Headline Classification with Neural BOW, LSTM

CS 4650 "Natural Language Processing" Project 1
Georgia Tech, Spring 2025 (Instructor: Weicheng Ma)

Welcome to the first full programming project for CS 4650! **To start, first make a copy of this notebook to your local drive, so you can edit it.**

If you want GPUs (which will improve training times), you can always change your instance type by going to Runtime -> Change runtime type -> Hardware accelerator.

In this project, we will be using PyTorch. If you are new to PyTorch, or simply want a refresher, we recommend you start by looking through these Introduction to PyTorch slides and this interactive PyTorch basics notebook. Additionally, this text sentiment notebook will provide some insight into working with PyTorch with a specific NLP task.

1. Load and preprocess data [10 points]

This project will be modeling a *classification task* for headlines from The Onion, a satirical news website. Our dataset contains headlines and whether they belong to The Onion or CNN. Given a headline, we want to predict whether it is Onion or not.

The following cell loads, pre-processes and tokenizes our OnionOrNot dataset.

```
!curl -so OnionOrNot.csv
https://raw.githubusercontent.com/lukefeilberg/onion/master/OnionOrNot
.csv

#
================================================================================
=====
# Run some setup code for this notebook. Don't modify anything in this
cell.
#
================================================================================
=====

import torch
import random, sys

RANDOM_SEED = 42
torch.manual_seed(RANDOM_SEED)
random.seed(RANDOM_SEED)

#
================================================================================
```

```
=====
# A quick note on CUDA functionality (and `.to(model.device)`):
# CUDA is a parallel GPU platform produced by NVIDIA and is used by
most GPU
# libraries in PyTorch. CUDA organizes GPUs into device IDs (i.e.,
"cuda:X" for GPU #X).
# "device" will tell PyTorch which GPU (or CPU) to place an object in.
Since
# collab only uses one GPU, we will use 'cuda' as the device if a GPU
is available
# and the CPU if not. You will run into problems if your tensors are
on different devices.
#
================================================================================
=====
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')

# Check what version of Python is running
print(sys.version)
```

```
3.12.0 (tags/v3.12.0:0fb18b0, Oct  2 2023, 13:03:39) [MSC v.1935 64
bit (AMD64)]
```

1.1 Dataset preprocessing functions

The following cell define some methods to clean the dataset, but feel free to take a look to see some of the operations it's doing.

```
#
================================================================================
=====
# Run some preprocessing code for our dataset. Don't modify anything
in this cell.
# This code was adapted from fast-bert.
#
================================================================================
=====

import re
import html

def spec_add_spaces(t: str) -> str:
    "Add spaces around / and # in `t`. \n"
    return re.sub(r"([/#\n])", r" \1 ", t)

def rm_useless_spaces(t: str) -> str:
    "Remove multiple spaces in `t`."
    return re.sub(" {2,}", " ", t)
```

```
def replace_multi_newline(t: str) -> str:
    return re.sub(r"(\n(\s)*){2,}", "\n", t)

def fix_html(x: str) -> str:
    "List of replacements from html strings in `x`."
    re1 = re.compile(r"  +")
    x = (
        x.replace("#39;", "'")
        .replace("amp;", "&")
        .replace("#146;", "'")
        .replace("nbsp;", " ")
        .replace("#36;", "$")
        .replace("\\n", "\n")
        .replace("quot;", "'")
        .replace("<br />", "\n")
        .replace('\\"', '"')
        .replace(" @.@ ", ".")
        .replace(" @-@ ", "-")
        .replace(" @,@ ", ",")
        .replace("\\", " \\ ")
    )
    return re1.sub(" ", html.unescape(x))

def clean_text(input_text):
    text = fix_html(input_text)
    text = replace_multi_newline(text)
    text = spec_add_spaces(text)
    text = rm_useless_spaces(text)
    text = text.strip()
    return text
```

1.2 Tokenize using NLTK

We will use our rule-based `clean_text` function to clean our raw text, then use the popular NLTK punkt tokenizer to convert text to individual sub-words. This will take a while because you have to download the pre-trained punkt tokenizer.

If you are interested: There's a long and diverse history of converting raw text to "tokens", and many available methods/algorithms (you can experiment with some recently trained ones, trained on a dynamic programming-based method called BPE, here).

```
#
=============================================================================
=====
# Tokenize using punkt. Don't modify anything in this cell.
#
=============================================================================
=====
```

```
import pandas as pd
import nltk
from tqdm import tqdm

nltk.download('punkt_tab')
nltk.download('punkt')
df = pd.read_csv("OnionOrNot.csv")
df["tokenized"] = df["text"].apply(lambda x:
nltk.word_tokenize(clean_text(x.lower())))
```

```
[nltk_data] Downloading package punkt_tab to C:\Users\Wei
[nltk_data]     Xuan\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt_tab is already up-to-date!
[nltk_data] Downloading package punkt to C:\Users\Wei
[nltk_data]     Xuan\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

We will use `pandas`, a popular library for data analysis and table manipulation, in this project to manage the dataset. For more information on usage, please refer to the Pandas documentation.

The primary data structure in Pandas is a `DataFrame`. The following cell will print out the basic information contained in our `DataFrame` structure, and the first few rows of our dataset.

```
# View the first few entries of our dataset
df.head()
```

```
                                                text  label  \
0  Entire Facebook Staff Laughs As Man Tightens P...      1
1  Muslim Woman Denied Soda Can for Fear She Coul...      0
2  Bold Move: Hulu Has Announced That They’re Gon...      1
3  Despondent Jeff Bezos Realizes He’ll Have To W...      1
4  For men looking for great single women, online...      1

                                           tokenized
0  [entire, facebook, staff, laughs, as, man, tig...
1  [muslim, woman, denied, soda, can, for, fear, ...
2  [bold, move, :, hulu, has, announced, that, th...
3  [despondent, jeff, bezos, realizes, he, ’, ll,...
4  [for, men, looking, for, great, single, women,...
```

Try to guess some examples! Is the task more difficult than you expected?

DataFrames can be indexed using `.iloc[]`. `iloc` uses interger based indexing and supports a single integer (`df.iloc[42]`), a list of integers (`df.iloc[[1, 5, 42]]`), or a slice (`df.iloc[7:42]`).

```
# E.g., get row 42 of our dataset
df.iloc[42]
```

```
text         Customers continued to wait at drive-thru even...
label                                                        0
tokenized    [customers, continued, to, wait, at, drive-thr...
Name: 42, dtype: object
```

1.3 Split the dataset into training, validation, and testing

Train/Test/Val Split - Now that we've loaded this dataset, we need to split the data into train, validation, and test sets.

A good explanation of why we need these different sets can be found in $\S$2.2.5 of Eisenstein but our high-level goal is to have a generalized model and have confidence in our results.

The *training set* is used to fit our model's learned parameters (weights and biases) to the task. The *validation set* (sometimes called development set) is used to verify our training jobs are minimizing loss on an unseen subset of the data and can also be used to help choose hyperparameters for our training setup. The *test set* is used to provide a final evaluation of our trained model (unbiased by development or training decisions), ideally providing some insight into how the model will perform in a scenario we cannot perfectly represent in our data (i.e., the real world). *Each of these sets should be disjoint from the others*, to prevent any leackage that could introduce bias in our evaluation metrics (in this case accuracy).

Model Vocabulary - We cannot directly feed sub-word token strings into a model! We need to create a "vocab map", which contains an ID for each unique token in our Onion dataset. This will be used as a "lookup" in the next few sections, since your PyTorch implementation will require first converting your Onion token representations to a list of sub-word IDs.

In the following cell, please implement `split_train_val_test` and `generate_vocab_map`.

```
#
============================================================================
=====
# Set constants for PAD and UNK. You will use these values, but DO NOT
change
# them, or import additional packages.

from collections import Counter
PADDING_VALUE = 0
UNK_VALUE     = 1

#
============================================================================
=====


def split_train_val_test(df, props=[.8, .1, .1]):
    """
    This method takes a dataframe and splits it into train/val/test
splits.
```

```
    It uses the props argument to split the dataset appropriately.

    Args:
      df (pd.DataFrame): A dataset as a Pandas DataFrame
      props (list): Proportions for each split in the order of [train,
validation, test].
                    the last value of the props array is repetitive,
but we've kept it for clarity.

    Returns:
      train_df (pd.DataFrame): Train DataFrame split.
      val_df (pd.DataFrame): Validation DataFrame split.
      test_df (pd.DataFrame): Test DataFramem split.
    """
    assert round(sum(props), 2) == 1 and len(props) >= 2
    train_df, test_df, val_df = None, None, None

    ### BEGIN YOUR CODE (~3-5 lines) ###
    ### Hint: You can use df.iloc to slice into specific indexes or
ranges.
    length = len(df)
    train_df = df.iloc[:int(props[0]*length)]
    val_df = df.iloc[int(props[0]*length):int(props[0]*length +
props[1]*length)]
    test_df = df.iloc[int(props[0]*length + props[1]*length):]
    ### END YOUR CODE ###

    return train_df, val_df, test_df


def generate_vocab_map(df, cutoff=2):
    """
    This method takes a dataframe and builds a vocabulary to unique
number map.
    It uses the cutoff argument to remove rare words occuring <=
cutoff times.
    *NOTE*: "" and "UNK" are reserved tokens in our vocab that will be
useful
    later. You'll also find the Counter imported for you to be useful
as well.

    Args:
      df (pd.DataFrame): The entire dataset this mapping is built from
      cutoff (int): We exclude words from the vocab that appear less
than or
                    eq to cutoff

    Returns:
      vocab (dict[str, int]):
        In vocab, each str is a unique token, and each dict[str] is a
```

```
        unique integer ID. Only elements that appear > cutoff times
appear
        in vocab.

      reversed_vocab (dict[int, str]):
        A reversed version of vocab, which allows us to retrieve
        words given their unique integer ID. This map will
        allow us to "decode" integer sequences we'll encode using
        vocab!
    """

    vocab          = {"": PADDING_VALUE, "UNK": UNK_VALUE}
    reversed_vocab = None

    ### BEGIN YOUR CODE (~5-15 lines) ###
    ### Hint: Start by iterating over df["tokenized"]
    c = Counter()

    for i in df["tokenized"]:
        c.update(i)

    for i in c:
        if c[i] > cutoff:
            vocab[i] = len(vocab)

    reversed_vocab = {v:k for k, v in vocab.items()}

    ### END YOUR CODE ###

    return vocab, reversed_vocab
```

With the methods you have implemented above, we can now split the dataset into training, validation, and testing sets and generate our dictionaries mapping from word tokens to IDs (and vice versa).

Note: The props list currently being used splits the dataset so that 80% of samples are used to train, and the remaining 20% are evenly split between training and validation. How you split your dataset is itself a major choice and something you would need to consider in your own projects. Can you think of why?

```
df                          = df.sample(frac=1)
train_df, val_df, test_df   = split_train_val_test(df, props=[.8, .1,
.1])
train_vocab, reverse_vocab = generate_vocab_map(train_df)

#
================================================================
=====
# This line of code will help test your implementation, the expected
output is
```

```
# the same distribution used in 'props' in the above cell. Try out
some
# different values to ensure it works, but for submission ensure you
use
# [.8, .1, .1]
#
==========================================================================
=====

(len(train_df) / len(df)), (len(val_df) / len(df)), (len(test_df) /
len(df))
```

```
(0.8, 0.1, 0.1)
```

1.4 Building a Dataset Class

PyTorch has custom Dataset Classes that have very useful extentions, we want to turn our current pandas DataFrame into a subclass of Dataset so that we can iterate and sample through it for minibatch updates. **In the following cell, fill out the `HeadlineDataset` class.** Refer to PyTorch documentation on Dataset Classes for help.

```
#
==========================================================================
=====
# Please do not change, or import additional packages.
from torch.utils.data import Dataset
#
==========================================================================
=====

class HeadlineDataset(Dataset):
  """
  This class takes a Pandas DataFrame and wraps in a PyTorch Dataset.
  Read more about Torch Datasets here:
  https://pytorch.org/tutorials/beginner/basics/data_tutorial.html
  """

  def __init__(self, vocab, df, max_length=50):
    """
    Initialize this class with appropriate instance variables

    We would *strongly* recommend storing the dataframe itself as an
instance
    variable, and keeping this method very simple. Leave processing to
    __getitem__.

    Sometimes, however, it does make sense to preprocess in __init__.
If you
    are curious as to why, read the aside at the bottom of this cell.
```

```
    """

    ### BEGIN YOUR CODE (~3 lines) ###
    self.vocab = vocab
    self.df = df
    self.maxLength = max_length
    return
    ### END YOUR CODE ###

  def __len__(self):
    """
    Return the length of the dataframe instance variable
    """
    df_len = None

    ### BEGIN YOUR CODE (~1 line) ###
    df_len = len(self.df)
    ### END YOUR CODE ###

    return df_len

  def __getitem__(self, index: int):
    """
    Converts a dataframe row (row["tokenized"]) to an encoded torch
LongTensor,
    using our vocab map created using generate_vocab_map. Restricts
the encoded
    headline length to max_length.

    The purpose of this method is to convert the row - a list of words
- into
    a corresponding list of numbers.

    i.e. using a map of {"hi": 2, "hello": 3, "UNK": 0}
    this list ["hi", "hello", "NOT_IN_DICT"] will turn into [2, 3, 0]

    Returns:
      tokenized_word_tensor (torch.LongTensor):
        A 1D tensor of type Long, that has each token in the dataframe
mapped to
        a number. These numbers are retrieved from the vocab_map we
created in
        generate_vocab_map.

        **IMPORTANT**: if we filtered out the word because it's
infrequent (and
        it doesn't exist in the vocab) we need to replace it w/ the
UNK token.

      curr_label (int):
```

```
        Binary 0/1 label retrieved from the DataFrame.
    """
    tokenized_word_tensor = None
    curr_label            = None

    ### BEGIN YOUR CODE (~3-7 lines) ###
    temp = []
    for i in self.df.iloc[index]['tokenized']:
        temp.append(self.vocab.get(i, UNK_VALUE))
    if len(temp) > self.maxLength:
        temp = temp[:self.maxLength]
    else:
        temp = temp + [PADDING_VALUE] * (self.maxLength - len(temp))
    tokenized_word_tensor = torch.LongTensor(temp)
    curr_label = self.df.iloc[index]['label']
    ### END YOUR CODE ###

    return tokenized_word_tensor, curr_label


#
================================================================================
=====
# Completely optional aside on preprocessing in __init__.
#
# Sometimes the compute bottleneck actually ends up being in
__getitem__.
# In this case, you'd loop over your dataset in __init__, passing data
# to __getitem__ and storing it in another instance variable. Then,
# you can simply return the preprocessed data in __getitem__ instead
of
# doing the preprocessing.
#
# There is a tradeoff though: can you think of one?
#
================================================================================
=====

from torch.utils.data import RandomSampler

train_dataset = HeadlineDataset(train_vocab, train_df)
val_dataset   = HeadlineDataset(train_vocab, val_df)
test_dataset  = HeadlineDataset(train_vocab, test_df)

# Now that we're wrapping our dataframes in PyTorch datsets, we can
make use of
# PyTorch Random Samplers, they'll define how our DataLoaders sample
elements
# from the HeadlineDatasets
```

```
train_sampler = RandomSampler(train_dataset)
val_sampler   = RandomSampler(val_dataset)
test_sampler  = RandomSampler(test_dataset)
```

1.5 Finalizing our DataLoader

We can now use PyTorch `DataLoader` to batch our data for us. **In the following cell, please implement `collate_fn`.** Refer to PyTorch documentation on `DataLoader` for help.

```
#
=============================================================================
=====
# Please do not change, or import additional packages.
from torch.nn.utils.rnn import pad_sequence
#
=============================================================================
=====

def collate_fn(batch, padding_value=PADDING_VALUE):
  """
  This function is passed as a parameter to Torch DataSampler.
collate_fn collects
  batched rows, in the form of tuples, from a DataLoader and applies
some final
  pre-processing.

  Objective:
    In our case, we need to take the batched input array of 1D
tokenized_word_tensors,
    and create a 2D tensor that's padded to be the max length from all
our tokenized_word_tensors
    in a batch. We're moving from a Python array of tuples, to a
padded 2D tensor.

    *HINT*: you're allowed to use torch.nn.utils.rnn.pad_sequence
(ALREADY IMPORTED)

    Finally, you can read more about collate_fn here:
https://pytorch.org/docs/stable/data.html

  Args:
    batch: PythonArray[tuple(tokenized_word_tensor: 1D
Torch.LongTensor, curr_label: int)]
           len(batch) == BATCH_SIZE

  Returns:
    padded_tokens: 2D LongTensor of shape (BATCH_SIZE, max len of all
tokenized_word_tensor))
    y_labels: 1D FloatTensor of shape (BATCH_SIZE)
```

```
  """
  padded_tokens, y_labels = None, None

  ### BEGIN YOUR CODE (~4-8 lines) ###
  token_tensor = [i[0] for i in batch]
  labels = [i[1] for i in batch]
  padded_tokens = pad_sequence(token_tensor, batch_first = True,
padding_value=padding_value)
  y_labels = torch.FloatTensor(labels)
  ### END YOUR CODE ###

  return padded_tokens, y_labels

from torch.utils.data import DataLoader
BATCH_SIZE = 16

train_iterator = DataLoader(train_dataset, batch_size=BATCH_SIZE,
sampler=train_sampler, collate_fn=collate_fn)
val_iterator   = DataLoader(val_dataset, batch_size=BATCH_SIZE,
sampler=val_sampler, collate_fn=collate_fn)
test_iterator  = DataLoader(test_dataset, batch_size=BATCH_SIZE,
sampler=test_sampler, collate_fn=collate_fn)

#
================================================================================
=====
# Use this to test your collate_fn implementation.
#
# You can look at the shapes of x and y or put print statements in
collate_fn
# while running this snippet
#
================================================================================
=====

for x, y in test_iterator:
    print(x, y)
    print(f'x: {x.shape}')
    print(f'y: {y.shape}')
    break
test_iterator = DataLoader(test_dataset, batch_size=BATCH_SIZE,
sampler=test_sampler, collate_fn=collate_fn)
```

```
tensor([[6222, 3096,    1,   46, 2201,  402,  303,  330,  937, 6148,
223,   97,
         2297,    0,    0,    0,    0,    0,    0,    0,    0,    0,
0,    0,
            0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
0,    0,
```

```
            0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
0,    0,
            0,    0],
        [4880,  112, 1143, 4318, 2275,    8, 6069,    0,    0,    0,
0,    0,
            0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
0,    0,
            0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
0,    0,
            0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
0,    0,
            0,    0],
        [  86,  318, 1338, 3857, 7158,  160, 1362,   16,    1,    0,
0,    0,
            0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
0,    0,
            0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
0,    0,
            0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
0,    0,
            0,    0],
        [2471,  259,  885,  886,  221,   97,    1, 1496,   16, 1777,
661,  230,
         6781,   14, 5977,  885,  886,    0,    0,    0,    0,    0,
0,    0,
            0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
0,    0,
            0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
0,    0,
            0,    0],
        [ 385, 4112, 3736, 2589, 1773, 3655, 1788,  160,    1, 3856,
66,    0,
            0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
0,    0,
            0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
0,    0,
            0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
0,    0,
            0,    0],
        [ 158,   91, 5292,    0,    0,    0,    0,    0,    0,    0,
0,    0,
            0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
0,    0,
            0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
0,    0,
            0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
0,    0,
            0,    0],
        [5331,  560, 1034, 2473,   18, 8156,  106, 2802, 7333,    1,
```

```
1,    0,
           0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
0,    0,
           0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
0,    0,
           0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
0,    0,
           0,    0],
         [ 186, 2960, 7488,    4,  891,   14,    1,  937,  410,  207,
54,    1,
          5662,    0,    0,    0,    0,    0,    0,    0,    0,    0,
0,    0,
           0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
0,    0,
           0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
0,    0,
           0,    0],
         [ 459,  460,  152,  884,  416,  789,  115,    1,   66,  115,
542,    1,
             66,    0,    0,    0,    0,    0,    0,    0,    0,    0,
0,    0,
           0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
0,    0,
           0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
0,    0,
           0,    0],
         [3027,   20,    1, 3294,  293,   43, 3711,   14,   60, 3060,
106, 1771,
           14, 6254, 5131,    0,    0,    0,    0,    0,    0,    0,
0,    0,
           0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
0,    0,
           0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
0,    0,
           0,    0],
         [   1,  119, 2168,    4, 2335,   16, 1395,    1, 1050, 1672,
0,    0,
           0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
0,    0,
           0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
0,    0,
           0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
0,    0,
           0,    0],
         [2186, 8923, 4422, 4423,  599,  561,   97, 1921,   20,  270,
1,   86,
           0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
0,    0,
           0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
```

```
0,    0,
            0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
0,    0,
            0,    0],
        [  54,   63,   29,  395,   14,  215,  581,  753, 4955,  115,
221, 2075,
           14,    1, 6235,  448, 2725, 8471,    0,    0,    0,    0,
0,    0,
            0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
0,    0,
            0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
0,    0,
            0,    0],
        [1484, 1485,  152,  366, 3580,   91,  785, 1456,   20,  104,
2227,  399,
           78,  339,  366,    0,    0,    0,    0,    0,    0,    0,
0,    0,
            0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
0,    0,
            0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
0,    0,
            0,    0],
        [ 849,  671,  261,  764, 1924, 3078,  456,  237, 7019, 1042,
0,    0,
            0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
0,    0,
            0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
0,    0,
            0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
0,    0,
            0,    0],
        [ 785,  152, 3210,  419,   12,   40,   52, 1938, 5092,   18,
1119, 1771,
          397, 8399,    0,    0,    0,    0,    0,    0,    0,    0,
0,    0,
            0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
0,    0,
            0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
0,    0,
            0,    0]]) tensor([0., 0., 0., 1., 0., 1., 0., 0., 0., 0.,
1., 0., 0., 0., 0., 1.])
x: torch.Size([16, 50])
y: torch.Size([16])
```

2. Modeling [10 points]

Now that we have a clean dataset and a useful PyTorch `DataLoader` object, we can begin building a model for our task! In the following code block, you will build a feed-forward neural network implementing a neural bag-of-words baseline, `NBOW-RAND`, described in $\S$2.1 of this

paper. You may find the PyTorch `torch.nn` docs useful for understanding the different layers and this PyTorch sequence models tutorial for how to put together `torch.nn` layers.

The core intuition behind `NBOW-RAND` is that after we embed each word for our input, we average the embeddings to produce a single vector that hopefully averages the information across all embeddings. Formally, we first convert each document of length n tokens into a matrix of $n \times d$, where d is the dimension of the token embedding. Then we average all embeddings to produce a vector of length d.

If you are new to PyTorch, ensuring your matrix operations are correct is often the most common source of errors. Keep in mind how the dimensions change and what each axes represents. Your documents will be passed in as minibatches, so be careful when selecting which axes to apply certain operations. Feel free to experiment with the architecture of this network outside of the basic `NBOW-RAND` setup (such as adding in other layers) to see how this changes your results.

2.1 Define the NBOW model class

```
#
================================================================================
=====
# Please do not change, or import additional packages.
import torch.nn as nn
#
================================================================================
=====

class NBOW(nn.Module):
  def __init__(self, vocab_size, embedding_dim):
    """
    Instantiate layers for your model.
    Your model architecture will be a feed-forward neural network.

    You will need 3 nn.Modules at minimum
     1. An embeddings layer (see nn.Embedding)
     2. A linear layer (see nn.Linear)
     3. A sigmoid output (see nn.Sigmoid)

    HINT: In the forward step, the BATCH_SIZE is the first dimension.
    """
    super().__init__()

    ### BEGIN YOUR CODE (~4 lines) ###
    self.embedding = nn.Embedding(vocab_size, embedding_dim)
    self.linear = nn.Linear(embedding_dim, 1)
    self.sigmoid = nn.Sigmoid()

    ### END YOUR CODE ###

  def forward(self, x):
    """
```

```
    Complete the forward pass of the model.

    Use the output of the embedding layer to create the average
vector,
    which will be input into the linear layer.

    Args:
      x: 2D LongTensor of shape (BATCH_SIZE, max len of all
tokenized_word_tensor))
         This is the same output that comes out of the collate_fn
function you completed
    """
    ### BEGIN YOUR CODE (~4-5 lines) ###
    x = self.embedding(x)
    x = torch.mean(x, dim = 1)
    x = self.linear(x)
    x = self.sigmoid(x)

    return x
    ### END YOUR CODE ###
```

2.2 Initialize the NBOW classification model

Since the NBOW model is rather basic, there is only one meaningful hyperparameter w.r.t. model architecture: the size of the embedding dimension (`embedding_dim`). (We also see a `vocab_size` parameter here, but this only a by-product on our cutoff for infrequent tokens, there also may more hyperparameters if you modified the architecture, such as adding a linear layer).

Remember the CUDA discussion in the first cell of this notebook? Here the `.to(device)` is where that discussion becomes relevant (if `device=='cuda'`, PyTorch will perform the matrix operations on GPU). If you recieve a mismatch error, your tensors may be on different devices.

```
model = NBOW(
  vocab_size    = len(train_vocab.keys()),
  embedding_dim = 300
).to(device)
```

2.3 Instantiate the loss function and optimizer

Please select and instantiate an appropriate loss function and optimizer.

Hint: What loss functions are availible for binary classification? Feel free to look at the torch.nn docs on loss functions for help!

```
# While we import Adam for you, you may try / import other optimizers
as well
from torch.optim import Adam
```

```
criterion, optimizer = None, None

### BEGIN YOUR CODE ###
criterion = nn.BCELoss()
optimizer = Adam(model.parameters())
### END YOUR CODE ###
```

Now that we have a NBOW model, a loss function, optimizer and dataset, we can begin training!

3. Training and Evaluation [10 points]

We will now instantiate a `train_loop`, and a `val_loop` to evaluate our model at each epoch.

Fill out the train and test loops below. Treat real headlines as `False`, and Onion headlines as `True`.

```
def train_loop(model, criterion, optim, iterator):
  """
  Returns the total loss calculated from criterion
  """
  model.train()
  total_loss = 0
  for x, y in tqdm(iterator):
    ### BEGIN YOUR CODE (~6 lines) ###
    x, y = x.to(device), y.to(device)
    optim.zero_grad()
    y_pred = model(x)
    loss = criterion(y_pred, y.unsqueeze(1))
    loss.backward()
    optim.step()

    total_loss += loss.item()
    ### END YOUR CODE ###

  return total_loss


def val_loop(model, iterator):
  """
  Returns:
    true (List[bool]): All the ground truth values taken from the
dataset iterator
    pred (List[bool]): All model predictions.
  """
  true, pred = [], []

  ### BEGIN YOUR CODE (~8 lines) ###
  model.eval()

  for x, y in tqdm(iterator):
```

```
    y_pred = model(x)
    pred.extend(y_pred.round().tolist())
    true.extend(y.bool().tolist())

  ### END YOUR CODE ###

  return true, pred
```

3.1 Define the evaluation metrics

We will also need evaluation metrics to tell us how well our model is doing on the validation set at each epoch and later how well the model does on the held-out test set. You may find $\ S$4.4.1 of Eisenstein useful for these questions.

Complete the functions in the following cell.

```
# Note: You will not need to import anything to implement these
functions.

def accuracy(true, pred):
  """
  Calculate the ratio of correct predictions.

  Args:
    true (List[bool]): ground truth
    pred (List[bool]): model predictions

  Returns:
    acc (float): percent accuracy with range [0, 1]
  """
  acc = None
  ### BEGIN YOUR CODE (~2-5 lines) ###
  pred = [i[0] for i in pred]
  num_correct = 0

  for i in range(len(true)):
      if true[i] == pred[i]:
         num_correct += 1
  acc = num_correct / len(true)


  ### END YOUR CODE ###
  return acc


def binary_f1(true, pred, selected_class=True):
  """
  Calculate F-1 scores for a binary classification task.

  Args:
```

```
    true (List[bool]): ground truth
    pred (List[bool]): model predictions
    selected_class (bool): the selected class the F-1 is being
calculated for.

  Returns:
    f1 (float): F-1 score between [0, 1]
  """
  f1 = None
  ### BEGIN YOUR CODE (~10-15 lines) ###
  tp, fp, fn = 0, 0, 0
  pred = [i[0] for i in pred]

  for i in range(len(true)):
      if pred[i] == selected_class:
        if true[i] == selected_class:
           tp += 1
        else:
           fp += 1
      elif true[i] == selected_class:
        fn += 1

  precision = tp / (tp + fp) if (tp + fp) > 0 else 0
  recall = tp / (tp + fn) if (tp + fn) > 0 else 0

  f1 = 2 * (precision * recall) / (precision + recall) if (precision +
recall) > 0 else 0

  ### END YOUR CODE ###
  return f1


def binary_macro_f1(true, pred):
  """
  Calculate averaged F-1 for all selected (true/false) classes.

  Args:
    true (List[bool]): ground truth
    pred (List[bool]): model predictions
  """
  averaged_macro_f1 = None
  ### BEGIN YOUR CODE (~1 line) ###
  averaged_macro_f1 = (binary_f1(true, pred, True) + binary_f1(true,
pred, False)) / 2

  ### END YOUR CODE ###
  return averaged_macro_f1

#
==========================================================================
```

```
=====
# To test your eval implementation, we will evaluate the untrained
model on our
# dev dataset. It will do pretty poorly (it's untrained), but the
exact performance
# will be random since the initialization of the model parameters is
random.
#
================================================================================
=====

true, pred = val_loop(model, val_iterator)
print(f'Binary Macro F1: {binary_macro_f1(true, pred)}')
print(f'Accuracy: {accuracy(true, pred)}')
```

```
100%|██████████| 150/150 [00:00<00:00, 338.52it/s]

Binary Macro F1: 0.2707383773928897
Accuracy: 0.37125
```

4. Full Training Run [1 point]

Now we can perform a full run and see the model fit our loss. If everything goes correctly, you should be able to achieve a validation F1 score of at least `0.80`

Feel free to adjust the number of epochs to prevent overfitting or underfitting and to play with your model hyperparameters/optimizer & loss function.

```
TOTAL_EPOCHS = 10
for epoch in range(TOTAL_EPOCHS):
    train_loss = train_loop(model, criterion, optimizer,
train_iterator)
    true, pred = val_loop(model, val_iterator)
    print(f"EPOCH: {epoch}")
    print(f"TRAIN LOSS: {train_loss}")
    print(f"VAL F-1: {binary_macro_f1(true, pred)}")
    print(f"VAL ACC: {accuracy(true, pred)}")
```

```
100%|██████████| 1200/1200 [00:14<00:00, 85.67it/s]
100%|██████████| 150/150 [00:00<00:00, 495.24it/s]

EPOCH: 0
TRAIN LOSS: 677.1727117598057
VAL F-1: 0.7426842290579556
VAL ACC: 0.7829166666666667

100%|██████████| 1200/1200 [00:13<00:00, 91.52it/s]
100%|██████████| 150/150 [00:00<00:00, 528.81it/s]
```

```
EPOCH: 1
TRAIN LOSS: 462.75611308962107
VAL F-1: 0.8414387765469298
VAL ACC: 0.85125

100%|██████████| 1200/1200 [00:13<00:00, 90.17it/s]
100%|██████████| 150/150 [00:00<00:00, 304.37it/s]

EPOCH: 2
TRAIN LOSS: 361.79292799532413
VAL F-1: 0.8552835005213388
VAL ACC: 0.8658333333333333

100%|██████████| 1200/1200 [00:13<00:00, 86.24it/s]
100%|██████████| 150/150 [00:00<00:00, 397.75it/s]

EPOCH: 3
TRAIN LOSS: 307.1231104284525
VAL F-1: 0.8608319158806248
VAL ACC: 0.87

100%|██████████| 1200/1200 [00:16<00:00, 72.10it/s]
100%|██████████| 150/150 [00:00<00:00, 270.41it/s]

EPOCH: 4
TRAIN LOSS: 270.68400552496314
VAL F-1: 0.8592179722755053
VAL ACC: 0.8691666666666666

100%|██████████| 1200/1200 [00:17<00:00, 66.75it/s]
100%|██████████| 150/150 [00:00<00:00, 326.72it/s]

EPOCH: 5
TRAIN LOSS: 241.15441867522895
VAL F-1: 0.858870309094558
VAL ACC: 0.86875

100%|██████████| 1200/1200 [00:17<00:00, 68.22it/s]
100%|██████████| 150/150 [00:00<00:00, 345.93it/s]

EPOCH: 6
TRAIN LOSS: 217.62708221375942
VAL F-1: 0.859418092384102
VAL ACC: 0.8691666666666666

100%|██████████| 1200/1200 [00:17<00:00, 70.19it/s]
100%|██████████| 150/150 [00:00<00:00, 288.55it/s]

EPOCH: 7
TRAIN LOSS: 199.39884072169662
VAL F-1: 0.8581872921932667
VAL ACC: 0.86875
```

```
100%|██████████| 1200/1200 [00:18<00:00, 65.94it/s]
100%|██████████| 150/150 [00:00<00:00, 366.56it/s]

EPOCH: 8
TRAIN LOSS: 182.3571298168972
VAL F-1: 0.8587626996151061
VAL ACC: 0.8679166666666667

100%|██████████| 1200/1200 [00:16<00:00, 72.56it/s]
100%|██████████| 150/150 [00:00<00:00, 359.50it/s]

EPOCH: 9
TRAIN LOSS: 167.30877292482182
VAL F-1: 0.8572131688400193
VAL ACC: 0.8670833333333333
```

We can also look at the models performance on the held-out test set, using the same `val_loop` from earlier.

```
true, pred = val_loop(model, test_iterator)
print(f"TEST F-1: {binary_macro_f1(true, pred)}")
print(f"TEST ACC: {accuracy(true, pred)}")

100%|██████████| 150/150 [00:00<00:00, 339.43it/s]

TEST F-1: 0.8614266662854778
TEST ACC: 0.87125
```

5. Analysis [5 points]

While modeling and accuracy are a great signal that our model is working in our specific task setup, an inspection of what the model is classifying (particularly its errors), can allow us to hypothesize about what is going on, why it works, and how to improve.

5.1 Impact of Vocab Size

Question: *What happens to the vocab size as you change the cutoff in the cell below? Can you explain this in the context of Zipf's Law?*

Answer: The size of the vocab decreases as cutoff increases. This is because increasing the cutoff results in only including words that appear at least a certain number of times, hence excluding rare words. Zipf's law notes that frequency of any word is inversely proportional to its rank in the frequency table, where vast majority of words will have lower frequencies and drop-off as we move down the list of ranked words.

```
tmp_vocab, _ = generate_vocab_map(train_df, cutoff = 3)
len(tmp_vocab)
```

```
7623
```

5.2 Error Analysis

Can you describe what cases the model is getting wrong in the witheld test-set?

To do this, you will need to create a new `val_train_loop_incorrect` which returns incorrect sequences **and** you will need to decode these sequences back into words. You have already created a map that can convert encoded sequences back to regular English (`reverse_vocab`).

```
def val_train_loop_incorrect(model, iterator):
  """
  Implement this however you like! It should look very similar to
val_loop.
  Pass the test_iterator through this function to look at errors in
the test set.
  """

  model.eval()

  incorrect = []

  for x, y in tqdm(iterator):
    y_pred = model(x)
    y_pred = y_pred.round()

    for i in range(len(y_pred)):
      if y_pred[i] != y[i]:
          indices = x[i].tolist()
          indices = [i for i in indices if i != PADDING_VALUE]
          words = " ".join([reverse_vocab.get(i, "UNK") for i in
indices])

          incorrect.append(words)

  return incorrect

val_train_loop_incorrect(model, test_iterator)
```

```
100%|██████████| 150/150 [00:00<00:00, 313.75it/s]

['buzzfeed hires clickhole editor',
 'is your interior designer putting your life at risk ?',
 'new haven police officer UNK like dog , tricks suspects into UNK',
 'UNK couple saves money by making own porn',
 'measles UNK must pay doctor UNK',
```

```
 'kelly UNK makes racial UNK while slamming trump for racial UNK',
 "government agencies have to tear down their websites , even if it 's cheaper to leave them up",
 'man on verge of UNK instead turns to god',
 'UNK corpse of jeremy UNK attends UNK board meeting',
 'father of brooklyn teen who died on class field trip gets call asking why son has been UNK',
 'seasons turn UNK from the one that kills old people to the one that kills homeless people',
 'america has found a way to UNK water',
 'melania trump would have been UNK for deportation under new immigration rules',
 'UNK fans put colin kaepernick up for sale on amazon',
 'the oldest person in the world UNK her long life to eating eggs and being single',
 'controversial theory suggests aliens may have built ancient egypt ’ s UNK UNK',
 'latest attack : isis just changed its name to ‘ google ’',
 'national guard UNK hunted homeless with UNK guns',
 'kavanaugh packing gun at congressional hearing in case parkland father tries to shake his hand again',
 'usa today : UNK conflict severely limits tourism in afghanistan',
 'school administration reminds female students bulletproof UNK must cover UNK',
 "by the time UNK harper 's UNK contract UNK , UNK will be a lonely , old , useless UNK",
 'nasa mars rover accidentally draws penis on red planet',
 'most americans would like to punch donald trump',
 'UNK , christ . not another UNK full of urine .',
 'world ’ s top UNK from ‘ UNK is now UNK , will be UNK',
 'your horoscopes – week of february 28 , 2017',
 'UNK might be hidden homosexuals',
 'innocent man ends up UNK with UNK cop that framed him',
 'breaking : drunk teen going 100 mph down UNK highway is UNK',
 'obese man doesn ’ t understand why he can ’ t lose weight despite his healthy , UNK diet',
 '‘ UNK ’ tops amazon UNK list',
 'UNK mix-up puts tony UNK in middle of UNK',
 '‘ men are not UNK , ’ says woman who has no idea what it like to take two whole UNK to get to your clothing section at UNK',
 'scott pruitt defends use of 1st armored division for trip to UNK',
 'new UNK school opens for students with interest in receiving UNK education',
 'if first lady michelle obama could pick any other job , ‘ i would be beyonce ’',
 'donald trump jr. takes son on hunting trip in national zoo',
 'paul ryan ’ s first challenge as house speaker : getting the smell of smoke left by boehner out of the speaker ’ s office',
 'people help themselves to UNK of thousands of onions found in middle
```

```
of desert',
 'UNK UNK fends off burglar with back UNK',
 'police : student had UNK bad plans for school shooting',
 'everyone in pride parade straight',
 'ancient UNK erection UNK in amber',
 'UNK informs george h.w . bush that dying so soon after wife would
really boost UNK rating',
 'gang of UNK UNK UNK UNK by UNK police following UNK shop assault',
 'white house claims iran behind attack on nancy UNK',
 'forgotten sour cream leads to milwaukee fast food shooting',
 'earth ’ s last male northern white rhino gets personal armed UNK',
 'man in gorilla suit shot with UNK UNK when vet confused him for real
gorilla',
 'news : a UNK of justice : a shocking study has found that as many as
1 in 10 people burned at the stake for witchcraft is falsely accused',
 'UNK : stay with your UNK UNK UNK face ‘ discipline ’',
 'UNK to britain : massive power UNK caused by millions of people
simultaneously making tea',
 'UNK confirms UNK dragon from ‘ UNK ’ pregnant',
 'comcast now just calling its customers assholes to their faces',
 'the secret to my UNK marriage is trust , respect , and threatening
to kill myself if she leaves',
 'depression , UNK UNK combine forces to produce UNK UNK UNK',
 'forest service considering explosives to get rid of frozen cows in
colorado mountain cabin .',
 'fema UNK emergency UNK pills for residents stranded by hurricane
florence',
 'painting hanging in UNK store must be founder of the UNK army',
 'study finds medical marijuana effective for UNK long-term pain over
jerry UNK ’ s death',
 'amputee inspires others not to lose limbs',
 'baltimore craigslist poster offering ‘ pokémon go UNK ’ services',
 'dea never checked if its massive surveillance operations are legal',
 'truck full of UNK UNK on u.s. 101 , UNK UNK with UNK sea creatures',
 'pope francis offers molested kids 10 % off at vatican city gift
shop',
 'kentucky players UNK over losing UNK season bonuses',
 'team UNK 2 poster mistaken for us propaganda on russian state
television',
 'man UNK cashier for putting chips and UNK goods in same grocery
bag',
 'lean in , queen ! her husband told her the only room off limits in
his UNK estate was his private study , but she went in anyway and
totally owned her horrible fate',
 'i had a terrible experience at this restaurant because i am a
terrible person',
 'u.s. will not seek to prosecute ancient tribe who murdered UNK john
allen UNK',
 'kanye west doesn ’ t like appearing on keeping up with the
```

```
kardashians because he has issues with the UNK',
 'ted cruz asks central park UNK cab driver how much it costs to UNK
horse for an hour',
 'man somehow getting worse at sex',
 'neither the time nor place : this girl wrote that she and her
boyfriend have had their ups and downs in the UNK of the instagram she
posted for his birthday',
 'UNK ben UNK claims',
 'UNK government UNK students from UNK in showers by handing out
digital UNK to limit shower time',
 'disturbing teen trend : UNK across the country are getting together
weekly to worship a dead man named jesus christ',
 'warren buffett eats like a UNK as they have UNK death rate',
 'u.s. protects already extinct UNK UNK',
 '‘ i ’ m going to get you UNK ’ : UNK chases man who UNK her in
cambridge',
 'life : first we gave this girl a barbie . then we gave her a doll
with normal UNK . then we gave her a doll with goat UNK',
 'popeyes UNK chick-fil-a UNK with new sandwich featuring dan UNK ’ s
battered , fried loved ones',
 "doctors say average heart attack victim does n't UNK at chest nearly
UNK enough",
 'study : most serial UNK did not receive toy every time they went to
store as kids',
 'elon musk hires onion writers for project . ( not an onion
headline )',
 "us taxpayers getting cut of UNK of the christ ' UNK",
 'eric UNK : schools ‘ pushing the liberal agenda ’ by teaching UNK',
 '‘ the UNK ’ turns 20',
 'gop website declares mike pence winner of vp debate before it
begins',
 'such is life : three spaghetti UNK and two spaghetti wins !',
 '300,000 pounds of rat meat sold as chicken wings across america',
 'panicked , UNK pope UNK UNK ban on abortion',
 "hundreds gather to stare at UNK construction site hole and say UNK '
like UNK wilson",
 'UNK duncan spends visit to local elementary school looking at ufo
books in library',
 'UNK at this mess : french park trains UNK to pick up litter',
 'indiana couple UNK goal of visiting every cracker barrel',
 'UNK chan UNK UNK UNK solar panel efficiency by a massive 22 %',
 'i ’ m so glad uncle joe is UNK again',
 'fire UNK factory destroyed in massive blaze',
 'part of the 1 % : bernie sanders is UNK in the polls after a dna
test revealed that he ’ s king',
 'image of kissing UNK has been named the single work of art that best
UNK british identity',
 'mom UNK into school office and slaps wrong child',
 'louis UNK . fan disappointed at lack of UNK power games in new
```

```
material',
 'the roomba for UNK is really UNK off astronomers',
 'this guy was loved so much both wife , girlfriend place UNK in
newspaper',
 'jay-z tried to have UNK ’ s name changed',
 'town in new jersey to fine UNK for driving through town',
 'UNK legendary actor christopher lee set to unleash a metal album
next week',
 'guantanamo guards beat a prisoner into brain injuries who later UNK
out to be an undercover guard who was taking part in a training
exercise',
 'UNK hammer not actually a fan of UNK',
 "i ca n't stand it when jews talk during movies",
 'UNK battle furiously over jennifer UNK',
 'sean spicer given own press secretary to answer media ’ s questions
about his controversial statements',
 '8 families find out they have been paying respects to the wrong UNK
for 39 years',
 'UNK UNK rocks back and forth in UNK while watching arby ’ s clap
back at burger king on twitter',
 'study : women fake UNK to increase sexual UNK',
 'study : 7 of 10 most UNK u.s. hospitals are UNK',
 'japanese family puts aging robot in retirement home',
 'habitat for humanity investigated for working conditions after UNK
UNK collapses on site',
 "trump on black supporter : UNK at my african-american over here '",
 "`` on UNK , it probably was n't the best decision '' - man regrets
buying 7,000 lance armstrong UNK tips UNK",
 "UNK UNK out to prove he 's worth UNK contract",
 'former lovers meet in coffee shop for one last UNK',
 'ice argues migrants in camps are free to die at any time',
 'for the sixth time in one week , man shot at gun show',
 'police subject man to 8 anal searches after minor traffic
violation',
 'UNK black unveils the song we ’ ve all been waiting for : ‘ saturday
’',
 'depressed businessman takes 16 power naps a day',
 'kellyanne conway decides to lay low until rule of law dies down',
 'little miss hispanic delaware stripped of title , because she ’ s
not latina enough',
 'UNK UNK UNK pray owner gets job soon',
 "birth horror : baby 's head torn off during birth",
 'russian lawyer admits to repeatedly informing kremlin of trump
campaign ’ s UNK',
 'UNK UNK UNK UNK after learning bob UNK wrote a president book
without him',
 'high court to doctors : write UNK UNK',
 "nra says mass shootings just the UNK price of protecting people 's
freedom to commit mass shootings",
```

```
 "trump : i always UNK that i was in the military '",
 "probe on UNK goat carcass will take up to a month to see if it 's a
UNK hybrid : official",
 'UNK UNK crab girlfriend wants to move in',
 'boyfriend ’ s UNK an UNK sleeping bag',
 "man does n't even do good job at sleeping",
 'goldman sachs hired by russia as corporate broker to boost image',
 'e3 attendees flee in terror after bethesda presentation UNK causes
UNK to UNK on convention floor',
 'butt of their jokes',
 '‘ grab her hand and put it right on your dick : ’ UNK successfully
UNK his terrible dating book',
 'real-life UNK : an app ’ s high score UNK someone a cow',
 "the top search result on the onion 's website",
 'UNK ‘ jeopardy ! ’ UNK says key to success is threatening other
contestants with UNK baseball bat during commercials',
 'one dead in hair UNK UNK',
 'tree counter is UNK by how many trees there are',
 'UNK , on reddit , dudes can ’ t stop talking about fucking UNK',
 'UNK father of UNK receives shocking news in the delivery room :
there are no babies',
 'manager of UNK taco bell / kfc secretly considers it mostly a taco
bell',
 'alex jones gets coffee dumped on him in seattle after chasing a guy
down on the street',
 'UNK turtle that UNK through its genitals added to endangered list',
 'air force removes god from chain of command',
 'milwaukee officer who got drunk , let child drive , up for
promotion',
 '10 cat UNK for the blind',
 'large UNK tarantula on the loose',
 "self-conscious panda swears it UNK UNK UNK to it as UNK '",
 'nfl reportedly asking music acts to pay for playing super bowl
halftime show',
 'denver ’ s flaming skull mayor announces plans to UNK magic
mushrooms',
 'why UNK join isis',
 'patriothole : wasting taxpayer money ? the white house reportedly
spends $ UNK a month on a free UNK trial obama forgot to cancel',
 'should the government stop dumping money into a giant hole ?
( youtube )',
 '‘ we will not repeat the mistakes of the 2016 election , ’ vows
nation still using internet',
 'inspiring rescue : this good samaritan in hawaii UNK through UNK to
rescue a dog from drowning',
 'kidnapped teen freed , though freedom is its own kind of prison , is
it not ?',
 'health scare prompts man to start UNK healthier',
 'trump UNK golf trophy to hurricane victims',
```

```
 'mark zuckerberg can ’ t believe india isn ’ t grateful for facebook
’ s free internet',
 '‘ ginger extremist ’ convicted in royal death plot so prince harry
can be king',
 'millionaire is worried it will be awkward when she demands money
from homeless man',
 'man practicing open carry law robbed of gun',
 "lyrics to carly UNK UNK 's next single to be UNK via online poll",
 'black and latina women scientists sometimes mistaken for UNK',
 'perfect UNK does not assault drunk woman',
 'report states dr. phil left recovering guest vodka in his dressing
room',
 'holocaust museum : please stop playing pokémon go here',
 'department of education hires art teacher to spread UNK across all
u.s. public schools',
 'london has already UNK its pollution limits for 2016',
 'rock fans outraged as bob UNK goes UNK',
 'UNK spiders cause UNK car recall for second time',
 'naked man waving american flag leads to large identity theft bust',
 'a beautiful reunion : this high school football player got the
surprise of a lifetime when he removed his helmet to reveal that he
was his father who had been fighting in afghanistan for the past 2
years',
 'news : security breach : edward snowden ’ s robot has been UNK into
the white house front door for 3 hours straight',
 'the onion said bill UNK wanted to join squad of UNK . then , UNK let
him in',
 'math journal accepts UNK paper UNK by computer program',
 'can a mother actually lift a car if her child is trapped under
it ?',
 'supreme court hears case of woman ticketed for not holding UNK UNK',
 'UNK UNK offers UNK cocaine | the onion',
 'UNK important for body : expert',
 '[ theonion ] this is not a dating site . largest in world online
search sex partners',
 'feeling bad about feeling bad can make you feel worse',
 'UNK finally goes too far , removes silent track for copyright
infringement',
 'african-american neighborhood UNK by ask murderer',
 'new express transplant list offers patients kidney or first
available UNK',
 "' i used to look up to you , ' shouts UNK flynn jr. running out of
room after learning father a UNK",
 'man discovers end of UNK UNK after UNK hours of UNK',
 'UNK UNK angels target businesses by posting UNK reviews',
 'man billed thousands for UNK he never received .',
 'lawyers identify dozens more bill cosby victims while UNK potential
UNK',
 '50 % of UNK would rather be UNK by a groundhog in congress',
```

```
 'new poll finds millennials far more likely to politically identify
as UNK than previous generations',
 'visitors to chinese zoo feel UNK after discovering new penguin
display UNK of UNK toys',
 'man pours all his UNK UNK into UNK , removing pizza from oven',
 'women still UNK in medical leadership by men with UNK , study
finds',
 'world of warcraft gamer dies after playing 19 hours straight',
 '6 things that could ’ ve been bought with the $ 1.5 trillion the
government spent developing the UNK fighter jet',
 "'the onion ' to halt UNK assault on trees",
 'new york train UNK reports suspicious UNK , turn out to be machines
used to report suspicious UNK',
 'UNK rich people now have as much UNK as 50 % of the rest of humanity
UNK',
 '‘ to defeat them , i must become them , ’ john kerry says while
putting on black face mask',
 "victim recalls UNK 's breasts , little else",
 "ten years later , cheney haunted by people he did n't UNK to kill in
iraq war",
 'upcoming ‘ game of thrones ’ battle reportedly took 55 days to
shoot',
 'donald trump stares UNK at tiny , aged penis in mirror before
putting on clothes , beginning day',
 'study : fat UNK doesn ’ t help obese people lose weight',
 'dick cheney vice presidential library opens in UNK , UNK underground
cave',
 'the incredible story of how an insurance company thinks a man burned
his house down from UNK away',
 'has prince william ever had a hot dog ? an investigation',
 'sexy women UNK new life into coffin making business',
 'doomsday clock pushed to one minute to UNK after arby ’ s threatens
launch of UNK UNK beef ’ n bacon melt',
 'charles UNK has ashes spread over UNK , UNK iraq',
 'colorado legalizes UNK fireworks',
 'un unveils design for floating city for 10,000 people',
 'new bill would limit abortion to cases where procedure necessary to
save promising political career',
 'medical crisis : george h.w . bush has been rushed to the hospital
for emergency lip UNK surgery',
 'this ‘ smart condom ’ will give UNK into your sex life you probably
didn ’ t want',
 'gaffe as civil service magazine prints poster telling parents to
shoot UNK children',
 "it 's an UNK horror . a 14-year-old girl with special needs
allegedly was raped at school after a teacher 's aide UNK her to act
as UNK to catch an accused sexual predator , a fellow student .",
 'newly discovered cave paintings suggest early man was battling a lot
of UNK demons',
```

```
 'beijing declares scary halloween costumes illegal',
 'jay-z vows not to lose touch with millionaire UNK on UNK throwback
track about buying first yacht',
 'UNK tight pants UNK sperm count',
 'fat kid avoids UNK by swimming with shirt',
 'severed head , UNK body found in same mississippi neighborhood',
 'doctor who made music videos in UNK room facing several UNK suits',
 'a publisher is turning the mueller report into a graphic novel',
 "the us just UNK al-qaeda ’ s job application form . it 's UNK
corporate .",
 'thanks for being so cool about everything -- vladimir putin',
 'trump boys gather UNK of comic books , candy bars for night hiding
from special prosecutors in UNK rose garden fort',
 'mysterious UNK skin disease continues to eat away at baby ’ s face
weeks after being kissed by ted cruz',
 'pope francis tells survivors those who cover up abuse are ‘ s * * t
’ ',
 'giant robot battle : who knew a duel between UNK UNK suits could be
so boring ?',
 'study UNK college education with brain tumor risk raises many
questions',
 'news : changing UNK : golden UNK is UNK itself as a cereal
exclusively for people who are grieving',
 'cracking story : french artist to UNK himself in rock for a week ,
then use body to hatch eggs',
 "trump ca n't recall saying he has one of the world 's best
memories",
 'every day , the tsa catches people smuggling UNK through x-ray
machines . why aren ’ t they doing anything to stop it ?',
 'trump UNK out at ap photographer who UNK empty chairs',
 'UNK confirms up to 100 % UNK in beef products',
 'obama : ‘ we tortured some folks ’ ',
 'touching : the nra is releasing a UNK line of ar-15 rifles to raise
money for the victims in parkland',
 'UNK in korea cross UNK solo',
 'kid rock apparently divorced UNK anderson because of UNK',
 '‘ i ’ m a UNK conservative , ’ says horrifying man 25 years from
now',
 'last thing government worker needed was agency labeling him ‘ UNK
’ ',
 'theresa may : trump told me to sue the eu',
 'drake slams rolling stone over losing cover to philip UNK UNK : ‘ i
’ m disgusted ’ ',
 'congressman who UNK secret service was rejected by secret service',
 '‘ sometimes things have to get worse before they get better , ’ says
man who accidentally turned shower UNK wrong way',
 'no UNK ! government records UNK with UNK tape , paper',
 'american baby names are somehow getting even worse',
 'the onion is UNK at $ 500 UNK , more than many of the UNK it UNK
```

```
( x-post r / til )',
 'massive semen explosion after blaze hits bull artificial UNK
facility , firefighters forced to dodge “ UNK ”',
 'gun control fail : this duck found a gun in a bush and is now
pushing it around the park with its UNK',
 'report : u.s. death rates from drugs , suicide , and alcohol have
UNK increased , but not in a cool rock and roll way',
 'man climbs on playground equipment to tell children where babies
come from',
 'a UNK , UNK UNK led to aliens : UNK marines ‘ weird ai',
 'soccer UNK shot and killed after showing player red card',
 'rats laugh when UNK UNK , top scientists reveal',
 'lawyer for martin shkreli UNK fees five thousand per cent',
 'hundreds of children terrified when movie theatre plays la UNK
instead of detective pikachu',
 'indiana governor insists new law has nothing to do with thing it UNK
intended to do',
 'UNK',
 'all flights grounded after faa officials suddenly realize that man
was not meant to fly',
 'first family gets pet UNK',
 'director seeking UNK unknown actress for next affair',
 'UNK jewish newspaper UNK female world leaders out of charlie UNK
march',
 'life : this one ’ s on her : this woman gave more than $ 120,000 to
an online dating scammer even though the guy had only UNK her ‘ hello
’ and never asked her for money',
 'everyone is totally just UNK it , all the time',
 'tinder users can now choose from 37 gender options',
 "god cites UNK in mysterious ways ' as UNK in killing of 3,000 UNK
new UNK",
 "peta crying UNK over signs that animals ca n't read .",
 'man tricked ex with abortion pill UNK',
 'study links binge eating to stress , UNK , depression , joy , UNK ,
anger , UNK',
 'woman feels UNK since growing out her beard , now looking for love',
 'new UNK UNK UNK in face of UNK .',
 'chicago police department to monitor all UNK with public using new
bullet UNK',
 'u.s. military defends controversial decision to test UNK volcano on
UNK civilians',
 'good samaritan : man shouts sex talk to boy stuck at bottom of
well',
 'a fifth of adults have forgotten how to do UNK or UNK',
 'nba will consider UNK games due to UNK attention UNK',
 "detectives UNK UNK anthony 's ' i killed my daughter ' UNK on
reddit",
 'cleveland hero charles UNK rewarded with burgers for life',
 'UNK family welcomes third child born on same day for third straight
```

```
year',
 'pimp my ride : hamas proudly shows off a tank , turns out to just be
a car',
 'mark UNK claims he would have hit 70 home runs without help of bat',
 'death officially a motherfucker',
 'grand jury indicted the man who filmed eric UNK ’ s killing',
 'gop rep. steve king questions UNK UNK to civilization',
 "new anti-smoking ads warn teens 'it 's gay to smoke '",
 'icy snowball can already tell it going to make 9-year-old cry',
 'fbi warns ‘ UNK UNK ’ UNK could be target for shootings by UNK UNK',
 "chuck e. cheese 's announces new lower prices , but the restaurants
will be UNK",
 'tearful justify holds press conference blaming failed drug test on
contaminated salt UNK']
```

Now that we have our incorrect sequences:
Question: *Can you describe what cases the model is getting wrong in the witheld test-set?*

Answer: The model seems to be replacing important entities/terms with 'UNK', due to low word frequency, which causes it to be unable to maintain context in sentences and hence incorrect outputs. This could point at a possibility where the cutoff was too high and key words were excluded from the vocab.

6. LSTM Model [Extra credit, 4 points]

6.1 Define the RecurrentModel class

Something that has been overlooked in this project (and a significant limitation of the bag-of-words approach) is the sequential structure of language: a word typically only has a clear meaning because of its relationship to the words before and after it in the sequence, and the feed-forward network of Part 2 cannot model this type of data. A solution to this, is the use of recurrent neural networks. These types of networks not only produce some output given some step from a sequence, but also update their internal state, hopefully "remembering" some information about the previous steps in the input sequence. Of course, they do have their own faults, but we'll cover this more thoroughly later in the semester.

Your task for the extra credit portion of this assignment, is to implement such a model below using a LSTM. Instead of averaging the embeddings as with the FFN in Part 2, you'll instead feed all of these embeddings to a LSTM layer, get its final output, and use this to make your prediction for the class of the headline.

```
class RecurrentModel(nn.Module):
  def __init__(self, vocab_size, embedding_dim, hidden_dim, \
               num_layers=1, bidirectional=True):
    """
    Instantiate layers for your model

    Your model architecture will be an optionally bidirectional LSTM,
followed
```

```
    by a linear + sigmoid layer.

    You will need 4 nn.Modules:
      1. An embeddings layer (see nn.Embedding)
      2. A bidirectional LSTM (see nn.LSTM)
      3. A Linear layer (see nn.Linear)
      4. A sigmoid output (see nn.Sigmoid)

    HINT: In the forward step, the BATCH_SIZE is the first dimension.
    HINT: Think about what happens to the linear layer's hidden_dim
size
          if bidirectional is True or False.
    """
    super().__init__()

    ### BEGIN YOUR CODE (~4 lines) ###
    self.embedding = nn.Embedding(vocab_size, embedding_dim)
    self.lstm = nn.LSTM(embedding_dim, hidden_dim, num_layers,
batch_first=True, bidirectional=bidirectional)
    self.linear = nn.Linear(2 * hidden_dim if bidirectional else
hidden_dim, 1)
    self.sigmoid = nn.Sigmoid()
    ### END YOUR CODE ###

  def forward(self, x):
    """
    Complete the forward pass of the model.

    Use the last timestep of the output of the LSTM as input to the
linear
    layer. This will only require some indexing into the correct
return
    from the LSTM layer.

    Args:
      x: 2D LongTensor of shape (BATCH_SIZE, max len of all
tokenized_word_tensor))
         This is the same output that comes out of the collate_fn
function you completed-
    """
    ### BEGIN YOUR CODE (~4-5 lines) ###
    x = self.embedding(x)

    _, (hidden, _) = self.lstm(x)

    if self.lstm.bidirectional:
        last_state = torch.cat((hidden[-2, :, :], hidden[-1, :, :]),
dim=-1)
    else:
        last_state = hidden[-1, :, :]
```

```
    x = self.linear(last_state)
    x = self.sigmoid(x)
    return x
    ### END YOUR CODE ###
```

Now that the `RecurrentModel` is defined, we will reinitialize our dataset iterators.

```
train_iterator = DataLoader(train_dataset, batch_size=BATCH_SIZE,
sampler=train_sampler, collate_fn=collate_fn)
val_iterator   = DataLoader(val_dataset, batch_size=BATCH_SIZE,
sampler=val_sampler, collate_fn=collate_fn)
test_iterator  = DataLoader(test_dataset, batch_size=BATCH_SIZE,
sampler=test_sampler, collate_fn=collate_fn)
```

6.2 Initialize the LSTM classification model

Next we need to initialize our new LSTM model, as well as define it's optimizer and loss function as we did for the FFNN. Feel free to use the same optimizer you did above, or see how this model reacts to different optimizers/learning rates than the FFNN.

```
lstm_model = RecurrentModel(vocab_size    = len(train_vocab.keys()),
                            embedding_dim = 300,
                            hidden_dim    = 300,
                            num_layers    = 1,
                            bidirectional = True).to(device)

lstm_criterion, lstm_optimizer = None, None

### BEGIN YOUR CODE ###
lstm_criterion = nn.BCELoss()
lstm_optimizer = Adam(model.parameters())
### END YOUR CODE ###
```

6.3 Training and Evaluation

Because the only difference between this model and the FFN is the internal structure, we can use the same methods as above to evaluate and train it. You should be able to achieve a validation F-1 score of at least `0.80` if everything went correctly.

Feel free to adjust the number of epochs to prevent overfitting or underfitting and to play with your model hyperparameters/optimizer & loss function.

```
#
================================================================
=====
# Pre-train to see what accuracy we can get with random parameters
#
================================================================
```

```
=====

true, pred = val_loop(lstm_model, val_iterator)
print(f'Binary Macro F1: {binary_macro_f1(true, pred)}')
print(f'Accuracy: {accuracy(true, pred)}')

100%|██████████| 150/150 [00:02<00:00, 61.40it/s]

---------------------------------------------------------------------------
-----
IndexError                                Traceback (most recent call
last)
Cell In[199], line 6
      1 #
=========================================================================
=====
      2 # Pre-train to see what accuracy we can get with random
parameters
      3 #
=========================================================================
=====
      5 true, pred = val_loop(lstm_model, val_iterator)
----> 6 print(f'Binary Macro F1: {binary_macro_f1(true, pred)}')
      7 print(f'Accuracy: {accuracy(true, pred)}')

Cell In[166], line 74, in binary_macro_f1(true, pred)
     72 averaged_macro_f1 = None
     73 ### BEGIN YOUR CODE (~1 line) ###
---> 74 averaged_macro_f1 = (binary_f1(true, pred, True) +
binary_f1(true, pred, False)) / 2
     76 ### END YOUR CODE ###
     77 return averaged_macro_f1

Cell In[166], line 47, in binary_f1(true, pred, selected_class)
     44 pred = [i[0] for i in pred]
     46 for i in range(len(true)):
---> 47     if pred[i] == selected_class:
     48       if true[i] == selected_class:
     49          tp += 1

IndexError: list index out of range

#
=========================================================================
=====
# Train your LSTM model
#
=========================================================================
=====
```

```
TOTAL_EPOCHS = 10
for epoch in range(TOTAL_EPOCHS):
    train_loss = train_loop(lstm_model, lstm_criterion,
lstm_optimizer, train_iterator)
    true, pred = val_loop(lstm_model, val_iterator)
    print(f"EPOCH: {epoch}")
    print(f"TRAIN LOSS: {train_loss}")
    print(f"VAL F-1: {binary_macro_f1(true, pred)}")
    print(f"VAL ACC: {accuracy(true, pred)}")
```

```
100%|██████████| 1200/1200 [00:57<00:00, 20.85it/s]
100%|██████████| 150/150 [00:02<00:00, 55.10it/s]

EPOCH: 0
TRAIN LOSS: 823.6590694189072
VAL F-1: 0.4525895691609978
VAL ACC: 0.6070833333333333

 55%|█████     | 658/1200 [00:35<00:29, 18.67it/s]

---------------------------------------------------------------------------
-----
KeyboardInterrupt                         Traceback (most recent call
last)
Cell In[193], line 7
      5 TOTAL_EPOCHS = 10
      6 for epoch in range(TOTAL_EPOCHS):
----> 7     train_loss = train_loop(lstm_model, lstm_criterion,
lstm_optimizer, train_iterator)
      8     true, pred = val_loop(lstm_model, val_iterator)
      9     print(f"EPOCH: {epoch}")

Cell In[165], line 13, in train_loop(model, criterion, optim,
iterator)
     11 y_pred = model(x)
     12 loss = criterion(y_pred, y.unsqueeze(1))
---> 13 loss.backward()
     14 optim.step()
     16 total_loss += loss.item()

File c:\Python312\Lib\site-packages\torch\_tensor.py:626, in
Tensor.backward(self, gradient, retain_graph, create_graph, inputs)
    616 if has_torch_function_unary(self):
    617     return handle_torch_function(
    618         Tensor.backward,
    619         (self,),
   (...)
    624         inputs=inputs,
    625     )
--> 626 torch.autograd.backward(
```

```
    627     self, gradient, retain_graph, create_graph, inputs=inputs
    628 )

File c:\Python312\Lib\site-packages\torch\autograd\__init__.py:347, in
backward(tensors, grad_tensors, retain_graph, create_graph,
grad_variables, inputs)
    342     retain_graph = create_graph
    344 # The reason we repeat the same comment below is that
    345 # some Python versions print out the first line of a multi-
line function
    346 # calls in the traceback and some print out the last line
--> 347 _engine_run_backward(
    348     tensors,
    349     grad_tensors_,
    350     retain_graph,
    351     create_graph,
    352     inputs,
    353     allow_unreachable=True,
    354     accumulate_grad=True,
    355 )

File c:\Python312\Lib\site-packages\torch\autograd\graph.py:823, in
_engine_run_backward(t_outputs, *args, **kwargs)
    821     unregister_hooks =
_register_logging_hooks_on_whole_graph(t_outputs)
    822 try:
--> 823     return Variable._execution_engine.run_backward(  # Calls
into the C++ engine to run the backward pass
    824         t_outputs, *args, **kwargs
    825     )  # Calls into the C++ engine to run the backward pass
    826 finally:
    827     if attach_logging_hooks:

KeyboardInterrupt:
```

```
#
==========================================================================
=====
# Evaluate your model on the held-out test set
#
==========================================================================
=====

true, pred = val_loop(lstm_model, test_iterator)
print(f"TEST F-1: {binary_macro_f1(true, pred)}")
print(f"TEST ACC: {accuracy(true, pred)}")
```

```
100%|██████████| 150/150 [00:02<00:00, 64.56it/s]
```

```
TEST F-1: 0.4409313003708133
TEST ACC: 0.59625
```

7. Submit Your Homework

This is the end of Project 1. Congratulations!

Now, follow the steps below to submit your homework in Gradescope:

1. Rename this ipynb file to `CS4650_p1_GTusername.ipynb`. Make sure all cells have been run. We recommend ensuring you have removed any extraneous cells & print statements, clearing all outputs, and using the Runtime --> Run all tool to make sure all output is update to date.
2. Click on the menu 'File' --> 'Download' --> 'Download .py'.
3. Click on the menu 'File' --> 'Download' --> 'Download .ipynb'.
4. Download the notebook as a .pdf document. Make sure the output from Parts 4 & 6.3 are captured so we can see how the loss, F1, & accuracy changes while training.
5. Upload all 3 files to Gradescope. Double check the files start with `CS4650_p1_*`, capitalization matters.