant-sweep-2** to Weights & Biases (docs)
Sweep page: https://wandb.ai/8225_team_/INF8225%20-%20TP3/sweeps/kob3h22b
View project at https://wandb.ai/8225_team_/INF8225%20-%20TP3
View sweep at https://wandb.ai/8225_team_/INF8225%20-%20TP3/sweeps/kob3h22b
View run at https://wandb.ai/8225_team_/INF8225%20-%20TP3/runs/lfbtmukp

```
Starting training for 6 epochs, using cuda.

Epoch 1
Train -   loss: 2.23     top-1: 0.55    top-5: 0.74    top-10: 0.80
Eval -    loss: 2.32     top-1: 0.53    top-5: 0.73    top-10: 0.79
Were you listening to the radio yesterday?
T'es-tu allé à la maison ?

Epoch 2
Train -   loss: 2.17     top-1: 0.55    top-5: 0.75    top-10: 0.81
Eval -    loss: 2.26     top-1: 0.54    top-5: 0.74    top-10: 0.80
Please turn over.
Prenez place, s'il vous plaît.

Epoch 3
Train -   loss: 2.16     top-1: 0.55    top-5: 0.75    top-10: 0.81
Eval -    loss: 2.23     top-1: 0.55    top-5: 0.74    top-10: 0.80
I'd like you to make one now.
J'aimerais que tu sois là.

Epoch 4
Train -   loss: 2.09     top-1: 0.56    top-5: 0.76    top-10: 0.82
Eval -    loss: 2.19     top-1: 0.55    top-5: 0.75    top-10: 0.81
My right leg hurts.
Ma lampe de la maison.

Epoch 5
Train -   loss: 2.04     top-1: 0.57    top-5: 0.77    top-10: 0.83
Eval -    loss: 2.16     top-1: 0.56    top-5: 0.75    top-10: 0.81
You shouldn't have paid the bill.
Tu n'aurais pas dû téléphoner.

Epoch 6
Train -   loss: 1.97     top-1: 0.58    top-5: 0.78    top-10: 0.84
Eval -    loss: 2.14     top-1: 0.56    top-5: 0.76    top-10: 0.81
Birch trees have white bark.
Des arbres portaient de la colline.
```

Waiting for W&B process to finish... **(success).**

Run history:

Train - loss	
Train - top-1	
Train - top-10	
Train - top-5	
Validation - loss	
Validation - top-1	
Validation - top-10	
Validation - top-5	

Run summary:

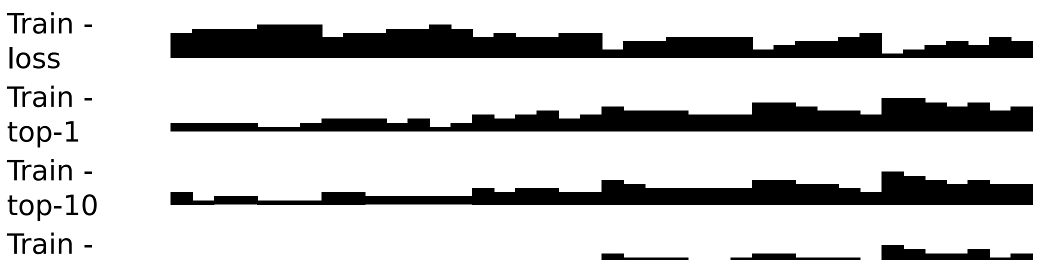
Train - loss	1.97076
Train - top-1	0.57857
Train - top-10	0.83682
Train - top-5	0.78123
Validation - loss	2.13888
Validation - top-1	0.56052
Validation - top-10	0.81467
Validation - top-5	0.75758

View run **valiant-sweep-2** at: https://wandb.ai/8225_team_/INF8225%20-%20TP3/runs/lfbtmukp
Synced 5 W&B file(s), 1 media file(s), 3 artifact file(s) and 1 other file(s)
Find logs at: `./wandb/run-20230410_073712-lfbtmukp/logs`

```
wandb: Agent Starting Run: cbb2ybco with config:
wandb:   batch_size: 512
wandb:   clip: 12
wandb:   dim_embedding: 656
wandb:   dim_hidden: 908
wandb:   dropout: 0.12753689232983317
wandb:   lr: 0.0008649158741300643
wandb:   n_heads: 10
wandb:   n_layers: 7
wandb: WARNING Ignored wandb.init() arg project when running a sweep.
```

```
Tracking run with wandb version 0.14.2
Run data is saved locally in /content/wandb/run-20230410_074547-cbb2ybco
Syncing run stoic-sweep-3 to Weights & Biases (docs)
Sweep page: https://wandb.ai/8225_team_/INF8225%20-%20TP3/sweeps/kob3h22b
View project at https://wandb.ai/8225_team_/INF8225%20-%20TP3
View sweep at https://wandb.ai/8225_team_/INF8225%20-%20TP3/sweeps/kob3h22b
View run at https://wandb.ai/8225_team_/INF8225%20-%20TP3/runs/cbb2ybco
Starting training for 6 epochs, using cuda.

Epoch 1
Train -    loss: 1.95     top-1: 0.58    top-5: 0.78    top-10: 0.84
Eval -     loss: 2.11     top-1: 0.56    top-5: 0.76    top-10: 0.82
Tom lives and works in Boston.
Tom habite jusqu'à Boston.

Epoch 2
Train -    loss: 1.90     top-1: 0.59    top-5: 0.79    top-10: 0.84
Eval -     loss: 2.10     top-1: 0.57    top-5: 0.76    top-10: 0.82
We thought that you were married.
Nous pensions que tu étais surpris.

Epoch 3
Train -    loss: 1.88     top-1: 0.59    top-5: 0.79    top-10: 0.85
Eval -     loss: 2.08     top-1: 0.57    top-5: 0.77    top-10: 0.82
I hope you're well paid.
J'espère que vous avez faim.

Epoch 4
Train -    loss: 1.82     top-1: 0.59    top-5: 0.80    top-10: 0.85
Eval -     loss: 2.07     top-1: 0.57    top-5: 0.77    top-10: 0.83
I rewrote it.
Je suis resté silencieux.

Epoch 5
Train -    loss: 1.83     top-1: 0.59    top-5: 0.80    top-10: 0.85
Eval -     loss: 2.04     top-1: 0.58    top-5: 0.78    top-10: 0.83
We all have kids.
Nous avons toutes nos enfants.

Epoch 6
Train -    loss: 1.81     top-1: 0.60    top-5: 0.80    top-10: 0.86
Eval -     loss: 2.03     top-1: 0.58    top-5: 0.78    top-10: 0.83
Let's not act rashly.
Ne bouge pas.
Waiting for W&B process to finish... (success).
```

Run history:



top-5	
Validation - loss	
Validation - top-1	
Validation - top-10	
Validation - top-5	

Run summary:

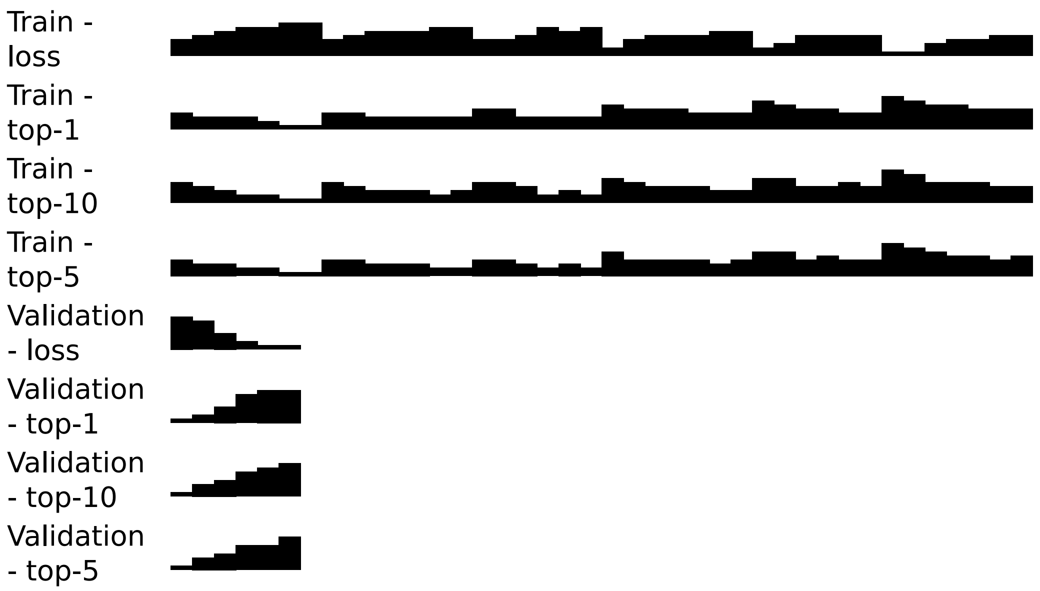
Train - loss	1.811
Train - top-1	0.59774
Train - top-10	0.85682
Train - top-5	0.80187
Validation - loss	2.03014
Validation - top-1	0.57855
Validation - top-10	0.83191
Validation - top-5	0.77738

```
View run stoic-sweep-3 at: https://wandb.ai/8225_team_/INF8225%20-%20TP3
/runs/cbb2ybco
Synced 5 W&B file(s), 1 media file(s), 3 artifact file(s) and 1 other file(s)
Find logs at: ./wandb/run-20230410_074547-cbb2ybco/logs
wandb: Agent Starting Run: pkqkwsiu with config:
wandb:   batch_size: 256
wandb:   clip: 4
wandb:   dim_embedding: 370
wandb:   dim_hidden: 634
wandb:   dropout: 0.27825276330947457
wandb:   lr: 0.0009833031838853794
wandb:   n_heads: 6
wandb:   n_layers: 8
wandb: WARNING Ignored wandb.init() arg project when running a sweep.
Tracking run with wandb version 0.14.2
Run data is saved locally in /content/wandb/run-20230410_075422-pkqkwsiu
Syncing run ancient-sweep-4 to Weights & Biases (docs)
Sweep page: https://wandb.ai/8225_team_/INF8225%20-%20TP3/sweeps/kob3h22b
View project at https://wandb.ai/8225_team_/INF8225%20-%20TP3
View sweep at https://wandb.ai/8225_team_/INF8225%20-%20TP3/sweeps/kob3h22b
View run at https://wandb.ai/8225_team_/INF8225%20-%20TP3/runs/pkqkwsiu
Starting training for 6 epochs, using cuda.

Epoch 1
Train -    loss: 1.78     top-1: 0.60    top-5: 0.80    top-10: 0.86
```

```
Eval -    loss: 2.02     top-1: 0.58    top-5: 0.78    top-10: 0.83
We'll be there in less than three hours.
Nous nous reverrons trois jours.

Epoch 2
Train -   loss: 1.74     top-1: 0.61    top-5: 0.81    top-10: 0.87
Eval -    loss: 2.01     top-1: 0.58    top-5: 0.78    top-10: 0.84
Tom's boss advanced him a week's wages.
Le chat de Tom a eu une attaque cardiaque.

Epoch 3
Train -   loss: 1.77     top-1: 0.60    top-5: 0.81    top-10: 0.86
Eval -    loss: 2.00     top-1: 0.58    top-5: 0.78    top-10: 0.84
I can't find my watch.
Je ne peux pas vendre mon appartement.

Epoch 4
Train -   loss: 1.70     top-1: 0.62    top-5: 0.82    top-10: 0.87
Eval -    loss: 1.98     top-1: 0.59    top-5: 0.79    top-10: 0.84
It's incredibly easy to cheat the system.
C'est intéressant.

Epoch 5
Train -   loss: 1.70     top-1: 0.62    top-5: 0.82    top-10: 0.87
Eval -    loss: 1.98     top-1: 0.59    top-5: 0.79    top-10: 0.84
We don't care what he does.
Nous ne prêtes pas quoi faire.

Epoch 6
Train -   loss: 1.67     top-1: 0.62    top-5: 0.82    top-10: 0.87
Eval -    loss: 1.98     top-1: 0.59    top-5: 0.79    top-10: 0.84
I slept a little during lunch break because I was so tired.
J'ai dormi trop froid hier soir.
```

Waiting for W&B process to finish... **(success).**

Run history:



Run summary:

Train - loss	1.66716
Train - top-1	0.62014
Train - top-10	0.87373
Train - top-5	0.82337
Validation - loss	1.97705
Validation - top-1	0.58895
Validation - top-10	0.84041
Validation - top-5	0.78763

View run **ancient-sweep-4** at: https://wandb.ai/8225_team_/INF8225%20-%20TP3/runs/pkqkwsiu
Synced 5 W&B file(s), 1 media file(s), 3 artifact file(s) and 1 other file(s)
Find logs at: `./wandb/run-20230410_075422-pkqkwsiu/logs`

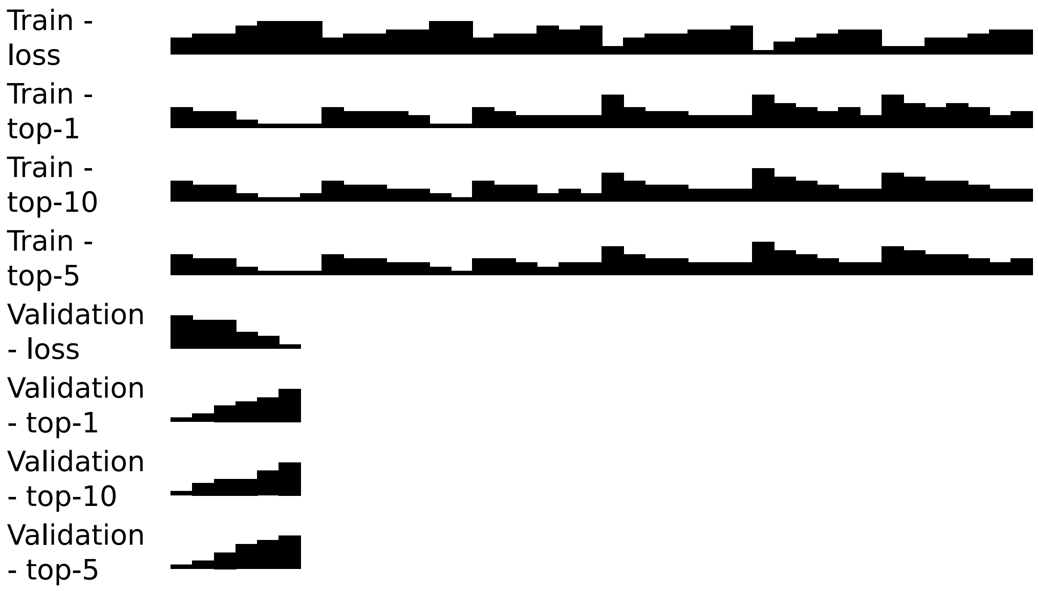
```
wandb: Agent Starting Run: bynjied2 with config:
wandb:   batch_size: 128
wandb:   clip: 11
wandb:   dim_embedding: 457
wandb:   dim_hidden: 73
wandb:   dropout: 0.22992617160098855
wandb:   lr: 0.0006994117711121344
wandb:   n_heads: 10
wandb:   n_layers: 7
wandb: WARNING Ignored wandb.init() arg project when running a sweep.
```

Tracking run with wandb version 0.14.2
Run data is saved locally in `/content/wandb/run-20230410_080306-bynjied2`
Syncing run **deft-sweep-5** to Weights & Biases (docs)
Sweep page: https://wandb.ai/8225_team_/INF8225%20-%20TP3/sweeps/kob3h22b
View project at https://wandb.ai/8225_team_/INF8225%20-%20TP3
View sweep at https://wandb.ai/8225_team_/INF8225%20-%20TP3/sweeps/kob3h22b
View run at https://wandb.ai/8225_team_/INF8225%20-%20TP3/runs/bynjied2

```
Starting training for 6 epochs, using cuda.

Epoch 1
Train -    loss: 1.64     top-1: 0.63    top-5: 0.83    top-10: 0.88
Eval -     loss: 1.96     top-1: 0.59    top-5: 0.79    top-10: 0.84
I think everything is functional.
Je pense que tout va bien.

Epoch 2
Train -    loss: 1.66     top-1: 0.62    top-5: 0.82    top-10: 0.87
Eval -     loss: 1.96     top-1: 0.59    top-5: 0.79    top-10: 0.84
Go now.
mieux maintenant.
```

```
Epoch 3
Train -   loss: 1.63    top-1: 0.63    top-5: 0.83    top-10: 0.88
Eval -    loss: 1.96    top-1: 0.59    top-5: 0.79    top-10: 0.84
He became a policeman.
Il devint professeur.

Epoch 4
Train -   loss: 1.62    top-1: 0.63    top-5: 0.83    top-10: 0.88
Eval -    loss: 1.95    top-1: 0.59    top-5: 0.79    top-10: 0.84
I'll be late for school!
Je serai de retour tôt. »

Epoch 5
Train -   loss: 1.58    top-1: 0.63    top-5: 0.83    top-10: 0.88
Eval -    loss: 1.94    top-1: 0.60    top-5: 0.79    top-10: 0.85
He was among those chosen.
Il a été complètement parvenu.

Epoch 6
Train -   loss: 1.59    top-1: 0.63    top-5: 0.83    top-10: 0.88
Eval -    loss: 1.93    top-1: 0.60    top-5: 0.80    top-10: 0.85
I don't know what I've been so afraid of.
Je ne sais pas ce que Tom sera heureux.
Waiting for W&B process to finish... (success).
```

Run history:



Run summary:

Train - loss	1.58603
Train - top-1	0.63178
Train - top-10	0.88436
Train -	0.83487

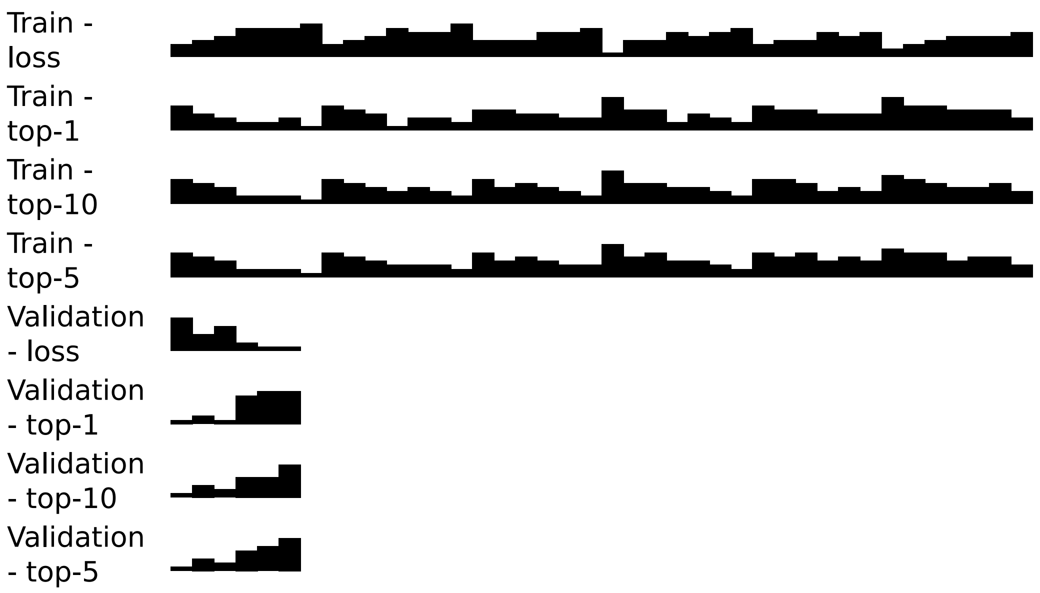
top-5	
Validation - loss	1.935
Validation - top-1	0.59726
Validation - top-10	0.84754
Validation - top-5	0.79509

```
View run deft-sweep-5 at: https://wandb.ai/8225_team_/INF8225%20-%20TP3
/runs/bynjied2
Synced 5 W&B file(s), 1 media file(s), 3 artifact file(s) and 1 other file(s)
Find logs at: ./wandb/run-20230410_080306-bynjied2/logs
wandb: Agent Starting Run: n78tw4wd with config:
wandb:   batch_size: 128
wandb:   clip: 7
wandb:   dim_embedding: 700
wandb:   dim_hidden: 289
wandb:   dropout: 0.13130109068684753
wandb:   lr: 0.0009352356160543272
wandb:   n_heads: 3
wandb:   n_layers: 6
wandb: WARNING Ignored wandb.init() arg project when running a sweep.
Tracking run with wandb version 0.14.2
Run data is saved locally in /content/wandb/run-20230410_081149-n78tw4wd
Syncing run good-sweep-6 to Weights & Biases (docs)
Sweep page: https://wandb.ai/8225_team_/INF8225%20-%20TP3/sweeps/kob3h22b
View project at https://wandb.ai/8225_team_/INF8225%20-%20TP3
View sweep at https://wandb.ai/8225_team_/INF8225%20-%20TP3/sweeps/kob3h22b
View run at https://wandb.ai/8225_team_/INF8225%20-%20TP3/runs/n78tw4wd
Starting training for 6 epochs, using cuda.

Epoch 1
Train -   loss: 1.55     top-1: 0.64    top-5: 0.84    top-10: 0.89
Eval -    loss: 1.94     top-1: 0.60    top-5: 0.80    top-10: 0.85
I have to get up anyways.
Il me faut être arrêté hier.

Epoch 2
Train -   loss: 1.59     top-1: 0.63    top-5: 0.84    top-10: 0.89
Eval -    loss: 1.92     top-1: 0.60    top-5: 0.80    top-10: 0.85
Do you need this book?
Avez-vous besoin de ce livre ?

Epoch 3
Train -   loss: 1.54     top-1: 0.64    top-5: 0.84    top-10: 0.89
Eval -    loss: 1.93     top-1: 0.60    top-5: 0.80    top-10: 0.85
I live in a small fishing village.
Je vis dans une petite île.

Epoch 4
Train -   loss: 1.56     top-1: 0.64    top-5: 0.84    top-10: 0.89
Eval -    loss: 1.92     top-1: 0.60    top-5: 0.80    top-10: 0.85
I'd appreciate your help.
J'ai apprécié votre aide.
```

```
J'ai apprécié votre aide.

Epoch 5
Train -    loss: 1.55     top-1: 0.64    top-5: 0.84    top-10: 0.89
Eval -     loss: 1.92     top-1: 0.60    top-5: 0.80    top-10: 0.85
While traveling in Europe, I was pickpocketed on a train.
En temps, je me frappe à sept heures.

Epoch 6
Train -    loss: 1.52     top-1: 0.64    top-5: 0.84    top-10: 0.89
Eval -     loss: 1.91     top-1: 0.60    top-5: 0.80    top-10: 0.85
I know what to expect.
Je sais ce que nous pouvons.
```

Waiting for W&B process to finish... **(success).**

Run history:



Run summary:

Train - loss	1.51933
Train - top-1	0.64206
Train - top-10	0.89156
Train - top-5	0.84346
Validation - loss	1.91465
Validation - top-1	0.60205
Validation - top-10	0.85042
Validation - top-5	0.79959

View run good-sweep-6 at: https://wandb.ai/8225_team_/INF8225%20-%20TP3

```
/runs/n78tw4wd
Synced 5 W&B file(s), 1 media file(s), 3 artifact file(s) and 1 other file(s)
Find logs at: ./wandb/run-20230410_081149-n78tw4wd/logs
wandb: Agent Starting Run: jcgivjoq with config:
wandb:   batch_size: 128
wandb:   clip: 12
wandb:   dim_embedding: 651
wandb:   dim_hidden: 481
wandb:   dropout: 0.07011145463813385
wandb:   lr: 0.0006600273023362729
wandb:   n_heads: 7
wandb:   n_layers: 9
wandb: WARNING Ignored wandb.init() arg project when running a sweep.
Tracking run with wandb version 0.14.2
Run data is saved locally in /content/wandb/run-20230410_082033-jcgivjoq
Syncing run dutiful-sweep-7 to Weights & Biases (docs)
Sweep page: https://wandb.ai/8225_team_/INF8225%20-%20TP3/sweeps/kob3h22b
View project at https://wandb.ai/8225_team_/INF8225%20-%20TP3
View sweep at https://wandb.ai/8225_team_/INF8225%20-%20TP3/sweeps/kob3h22b
View run at https://wandb.ai/8225_team_/INF8225%20-%20TP3/runs/jcgivjoq
Starting training for 6 epochs, using cuda.

Epoch 1
Train -   loss: 1.51     top-1: 0.65    top-5: 0.84    top-10: 0.89
Eval -    loss: 1.91     top-1: 0.60    top-5: 0.80    top-10: 0.85
He lied about his age.
Il a menti à son âge.

Epoch 2
Train -   loss: 1.49     top-1: 0.65    top-5: 0.85    top-10: 0.89
Eval -    loss: 1.91     top-1: 0.60    top-5: 0.80    top-10: 0.85
I can't understand this word.
Je ne comprends pas ce mot.

Epoch 3
Train -   loss: 1.50     top-1: 0.65    top-5: 0.85    top-10: 0.89
Eval -    loss: 1.91     top-1: 0.60    top-5: 0.80    top-10: 0.85
Thanks to all of you.
Merci pour tout.

Epoch 4
Train -   loss: 1.50     top-1: 0.65    top-5: 0.85    top-10: 0.89
Eval -    loss: 1.90     top-1: 0.61    top-5: 0.80    top-10: 0.85
Last summer, I had a chance to go to Boston, but I didn't go.
La nuit dernière, j'ai pu me rendre visite à Tom demain.

Epoch 5
Train -   loss: 1.44     top-1: 0.66    top-5: 0.85    top-10: 0.90
Eval -    loss: 1.90     top-1: 0.61    top-5: 0.80    top-10: 0.85
Tom and Mary decorated their house for Christmas.
Tom et Mary ont construit la veille de Noël.

Epoch 6
Train -   loss: 1.49     top-1: 0.65    top-5: 0.85    top-10: 0.90
Eval -    loss: 1.90     top-1: 0.61    top-5: 0.80    top-10: 0.85
```

```
Get to the point.
Sors de moi.
```

Waiting for W&B process to finish... **(success).**

Run history:

Train - loss	
Train - top-1	
Train - top-10	
Train - top-5	
Validation - loss	
Validation - top-1	
Validation - top-10	
Validation - top-5	

Run summary:

Train - loss	1.48706
Train - top-1	0.65022
Train - top-10	0.89503
Train - top-5	0.84735
Validation - loss	1.90222
Validation - top-1	0.60517
Validation - top-10	0.85285
Validation - top-5	0.80298

View run **dutiful-sweep-7** at: https://wandb.ai/8225_team_/INF8225%20-%20TP3/runs/jcgivjoq
Synced 5 W&B file(s), 1 media file(s), 3 artifact file(s) and 1 other file(s)
Find logs at: `./wandb/run-20230410_082033-jcgivjoq/logs`

```
wandb: Sweep Agent: Waiting for job.
wandb: Job received.
wandb: Agent Starting Run: svwo7xmb with config:
wandb:   batch_size: 128
wandb:   clip: 8
wandb:   dim_embedding: 577
wandb:   dim_hidden: 575
```

```
wandb:   dropout: 0.2536167158228334
wandb:   lr: 0.000770141803485162
wandb:   n_heads: 11
wandb:   n_layers: 6
wandb: WARNING Ignored wandb.init() arg project when running a sweep.
```
Tracking run with wandb version 0.14.2
Run data is saved locally in `/content/wandb/run-20230410_082922-svwo7xmb`
Syncing run **stoic-sweep-8** to Weights & Biases (docs)
Sweep page: https://wandb.ai/8225_team_/INF8225%20-%20TP3/sweeps/kob3h22b
View project at https://wandb.ai/8225_team_/INF8225%20-%20TP3
View sweep at https://wandb.ai/8225_team_/INF8225%20-%20TP3/sweeps/kob3h22b
View run at https://wandb.ai/8225_team_/INF8225%20-%20TP3/runs/svwo7xmb
```
Starting training for 6 epochs, using cuda.

Epoch 1
Train -    loss: 1.46     top-1: 0.65    top-5: 0.85    top-10: 0.90
Eval -     loss: 1.90     top-1: 0.61    top-5: 0.80    top-10: 0.85
I had to wait for Tom to finish.
J'ai essayé de sortir avec Tom.

Epoch 2
Train -    loss: 1.46     top-1: 0.65    top-5: 0.85    top-10: 0.90
Eval -     loss: 1.89     top-1: 0.61    top-5: 0.80    top-10: 0.85
Watching TV is a passive activity.
Le musée est un enfant gâté.

Epoch 3
Train -    loss: 1.47     top-1: 0.65    top-5: 0.85    top-10: 0.90
Eval -     loss: 1.89     top-1: 0.61    top-5: 0.80    top-10: 0.85
I wonder why women don't go bald.
Je me demande pourquoi nous sommes morts.

Epoch 4
Train -    loss: 1.45     top-1: 0.65    top-5: 0.85    top-10: 0.90
Eval -     loss: 1.89     top-1: 0.61    top-5: 0.81    top-10: 0.85
I found the picture you were looking for.
J'ai trouvé la photo que tu cherchais.

Epoch 5
Train -    loss: 1.44     top-1: 0.66    top-5: 0.85    top-10: 0.90
Eval -     loss: 1.89     top-1: 0.61    top-5: 0.81    top-10: 0.86
I don't have to apologize for what I said.
Je n'ai pas peur d'aider Tom.

Epoch 6
Train -    loss: 1.44     top-1: 0.66    top-5: 0.86    top-10: 0.90
Eval -     loss: 1.88     top-1: 0.61    top-5: 0.81    top-10: 0.86
What are you reading?
Que lisez-vous   ?
```
Waiting for W&B process to finish... **(success).**

Run history:

Train - loss

Train - top-1	
Train - top-10	
Train - top-5	
Validation - loss	
Validation - top-1	
Validation - top-10	
Validation - top-5	

Run summary:

Train - loss	1.43726
Train - top-1	0.65743
Train - top-10	0.9001
Train -	

```
wandb.agent("INF8225 - TP3/i736m3vi", train, count=10) # Transformer
```

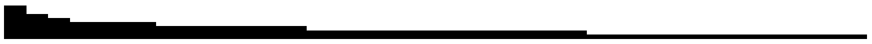
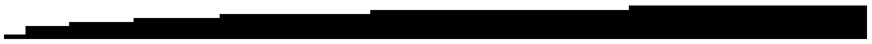
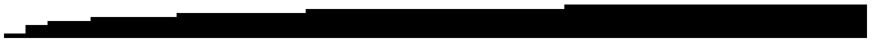
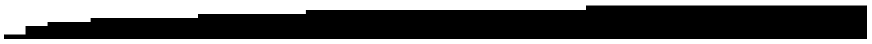
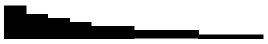
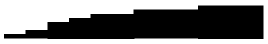
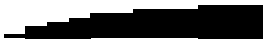
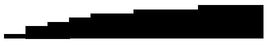
```
wandb: Agent Starting Run: swo5t39a with config:
wandb:   batch_size: 128
wandb:   clip: 11
wandb:   dim_embedding: 457
wandb:   dim_hidden: 73
wandb:   dropout: 0.22992617160098855
wandb:   lr: 0.0006994117711121344
wandb:   n_heads: 10
wandb:   n_layers: 7
wandb: Currently logged in as: ayzeg (8225_team_). Use `wandb login --relogin
wandb: WARNING Ignored wandb.init() arg project when running a sweep.
Tracking run with wandb version 0.14.2
Run data is saved locally in /content/wandb/run-20230410_090909-swo5t39a
Syncing run silvery-sweep-5 to Weights & Biases (docs)
Sweep page: https://wandb.ai/8225_team_/INF8225%20-%20TP3/sweeps/i736m3vi
View project at https://wandb.ai/8225_team_/INF8225%20-%20TP3
View sweep at https://wandb.ai/8225_team_/INF8225%20-%20TP3/sweeps/i736m3vi
View run at https://wandb.ai/8225_team_/INF8225%20-%20TP3/runs/swo5t39a
Starting training for 12 epochs, using cuda.

Epoch 1
Train -    loss: 3.21     top-1: 0.43    top-5: 0.61    top-10: 0.67
Eval -     loss: 3.10     top-1: 0.44    top-5: 0.62    top-10: 0.68
We're talking about you.
Nous sommes heureux.

Epoch 2
```

```
Train -   loss: 2.87     top-1: 0.47    top-5: 0.66    top-10: 0.72
Eval -    loss: 2.76     top-1: 0.49    top-5: 0.67    top-10: 0.73
I'm ready to try doing that.
Je suis sûr de le faire ça.

Epoch 3
Train -   loss: 2.64     top-1: 0.50    top-5: 0.69    top-10: 0.75
Eval -    loss: 2.51     top-1: 0.52    top-5: 0.71    top-10: 0.76
I wonder how long we'll have to wait.
Je me suis imaginé quelque chose.

Epoch 4
Train -   loss: 2.43     top-1: 0.54    top-5: 0.73    top-10: 0.78
Eval -    loss: 2.31     top-1: 0.55    top-5: 0.74    top-10: 0.79
Let's vote.
Commençons.

Epoch 5
Train -   loss: 2.23     top-1: 0.56    top-5: 0.75    top-10: 0.81
Eval -    loss: 2.16     top-1: 0.58    top-5: 0.76    top-10: 0.81
The moon emerged from behind the cloud.
Le sommet de la rivière.

Epoch 6
Train -   loss: 2.15     top-1: 0.58    top-5: 0.77    top-10: 0.82
Eval -    loss: 2.04     top-1: 0.59    top-5: 0.78    top-10: 0.83
We're currently experiencing some turbulence.
Nous sommes en train de bois.

Epoch 7
Train -   loss: 2.02     top-1: 0.59    top-5: 0.78    top-10: 0.83
Eval -    loss: 1.95     top-1: 0.61    top-5: 0.80    top-10: 0.84
How dare you accuse me of lying!
Comment oses-tu me réveiller.

Epoch 8
Train -   loss: 1.93     top-1: 0.61    top-5: 0.80    top-10: 0.85
Eval -    loss: 1.87     top-1: 0.62    top-5: 0.81    top-10: 0.85
What would you like?
Que ferais-tu ?

Epoch 9
Train -   loss: 1.88     top-1: 0.61    top-5: 0.81    top-10: 0.85
Eval -    loss: 1.82     top-1: 0.63    top-5: 0.81    top-10: 0.86
He showed courage in the face of great danger.
Il a montré dans la classe.

Epoch 10
Train -   loss: 1.83     top-1: 0.62    top-5: 0.81    top-10: 0.86
Eval -    loss: 1.77     top-1: 0.64    top-5: 0.82    top-10: 0.86
There was a food fight in the cafeteria.
Il y avait une nourriture.

Epoch 11
Train -   loss: 1.75     top-1: 0.63    top-5: 0.82    top-10: 0.87
```

```
Eval -    loss: 1.72     top-1: 0.64    top-5: 0.83    top-10: 0.87
Come back home.
Reviens la maison.

Epoch 12
Train -   loss: 1.73     top-1: 0.63    top-5: 0.83    top-10: 0.87
Eval -    loss: 1.69     top-1: 0.65    top-5: 0.83    top-10: 0.87
I'm not afraid of you anymore.
Je n'ai plus peur de vous.
```

Waiting for W&B process to finish... **(success).**

Run history:

Train - loss	
Train - top-1	
Train - top-10	
Train - top-5	
Validation - loss	
Validation - top-1	
Validation - top-10	
Validation - top-5	

Run summary:

Train - loss	1.72659
Train - top-1	0.63356
Train - top-10	0.87179
Train - top-5	0.82759
Validation - loss	1.68834
Validation - top-1	0.64815
Validation - top-10	0.87183
Validation - top-5	0.83275

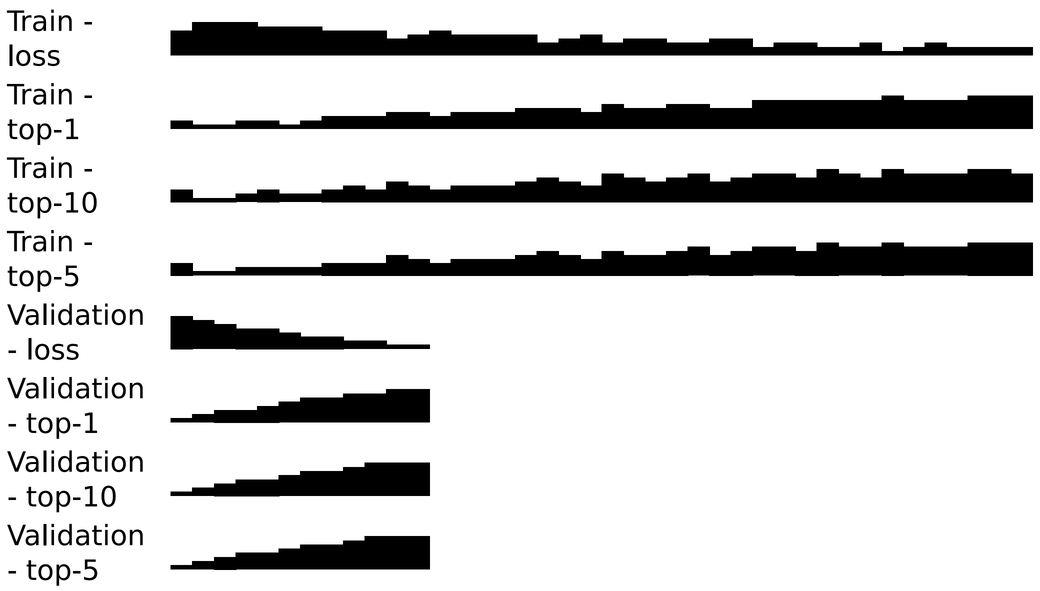
View run **silvery-sweep-5** at: https://wandb.ai/8225_team_/INF8225%20-%20TP3/runs/swo5t39a
Synced 5 W&B file(s), 1 media file(s), 3 artifact file(s) and 1 other file(s)
Find logs at: `./wandb/run-20230410_090909-swo5t39a/logs`

```
wandb: Agent Starting Run: x7xgxxu0 with config:
```

```
wandb: Agent Starting Run: x7xqxxu0 with config:
wandb:   batch_size: 128
wandb:   clip: 7
wandb:   dim_embedding: 700
wandb:   dim_hidden: 289
wandb:   dropout: 0.13130109068684753
wandb:   lr: 0.0009352356160543272
wandb:   n_heads: 3
wandb:   n_layers: 6
wandb: WARNING Ignored wandb.init() arg project when running a sweep.
Tracking run with wandb version 0.14.2
Run data is saved locally in /content/wandb/run-20230410_091819-x7xqxxu0
Syncing run ethereal-sweep-6 to Weights & Biases (docs)
Sweep page: https://wandb.ai/8225_team_/INF8225%20-%20TP3/sweeps/i736m3vi
View project at https://wandb.ai/8225_team_/INF8225%20-%20TP3
View sweep at https://wandb.ai/8225_team_/INF8225%20-%20TP3/sweeps/i736m3vi
View run at https://wandb.ai/8225_team_/INF8225%20-%20TP3/runs/x7xqxxu0
Starting training for 12 epochs, using cuda.

Epoch 1
Train -    loss: 1.66     top-1: 0.65    top-5: 0.84    top-10: 0.88
Eval -     loss: 1.65     top-1: 0.65    top-5: 0.84    top-10: 0.88
It's completely dark.
C'est complètement noir.

Epoch 2
Train -    loss: 1.65     top-1: 0.64    top-5: 0.84    top-10: 0.88
Eval -     loss: 1.63     top-1: 0.66    top-5: 0.84    top-10: 0.88
Have you already fed the dog?
As-tu déjà nourri le chien ?

Epoch 3
Train -    loss: 1.59     top-1: 0.66    top-5: 0.84    top-10: 0.89
Eval -     loss: 1.60     top-1: 0.66    top-5: 0.84    top-10: 0.88
Where did you learn to speak French?
Où as-tu appris à parler français ?

Epoch 4
Train -    loss: 1.57     top-1: 0.66    top-5: 0.85    top-10: 0.89
Eval -     loss: 1.58     top-1: 0.66    top-5: 0.85    top-10: 0.88
You broke the washing machine.
Tu as enfreint la machine.

Epoch 5
Train -    loss: 1.60     top-1: 0.66    top-5: 0.85    top-10: 0.88
Eval -     loss: 1.56     top-1: 0.67    top-5: 0.85    top-10: 0.89
Tom turned right.
Tom a tourné raison.

Epoch 6
Train -    loss: 1.51     top-1: 0.67    top-5: 0.86    top-10: 0.89
Eval -     loss: 1.54     top-1: 0.67    top-5: 0.85    top-10: 0.89
You don't speak French, do you?
Tu ne parles pas français, si ?

Epoch 7
```

```
Train -   loss: 1.50     top-1: 0.67    top-5: 0.86    top-10: 0.89
Eval -    loss: 1.52     top-1: 0.68    top-5: 0.86    top-10: 0.89
We're all mothers.
Nous sommes toutes fausses.

Epoch 8
Train -   loss: 1.51     top-1: 0.67    top-5: 0.86    top-10: 0.90
Eval -    loss: 1.51     top-1: 0.68    top-5: 0.86    top-10: 0.89
They're bad.
Ils sont mauvais.

Epoch 9
Train -   loss: 1.49     top-1: 0.67    top-5: 0.86    top-10: 0.90
Eval -    loss: 1.50     top-1: 0.68    top-5: 0.86    top-10: 0.89
I always said no.
Je n'ai toujours rien dit.

Epoch 10
Train -   loss: 1.46     top-1: 0.67    top-5: 0.86    top-10: 0.90
Eval -    loss: 1.48     top-1: 0.68    top-5: 0.86    top-10: 0.90
Tom and Mary usually speak French to each other.
Tom et Marie parlent généralement français.

Epoch 11
Train -   loss: 1.43     top-1: 0.68    top-5: 0.87    top-10: 0.90
Eval -    loss: 1.47     top-1: 0.69    top-5: 0.86    top-10: 0.90
I feel like I have to be there.
J'ai l'impression de pouvoir être là.

Epoch 12
Train -   loss: 1.42     top-1: 0.68    top-5: 0.87    top-10: 0.90
Eval -    loss: 1.46     top-1: 0.69    top-5: 0.86    top-10: 0.90
I'm as tired as tired can be.
Je suis aussi fatigué que possible.
Waiting for W&B process to finish... (success).
```

Run history:



Run summary:

Train - loss	1.4232
Train - top-1	0.67939
Train - top-10	0.90422
Train - top-5	0.86736
Validation - loss	1.46373
Validation - top-1	0.68617
Validation - top-10	0.89791
Validation - top-5	0.86383

View run **ethereal-sweep-6** at: https://wandb.ai/8225_team_/INF8225%20-%20TP3/runs/x7xqxxu0
Synced 5 W&B file(s), 1 media file(s), 3 artifact file(s) and 1 other file(s)
Find logs at: ./wandb/run-20230410_091819-x7xqxxu0/logs

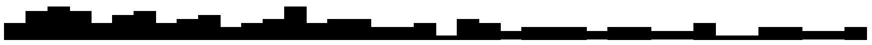
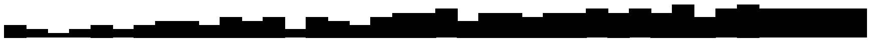
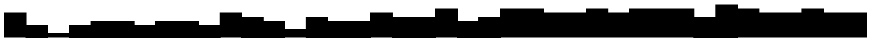
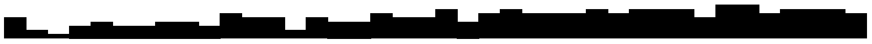
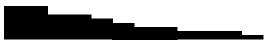
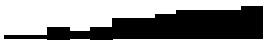
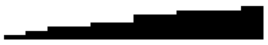
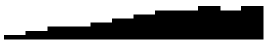
```
wandb: Agent Starting Run: xc4uydyh with config:
wandb:   batch_size: 128
wandb:   clip: 12
wandb:   dim_embedding: 651
wandb:   dim_hidden: 481
wandb:   dropout: 0.07011145463813385
wandb:   lr: 0.0006600273023362729
wandb:   n_heads: 7
wandb:   n_layers: 9
wandb: WARNING Ignored wandb.init() arg project when running a sweep.
```

Tracking run with wandb version 0.14.2
Run data is saved locally in /content/wandb/run-20230410_092730-xc4uydyh
Syncing run **ethereal-sweep-7** to Weights & Biases (docs)
Sweep page: https://wandb.ai/8225_team_/INF8225%20-%20TP3/sweeps/i736m3vi
View project at https://wandb.ai/8225_team_/INF8225%20-%20TP3
View sweep at https://wandb.ai/8225_team_/INF8225%20-%20TP3/sweeps/i736m3vi
View run at https://wandb.ai/8225_team_/INF8225%20-%20TP3/runs/xc4uydyh

```
Starting training for 12 epochs, using cuda.

Epoch 1
Train -   loss: 1.40     top-1: 0.69    top-5: 0.87    top-10: 0.91
Eval -    loss: 1.45     top-1: 0.69    top-5: 0.87    top-10: 0.90
May I sit next to you?
Puis-je t'asseoir à côté ?

Epoch 2
Train -   loss: 1.39     top-1: 0.68    top-5: 0.87    top-10: 0.91
Eval -    loss: 1.45     top-1: 0.69    top-5: 0.87    top-10: 0.90
She has nothing to do with that affair.
Elle n'a rien à faire avec ça.
```

```
Epoch 3
Train -   loss: 1.38     top-1: 0.69    top-5: 0.87    top-10: 0.91
Eval -    loss: 1.43     top-1: 0.69    top-5: 0.87    top-10: 0.90
Tom is upbeat.
Tom est <unk>.

Epoch 4
Train -   loss: 1.42     top-1: 0.68    top-5: 0.87    top-10: 0.90
Eval -    loss: 1.44     top-1: 0.69    top-5: 0.87    top-10: 0.90
I quit.
J'ai démissionné.

Epoch 5
Train -   loss: 1.34     top-1: 0.70    top-5: 0.88    top-10: 0.91
Eval -    loss: 1.42     top-1: 0.69    top-5: 0.87    top-10: 0.90
Tom and Mary were very busy.
Tom et Marie étaient très occupée.

Epoch 6
Train -   loss: 1.36     top-1: 0.69    top-5: 0.88    top-10: 0.91
Eval -    loss: 1.41     top-1: 0.70    top-5: 0.87    top-10: 0.90
Did you clinch the deal?
As-tu reçu les pieds ?

Epoch 7
Train -   loss: 1.35     top-1: 0.69    top-5: 0.88    top-10: 0.91
Eval -    loss: 1.40     top-1: 0.70    top-5: 0.87    top-10: 0.90
This is silly.
C'est bête.

Epoch 8
Train -   loss: 1.31     top-1: 0.70    top-5: 0.88    top-10: 0.92
Eval -    loss: 1.40     top-1: 0.70    top-5: 0.87    top-10: 0.91
Why would you do that without telling us?
Pourquoi ferais-tu ça sans nous dire ?

Epoch 9
Train -   loss: 1.28     top-1: 0.71    top-5: 0.89    top-10: 0.92
Eval -    loss: 1.39     top-1: 0.70    top-5: 0.87    top-10: 0.91
I know it won't be easy to do that.
Je sais que ça ne sera pas facile à faire ça.

Epoch 10
Train -   loss: 1.30     top-1: 0.70    top-5: 0.88    top-10: 0.92
Eval -    loss: 1.38     top-1: 0.70    top-5: 0.87    top-10: 0.91
Tom doesn't cook well.
Tom ne parle pas bien.

Epoch 11
Train -   loss: 1.29     top-1: 0.70    top-5: 0.88    top-10: 0.92
Eval -    loss: 1.38     top-1: 0.70    top-5: 0.87    top-10: 0.91
I think I've seen enough.
Je pense que j'ai déjà vu.

Epoch 12
Epoch 12
Train -   loss: 1.29     top-1: 0.70    top-5: 0.88    top-10: 0.92
Eval -    loss: 1.38     top-1: 0.70    top-5: 0.88    top-10: 0.91
I'll tell you only what you need to know.
Je te dirai à ce que tu as besoin de savoir.
```
Waiting for W&B process to finish... **(success).**

Run history:

Train - loss	
Train - top-1	
Train - top-10	
Train - top-5	
Validation - loss	
Validation - top-1	
Validation - top-10	
Validation - top-5	

Run summary:

Train - loss	1.28716
Train - top-1	0.70235
Train - top-10	0.91854
Train - top-5	0.88329
Validation - loss	1.37801
Validation - top-1	0.70341
Validation - top-10	0.90736
Validation - top-5	0.87535

View run **ethereal-sweep-7** at: https://wandb.ai/8225_team_/INF8225%20-%20TP3/runs/xc4uydyh
Synced 5 W&B file(s), 1 media file(s), 3 artifact file(s) and 1 other file(s)
Find logs at: `./wandb/run-20230410_092730-xc4uydyh/logs`

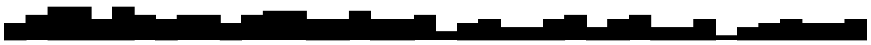
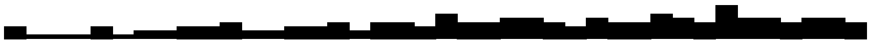
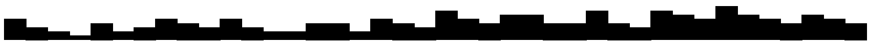
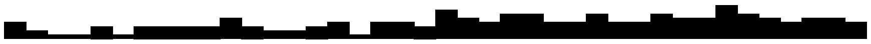
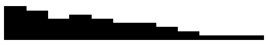
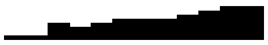
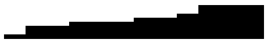
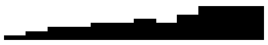
```
wandb: Agent Starting Run: 5fih3buq with config:
wandb:   batch_size: 128
wandb:   clip: 8
wandb:   dim_embedding: 577
wandb:   dim_hidden: 575
```

```
wandb:   dropout: 0.253616715822834
wandb:   lr: 0.000770141803485162
wandb:   n_heads: 11
wandb:   n_layers: 6
wandb: WARNING Ignored wandb.init() arg project when running a sweep.
Tracking run with wandb version 0.14.2
Run data is saved locally in /content/wandb/run-20230410_093639-5fih3buq
Syncing run expert-sweep-8 to Weights & Biases (docs)
Sweep page: https://wandb.ai/8225_team_/INF8225%20-%20TP3/sweeps/i736m3vi
View project at https://wandb.ai/8225_team_/INF8225%20-%20TP3
View sweep at https://wandb.ai/8225_team_/INF8225%20-%20TP3/sweeps/i736m3vi
View run at https://wandb.ai/8225_team_/INF8225%20-%20TP3/runs/5fih3buq
Starting training for 12 epochs, using cuda.

Epoch 1
Train -    loss: 1.28     top-1: 0.71    top-5: 0.88    top-10: 0.92
Eval -     loss: 1.37     top-1: 0.70    top-5: 0.88    top-10: 0.91
Why does my dog hate Tom?
Pourquoi mon chien   ?

Epoch 2
Train -    loss: 1.29     top-1: 0.70    top-5: 0.88    top-10: 0.92
Eval -     loss: 1.37     top-1: 0.70    top-5: 0.88    top-10: 0.91
I think it's time for a beer.
Je pense qu'il est temps pour une bière.

Epoch 3
Train -    loss: 1.26     top-1: 0.71    top-5: 0.89    top-10: 0.92
Eval -     loss: 1.36     top-1: 0.71    top-5: 0.88    top-10: 0.91
Can I ask you for a favor?
Puis-je vous demander une faveur ?

Epoch 4
Train -    loss: 1.25     top-1: 0.71    top-5: 0.89    top-10: 0.92
Eval -     loss: 1.36     top-1: 0.71    top-5: 0.88    top-10: 0.91
Tom has decided to leave the company.
Tom a décidé de partir de l'entreprise.

Epoch 5
Train -    loss: 1.28     top-1: 0.71    top-5: 0.88    top-10: 0.92
Eval -     loss: 1.36     top-1: 0.71    top-5: 0.88    top-10: 0.91
I've made a list of things I'd like to buy.
J'ai fait une liste de choses dont j'aimerais acheter.

Epoch 6
Train -    loss: 1.23     top-1: 0.71    top-5: 0.89    top-10: 0.92
Eval -     loss: 1.35     top-1: 0.71    top-5: 0.88    top-10: 0.91
This is all a conspiracy.
C'est tout une conspiration.

Epoch 7
Train -    loss: 1.24     top-1: 0.71    top-5: 0.89    top-10: 0.92
Eval -     loss: 1.35     top-1: 0.71    top-5: 0.88    top-10: 0.91
That's just what I need.
C'est juste ce que j'ai besoin.
```

```
Epoch 8
Train -   loss: 1.23     top-1: 0.71    top-5: 0.89    top-10: 0.92
Eval -    loss: 1.35     top-1: 0.71    top-5: 0.88    top-10: 0.91
You promised you'd stay.
Tu avais promis que tu restes.

Epoch 9
Train -   loss: 1.22     top-1: 0.72    top-5: 0.89    top-10: 0.92
Eval -    loss: 1.34     top-1: 0.71    top-5: 0.88    top-10: 0.91
Where did you go?
Où êtes-vous allées ?

Epoch 10
Train -   loss: 1.25     top-1: 0.71    top-5: 0.89    top-10: 0.92
Eval -    loss: 1.33     top-1: 0.71    top-5: 0.88    top-10: 0.91
Tom smelled something.
Tom sentait quelque chose.

Epoch 11
Train -   loss: 1.22     top-1: 0.71    top-5: 0.89    top-10: 0.92
Eval -    loss: 1.33     top-1: 0.71    top-5: 0.88    top-10: 0.91
I'm not used to the heat.
Je ne suis pas habituée à la chaleur.

Epoch 12
Train -   loss: 1.20     top-1: 0.72    top-5: 0.89    top-10: 0.92
Eval -    loss: 1.33     top-1: 0.71    top-5: 0.88    top-10: 0.91
Why didn't you call me last night?
Pourquoi n'as-tu pas appelée hier soir ?
```

Waiting for W&B process to finish... **(success).**

Run history:

Train - loss	
Train - top-1	
Train - top-10	
Train - top-5	
Validation - loss	
Validation - top-1	
Validation - top-10	
Validation - top-5	

Run summary:

Train -

loss	1.20497
Train - top-1	0.71944
Train - top-10	0.92491
Train - top-5	0.89224
Validation - loss	1.33232
Validation - top-1	0.71276
Validation - top-10	0.91254
Validation - top-5	0.88194

View run **expert-sweep-8** at: https://wandb.ai/8225_team_/INF8225%20-%20TP3/runs/5fih3buq
Synced 5 W&B file(s), 1 media file(s), 3 artifact file(s) and 1 other file(s)
Find logs at: ./wandb/run-20230410_093639-5fih3buq/logs

```
wandb: Agent Starting Run: xa4xttau with config:
wandb:   batch_size: 256
wandb:   clip: 7
wandb:   dim_embedding: 493
wandb:   dim_hidden: 422
wandb:   dropout: 0.19391275354122212
wandb:   lr: 0.000645164751201379
wandb:   n_heads: 3
wandb:   n_layers: 3
wandb: WARNING Ignored wandb.init() arg project when running a sweep.
```

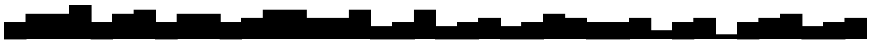
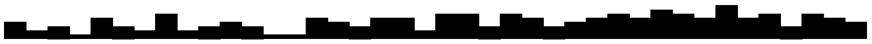
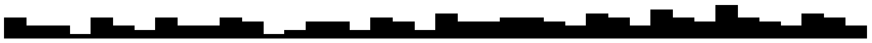
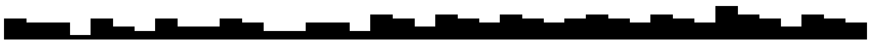
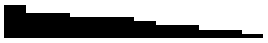
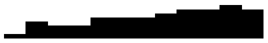
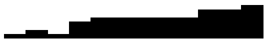
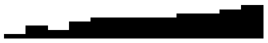
Tracking run with wandb version 0.14.2
Run data is saved locally in /content/wandb/run-20230410_094550-xa4xttau
Syncing run **crimson-sweep-9** to Weights & Biases (docs)
Sweep page: https://wandb.ai/8225_team_/INF8225%20-%20TP3/sweeps/i736m3vi
View project at https://wandb.ai/8225_team_/INF8225%20-%20TP3
View sweep at https://wandb.ai/8225_team_/INF8225%20-%20TP3/sweeps/i736m3vi
View run at https://wandb.ai/8225_team_/INF8225%20-%20TP3/runs/xa4xttau

```
Starting training for 12 epochs, using cuda.

Epoch 1
Train -   loss: 1.22     top-1: 0.72    top-5: 0.89    top-10: 0.92
Eval -    loss: 1.33     top-1: 0.71    top-5: 0.88    top-10: 0.91
She hates him.
Elle lui déteste.

Epoch 2
Train -   loss: 1.21     top-1: 0.72    top-5: 0.89    top-10: 0.93
Eval -    loss: 1.32     top-1: 0.72    top-5: 0.88    top-10: 0.91
I'm so thirsty.
J'ai tellement soif.

Epoch 3
Train -   loss: 1.19     top-1: 0.72    top-5: 0.90    top-10: 0.93
Eval -    loss: 1.32     top-1: 0.71    top-5: 0.88    top-10: 0.91
```

```
Do your parents leave you home alone?
Vos parents me quittez la maison ?

Epoch 4
Train -   loss: 1.21     top-1: 0.72    top-5: 0.89    top-10: 0.93
Eval -    loss: 1.32     top-1: 0.71    top-5: 0.88    top-10: 0.91
Don't be such a cheapskate.
Ne sois pas si mignonne.

Epoch 5
Train -   loss: 1.21     top-1: 0.72    top-5: 0.89    top-10: 0.93
Eval -    loss: 1.32     top-1: 0.72    top-5: 0.88    top-10: 0.91
Do you think you could help me do that?
Pensez-vous que tu pourrais m'aider ?

Epoch 6
Train -   loss: 1.19     top-1: 0.72    top-5: 0.89    top-10: 0.93
Eval -    loss: 1.32     top-1: 0.72    top-5: 0.88    top-10: 0.91
Tom speaks French much better than Mary does.
Tom parle beaucoup mieux que Marie.

Epoch 7
Train -   loss: 1.20     top-1: 0.72    top-5: 0.89    top-10: 0.92
Eval -    loss: 1.32     top-1: 0.72    top-5: 0.88    top-10: 0.91
She handed me the letter without saying anything.
Elle m'a tendu la lettre sans dire.

Epoch 8
Train -   loss: 1.16     top-1: 0.73    top-5: 0.90    top-10: 0.93
Eval -    loss: 1.31     top-1: 0.72    top-5: 0.88    top-10: 0.91
I can give you something for your pain.
Je peux te donner quelque chose pour votre douleur.

Epoch 9
Train -   loss: 1.17     top-1: 0.72    top-5: 0.90    top-10: 0.93
Eval -    loss: 1.31     top-1: 0.72    top-5: 0.88    top-10: 0.91
The competition has become fierce.
La concurrence a été en vain.

Epoch 10
Train -   loss: 1.18     top-1: 0.73    top-5: 0.90    top-10: 0.93
Eval -    loss: 1.31     top-1: 0.72    top-5: 0.89    top-10: 0.91
The cat is sitting on the table.
Le chat est assis sur la table.

Epoch 11
Train -   loss: 1.18     top-1: 0.72    top-5: 0.90    top-10: 0.93
Eval -    loss: 1.31     top-1: 0.72    top-5: 0.89    top-10: 0.92
Doesn't that car look familiar?
N'interromps-t-il pas cette voiture   ?

Epoch 12
Train -   loss: 1.15     top-1: 0.73    top-5: 0.90    top-10: 0.93
Eval -    loss: 1.30     top-1: 0.72    top-5: 0.89    top-10: 0.92
What could possibly go wrong?
Que pourraient arriver ?
```

Waiting for W&B process to finish... **(success).**

Run history:

Train - loss	
Train - top-1	
Train - top-10	
Train - top-5	
Validation - loss	
Validation - top-1	
Validation - top-10	
Validation - top-5	

Run summary:

Train - loss	1.15307
Train - top-1	0.72938
Train - top-10	0.93084
Train - top-5	0.90008
Validation - loss	1.30386
Validation - top-1	0.71872
Validation - top-10	0.91533
Validation - top-5	0.88612

View run **crimson-sweep-9** at: https://wandb.ai/8225_team_/INF8225%20-%20TP3/runs/xa4xttau
Synced 5 W&B file(s), 1 media file(s), 3 artifact file(s) and 1 other file(s)
Find logs at: ./wandb/run-20230410_094550-xa4xttau/logs

```
wandb: Agent Starting Run: yi8cg81f with config:
wandb:   batch_size: 256
wandb:   clip: 9
wandb:   dim_embedding: 402
wandb:   dim_hidden: 144
wandb:   dropout: 0.2394009804916092
wandb:   lr: 0.0009383851854113872
wandb:   n_heads: 11
wandb:   n_layers: 8
```

```
wandb: WARNING Ignored wandb.init() arg project when running a sweep.
Tracking run with wandb version 0.14.2
Run data is saved locally in /content/wandb/run-20230410_095501-yi8cg81f
Syncing run autumn-sweep-10 to Weights & Biases (docs)
Sweep page: https://wandb.ai/8225_team_/INF8225%20-%20TP3/sweeps/i736m3vi
View project at https://wandb.ai/8225_team_/INF8225%20-%20TP3
View sweep at https://wandb.ai/8225_team_/INF8225%20-%20TP3/sweeps/i736m3vi
View run at https://wandb.ai/8225_team_/INF8225%20-%20TP3/runs/yi8cg81f
Starting training for 12 epochs, using cuda.

Epoch 1
Train -    loss: 1.16     top-1: 0.73    top-5: 0.90    top-10: 0.93
Eval -     loss: 1.30     top-1: 0.72    top-5: 0.89    top-10: 0.92
Do we have a deal here?
Avons-nous une décision ici ?

Epoch 2
Train -    loss: 1.14     top-1: 0.73    top-5: 0.90    top-10: 0.93
Eval -     loss: 1.30     top-1: 0.72    top-5: 0.89    top-10: 0.92
The whole soccer team was on cloud nine after winning the championship.
Toute la football était le fruit au regard après le championnat.

Epoch 3
Train -    loss: 1.15     top-1: 0.73    top-5: 0.90    top-10: 0.93
Eval -     loss: 1.30     top-1: 0.72    top-5: 0.89    top-10: 0.92
How arrogant!
Comme c'est arrogant !

Epoch 4
Train -    loss: 1.14     top-1: 0.73    top-5: 0.90    top-10: 0.93
Eval -     loss: 1.29     top-1: 0.72    top-5: 0.89    top-10: 0.92
I'm sure this is only temporary.
Je suis certaine que c'est seulement temporaire.

Epoch 5
Train -    loss: 1.14     top-1: 0.73    top-5: 0.90    top-10: 0.93
Eval -     loss: 1.29     top-1: 0.72    top-5: 0.89    top-10: 0.92
Which one of them was it?
Laquelle d'entre eux était ?

Epoch 6
Train -    loss: 1.17     top-1: 0.72    top-5: 0.90    top-10: 0.93
Eval -     loss: 1.29     top-1: 0.72    top-5: 0.89    top-10: 0.92
Where are they sending us?
Où nous téléphone nous a embrassé ?

Epoch 7
Train -    loss: 1.15     top-1: 0.73    top-5: 0.90    top-10: 0.93
Eval -     loss: 1.29     top-1: 0.72    top-5: 0.89    top-10: 0.92
I saw my sister tear up the letter.
J'ai vu ma sœur à la lettre.

Epoch 8
Train -    loss: 1.13     top-1: 0.73    top-5: 0.90    top-10: 0.93
Eval -     loss: 1.29     top-1: 0.72    top-5: 0.89    top-10: 0.92
```

```
This mouse was killed by my cat.
Cette souris a été tué par mon chat.

Epoch 9
Train -   loss: 1.13     top-1: 0.73    top-5: 0.90    top-10: 0.93
Eval -    loss: 1.29     top-1: 0.72    top-5: 0.89    top-10: 0.92
None of the money is yours.
Rien de l'argent n'est le tien.

Epoch 10
Train -   loss: 1.11     top-1: 0.73    top-5: 0.91    top-10: 0.94
Eval -    loss: 1.29     top-1: 0.72    top-5: 0.89    top-10: 0.92
The floor has to be scrubbed.
Le sol doit être récuré.

Epoch 11
Train -   loss: 1.11     top-1: 0.73    top-5: 0.90    top-10: 0.93
Eval -    loss: 1.29     top-1: 0.72    top-5: 0.89    top-10: 0.92
My mother put a large vase on the shelf.
Ma mère a mis un grand vase sur l'étagère.

Epoch 12
Train -   loss: 1.12     top-1: 0.73    top-5: 0.90    top-10: 0.93
Eval -    loss: 1.29     top-1: 0.72    top-5: 0.89    top-10: 0.92
I was wrong about that.
J'ai eu tort à ça.
```

Waiting for W&B process to finish... **(success).**

Run history:

Train - loss	
Train - top-1	
Train - top-10	
Train - top-5	
Validation - loss	
Validation - top-1	
Validation - top-10	
Validation - top-5	

Run summary:

Train - loss	1.12043
Train - top-1	0.73397
Train -	

top-10	0.93225
Train - top-5	0.90147
Validation - loss	1.28651
Validation - top-1	0.7235
Validation - top-10	0.91741
Validation - top-5	0.88893

View run **autumn-sweep-10** at: https://wandb.ai/8225_team_/INF8225%20-%20TP3/runs/yi8cg81f
Synced 5 W&B file(s), 1 media file(s), 3 artifact file(s) and 1 other file(s)
Find logs at: `./wandb/run-20230410_095501-yi8cg81f/logs`

```
wandb: Sweep Agent: Waiting for job.
wandb: Job received.
wandb: Agent Starting Run: ipjf6aiz with config:
wandb:   batch_size: 512
wandb:   clip: 12
wandb:   dim_embedding: 451
wandb:   dim_hidden: 428
wandb:   dropout: 0.2263208499636016
wandb:   lr: 0.0006378170606560637
wandb:   n_heads: 10
wandb:   n_layers: 8
wandb: WARNING Ignored wandb.init() arg project when running a sweep.
```

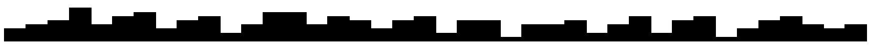
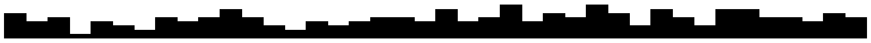
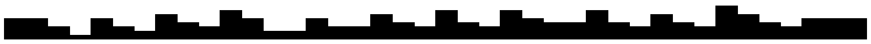
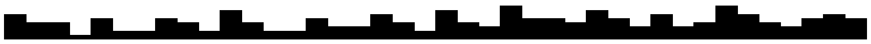
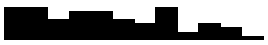
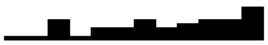
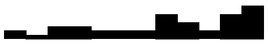
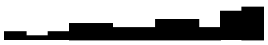
Tracking run with wandb version 0.14.2
Run data is saved locally in `/content/wandb/run-20230410_100422-ipjf6aiz`
Syncing run **usual-sweep-11** to Weights & Biases (docs)
Sweep page: https://wandb.ai/8225_team_/INF8225%20-%20TP3/sweeps/i736m3vi
View project at https://wandb.ai/8225_team_/INF8225%20-%20TP3
View sweep at https://wandb.ai/8225_team_/INF8225%20-%20TP3/sweeps/i736m3vi
View run at https://wandb.ai/8225_team_/INF8225%20-%20TP3/runs/ipjf6aiz

```
Starting training for 12 epochs, using cuda.

Epoch 1
Train -    loss: 1.13     top-1: 0.73    top-5: 0.90    top-10: 0.93
Eval -     loss: 1.29     top-1: 0.72    top-5: 0.89    top-10: 0.92
Would you like to travel to the United States?
Aimerais-tu voyager aux États-Unis ?

Epoch 2
Train -    loss: 1.12     top-1: 0.73    top-5: 0.90    top-10: 0.93
Eval -     loss: 1.29     top-1: 0.72    top-5: 0.89    top-10: 0.92
We trust him.
Nous l'avons confiance.

Epoch 3
Train -    loss: 1.09     top-1: 0.74    top-5: 0.91    top-10: 0.94
Eval -     loss: 1.28     top-1: 0.73    top-5: 0.89    top-10: 0.92
I prefer reading books to watching television.
Je préfère lire des livres à regarder la télévision.
```

```
Epoch 4
Train -   loss: 1.12     top-1: 0.73    top-5: 0.90    top-10: 0.93
Eval -    loss: 1.28     top-1: 0.72    top-5: 0.89    top-10: 0.92
I thought Tom might want to do that today.
Je pensais que Tom pourrait faire cela aujourd'hui.

Epoch 5
Train -   loss: 1.09     top-1: 0.74    top-5: 0.91    top-10: 0.94
Eval -    loss: 1.28     top-1: 0.72    top-5: 0.89    top-10: 0.92
Don't approach the dog.
Ne vous approchez pas.

Epoch 6
Train -   loss: 1.07     top-1: 0.74    top-5: 0.91    top-10: 0.94
Eval -    loss: 1.28     top-1: 0.72    top-5: 0.89    top-10: 0.92
You're the richest man I know.
Vous êtes le plus riche que je connaisse.

Epoch 7
Train -   loss: 1.10     top-1: 0.74    top-5: 0.91    top-10: 0.93
Eval -    loss: 1.28     top-1: 0.73    top-5: 0.89    top-10: 0.92
You are fabulous.
Tu es fabuleux.

Epoch 8
Train -   loss: 1.08     top-1: 0.74    top-5: 0.91    top-10: 0.94
Eval -    loss: 1.29     top-1: 0.72    top-5: 0.89    top-10: 0.92
Help yourself to these cookies.
Aidez-vous à ces biscuits.

Epoch 9
Train -   loss: 1.10     top-1: 0.73    top-5: 0.91    top-10: 0.94
Eval -    loss: 1.28     top-1: 0.73    top-5: 0.89    top-10: 0.92
He solved all those problems with ease.
Il a résolu ces problèmes avec lui.

Epoch 10
Train -   loss: 1.09     top-1: 0.74    top-5: 0.91    top-10: 0.94
Eval -    loss: 1.28     top-1: 0.73    top-5: 0.89    top-10: 0.92
I lay on my bed.
J'étais étendu sur mon lit.

Epoch 11
Train -   loss: 1.09     top-1: 0.74    top-5: 0.91    top-10: 0.94
Eval -    loss: 1.28     top-1: 0.73    top-5: 0.89    top-10: 0.92
Tom robbed a bank in Boston.
Tom a volé une banque à Boston.

Epoch 12
Train -   loss: 1.08     top-1: 0.74    top-5: 0.91    top-10: 0.94
Eval -    loss: 1.27     top-1: 0.73    top-5: 0.89    top-10: 0.92
I'm not always so lenient.
Je ne suis pas toujours aussi excité.
```

Waiting for W&B process to finish... **(success).**

Run history:

Train - loss	
Train - top-1	
Train - top-10	
Train - top-5	
Validation - loss	
Validation - top-1	
Validation - top-10	
Validation - top-5	

Run summary:

Train - loss	1.07839
Train - top-1	0.73806
Train - top-10	0.93647
Train - top-5	0.90798
Validation - loss	1.27408
Validation - top-1	0.72695
Validation - top-10	0.91922
Validation - top-5	0.89065

```
View run usual-sweep-11 at: https://wandb.ai/8225_team_/INF8225%20-%20TP3/runs/ipjf6aiz
Synced 5 W&B file(s), 1 media file(s), 3 artifact file(s) and 1 other file(s)
Find logs at: ./wandb/run-20230410_100422-ipjf6aiz/logs
wandb: Agent Starting Run: xqadt05t with config:
wandb:   batch_size: 512
wandb:   clip: 12
wandb:   dim_embedding: 656
wandb:   dim_hidden: 908
wandb:   dropout: 0.12753689232983317
wandb:   lr: 0.0008649158741300643
wandb:   n_heads: 10
wandb:   n_layers: 7
wandb: WARNING Ignored wandb.init() arg project when running a sweep.
Tracking run with wandb version 0.14.2
```

```
Run data is saved locally in /content/wandb/run-20230410_101331-xqadt05t
Syncing run likely-sweep-3 to Weights & Biases (docs)
Sweep page: https://wandb.ai/8225_team_/INF8225%20-%20TP3/sweeps/i736m3vi
View project at https://wandb.ai/8225_team_/INF8225%20-%20TP3
View sweep at https://wandb.ai/8225_team_/INF8225%20-%20TP3/sweeps/i736m3vi
View run at https://wandb.ai/8225_team_/INF8225%20-%20TP3/runs/xqadt05t
Starting training for 12 epochs, using cuda.

Epoch 1
Train -   loss: 1.10     top-1: 0.73    top-5: 0.91    top-10: 0.94
Eval -    loss: 1.28     top-1: 0.73    top-5: 0.89    top-10: 0.92
Do you know when she will come?
Sais-tu quand elle viendra ?

Epoch 2
Train -   loss: 1.06     top-1: 0.74    top-5: 0.91    top-10: 0.94
Eval -    loss: 1.27     top-1: 0.73    top-5: 0.89    top-10: 0.92
Maybe I need a new assistant.
Peut-être ai-je besoin d'un nouvel assistant.

Epoch 3
Train -   loss: 1.08     top-1: 0.74    top-5: 0.91    top-10: 0.94
Eval -    loss: 1.27     top-1: 0.73    top-5: 0.89    top-10: 0.92
If I'd known, I would've told you.
Si j'avais su, je t'aurais dit.

Epoch 4
Train -   loss: 1.07     top-1: 0.74    top-5: 0.91    top-10: 0.94
Eval -    loss: 1.27     top-1: 0.73    top-5: 0.89    top-10: 0.92
Tom can speak French better than you.
Tom parle mieux le français que toi.

Epoch 5
Train -   loss: 1.08     top-1: 0.73    top-5: 0.91    top-10: 0.94
Eval -    loss: 1.27     top-1: 0.73    top-5: 0.89    top-10: 0.92
Have you ever been on TV?
As-tu jamais été directement à la télé ?

Epoch 6
Train -   loss: 1.07     top-1: 0.74    top-5: 0.91    top-10: 0.94
Eval -    loss: 1.28     top-1: 0.73    top-5: 0.89    top-10: 0.92
It's bad weather, to be sure, but we've seen worse.
C'est mauvais temps, mais nous sommes certains.

Epoch 7
Train -   loss: 1.05     top-1: 0.75    top-5: 0.91    top-10: 0.94
Eval -    loss: 1.27     top-1: 0.73    top-5: 0.89    top-10: 0.92
You really don't get it, do you?
Tu ne comprends vraiment pas, n'est-ce pas ?

Epoch 8
Train -   loss: 1.08     top-1: 0.74    top-5: 0.91    top-10: 0.94
Eval -    loss: 1.27     top-1: 0.73    top-5: 0.89    top-10: 0.92
What color are they?
Quelle couleur sont-ils ?
```

```
Epoch 9
Train -   loss: 1.06     top-1: 0.74    top-5: 0.91    top-10: 0.94
Eval -    loss: 1.27     top-1: 0.73    top-5: 0.89    top-10: 0.92
I don't care why Tom did it. I'm just glad he did it.
Je me fiche de Tom.

Epoch 10
Train -   loss: 1.09     top-1: 0.74    top-5: 0.91    top-10: 0.94
Eval -    loss: 1.27     top-1: 0.73    top-5: 0.89    top-10: 0.92
He will travel abroad next year.
Il va voyager à l'année prochaine.

Epoch 11
Train -   loss: 1.07     top-1: 0.74    top-5: 0.91    top-10: 0.94
Eval -    loss: 1.26     top-1: 0.73    top-5: 0.89    top-10: 0.92
Did you remember to buy bread?
Avez-vous pensé à acheter de pain ?

Epoch 12
Train -   loss: 1.06     top-1: 0.74    top-5: 0.91    top-10: 0.94
Eval -    loss: 1.26     top-1: 0.73    top-5: 0.89    top-10: 0.92
Can't you tell us anything?
Ne peux-tu pas nous dire ?
```

Waiting for W&B process to finish... **(success).**

Run history:

Train - loss	
Train - top-1	
Train - top-10	
Train - top-5	
Validation - loss	
Validation - top-1	
Validation - top-10	
Validation - top-5	

Run summary:

Train - loss	1.06358
Train - top-1	0.7438
Train - top-10	0.93882
Train -	0.90913

top-5	0.90913
Validation - loss	1.2639
Validation - top-1	0.72841
Validation - top-10	0.91993
Validation - top-5	0.89153

View run **likely-sweep-3** at: https://wandb.ai/8225_team_/INF8225%20-%20TP3/runs/xqadt05t
Synced 5 W&B file(s), 1 media file(s), 3 artifact file(s) and 1 other file(s)
Find logs at: `./wandb/run-20230410_101331-xqadt05t/logs`

```
wandb: Agent Starting Run: 2ceuhwui with config:
wandb:   batch_size: 256
wandb:   clip: 4
wandb:   dim_embedding: 370
wandb:   dim_hidden: 634
wandb:   dropout: 0.27825276330947457
wandb:   lr: 0.0009833031838853794
wandb:   n_heads: 6
wandb:   n_layers: 8
wandb: WARNING Ignored wandb.init() arg project when running a sweep.
```

Tracking run with wandb version 0.14.2
Run data is saved locally in `/content/wandb/run-20230410_102241-2ceuhwui`
Syncing run **proud-sweep-4** to Weights & Biases (docs)
Sweep page: https://wandb.ai/8225_team_/INF8225%20-%20TP3/sweeps/i736m3vi
View project at https://wandb.ai/8225_team_/INF8225%20-%20TP3
View sweep at https://wandb.ai/8225_team_/INF8225%20-%20TP3/sweeps/i736m3vi
View run at https://wandb.ai/8225_team_/INF8225%20-%20TP3/runs/2ceuhwui

```
Starting training for 12 epochs, using cuda.

Epoch 1
Train -   loss: 1.05     top-1: 0.75    top-5: 0.91    top-10: 0.94
Eval -    loss: 1.27     top-1: 0.73    top-5: 0.89    top-10: 0.92
As soon as I have it, I'll forward it to you.
Dès que je l'ai hâte de vous.

Epoch 2
Train -   loss: 1.07     top-1: 0.74    top-5: 0.91    top-10: 0.94
Eval -    loss: 1.27     top-1: 0.73    top-5: 0.89    top-10: 0.92
We talked about what we could do.
Nous avons parlé de ce que nous pourrions faire.

Epoch 3
Train -   loss: 1.05     top-1: 0.75    top-5: 0.91    top-10: 0.94
Eval -    loss: 1.27     top-1: 0.73    top-5: 0.89    top-10: 0.92
I think Tom is here.
Je pense que Tom est là.

Epoch 4
Train -   loss: 1.03     top-1: 0.75    top-5: 0.91    top-10: 0.94
Eval -    loss: 1.27     top-1: 0.73    top-5: 0.89    top-10: 0.92
What year were you born?
```

```
En quelle année es-tu né ?

Epoch 5
Train -   loss: 1.06     top-1: 0.74    top-5: 0.91    top-10: 0.94
Eval -    loss: 1.26     top-1: 0.73    top-5: 0.89    top-10: 0.92
Don't say a word to anyone.
Ne dites pas un mot à qui que ce soit.

Epoch 6
Train -   loss: 1.05     top-1: 0.75    top-5: 0.91    top-10: 0.94
Eval -    loss: 1.27     top-1: 0.73    top-5: 0.89    top-10: 0.92
The doctor says she suffers from rheumatism.
Le docteur dit qu'elle souffre d'accepter.

Epoch 7
Train -   loss: 1.04     top-1: 0.75    top-5: 0.91    top-10: 0.94
```

Colab paid products - Cancel contracts here