In [ ]:
```
import numpy as np
import pandas as pd
import joblib
import math
from copy import deepcopy

import torch
from torch.utils.data import Dataset
from torch.utils.data import DataLoader
import torch.nn as nn
import torch.nn.functional as F
import torch.optim as optim
from sklearn.preprocessing import LabelEncoder

import matplotlib.pyplot as plt
import seaborn as sns
```

load raw datasets

load tweets

In [ ]:
```
# func to load_dataset into a list of lists of (word, tag) tuples (eac
h inner list is a tweet)

def load_dataset(data_path):

    tweets = []
    vocab = set()
    tags = set()

    with open(data_path, 'r') as file:

        tweet = []

        for i, line in enumerate(file):

            # if line is empty, store current tweet and start a new tw
eet
            if line in ['\n']:
                tweets.append(tweet)
                tweet = []

            # otherwise, append new word and tag to current tweet as a
tuple
            else:
                word, tag = line.strip('\n').split('\t')  # split stri
ng into word and tag
                vocab.add(word)
                tags.add(tag)
```

```
                tweet.append((word, tag))

    return tweets, vocab, tags
```

In [ ]:
```
# load datasets

twpos_train, vocab_train, tags_train = load_dataset('/content/drive/MyDrive/postag/data/twpos-data/twpos-train.tsv')
twpos_dev, vocab_dev, tags_dev = load_dataset('/content/drive/MyDrive/postag/data/twpos-data/twpos-dev.tsv')
twpos_devtest, vocab_devtest, tags_devtest = load_dataset('/content/drive/MyDrive/postag/data/twpos-data/twpos-devtest.tsv')

print(f'twpos_train: {len(twpos_train)}, vocab_train: {len(vocab_train)}\ntwpos_dev: {len(twpos_dev)}, vocab_dev: {len(vocab_dev)}\ntwpos_devtest: {len(twpos_devtest)}, vocab_devtest: {len(vocab_devtest)}')
```

```
twpos_train: 1173, vocab_train: 4420
twpos_dev: 327, vocab_dev: 1750
twpos_devtest: 327, vocab_devtest: 1705
```

In [ ]:
```
# get all_vocab in train, dev, and devtest
all_vocab = list(vocab_train.union(vocab_dev).union(vocab_devtest))
all_vocab += ['<s>', '</s>']   # add beginning and end of sentence markers
print(len(all_vocab))

# get all_tags in train, dev, and devtest
all_tags = list(tags_train.union(tags_dev).union(tags_devtest))
print(len(all_tags))
```

```
5991
25
```

In [ ]:
```
# load orig datasets

orig_train, _, _ = load_dataset('/content/drive/MyDrive/postag/data/twpos-data/orig-train.tsv')
orig_dev, _, _ = load_dataset('/content/drive/MyDrive/postag/data/twpos-data/orig-dev.tsv')
orig_devtest, _, _ = load_dataset('/content/drive/MyDrive/postag/data/twpos-data/orig-devtest.tsv')

print(f'orig_train: {len(orig_train)}\norig_dev: {len(orig_dev)}\norig_devtest: {len(orig_devtest)}')
```

```
orig_train: 1173
orig_dev: 327
orig_devtest: 327
```

load embeddings

In [ ]:
```
# load pretrained embeddings

emb_pretrained_vocab = []
emb_pretrained = []

with open('/content/drive/MyDrive/postag/data/twitter-embeddings.txt',
'r') as file:

  for i, line in enumerate(file):

    line_split = line.strip().split()

    emb_pretrained_vocab.append(line_split[0])
    emb_pretrained.append(list(map(float, line_split[1:])))

emb_pretrained = torch.tensor(emb_pretrained)
print(len(emb_pretrained_vocab), emb_pretrained.shape)
```

```
30001 torch.Size([30001, 50])
```

encoders

word & tag encoders

In [ ]:
```
# func: get idx in emb matrix given a word
def get_word2idx(vocab_list):
  word2idx = {}
  for i, word in enumerate(vocab_list):
    word2idx[word] = i
  return word2idx
```

In [ ]:
```
# for encoding words in context windows
word2idx_all_vocab = get_word2idx(all_vocab)
word2idx_emb_pretrained_vocab = get_word2idx(emb_pretrained_vocab)
# tag2idx = get_word2idx(all_tags)
```

In [ ]:
```
# for encoding targets
le = LabelEncoder()
le.fit(all_tags)
```

Out[ ]:
```
▾ LabelEncoder

LabelEncoder()
```

encoded data class

In [ ]:

```
# reference: https://pytorch.org/tutorials/beginner/basics/data_tutorial.html


class POSDataset(Dataset):


    def __init__(self, dataset:list, dataset_orig:list, word2idx:dict, tag2idx:LabelEncoder(), w:int, feature_funcs:list=None):

        """
        wins; center_words; center_words_orig; tags; tags_encoded
        """

        wins = []
        center_words = []
        tags = []
        center_words_orig = []   # from the orig-* files

        # encode context window and center word featuress
        for tweet, tweet_orig in zip(dataset, dataset_orig):

            # process every center word in each tweet
            for i, (word, tag) in enumerate(tweet):

                # center word for curr obs
                center_words.append(word)

                # orig center word for curr obs
                center_words_orig.append(tweet_orig[i][0])

                # target of curr obs
                tags.append(tag)

                # construct win: idx for words in context window
                win = []
                for i in range(i-w, i+w+1):
                    if i < 0:   # if before fist token
                        try: win.append(word2idx['<s>'])
                        except: win.append(word2idx['</s>'])   # if '<s>' not in emb vocab, use emb for '</s>'
                    elif i > len(tweet)-1:    # if after last token
                        win.append(word2idx['</s>'])
                    else:
                        try: win.append(word2idx[tweet[i][0]])
                        except: win.append(word2idx['UUUNKKK'])  # use emb for unknown words
                wins.append(win)

        # encode all target tags
        tags_encoded = tag2idx.transform(tags)

        # data type
        wins = torch.tensor(wins)
        center_words = np.array(center_words)
        center_words_orig = np.array(center_words_orig)
```

```
        tags_encoded = torch.tensor(tags_encoded)
        tags = np.array(tags)

        # construct features from center_words_orig using feature_func
s
        if feature_funcs is not None:
            features = np.column_stack([np.vectorize(func)(center_word
s_orig) for func in feature_funcs])
            wins = torch.cat((wins, torch.tensor(features)), dim=1)

        # set attributes
        self.wins = wins
        self.center_words = center_words
        self.center_words_orig = center_words_orig
        self.tags_encoded = tags_encoded
        self.tags = tags


    def __len__(self):
        return len(self.wins)


    def __getitem__(self, idx):
        return self.wins[idx], self.tags_encoded[idx]
```

instantiate encoded datasets

In [ ]:
```
# encode datasets


# w = 0, all vocab encoding
train_w0_allvocab = POSDataset(dataset=twpos_train, dataset_orig=orig_
train, word2idx=word2idx_all_vocab, tag2idx=le, w=0, feature_funcs=Non
e)
dev_w0_allvocab = POSDataset(dataset=twpos_dev, dataset_orig=orig_dev,
word2idx=word2idx_all_vocab, tag2idx=le, w=0, feature_funcs=None)
devtest_w0_allvocab = POSDataset(dataset=twpos_devtest, dataset_orig=o
rig_devtest, word2idx=word2idx_all_vocab, tag2idx=le, w=0, feature_fun
cs=None)

# w = 1, all vocab encoding
train_w1_allvocab = POSDataset(dataset=twpos_train, dataset_orig=orig_
train, word2idx=word2idx_all_vocab, tag2idx=le, w=1, feature_funcs=Non
e)
dev_w1_allvocab = POSDataset(dataset=twpos_dev, dataset_orig=orig_dev,
word2idx=word2idx_all_vocab, tag2idx=le, w=1, feature_funcs=None)
devtest_w1_allvocab = POSDataset(dataset=twpos_devtest, dataset_orig=o
rig_devtest, word2idx=word2idx_all_vocab, tag2idx=le, w=1, feature_fun
cs=None)


# w = 0, pretrained 30k vocab encoding
train_w0_30k = POSDataset(dataset=twpos_train, dataset_orig=orig_trai
n, word2idx=word2idx_emb_pretrained_vocab, tag2idx=le, w=0, feature_fu
ncs=None)
dev_w0_30k = POSDataset(dataset=twpos_dev, dataset_orig=orig_dev, word
2idx=word2idx_emb_pretrained_vocab, tag2idx=le, w=0, feature_funcs=Non
e)
devtest_w0_30k = POSDataset(dataset=twpos_devtest, dataset_orig=orig_d
evtest, word2idx=word2idx_emb_pretrained_vocab, tag2idx=le, w=0, featu
re_funcs=None)

# w = 1, pretrained 30k vocab encoding
train_w1_30k = POSDataset(dataset=twpos_train, dataset_orig=orig_trai
n, word2idx=word2idx_emb_pretrained_vocab, tag2idx=le, w=1, feature_fu
ncs=None)
dev_w1_30k = POSDataset(dataset=twpos_dev, dataset_orig=orig_dev, word
2idx=word2idx_emb_pretrained_vocab, tag2idx=le, w=1, feature_funcs=Non
e)
devtest_w1_30k = POSDataset(dataset=twpos_devtest, dataset_orig=orig_d
evtest, word2idx=word2idx_emb_pretrained_vocab, tag2idx=le, w=1, featu
re_funcs=None)
```

1.1 baseline neural network tagger

model architecture

In [ ]:
```
# references:
```

```
# - https://pytorch.org/tutorials/beginner/blitz/neural_networks_tutorial.html
# - https://pytorch.org/docs/stable/generated/torch.nn.Embedding.html
# - https://discuss.pytorch.org/t/how-to-create-mlp-model-with-arbitrary-number-of-hidden-layers/13124/2
# - https://www.deeplearningwizard.com/deep_learning/practical_pytorch/pytorch_feedforward_neuralnetwork/
# - https://machinelearningmastery.com/activation-functions-in-pytorch/


class FeedForwardNN(nn.Module):


    def __init__(self, w, vocab_size, emb_dim, nfeatures, layer_sizes:list, layer_acts:list, pretrained_emb=None, emb_freeze=False):

        # call parent constructor
        super(FeedForwardNN, self).__init__()

        # set initial embeddings
        if pretrained_emb is not None:
            self.emb = nn.Embedding.from_pretrained(pretrained_emb, freeze=emb_freeze)
        else:   # randomly init embeddings
            self.emb = nn.Embedding(vocab_size, emb_dim)
            self.emb.weight.data.uniform_(-0.01, 0.01)

        # set embeddings' dimensionality
        self.emb_dim = self.emb.weight.shape[1]

        # set total num of words in win
        self.w = 1 + 2 * w

        # set input layer dimensionality
        in_size = self.emb_dim * self.w + nfeatures

        # construct layers (last layer is output layer)
        self.layers = nn.ModuleList()
        for i, layer_size in enumerate(layer_sizes):
            if i == 0:
                layer = nn.Linear(in_size, layer_size)
                layer.weight.data.uniform_(-0.01, 0.01)
                layer.bias.data.zero_()
                self.layers.append(layer)
                # self.layers.append(nn.Linear(in_size, layer_size))
            else:
                layer = nn.Linear(layer_sizes[i-1], layer_size)
                layer.weight.data.uniform_(-0.01, 0.01)
                layer.bias.data.zero_()
                self.layers.append(layer)
                # self.layers.append(nn.Linear(layer_sizes[i-1], layer_size))

        # set each layer's activation function
        self.layer_acts = layer_acts


    def forward(self, x):

        # print('before', x.shape, x[:, :self.w], x[:, self.w:])
```

```
        # retrieve context word embeddings and concat horizontally; co
ncat additional features to the right
        x = torch.cat(
            (
                self.emb(x[:, :self.w]).view((x.shape[0], -1)),   # wo
rd embeddings
                x[:, self.w:]     # additional features
            ),
            dim=1
        )

        # print('after', x.shape)

        # forward pass
        for i, layer in enumerate(self.layers):
            x = layer(x)
            x = self.layer_acts[i](x)

        return x
```

train and eval func

In [ ]:

```
# run one epoch of training

def train1epoch(model, optimizer, criterion, train_dataloader):

    # turn on training mode
    model.train()

    # reset epoch_loss tracker
    epoch_loss = 0

    # iterate through mini-batches
    for xtrain_batch, ytrain_batch in train_dataloader:

        optimizer.zero_grad()   # zero the gradient buffers
        output = model(xtrain_batch)
        loss = criterion(output, ytrain_batch)
        loss.backward()
        optimizer.step()   # does the update

        epoch_loss += loss.item()

    print(f'  epoch loss: {epoch_loss}')
    return epoch_loss
```

In [ ]:

```
# eval

def eval(model, eval_data):

    # turn on eval mode
    model.eval()

    # turn off gradient calc to reduce memory consumption
    with torch.no_grad():

        # get model predictions on eval_data
        ypred = torch.argmax(model(eval_data.wins), dim=1)

        # count correct predictions
        ycorrect = torch.sum(torch.eq(ypred, eval_data.tags_encoded)).
item()

        # total num of obs in eval_data
        ytotal = len(eval_data.tags_encoded)

        # compute accuracy
        yaccu = ycorrect / ytotal

    print(f'  accuracy: {yaccu}')
    return yaccu
```

train & eval wrapper

In [ ]:

```
# wrapper for train & eval
# reference: https://pytorch.org/tutorials/beginner/saving_loading_models.html

def main_process(model, name, optimizer, criterion, train_data, batch_size, shuffle, val_data, test_data, max_epochs=10, early_stopping=3):


    # create batched iterator for train_data
    train_dataloader = DataLoader(train_data, batch_size=batch_size, shuffle=shuffle)


    # initialize vars: track metrics
    epoch_losses = []
    train_evals = []
    dev_evals = []

    # initialize vars: track best model
    best_dev_eval = 0
    best_model_epoch = -1

    # train and eval
    for epoch in range(max_epochs):

        print(f'epoch {epoch+1}')

        # train
        epoch_loss = train1epoch(
                            model=model,
                            optimizer=optimizer,
                            criterion=criterion,
                            train_dataloader=train_dataloader
                        )
        epoch_losses.append(epoch_loss)

        # eval on training set
        train_eval = eval(model=model, eval_data=train_data)
        train_evals.append(train_eval)

        # eval on dev set
        dev_eval = eval(model=model, eval_data=val_data)
        dev_evals.append(dev_eval)

        # update best model based on dev eval
        if dev_eval > best_dev_eval:

            # save state_dict of best model so far
            torch.save(model.state_dict(), '/content/drive/MyDrive/postag/models/'+name+'_best.pth.tar')

            # update which epoch best_model is from
            best_model_epoch = epoch

            # update best_dev_accu
            best_dev_eval = dev_eval
```

```
        print(f'  best model from epoch {best_model_epoch+1}')

        # early stopping based on dev eval
        if early_stopping is not None:
            if epoch - best_model_epoch >= early_stopping:
                print('========= EARLY STOPPING =========')
                break


    # load state_dict of best model (modifies input model in place)
    print(f'load best model')
    model.load_state_dict(torch.load('/content/drive/MyDrive/postag/mo
dels/'+name+'_best.pth.tar'))

    # eval best model on devtest set
    print(f'eval best model on devtest')
    devtest_eval = eval(model=model, eval_data=test_data)


    return epoch_losses, train_evals, dev_evals, devtest_eval
```

run model: w=0, all vocab

In [ ]:
```
# instantiate model: single hidden layer 128 with tanh nonlinearity, w
=0, all vocab (random init)
tagger_w0 = FeedForwardNN(w=0, vocab_size=len(all_vocab), emb_dim=50,
nfeatures=0,
                          layer_sizes=[128, len(all_tags)],  # last la
yer is the output layer
                          layer_acts=[nn.Tanh(), nn.Identity()],   # n
n.CrossEntropyLoss() already includes softmax transformation
                          pretrained_emb=None, emb_freeze=False)

# instantiate optimizer
sgd = optim.SGD(tagger_w0.parameters(), lr=0.02)

# train and eval
epoch_losses, train_evals, dev_evals, devtest_eval = main_process(
                                            model=tagger_w0,
                                            name='tagger_w0',  # file na
me used for using checkpoint
                                            optimizer=sgd,
                                            criterion=nn.CrossEntropyLos
s(),   # objective: log loss
                                            train_data=train_w0_allvoca
b,
                                            batch_size=1,
                                            shuffle=True,
                                            val_data=dev_w0_allvocab,
                                            test_data=devtest_w0_allvoca
b,
                                            max_epochs=20,
                                            early_stopping=3   # when de
v eval doesn't improve for 3 consecutive epochs
                                           )
```

```
epoch 1
  epoch loss: 46360.52154862881
  accuracy: 0.13736135434909516
  accuracy: 0.13690105787181084
  best model from epoch 1
epoch 2
  epoch loss: 33964.24647471821
  accuracy: 0.5658493870402802
  accuracy: 0.5328769964737606
  best model from epoch 2
epoch 3
  epoch loss: 22743.718198784278
  accuracy: 0.7480443666082895
  accuracy: 0.7033810412777433
  best model from epoch 3
epoch 4
  epoch loss: 14901.21043849986
  accuracy: 0.8575014594279042
  accuracy: 0.750674133997096
  best model from epoch 4
epoch 5
  epoch loss: 11242.088181002444
  accuracy: 0.896322241681261
  accuracy: 0.7598008711885501
  best model from epoch 5
epoch 6
  epoch loss: 8556.753457576851
  accuracy: 0.9193228254524226
  accuracy: 0.7728686994399502
  best model from epoch 6
epoch 7
  epoch loss: 6902.9516340345435
  accuracy: 0.9268534734384122
  accuracy: 0.7768097904998963
  best model from epoch 7
epoch 8
  epoch loss: 6002.3917924894195
  accuracy: 0.9207822533566842
  accuracy: 0.770172163451566
  best model from epoch 7
epoch 9
  epoch loss: 5477.893790991569
  accuracy: 0.9256275539988325
  accuracy: 0.7778469197261979
  best model from epoch 9
epoch 10
  epoch loss: 5132.106349430047
  accuracy: 0.9288382953882078
  accuracy: 0.7782617714167185
  best model from epoch 10
epoch 11
  epoch loss: 4793.107766985544
  accuracy: 0.9279042615294805
  accuracy: 0.776602364654636
  best model from epoch 10
epoch 12
  epoch loss: 4591.493362655194
  accuracy: 0.927729130180969
  accuracy: 0.7741132545115121
  best model from epoch 10
epoch 13
```

```
  epoch loss: 4470.116119947081
  accuracy: 0.9201401050788091
  accuracy: 0.7687201825347438
  best model from epoch 10
========= EARLY STOPPING =========
load best model
eval best model on devtest
  accuracy: 0.7909032118991162
```

With a window size of 0, the best taggging accuracy on DEV is 77.83% from epoch 10; this best model has a tagging accuracy of 79.09% on DEVTEST.
(In the cell outputs above, the first accuracy score in each epoch is the accuracy on TRAIN, and the second accuracy score in each epoch is the accuracy on DEV. Same below.)

run model: w=1, all vocab

In [ ]:

```
# instantiate model: single hidden layer 128 with tanh nonlinearity, w
=1, all vocab (random init)
tagger_w1 = FeedForwardNN(w=1, vocab_size=len(all_vocab), emb_dim=50,
nfeatures=0,
                          layer_sizes=[128, len(all_tags)],  # last la
yer is the output layer
                          layer_acts=[nn.Tanh(), nn.Identity()],   # n
n.CrossEntropyLoss() already includes softmax transformation
                          pretrained_emb=None, emb_freeze=False)

# instantiate optimizer
sgd = optim.SGD(tagger_w1.parameters(), lr=0.02)

# train and eval
epoch_losses, train_evals, dev_evals, devtest_eval = main_process(
                                                        model=tagger
_w1,
                                                        name='tagger
_w1',  # file name used for using checkpoint
                                                        optimizer=sg
d,
                                                        criterion=n
n.CrossEntropyLoss(),   # objective: log loss
                                                        train_data=t
rain_w1_allvocab,
                                                        batch_size=
1,
                                                        shuffle=Tru
e,
                                                        val_data=dev
_w1_allvocab,
                                                        test_data=de
vtest_w1_allvocab,
                                                        max_epochs=2
0,
                                                        early_stoppi
ng=3   # when dev eval doesn't improve for 3 consecutive epochs
                                                       )
```

```
epoch 1
  epoch loss: 46358.82570374012
  accuracy: 0.1514302393461763
  accuracy: 0.1557768097904999
  best model from epoch 1
epoch 2
  epoch loss: 29023.68808175315
  accuracy: 0.7378867483946293
  accuracy: 0.701929060360921
  best model from epoch 2
epoch 3
  epoch loss: 12888.414400380258
  accuracy: 0.8904261529480444
  accuracy: 0.7826177141671853
  best model from epoch 3
epoch 4
  epoch loss: 7965.6317956046805
  accuracy: 0.9298890834792761
  accuracy: 0.800663762704833
  best model from epoch 4
epoch 5
  epoch loss: 6233.374623492888
  accuracy: 0.9455341506129598
  accuracy: 0.8039825762289982
  best model from epoch 5
epoch 6
  epoch loss: 4933.45225395245
  accuracy: 0.9447752481027437
  accuracy: 0.7919518772038996
  best model from epoch 5
epoch 7
  epoch loss: 3884.0792413342547
  accuracy: 0.960128429655575
  accuracy: 0.7979672267164488
  best model from epoch 5
epoch 8
  epoch loss: 3049.7473814702535
  accuracy: 0.9688266199649738
  accuracy: 0.8025305953121759
  best model from epoch 5
========= EARLY STOPPING =========
load best model
eval best model on devtest
  accuracy: 0.810303944815693
```

With a window size of 1, the best taggging accuracy on DEV is 80.40% from epoch 5; this best model has a tagging accuracy of 81.03% on DEVTEST.
(In the cell outputs above, the first accuracy score in each epoch is the accuracy on TRAIN, and the second accuracy score in each epoch is the accuracy on DEV.)

1.2 feature engineering

examine model errors

In [ ]:
```
# load best tagger so far
tagger_w1 = FeedForwardNN(w=1, vocab_size=len(all_vocab), emb_dim=50,
nfeatures=0,
                          layer_sizes=[128, len(all_tags)],  # last la
yer is the output layer
                          layer_acts=[nn.Tanh(), nn.Identity()],   # n
n.CrossEntropyLoss() already includes softmax transformation
                          pretrained_emb=None, emb_freeze=False)
tagger_w1.load_state_dict(torch.load('/content/drive/MyDrive/postag/mo
dels/tagger_w1_best.pth.tar'))
```

Out[ ]:
```
<All keys matched successfully>
```

In [ ]:

```
# check the errors the best tagger so far made on *DEV*

# get a mask for correct predictions
ycorrect_mask = torch.eq(
    torch.argmax(tagger_w1(dev_w1_allvocab.wins), dim=1),  # pred
    dev_w1_allvocab.tags_encoded  # true
)

# get center words with wrong pred
yerror = dev_w1_allvocab.center_words_orig[~ycorrect_mask]
print([e for e in yerror])
```

['@ciaranyree', 'it', 'was', 'on', ',', 'one', 'of', 'the', 'player
s', 'and', 'his', 'wife', 'own', 'burger', 'RT', '@TheRealQuailman',
':', 'Currently', 'laughing', 'at', '.', 'RT', '@HollywoodOompa',
':', 'Sat', '6', 'ill', 'be', 'at', 'Nashville', 'center', 'stage',
'for', 'the', 'ice', 'kream', 'for', '<<', "it's", 'the', 'music',
'center', 'center', 'You', "don't", 'know', 'my', 'struggle', 'Win
d', '2.0', 'mph', '.', 'Barometer', '29.676', ',', 'Rising', '.', 'T
emperature', '56.3', '.', 'Rain', 'today', '0.00', '.', '45%', '@Shi
versTheNinja', 'forgive', 'me', 'for', 'blowing', 'up', 'your', 'com
ment', 'section', '.', 'New', ':', 'How', 'CAN', 'you', 'mend', 'a',
'broken', '?', 'Really', '?', 'Please', '?:', 'http://bit.ly/9RgG9
L', '@justinbieber', 'can', 'u', 'follow', 'me', 'please', '???', 'i
t', 'mean', 'the', 'world', 'to', 'me', '!!', ':)', '@JoycieeLovesUu
u', '=)', 'Lls', '@carolyncallahan', 'I', 'knew', 'it', 'last', 'nig
ht', ',', 'but', "didn't", 'bother', 'calling', 'Shawn', 'because',
"I'd", 'just', 'be', 'working', 'on', 'it', 'this', 'morning', 'w/',
'same', 'info', '.', 'Senate', '#ArtsGrades', 'are', 'in', '!', 'Se
e', 'who', 'passed', 'and', 'who', 'made', 'the', 'Dirty', '.', '#ar
ts', 'http://t.co/BAh2iUL', 'via', '@ArtsActionFund', 'RT', '@guarna
schelli', ':', 'I', 'want', 'to', 'at', 'a', 'in', 'a', 'diner', 'an
d', 'watch', 'the', 'news', 'and', 'out', 'over', 'some', 'fresh',
'pancakes', ',', 'with', 'rum', ',', 'cri', '...', '29p', '11r', 'Pa
l', 'went', 'da', 'fuck', 'off', '@comicsguy024', 'I', "don't", 'us
e', 'due', 'to', 'the', 'lack', 'of', 'a', 'good', 'Twitter', 'exten
sion', '(', 'and', 'extension', 'behavior', 'in', 'general)', '.',
'Also', ',', 'it', 'looks', 'weird', 'So', "who's", 'going', 'to',
'the', 'Ethernet', 'Expo', 'next', 'week', 'in', '?', "I'm", 'addict
ed', 'to', 'that', '.', 'X', ')', 'RT', '@DarCoxaj', ':', 'TETRIS',
'!', '(:', '"', '@gcouros', 'I', 'see', 'that', ',', 'regretfully',
'I', 'was', 'tied', 'up', ',', 'Physed', 'at', '-', 'have', 'a', 'go
od', 'one', 'Costume', 'ready', '!', 'Where', 'to', 'go', 'for', 'Ha
lloween', 'on', 'and', 'sat', '...', 'Thinking', 'pyramid', 'on', 's
at', '...', 'Confirmed', 'no', 'phone', ',', 'now', 'I', 'have', 't
o', 'go', 'to', 'school', 'tomorrow', 'so', 'I', "don't", 'get', 't
o', 'see', '@taybby21', 'till', 'the', 'weekend', 'or', '.', '/',
':', "Mom's", '.', '10/27-', 'Make', 'sure', 'to', 'get', 'your', 'I
R', 'uploaded', '...', 'AND', 'dress', 'up', 'for', 'Halloween', 'o
n', 'Friday', '!!', '"', 'Because', 'if', 'seeing', 'is', 'believin
g', ',', 'then', 'that', 'we', 'have', 'lost', 'our', '..."', '#fb',
'@GUUuh_POwEr', 'u', 'from', 'Nashville', '?', 'RT', '@Gromit01',
':', 'Get', 'into', 'an', 'argument', 'with', 'your', 'tattooist',
'?', 'What', 'could', 'possibly', 'go', 'wrong', '?', 'http://tinyur
l.com/2e2z82n', 'wow', 'Jay-Z', 'Responds', 'To', 'Beyonce', 'Pregna
ncy', '|', 'The', 'http://bit.ly/aMjMsZ', '@MyName_IsAndre', 'lol',
'boy', 'shut', 'up', '!', 'I', 'wonder', 'if', 'you', 'realize', 'w
e', 'were', 'YOU', '.', 'RT', '@xerxeslux', ':', 'Charice', '!!', '#
breakoutmusicartist', '#PeoplesChoice', '@OfficialCharice', 'http://
bit.ly/aDTggc', '#FOLLOW', '---->', '@_FashionCRAZED', '@MisGeneviev
e', '@LauraAshley913', '@NishaGotEmLoced', '@iT0NGUEuSQUIRT', '@SoLa
cedUp_', '@Ggs_PrettyInked', '@Twon_GotStaCCs', 'RT', '@stevedupe',
':', 'Holy', 'crap', '-', 'going', 'to', 'war', 'http://www.reuters.
com/article/idUSN2618043220101027', 'Ignite', 'to', 'Write', ':', 'T
his', 'Photo', 'Needs', 'a', 'Title', '!:', 'This', 'photograph', 'c
ertainly', 'needs', 'a', 'title', '!', 'Can', 'you', 'give', 'it',
'one', '?', 'It', 'se', '...', 'http://bit.ly/bz8n9w', '@thehunterno
rris', 'nah', 'man', 'your', 'a', '#biebro', "don't", 'think', 'I',
"didn't", 'save', 'that', 'phone', 'call', '!', ';)', "What's", 'sca
rier', 'than', 'fake', 'blood', ',', 'guts', 'and', '?', 'How', 'muc
h', 'they', 'can', 'cost', ':', 'S', 'So', 'try', 'these', 'homemad
e', 'solutions', '!', 'http://tinyurl.com/27gdfc4', 'RT', '@CristalF
lo', ':', 'I', 'know', 'it', 'gets', 'repetitive', 'but', '.....',

'I', 'love', 'my', 'life', '76', '??', 'AMERICA', ',', 'FUCK', 'YEA
H', '!', '@HuffingtonPost', 'Awesome', 'Old', 'Man', 'Makes', 'A',
'With', 'A', 'Gun', '(', ')', 'http://huff.to/cgyuKq', '@PleazeBelie
veMe', 'lmao', 'u', 'should', 'see', 'my', 'hand', 'motions', 'We',
'went', '2', 'long', 'without', 'talkin', 'bout', 'some', 'shit',
'I', "don't", 'care', '2', 'discuss', '.', 'I', 'guess', 'its', 'tha
t', 'time', '.', 'The', 'peace', 'was', 'good', 'while', 'it', 'last
ed', '*', 'sighs', 'Am', 'I', 'ur', 'one', 'and', 'only', '?', 'Am',
'I', 'the', 'reason', 'you', 'breathe', ',', 'or', 'am', 'I', 'the',
'reason', 'you', '#cry', '?', '#np', "can't", 'get', '@brutha', '♥',
'the', 'blog', 'updates', '...', 'http://fb.me/KCCi1JuT', 'coverag
e', 'of', 'and', 'Friends', 'Concert', '...', 'I', 'love', 'him', 'a
nd', 'he', 'love', 'me', '...', 'aint', 'nuthin', 'u', 'can', 'tel
l', 'us', '...', 'u', 'can', 'try', 'all', 'u', 'want', 'to', 'thi
s', 'bond', 'too', 'strong', '...', 'so', 'imma', 'n…', 'http://lnk.
ms/DcpY6', '@jesseecahh', 'him', ',', 'lmfao', '-', 'But', 'of', 'hi
s', 'fat', 'ass', 'wants', 'treats', 'so', 'either', 'way', 'I', 'ha
d', 'to', 'get', 'up', '.', '@theGypsy', 'my', 'comment', '(', 'curr
ently', 'awaiting', 'moderation', ')', 'probably', 'chase', 'them',
'away', ';)', 'WAIT', 'A', 'SECOND', "IT'S", 'ON', 'MY', 'PHONE', 'H
AMMER', 'U', 'ARE', 'MAD', 'GAY', 'AND', 'U', 'NOT', 'DO', 'ANYTHIN
G', 'u', 'mad', 'I', 'am', 'illuminate', 'I', 'am', 'king', 'Augustu
s', 'say', 'it', 'to', 'me', 'RT', '@ObserverDallas', ':', 'with',
'is', 'playing', 'at', 'this', '!', 'Want', 'to', 'be', 'a', 'person
al', 'guest', 'of', 'Observer', '?', 'RT', 'to', 'win', 't', '...',
'I', 'love', 'my', 'she', 'lettin', 'me', 'out', 'just', 'in', 'tim
e', 'for', 'my', '#Colts', 'game', 'siiiiiced', 'RT', '@TerrenceJ10
6', ':', 'I', 'can', 'say', 'that', 'tonights', '106', 'is', 'gonn
a', 'be', 'a', '!', '@NICKIMINAJ', '&', 'Janet', 'on', 'the', 'sam
e', 'show', '?', '#goingtoworkontime', '!', 'World', 'Cup', "socce
r's", 'psychic', 'octopus', 'Paul', 'dies', 'in', 'http://bit.ly/d6q
TGc', 'Ring', '!', '#lakers', 'Slightly', 'Stoopid', '@', 'ACL', 'wi
th', 'Slacker', 'part', '2', ':', 'http://fb.me/Ai9Qfi5J', 'first',
'game', 'on', 'the', '27th', 'vs', '.', 'Houston', 'This', 'is', 'st
artin', 'to', 'get', 'borin', '...', '@Joigadoi', 'thanks', ':D', 'E
ast', 'Bay', '—', '10/26', 'http://nblo.gs/9H1z8', 'Happy', 'Year',
'of', '!', 'What', 'better', 'way', 'to', 'than', 'in', 'to', "CropL
ife's", 'Biodiversity', 'World', 'Tour', '.', '#BWT2010', 'Finds',
'And', 'And', 'Takes', 'Social', ':', '“People', 'like', 'stumblin
g', 'videos', 'more', 'than', 'web', '...', 'http://bit.ly/c4gny7',
"I'm", 'not', 'a', 'boy', '-_-', '@WilliefknUnique', 'hahah', 'wel
l', 'that', 'a', 'diff', 'story', 'then', ':P', 'lmao', 'he', 'doesn
t', 'wanna', 'talk', 'to', 'ya', '.', 'hes', 'talkin', 'to', 'me',
'!', ';)', 'We', 'has', 'real', 'internets', 'at', 'laaaaaaaaast',
'!', '{{', 'Is', 'only', 'in', '@s', 'while', 'watching', 'Get', 'Hi
m', 'to', 'the', '!', '}}', 'RT', '@liltunechi', ':', '8', 'days',
'until', 'they', '#FreeWeezy', '.', 'Shoutout', '#FreeWeezy', 'all',
'day', 'everyday', 'until', "he's", 'home', '!!', 'RT', '@SugaHunniI
ceTea', ':', 'But', "I'm", 'going', 'to', 'praise', 'at', 'my', 'low
est', 'point', '.', 'Amen', 'Super', 'for', 'our', 'Halloweeen', 'to
morrowwww', ':)', 'TRICK', 'OR', 'TREAT', '!', '@SkyHearDOTcom', 'Yo
ur', 'local', 'and', "that's", 'cool', ',', 'hit', 'me', 'up', 'dire
ct', 'would', 'like', 'to', 'discuss', 'where', 'this', 'is', 'goin
g', '...', 'Now', 'I', 'to', 'go', 'home', 'and', 'make', 'nachos',
'with', 'this', 'probably', 'bad', 'for', 'me', 'cheese', 'sauce',
'.', '#mmmm', 'Announce', 'Partnership', 'with', 'USGN', 'http://ow.
ly/19EQDn', '@monaFCKN_lisa', '#oneword', ':', 'LAZYASFUCK', 'lol',
'RT', '@yungboikortny', 'Ray', 'Allen', 'Look', 'Just', 'Like', 'M
y', 'Dad', '!', 'Lol', '<==', '#FlagOnThePlay', '@JeffroMusic', '@ar
lene_chang', 'she', 'cant', 'Cus', 'she', 'probably', 'the', 'same',
'size', 'as', 'the', 'bop', 'haha', 'well', 'that', 'strange', '!',

'hahahahhaa', '.', 'airlines', 'from', 'sta', 'http://www.bestcoupon
for.com/coupon/266684.aspx', 'Support', 'Pray', 'For', 'Indonesia',
',', 'add', 'a', 'to', 'your', 'now', '!', '-', 'http://twb.ly/cHMmw
J', '#PrayForIndonesia', '@PimpThoughts', 'the', 'invite', 'was', 'a
ctually', '2', 'lure', "'", 'em', 'in2', 'the', 'den', 'of', 'lion
s', '.', 'She', 'assured', 'me', "they'd", 'bring', 'napkins', '2',
'wipe', 'away', 'tears', 'frm', '#roastin', '@ayoooitsRICA', '(',
')', 'a', 'person', 'from', 'seeing', 'the', 'the', 'in', 'others',
'.', 'I', 'think', 'I', 'get', 'fever', '.', 'Geez', '.', 'RT', '@Ma
nsionOct29', ':', 'BLACK', 'PARTY', 'HITS', '98', 'TODOAY', '!', '70
46614311', 'FOR', 'INFO', '@Danithepoet', '@DanniGyrl1', '@daRealCit
yboi', '@Darling_IamWarr', '@daveknowspdx', 'We', 'are', 'making',
'a', 'difference', 'already', ':', 'http://bit.ly/c8fLnr', 'RT', '@d
junique23', ':', '@MRMcGill85', 'LOL', 'HE', 'CANT', 'WIN', 'A', 'GA
ME', 'BY', 'HIS', "SELF(that's", 'why', 'he', 'got', 'the', 'other',
'2', '!!', ')(', 'RT', '@Pixer23', 'Fun', 'times', 'http://twitpic.c
om/317itr', '<', 'love', 'this', 'haha', 'heelllooo', 'Just', 'out',
"who's", 'been', 'callin', 'me', 'private', 'n', 'hangin', 'up',
',', 'and', 'all', 'i', 'have', 'to', 'say', 'is', '...', 'get', 'th
e', 'fuck', 'outta', 'here', '!', 'lmao', '@shaeken', 'what', 'me',
'.....', 'Cry', 'cry', 'What', 'you', 'see', 'is', 'what', 'you', 'g
et', 'Part', 'Builder', 'I', ':', 'Define', 'http://bit.ly/bZyU80',
'RT', '@sultrysole', ':', 'the', 'way', 'some', 'guys', 'talk', 'gir
ls', 'makes', 'me', 'cringe', 'How', 'does', 'Morrison', 'have',
'a', 'ring', ',', 'and', 'Miller', "doesn't", 'have', 'one', '?', '#
mysteriesofHOOPS', '|', ',', 'instant', 'screenshots', 'and', 'scree
ncasts', ',', 'home', 'http://t.co/c6alsyO', 'via', '@AddThis', "╚☆s
houtout's™☆╝>>>>", '@allilove_SODMG', 'Animation', 'of', 'yesterday
s', 'historic', '.', 'http://severeplains.com/images/102610_surfacel
ow.html', '@chewthis_bash', 'lol', 'how', '?', "I'm", 'never', 'smal
l', 'minded', 'RT', '@CynthiaBoS', ':', 'Grrrrrrrr', '!', '/', 'Gua
u', '!', 'I', 'love', 'dwayne', 'wade', '!!!!', 'I', 'wanna', 'marr
y', 'him', '!!!', ';)', 'RT', '@KevinDing', ':', 'Shannon', 'is', 't
onight', ':', 'points', 'in', '17', 'minutes', ',', '6', 'of', '8',
'shooting', ',', '4', 'of', '5', 'from', 'deep', '--', 'plus', 'som
e', 'key', '.', 'RT', '@ProSeverusSnape', ':', '“All', 'that', 'glit
ters', 'is', 'not', 'True', '.', 'Sometimes', 'it’s', 'pretending',
'to', 'be', 'vampires', '.', 'in', 'point', ',', 'Edward', '.', 'Wha
t', '...', "I've", 'had', 'a', 'few', 'requests', 'that', 'I', 're-e
nable', 'on', 'on', 'my', '.', 'Looking', 'for', 'rigs', 'for', 'm
y', '#7D', 'If', 'any1', 'can', 'point', 'me', 'in', 'the', 'thatd',
'be', 'great', '!', '#Video', '#Ikan', '#RedRockMicro', '#Zacuto',
'#IDC', '#eBay', 'RT', '@zwriter', ':', '@dustinlong', 'USAC', 'pled
ged', '10¢', 'for', 'Shane', 'Recovery', 'Fund', 'for', 'every', 'ne
w', 'FB', 'fan-', 'link', ':', 'http://on.fb.me/bnqidV', '.', 'Let
s', 'get', '...', 'People', 'always', 'what', 'they', "don't", 'kno
w', '.', 'Back', 'to', 'real', 'life', ':', 'picking', 'up', 'Camp',
'Kern', 'fundraiser', 'pizzas', 'tonight', ',', 'hoping', 'to', 'sta
rt', 'on', 'bathroom', 'tile', 'tomorrow', 'and', 'laundry', '...',
'always', 'laundry', '..', '@DjDSTRONG', 'I', 'would', 'of', 'swor
e', 'u', 'said', 'that', 'yesterday', ':/', 'lol', 'Before', 'I', 'g
et', 'started', 'on', 'this', 'post', 'me', 'make', 'one', 'thing',
'absolutely', 'clear', '.', 'I', 'am', 'a', '.', 'My', 'is', 'to',
'develo', '-', 'http://bit.ly/b50fIF', 'RT', '@RealSkipBayless',
':', 'Are', 'you', 'SURE', 'this', 'was', 'Big', '3', 'vs', '.', 'Bi
g', '3', '?', 'has', 'played', 'up', 'in', 'Toronto', ',', 'away',
'from', ',', 'pressure', '.', 'This', 'stage', 'loo', '...', 'is',
'already', 'on', 'the', '106', 'top', 'ten', 'countdown', ',', 'it',
'really', 'is', 'mind', 'blowing', '.', '@justinbieber', 'i', 'was',
'going', 'to', 'go', 'to', 'your', 'concert', 'in', 'and', 'i', 'cou
ldnt', 'because', 'i', 'couldnt', 'get', 'a', ':((', 'i', 'wish', 'w

```
ent', '.', 'RT', '@A_Rrod', 'WARNER', 'CABLE', 'WHAT', 'THE', 'FUC
K', 'IS', 'THE', 'FUCKIN', 'PROBLEM', 'It', 'needs', 'to', 'be', 'wr
estling', 'season', 'so', 'I', 'can', 'concentrate', 'on', 'other',
'than', 'this', '.', 'Real', 'niqqa', 'stfu', 'RT', '@eklecticx',
':', 'Chocolate', 'Mod', 'Beatle', 'Boots', 'by', 'eklecticxplosio
n', 'on', 'http://bit.ly/c95nZY', '#Etsy', '@BeliebinKPerry', 'wha
t', 'is', '?', '@Miz_Kellie', 'wat', 'cha', 'doin', 'flirty', 'apron
s', "mother's", '40%', 'off', 'sale', ',', 'code', 'fa-4110', '.',
'http://www.bestcouponfor.com/coupon/251486.aspx', 'Teaching', '.',
'Fuck', 'you', 'America', '...', 'stfu', 'IS', 'NOT', 'silent',
'...', 'and', 'we', 'still', 'up', 'NIGGA', 'WHAT', '!!!', '@Misguid
edGhost', 'me', '=)', '@DrakeswifeTiffy', 'yea', 'you', '!', ':)',
'I', "ain't", 'got', 'time', '..', 'bye', 'bye', '.', 'A', 'day', 'w
ithout', 'you', 'is', 'like', 'a', 'Year', 'Without', 'Rain', '<3',
'@irenitapetty4', 'First', 'concert', '?', 'Ohmygosh', 'awww', '<3',
'.', 'How', 'it', 'was', '?', ':D', 'RT', '@trini87', ':', 'I', "do
n't", 'see', 'any', 'fans', 'on', 'my', '...', 'all', 'I', 'see', 'i
s', '#teamlakers', '...', 'is', '...', 'do', 'rocket', 'fans', 'exis
t', '?', 'MC', 'Hammer', 'beefing', 'with', 'Jay-z', '!!??', 'Lmao',
'wait', 'no', '"', 'King', 'Hammer', '"', 'RT', '@kpopidol', ':', 'T
rue', ':', "they're", 'who', 'accept', 'any', "member's", 'decisio
n', '!!!', 'if', 'u', 'are', 'true', 'shawol', '!!!', "i'm", 'suppor
ting', 'them', 'even', 'i', 'fell', 'a', 'bit', '...', 'RT', '@grung
ereport', ':', 'GrungeReport', ':', 'DAVE', '&', 'NOVOSELIC', 'OF',
'CONFIRMED', 'TO', 'REUNITE', 'FOR', 'A', 'SONG', 'ON', 'NEW', 'FO
O', 'ALBUM', 'http', ':/', '...', 'Love', '@GansevoortPark', '!', 'T
hanks', 'for', 'taking', 'awesome', 'of', 'our', ':)', '@anidasabrin
a', '!!', ':)', "Today's", 'MetroTube', ':', 'An', 'oddly', 'touchin
g', 'electro', 'to', 'celebrity', 'mugshots', '—', 'it', 'almost',
'makes', 'you', 'feel', 'bad', 'for', 'them', 'http://cot.ag/dfjDP
f', 'RT', '@grantimahara', ':', "I'm", 'glad', 'we', 'in', 'a', 'wor
ld', 'where', 'a', 'little', 'kid', 'can', 'wear', 'a', 'cape', 't
o', 'the', 'airport', '.', '@wax_ecstatic', 'Replace', 'Mansfield',
'with', 'Azle', '&', 'I', 'can', 'totally', '!', 'You', 'smile',
'I', 'smile', '!', 'And', 'from', 'now', 'to', 'my', 'very', 'last',
'breathThis', 'day', "I'll", 'cherishYou', 'look', 'so', 'beautifu
l', 'in', 'whiteTonight', 'Great', 'touring', 'with', 'Lt', '.',
'.', 'We', 'visited', 'with', 'at', 'White', 'Oaks', '...', 'http://
fb.me/KkGbtk4W', 'ROTFLMAO', 'my', 'nigga', '@Qui_to_Success', 'wa
s', 'gonna', 'in', 'on', 'that', '#TT', '#MyExGirlfriend', '.', 'Lo
l', "I'm", 'over', 'diein', 'lol', 'RT', '@TIPStrategies', ':', 'R
T', '@jen_martinez', ':', 'Dept', 'of', ',', 'grants', '$500k', 't
o', 'metrics', 'for', '#econdev', 'http://ow.ly/2ZYzd', 'RT', '@Solo
dagod', ':', 'Miami', 'put', 'a', 'fork', 'in', 'it', '...', '@theun
abeefer', 'Man', ',', 'would', 'you', 'look', 'at', 'that', 'scener
y', '.', 'A', 'man', 'could', 'die', 'happy', 'with', 'a', 'like',
'that', '.', '#potsandpans', 'My', 'pen', 'died', '....', "it's", 't
he', 'only', 'pen', "i've", 'ever', 'used', 'in', '.', "I'm", 'los
t', 'with', 'out', 'you', '.', '#win', 'Witch', 'by', 'Kalayna', 'Pr
ice', 'http://fang-tasticbooks.blogspot.com/2010/10/guest-blog-and-g
iveaway-with-kalayna.html', '@jsun187', 'lol', '@', '.', 'Congrats',
'family', '!', 'RT', '@chimaincalgary', "Ford's", 'on', '@CBCAsItHap
pens', 'http://bit.ly/cq3ZZk', '#voteto', '#yyccc', '|', 'Interestin
g', 'juxtaposed', 'to', "Shaheen's", 'speech', 'It', 'is', 'a', 'goo
d', 'day', 'to', 'get', 'inspired', 'on', '@TODAYSHOW', '-', '@edwar
d_burns', ',', '@ApoloOhno', 'and', '@cakelovewarren', '.', "I'm",
'totally', 'the', 'movie', '&', 'books', '!', 'Main', 'Girl', '>',
'Phone', 'full', 'of', 'bitches', '.', 'RT', '@AMorningOttawa', ':',
'UNITED', 'UPDATE', ':', 'So', 'far', 'you', 'have', 'helped', 'rais
e', '15.7', 'M', 'dollars', '.', 'The', 'is', '33.1', 'M', 'and', 'i
t', 'wraps', 'up', 'Dec', '2nd', '.', '@shaunmenary', 'Shaun', '!',
```

```
'@jamieregier', 'I', 'always', 'do', '.', 'kings', 'is', 'unwalkabl
e', '!', '(', '@djcamilo', 'voice', ')', 'for', '@mousebuddens', '#s
upportrealhiphop', '!', '@diego_goLden', 'lol', '?', 'RT', '@FruitPu
nchYoAss', '@DoinMeDailey', 'u', 'down', 'lol', '<<<', 'lol', 'doi
n', 'it', 'already', 'Ray', 'ray', 'all', 'day', 'out', '@jayclipp',
'.', 'It', 'is', 'so', 'perfect', '.', 'Vintage', 'with', 'a', 'caus
e', '.', ':)', '@NumniimzZ', 'morning', '<3', '@HighImpactDsgns', 'o
nyx', '???', 'onyd', '=', 'oh', 'no', 'you', "didn't", '.', 'i', 'th
ink', 'you', 'made', 'that', 'one', 'up', '@sotinafunk', 'no', 'on
e', 'cares', '.', 'are', 'used', 'to', 'foolish', 'for', 'attentio
n', '.', "they'll", 'just', '*', 'shrug', '*', "'", 'Rent', 'Too',
'Damn', "'", 'NY', 'Jimmy', 'McMillan', 'inspires', 'talking', '-',
'The…', 'http://goo.gl/fb/1Y22x', '#palin', '#teaparty', '@SBGVoltag
e', 'Just', 'to', 'clarify', 'when', 'you', 'cleared', 'your', 'brow
ser', 'cookies', 'you', 'also', 'cleared', 'the', 'browser', 'cach
e', '?', '^SM', 'RT', '@FrankAdman', ':', 'Name', 'says', 'it', 'al
l', ':', 'http://badnewsrobot.com', 'Not', 'a', 'scam', '.', "It's",
'for', 'real', '.', 'Might', 'be', 'the', 'only', 'Robot', 'you', 'c
an', 'trust', 'on', 'the', '...', 'I', 'pick', 'my', 'nose', 'wit',
'my', 'penis', 'Well', 'my', 'night', 'away', 'from', 'here', "did
n't", 'last', 'too', 'long', '.', 'I', 'want', 'the', 'new', '3', 'c
oming', 'out', 'for', 'Xbox', '360', '-', 'so', 'who', 'wants', 't
o', 'buy', 'it', 'for', 'me', '?!', 'LoL', 'why', 'did', 'they', 'ad
d', 'league', 'to', '???', 'I', 'miss', 'Jenkins', 'being', 'with',
'us', 'everyday', ';(', 'RT', '@AAACarolinas', ':', 'Looking', '4',
'a', 'new', '?', '(', '@studiobanks', ',', 'maybe', '?', ')', 'AAA',
'Auto', 'Sales', 'makes', 'it', 'easy', 'http://trp.la/byLOZD', '2
8', 'Inspirational', 'Examples', 'of', 'Well', 'Designed', 'Contac
t', 'http://bit.ly/9ThfCn', 'Lol', 'we', 'dumb', '...', 'But', 'i',
'like', 'that', 'xnevershoutbrianna', 'asked', ':', 'I'M', 'SORRY',
'!', 'ITS', 'TEN', 'HER', 'AND', 'I', 'HAS', 'SCHOOL', 'TOMORROW',
'AND', 'I', 'GOTTA', 'A', 'SHOWER', '.', 'WAIT', 'WAIT', '...', 'htt
p://tumblr.com/xjcncmwl1', '#PrayForIndonesia', ',', 'RT', '@visuals
urgery', ':', '@dheanaya', '@Ardhieeto', '@BinalleFabolous', '@hany_
oiz', '@RicadRici', '@ata86', '(', 'cont', ')', 'http://tl.gd/6m3nn
s', 'RT', '@iFuckDickNPussy', ':', '#SayNo2', 'flat', 'do', 'i', 'ha
ve', 'to', 'be', 'a', 'slut', 'to', 'get', 'attention', 'wtf', ':',
'http://yearbook.com/a/pgn12', '@NicoleDaboub', "that's", 'pretty',
'hawt', 'is', 'that', 'all', 'of', 'the', 'outfit', '?', 'Exactly',
'how', 'I', 'expected', 'it', 'I', 'wish', 'I', 'can', 'rewind', 'ti
me', 'n', 'go', 'way', 'back', 'to', 'when', 'I', 'was', 'playing',
'with', 'my', 'n', 'set', 'with', 'my', 'imaginary', 'it', 'was', 'm
uch', 'better', 'then', '.', 'Photo', ':', 'AWWWWW', '.', 'Honestl
y', 'Imma', 'Lakers', 'just', 'cuz', 'they', 'are', ':P', 'Joking',
'.', 'I', 'don't', 'even', 'have', 'a', 'favorite', ':/', 'http://tu
mblr.com/xssngkmfi', '@Bash_TI', '<33', 'Love', 'Yur', '@DjAnArchy',
'LMAOOOO', 'I', 'gotta', 'that', 'does', 'that', '?!?', 'Who', '??',
'Help', 'us', '500', '!', 'RT', '"', '#GlareX2Mender', 'products',
'!', 'All', 'this', 'in', 'one', '!', 'Details', 'here', 'http://bi
t.ly/d6UmZd', '.', 'RT/Follow', '@Vivitone', 'to', 'win', '!"', '@cu
tecanukgirl', '@NelizMD', 'Just', "don't", 'Nel', 'near', 'him',
'...', "she's", 'all', 'the', 'fine', 'art', 'of', 'decapation', 'la
tely', '*', 'cookoo', 'cookoo', '*', 'RT', '@soulcooljay', ':', 'Cou
ltrain', '—', 'Green', '-', 'almost', 'forgot', 'how', 'brilliant',
'this', 'was', '...', '@seymourliberty', '♫', 'http://blip.fm/~xxe8
8', 'This', 'lil', 'girl', 'got', 'some', 'unnecessary', 'tattoos',
'but', 'the', 'art', 'work', 'is', 'superb', '.', 'RT', '@MacCoverGi
rl', ':', 'RT', '@gracefuldelta', ':', 'allows', 'us', 'to', 'experi
ence', 'the', 'low', 'points', 'of', 'life', 'in', 'to', 'teach', 'u
s', '...', 'http://tmi.me/2DoEF', '@harvepierre', 'did', 'you', 'rec
eive', 'the', 'i', 'sent', 'earlier', 'today', '???', 'Let', 'me',
```

```
'know', 'por', 'favor', ':)', '@RedB3rryCARTER', 'but', 'you', 'gott
a', 'think', 'Nigga', 'they', 'teams', 'that', 'already', 'got', 'th
e', 'chemistry', 'not', 'lookn', '4', 'it', 'Find', 'great', 'domain
s', 'for', '!', 'Come', 'and', 'check', 'our', 'domains', 'for', 'se
ction', '.', 'List', 'your', 'domains', 'for', '$', '/year', 'htt
p://sns.ly/5Jx53', 'Carving', 'a', 'Pumpkin', 'With', 'a', 'Gun',
'[', ']:', 'This', 'man', 'is', '.', 'His', 'pump', '...', 'http://b
it.ly/dbTukl', '@agingjoy', 'Excuse', 'me', '?', 'Uppity', '?',
'::', 'confused', '::', 'GLC', '-', 'The', 'Light', 'http://ping.fm/
OJQ71', '@Bieberlove96', 'oh', 'I', 'sooo', 'am', '!!', 'Lol', 'wait
ing', 'to', 'see', 'if', "he's", 'gonna', 'me', 'back', '!', 'And',
'it', "doesn't", 'seem', 'like', 'it', '.', '#howcome', 'there', 'i
s', 'so', 'many', 'bops', 'at', 'TSU', 'RT', '@StopBeck', ':', '(',
'Yup', ',', 'here', 'comes', 'the', 'guilt)', '.', 'Remember', ',',
'your', 'tweets', '*', 'do', '*', '.', 'See', 'the', 'result', 'of',
'*', 'your', '*', 'efforts', 'here', ':', 'http://bit.ly/cM3pq3', 'R
T', '@Real_ESPNLeBrun', ':', 'RT', '@RealKyper', ':', 'news', '.',
'Early', 'are', 'that', '#Leafs', 'may', 'surgery', 'on', 'his', 'ha
nd', 'and', 'could', '...', '@lawmomma77', 'Say', 'what', '?', '@JOH
NSAVAAGE', 'oh', 'shit', 'its', 'beyonce', 'u', 'up', 'from', 'da',
'train', 'station', 'RT', '@MISTAKT', ':', 'RT', '@GQstatus', ':',
'RT', '@DJ_Ranga', '&', '@GQstatus', ',', '@MistaKT', '&', '@DBG93
6', 'present', '"', 'Get', 'Or', 'Go', 'Broke', '"', 'on', '@ThatCra
ck', '...', 'http://tmi.me/', '...', 'Just', 'watched', 'Iron', '2',
',', 'what', 'a', 'nearly-perfect', 'superhero', 'movie', '!', 'So',
'good', 'you', "don't", 'even', 'mind', 'Rourke', '!', '(', 'J/k',
',', 'he', 'was', 'great', 'too', ')', 'Agree', 'that', 'humanizin
g', 'brand', 'is', 'paramount', 'but', 'associating', 'w', '1', 'per
sonality', 'has', 'big', 'risk', ':', 'Dell', ',', ',', 'Tiger',
',', ',', ',', '...', '#brandchat', '@LilHimQueenB', '@CuteKidd23', '@Rig
hteousBoi', '@RipJasmyne425', '@dabossmane', '@duric2smooth', 'HATI
N', "that's", 'crazy', ',', 'you', 'unprepared', '.', "It's", 'alrea
dy', 'to', 'be', 'one', 'of', 'Those', 'Days', '.', 'I', 'got', 'wok
en', 'up', 'by', 'my', 'tv', 'falling', 'on', 'tha', 'floor', '.',
'Nice', ',', 'right', '?', '-_-', '@AndyMilonakis', 'congrats', 'o
n', 'the', '100k', 'Rogue', '.', 'No', '.', 'I', "didn't", 'do', 'i
t', 'yet', '.', 'ghdjfsshdfjg', '.', 'Should', 'I', '.', ',', 'righ
t', 'now', '?', 'Larry', 'To', 'Prove', 'HP's', 'New', 'CEO', 'Stol
e', 'His', 'Software', '-', 'http://tinyurl.com/272edn8', 'its', 're
ally', 'weird', 'how', 'FB', 'shows', '"', 'photo', 'memories', '"',
'of', 'the', 'hottest', 'girl', 'to', 'me', 'on', 'the', 'side', '@B
JNemeth', 'was', 'asking', 'walk', 'on', 'RT', '@FunnyOrFact', ':',
'Cheating', 'on', 'the', 'person', 'u', 'LOVE', '.', '#WhoDoesThat',
'R', 'E', 'T', 'W', 'E', 'E', 'T', 'if', 'u', 'DONT', 'cheat', '&',
"u're", 'loyal', '&', 'faithful', '.', '-', '@iRespectFemales', 'I',
'can', 'usually', 'handle', 'the', '.', 'The', 'day', 'of', 'the',
'Dallas', 'show', 'I', 'was', 'ok', 'but', 'today', 'I', "can't", 's
eem', 'to', 'get', '.', 'Jones', 'Get', '3D', 'Treatment', '&', 'Re-
Release', ':', 'Following', 'up', 'on', 'George', 'Lucas', "'", 'dec
ision', 'to', 're-release', 'the', 'Star', '...', 'http://bit.ly/99Z
yaY', 'Denise', 'Schump', 'chosen', 'for', 'Teacher', 'Tuesday', 'ho
nor', ':', '...', 'with', 'a', 'and', 'a', 'collection', 'of', 'gift
s', '.', 'Shump', 'said', 'she', 'w', '...', 'http://bit.ly/as5RPc',
'RT', '@ThatNicholas', ':', '10:55', 'ಠಠ', '@lewismd13', 'true', '.'
'Then', 'again', 'you', 'post', 'several', 'times', 'a', 'week', 'an
d', 'I', 'only', 'post', 'once', '.', 'So', 'have', 'to', 'get', 'i
t', 'all', 'in', 'haha', '.', 'This', 'next', 'one', 'is', 'shortis
h', 'My', 'head', 'is', 'killing', 'me', 'good', 'night', 'to', 'all
#', 'fuck', 'the', 'rest', 'of', 'yall', 'lol', 'I', 'kno', 'thats',
'husband', ':)', 'ily', 'win', 'or', 'lose', 'RT', '@Sir_Freshiest_
J', 'RT', '@KingJames', "wasn't", 'built', 'in', 'a', 'Day', '!', 'W
```

ork', 'http://tl.gd/6m0vlp', '@robertokaercher', 'Yes', '.', 'and',
'how', 'your', 'school', '?', '@ZCOOP', 'I', 'love', 'it', 'when',
'you', 'talk', 'contractor', '.', 'It', 'keeps', 'me', 'at', 'nigh
t', '.', 'RT', '@MichaelKors', ':', 'Check', 'me', 'out', 'as', 'a',
'devil', ',', 'age', '5', '.', 'Email', 'ur', 'halloween', 'to', 'ev
ents@michaelkors.com', ',', 'win', 'the', 'of', 'the', 'season',
'!', 'http://ow', '...', 'RT', '@FactsAboutBoys', ':', 'out', 'of',
'all', 'your', 'lies', ',', '"', 'i', 'love', 'you', '"', 'was', 'm
y', 'favorite', '.', '#factsaboutboys', 'New', 'tweetaway', '!', 'Fo
llow', '@Dermstore', '&', 'RT', 'to', 'enter', 'to', 'win', 'a', 'pr
ize', 'from', 'Obagi', '!', '1', 'winner', 'every', 'day', 'this',
'week', '!', 'http://budurl.com/DSobagi', 'Good', '!', 'Lmao', 'ever
ybody', 'tweeted', 'bout', 'just', 'now', '@Shells7474', 'same', 'ye
ar', '?', 'RT', '@KiLLa_iNk', ':', "Grown'n", 'up', 'as', 'a', 'ki
d', '...', 'I', 'thought', 'I', 'could', 'get', 'any', 'girl',
'...', 'who', 'was', 'I', "fool'n?!", 'The', 'top', 'priority', 'whe
n', 'moving', 'to', 'a', 'new', '?', 'Finding', 'a', 'dresser', ',',
'of', '.', 'Under', 'the', 'dryer', 'now', '.', "We'll", 'see', 'i
f', 'this', 'place', 'is', 'a', 'keeper', '!', 'to', 'Drop', 'OS',
'10.4', 'Tiger', 'Support', '?', 'Say', 'It', 'Isn't', 'So', '-', 'h
ttp://ht.ly/2ZsDA', '#education', '#teched', '#elearning', '#RayRa
y', '#ThatIsAll', '!.', 'man', 'real', 'i', 'fuck', 'wit', 'a', 'nik
ka', 'from', 'the', 'bottoms', 'imgood', '.', 'com', 'http://dld.bz/
y5fb', '@MClark_52', 'just', 'down', 'calmly', 'with', 'her', '&',
'tell', 'her', 'every', 'fucking', 'that', 'is', 'annoying', 'you',
'.', 'See', 'if', 'she', 'stays', '.', 'RT', '@imacsweb', ':', 'Deal
ers', ':', 'Before', 'you', 'get', 'sold', '"', 'an', 'for', 'that',
'",', 'out', 'what', 'smartphone', 'users', 'actually', 'do', 'htt
p://bit.ly/9Hyfcr', '(', 'via', '@eMa', '...', '@NickLikeWoe', 'YE
S', '.', '"', 'Actin', "'", 'like', 'a', ',', 'finna', 'get', 'you',
'hurt', '"', '-', 'Rihanna', '@MY_CHERRY_WET', 'head', 'They', 'to',
'do', 'an', 'episode', 'of', 'Cops', 'here', 'in', 'Cleveland', 'I',
'missed', 'house', 'of', 'glam', ':', '0', '(', 'in', 'the', 'room',
'before', '1', 'yess', '!', '!', 'time', 'til', 'practicee', 'No',
'injuries', 'in', 'three', 'shooting', 'incidents', 'in', 'Muskego
n', ':', 'Chronicle/Jeffrey', 'BallThe', 'owner', 'of', 'a', 'home',
'i', '...', 'http://bit.ly/bb1m4K', '@digg_dugg', 'since', 'I', "did
n't", 'know', 'what', 'the', 'you', 'were', 'in', 'the', 'first', 'p
lace', ',', 'I', 'overlooked', 'said', 'spelling', 'mistakes',
'...', '@InMyCrazyMind', 'i', 'have', 'to', 'turn', 'in', 'for', 'el
se', '-_-', 'RT', '@CapoRo722', ':', 'Lincecum', 'lookin', 'real',
'anxious', 'heyyyy', 'Hiiiiiiii', 'plzzzzzzzzz', 'plzzzzzzz', '+22',
'RT', '@gadling', ':', 'Five', 'tips', 'to', 'reduce', 'your', 'heal
th', 'risk', 'while', 'eating', 'food', '|', 'Gadling', '|', 'htt
p://aol.it/d9zm2R', "Couldn't", 'be', 'more', 'happy', 'for', '&',
'!!', 'Just', 'saw', 'a', 'for', 'the', 'new', 'goldeneye', 'AHHHH
H', '!!!!', 'If', 'i', 'die', 'dont', 'cry', 'just', 'get', 'and',
'fly', 'with', 'me', '..', 'Naw', 'we', 'pressd', 'da', 'nigga', 'h
e', 'did', 'some', 'slimey', 'shit', 'it', 'was', '2', 'of', 'us',
'four', 'of', 'dem', 'and', 'dey', 'was', 'some', 'brolik', 'grown',
'ass', '@Dope_Montana', '@DJJUSTN', 'yes', 'real', 'soon', '!!!', 'c
ant', 'wait', '!', '=)', '@bowlerhatlover', '@someonesmissing', 'tha
nk', 'u', '!', 'Watch', 'it', 'when', 'I', 'get', 'home', '!', ':p',
'Monaco', 'asking', 'customers', 'to', 'pics', 'for', 'its', 'new',
'blog', '&', 'they', 'supply', 'the', 'http://bit.ly/aXvNCf', 'Sugar
y', 'May', 'Raise', 'Diabetes', 'Risk', ':', 'http://bit.ly/cTwLpF',
'via', '@addthis', '#TDFAMILY', 'YOU', 'MUST', 'ME', 'IF', 'YOU', 'N
OT', 'BE', 'AT', 'EITHER', 'MEETING', '#Arcticmonkeys', '-ibetyouloo
kgoodonthedancefloor', '@Chikolarev', 'lol', 'ohhhhh', 'tru', 'tru',
'hahaha', 'gotta', 'stay', 'safe', '!!!!', 'I', 'just', 'put', 'extr
a', 'granola', 'in', 'my', 'cereal', '.', 'You', 'may', 'feel', 'a

```
s', 'if', 'you', 'have', 'out', 'of', 'time', 'because', 'there', 'a
re', '...', 'More', 'for', 'Gemini', 'http://twittascope.com/?sign=
3', 'kinda', 'pissed', 'but', 'still', 'got', 'til', 'friday', '@Jus
tyBiebsx3', 'Why', 'u', 'put', 'my', 'in', 'ur', 'tweets', '?', 'HAH
AHA', '#TeamLakers', '#TeamKobe', '#HandsDown', '#Nuffsaid', '@royal
prettygyal', 'good', 'gorgeous', '@FireMedic_Eric', 'Noooooooo', ':
(', '!', "don't", 'say', 'thaaaaat', '.', 'Dis', 'nigga', 'just', 'w
ent', 'on', 'First', '48', '!', 'UNC', 'lookin', "'", 'good', 'righ
t', 'now', '.', 'Might', 'start', 'this', '@MoeKhan19', 'Ha', '.',
'Get', 'one', 'of', 'them', '!', '@kiddotrue', 'fashionista.com', 'c
overed', 'the', 'story', '@kusshmeboricua', 'rt', 'http://twitpic.co
m/2yt9l6', 'HalloweenNight', '!', 'Photo', ':', 'http://tumblr.com/x
u0nbusxi', 'RT', '@J_Bieber_Facts', ':', 'Tweet', 'me', 'if', "you're
", 'online', 'and', 'bored', '!', ':)', 'Homeboy', 'Sandman', ':',
'"', 'Calm', 'Tornado', '"', '[', ']', 'http://bit.ly/9b85o3', 'RT',
'@iPennyAuctions', ':', '@Beezid', '-', 'BEEZID', 'is', 'celebratin
g', 'their', '1', 'year', 'Birthday', 'and', 'giving', 'YOU', 'the',
'greatest', 'gift', 'of', 'all', '!', 'out', 'Beezid™', 'Bu', '...',
'RT', '@akpierce', 'I', 'am', 'having', 'the', 'most', 'awesome/inap
propriate', 'conversation', 'with', '@pnuts_mama', '@rachelmariann
e', '@amylou890', 'right', 'now', '.', '#guesswhatabout', 'to', 'eve
ryone', 'who', 'likes', 'us', 'and', 'to', 'Erica', 'Haspel', 'for',
'being', 'the', '1,000', 'th', 'to', '"', 'Like', '"', 'our', 'pag
e', '!', 'LMFAOOOOOOOOOOOOOO', 'TANAYA', 'BRODY', 'LOOK', 'LIKE',
'A', 'RT', '@MayslesCinema', ':', '"', 'Garcia's', 'Playground', 'Ba
sketball', 'Film', 'Festival…', '100%', 'APPROVED', '"', 'http://bi
t.ly/cYKBtY', 'Early', 'TRIF', ':', 'Specific', '#2', 'no', 'longe
r', 'a', 'problem', ',', 'I', 'guess', '.', 'New', '#2', 'expected',
'this', 'weekend', '.', 'Agreed', '!!!', 'RT', '@RioOfLaw25', ':',
'It', "doesn't", 'to', 'much', 'to', 'turn', 'me', 'on', 'BUT', 'i
t', 'really', "don't", 'much', 'to', 'turn', 'me', 'OFF', '!', "B
J's", 'is', 'fuckin', 'gross', 'and', 'a', 'waste', 'of', 'RT', '@TB
ManOfTheYear', "I'm", 'gettin', 'a', 'lil', 'sad', '...', '<<', 'y',
'??', 'Ur', 'not', 'coming', '??', '"', "I'm", 'sure', 'Primo', 'i
s', 'very', 'motivational', '."', '@YeseniaJasmin', '=)', '@KOTCBLAD
E', 'my', 'bad', 'i', 'said', 'yes', 'sir', 'i', 'text', 'bac', 'o
n', 'my', 'go', 'thru', 'i', 'guess', 'Catching', 'up', 'with', 'Bel
a', '!', '(@', '48', '1221', '6th', 'Ave', ')', 'http://whrrl.com/e/
ibUyw', 'Vacation', 'In', 'My', 'Mind', 'the', 'movie', 'drops', 'Ma
rch', '17th', '.', 'Wrote', 'the', 'in', '09', "'", 'way', 'before',
'emo', 'man', 'made', 'his', 'movie', '.', '#FrosB4Hoes', '-Biggit
y', '@Boddingtons', 'Bone', 'taken', 'out', ',', 'tendon', '.',
"i'm", 'too', 'lazy', 'to', 'get', '&', 'get', 'dressed', '.', 'jus
t', 'too', 'lazy', ',', 'i', 'hope', 'you', 'understand', '.', 'Gott
a', 'love', 'getting', 'pulled', 'over', 'right', 'outside', 'your',
'office', '...', '#goodstarttotheday', ':-\\', '@MrBluz', 'lets', 'S
upport', 'Breast', 'Cancer', 'click', 'on', 'the', 'link', '$5', 'fr
om', 'each', 'sale', 'donated', 'http://ht.ly/30xrt?=njk2', 'My', 'c
ousin', 'is', 'deff', 'losing', 'ha', 'fckn', 'mind', '!', 'smh', 'T
ried', 'to', 'read', 'more', 'of', 'Amy', "McKay's", "'", 'The',
"',", 'but', 'it', 'just', "isn't", 'the', 'same', 'without', '@Dreh
al', 'reading', 'it', 'aloud', '.', '@sambolicious69', 'yess', 'i',
'am', '@MiGLBeatz', 'http://twitpic.com/31d68i', 'New', 'blog', 'pos
t', ':', 'Camping', 'creates', 'strong', 'family', 'bonds', 'http://
bit.ly/cC4oRK', '@RyanGerren', 'close', 'enough', 'to', 'smell', 'it
tt', 'RT', '@vjdaniel', ':', '#prayforindonesia', 'http://myloc.me/d
vLRd', 'Halloween', 'Activities', 'http://ping.fm/QIhmX', 'RT', '@He
yKikO', 'Every', 'girl', 'lives', 'for', 'the', '"', 'unexpected',
'hugs', 'from', 'behind', '"', '<', 'I', "wouldn't", 'say', '"',
'"...', 'but', 'they', 'r', 'nice', 'How', 'to', 'Create', 'the', 'P
erfect', 'Mudroom', '(', '10', 'photos', '):', 'A', 'mudroom', 'is',
```

'a', 'terrific', 'room', 'to', 'have', 'in', 'your', 'a', '...', 'ht
tp://bit.ly/bZiTd2', '#Home', 'Indonesians', 'try', 'to', 'return',
'to', 'homes', 'on', 'Mount', 'Merapi', '—', 'BBC', ':', 'The', 'tr
y', 'to', 'return', 't', '...', 'http://bit.ly/9RqIbO', 'JK', '@clin
teraction', 'looking', 'for', 'the', 'same', 'Life', 'is', 'like',
'photography', ',', 'we', 'use', 'the', 'negatives', 'to', '.', 'R
T', '@petershankman', ':', 'Surrounded', 'by', 'in', 'uncomfortable-
looking', 'ties', 'at', 'ORD', '.', "I'm", 'in', 'a', 'and', 'jean
s', ',', 'various', 'entities', 'that', 'I', '...', '@wood_thrush',
"I'm", 'just', 'a', 'normal', 'girl', ':', '3', 'Great', 'collectio
n', 'of', '...', 'http://bit.ly/9mTZk0', 'RT', '@GlobalFundWomen',
'Our', 'time', 'is', 'now', '!', 'Support', 'women', 'as', 'full',
'peacemaking', 'partners', '.', '#Make1325real', '&', 'sign', 'the',
'petition', ':', 'http://bit.ly/b0Rakg', '@lilmetch', 'Oh', '!', 'To
day', 'is', 'day', '!', 'Go', '@Rangers', '@ishakeitup', 'I', 'was',
'really', 'hoping', "y'all", 'would', 'be', 'driving', 'through', 'A
lbuquerque', 'on', 'Halloween', '...', 'I', 'turn', '21', 'and', "yo
u're", 'the', 'authority', 'on', 'cocktails', '.', '@MelanieDenmar
k', 'im', 'Tryin', 'to', 'out', 'the', 'But', 'is', 'the', 'ONLY',
'one', 'with', 'the', 'RT', '@martinquest', 'Wikipedia', 'is', 'grea
t', 'idk', 'what', 'anybody', 'says', '.', 'anyone', 'that', 'disagr
ees', 'is', 'a', 'fucking', '#mark', '#trickassbuster', '★ⓕⓤⓒⓚⓘⓝ
ⓞⓦⓝⓞⓦ★', '@iEATiT_n_BEATiT', '@iT0NGUEuSQUIRT', '@AmayaLei', '@ii
NenaBoo', '@BarbaraTheDoll', '@LookMa_AllGolds', '@SexyazzCC', '@IFu
kdHerMouth', '@NZAfro', 'And', 'yall', 'knew', 'heat', "wasn't", 'fu
ckin', 'wit', 'the', '@jazzyjaztho', 'hahaha', 'lmao', '@', 'mutha',
'bustin', '..', 'and', 'why', 'not', 'that', 'seems', 'like', 'a',
'fun', 'game', 'pahaha', 'jkaay', 'jkaay', 'and', 'Advanced', 'Techn
ology', '/', 'Jaeger', 'and', ':', 'Shenzhen', 'based', 'and',
'...', 'http://bit.ly/avRpiC', 'If', 'you', 'reading', 'this', 'Mars
hall', 'blog', ',', 'you', "can't", 'really', 'understand', 'media',
',', 'sorry', '.', 'http://marshallandme.com/', 'RT', '♥', 'it', 'lo
l', '@Obedbrown', ':', 'Thanks', 'boo', '!', ';-)', 'Gotta', 'keep',
'the', 'GOOD', 'in', 'ALL', 'areas', '!', 'LOL', 'RT', '@CherieMCamp
bell', '*', 'great', 'choice', 'of', '*', 'lmao', '@MsYellaMulann',
'@Ra_StayViolatin', 'call', 'us', 'now', 'lol', 'y', 'not', 'I', 'ju
st', 'won', 'this', 'free', 'auction', ':', '3-bikes', 'an', 'a', 'c
addy', 'combo', '!!', 'http://listia.com/148HW?r=36159', '@GlassMany
Colors', 'Thank', 'you', 'so', 'much', 'for', 'the', 'mention', '&',
'!', 'Make', 'my', 'day', '!!', '@AUsPriceless', 'come', 'on', 'ou
t', 'there', '...', "don't", 'b', 'scared', 'now', '..', 'A', 'Chanc
e', 'To', 'Win', 'A', 'Pocket', 'Devil', 'HD', 'Promo', 'Code', 'Wit
h', 'A', 'Retweet', 'Or', 'Comment', 'http://t.co/yqyJ3kV', 'via',
'@appadvice', '#teamlakers', "ya'll", 'ready', '??', ':)', 'Be', 'th
e', 'first', 'to', 'know', "what's", 'going', 'on', '!', 'out', 'th
e', 'November', '"', "What's", 'the', 'Buzz', '"', 'here', 'before',
"it's", 'even', 'back', 'from', '...', 'http://fb.me/xZFqpwxC', '#gl
ee', '!!', 'we', 'have', 'a', 'game', 'folks', '.', '@_Zaylito901',
'u', 'already', 'know', '!', '@primesuspect', 'I', 'am', 'glad',
'u', 'to', 'make', 'it', '.', '#backchannel', 'Overcast', 'and', '5
5', 'F', 'at', 'Presque', 'Isle', ',', 'ME', 'Winds', 'are', 'at',
'9.2', 'MPH', '(', '8', 'KT)', '.', 'The', 'is', '88%', '.', 'The',
'wind', 'chill', 'is', '52', '.', 'La', '@dartmedia', "ya'll", 'to',
'make', 'official', "api's", 'like', '@sfbart', 'does', 'for', 'deve
lopers', 'to', 'tie', 'into', 'RT', '@Jmack37', ':', 'doing', 'anyth
ing', 'cuz', 'ballin', 'on', 'this', "2k11<Haven't", 'made', 'up',
'my', 'mind', ',', 'worth', 'the', '?', 'I', 'would', 'like', 'to',
'thank', 'v', 'from', 'meijer', 'for', 'telling', 'me', 'geocachin
g', 'RT', '@matt_pc', ':', '@viviannereim', 'An', 'it', 'follows',
"Bello's", 'top', 'journalism', 'rule', ':', 'If', 'there', 'is', 'a
n', 'animal', ',', 'always', 'get', "it's", 'name', '.', 'Makes', 't

```
he', 'story', '1,000', '...', 'TORNADO', 'issued', 'for', 'of', 'cou
nty', 'in', 'until', '04:45', 'ET', '-', 'http://s.wx4.me/KRAHT002
2', '@squabtweets', 'It', 'was', 'a', 'short', ',', 'but', 'good',
'life', '.', 'How', 'many', 'of', 'us', 'can', 'say', "we've", 'bee
n', 'an', 'under-water', 'in', 'our', 'life', '?!', 'A', 'winner',
'to', 'the', 'end', '!', 'still', 'up', '4', 'no', 'reason', 'tho',
'@sabrina_hudgins', 'First', 'a', 'window', 'now', 'keys', '.', "Wha
t's", 'next', '..', '#js']
```

feature encoders

In [ ]:
```
# num characters: len()

# contains special char?
def special_char(word):
    return 0 if word.isalnum() else 1

def special_char_at(word):
    return 1 if ('@' in word) else 0

def special_char_hash(word):
    return 1 if ('#' in word) else 0

# contrains RT?
def RT(word):
    return 1 if ('RT' in word) else 0

# contrains URL?
def url(word):
  return 1 if ('http' in word) else 0
```

model with additional features

I will add 6 features computed on the original center word:

- the length of the center word
- whether it contains any special characters
- whether it contains the specific special character "@"
- whether it contains the specific special character "#"
- whether it contains "RT"
- whether it contrains URL

I will train the model with a window size of 0 and a window size of 1.

w=0, all vocab

In [ ]:

```
# construct datasets
train_w0_allvocab_addfeat = POSDataset(dataset=twpos_train, dataset_or
ig=orig_train, word2idx=word2idx_all_vocab, tag2idx=le, w=0,
                                       feature_funcs=[len, special_cha
r, special_char_at, special_char_hash, RT, url])
dev_w0_allvocab_addfeat = POSDataset(dataset=twpos_dev, dataset_orig=o
rig_dev, word2idx=word2idx_all_vocab, tag2idx=le, w=0,
                                     feature_funcs=[len, special_char,
special_char_at, special_char_hash, RT, url])
devtest_w0_allvocab_addfeat = POSDataset(dataset=twpos_devtest, datase
t_orig=orig_devtest, word2idx=word2idx_all_vocab, tag2idx=le, w=0,
                                         feature_funcs=[len, special_c
har, special_char_at, special_char_hash, RT, url])
```

In [ ]:

```
# instantiate model: single hidden layer 128 with tanh nonlinearity, w
=0, all vocab (random init)
tagger_w0_addfeat = FeedForwardNN(w=0, vocab_size=len(all_vocab), emb_
dim=50, nfeatures=6,
                            layer_sizes=[128, len(all_tags)],  # last la
yer is the output layer
                            layer_acts=[nn.Tanh(), nn.Identity()],   # n
n.CrossEntropyLoss() already includes softmax transformation
                            pretrained_emb=None, emb_freeze=False)

# instantiate optimizer
sgd = optim.SGD(tagger_w0_addfeat.parameters(), lr=0.02)

# train and eval
epoch_losses, train_evals, dev_evals, devtest_eval = main_process(
                                                                  model=tagger
_w0_addfeat,
                                                                  name='tagger
_w0_addfeat',  # file name used for using checkpoint
                                                                  optimizer=sg
d,
                                                                  criterion=n
n.CrossEntropyLoss(),   # objective: log loss
                                                                  train_data=t
rain_w0_allvocab_addfeat,
                                                                  batch_size=
1,
                                                                  shuffle=Tru
e,
                                                                  val_data=dev
_w0_allvocab_addfeat,
                                                                  test_data=de
vtest_w0_allvocab_addfeat,
                                                                  max_epochs=2
0,
                                                                  early_stoppi
ng=3   # when dev eval doesn't improve for 3 consecutive epochs
                                                                 )
```

```
epoch 1
  epoch loss: 31060.799917872762
  accuracy: 0.6065382370110917
  accuracy: 0.5978012860402406
  best model from epoch 1
epoch 2
  epoch loss: 17979.103977279097
  accuracy: 0.6153531815528313
  accuracy: 0.6038166355527899
  best model from epoch 2
epoch 3
  epoch loss: 13492.37095418881
  accuracy: 0.805312317571512
  accuracy: 0.7255756067205974
  best model from epoch 3
epoch 4
  epoch loss: 10987.344492305776
  accuracy: 0.8763572679509632
  accuracy: 0.7705870151420867
  best model from epoch 4
epoch 5
  epoch loss: 9082.45047461877
  accuracy: 0.9160537069468768
  accuracy: 0.7929890064302012
  best model from epoch 5
epoch 6
  epoch loss: 7938.7323304998135
  accuracy: 0.9033858727378867
  accuracy: 0.777224642190417
  best model from epoch 5
epoch 7
  epoch loss: 7117.466688679444
  accuracy: 0.9039112667834209
  accuracy: 0.7749429578925534
  best model from epoch 5
epoch 8
  epoch loss: 6382.096051272018
  accuracy: 0.9169877408056042
  accuracy: 0.7838622692387471
  best model from epoch 5
========= EARLY STOPPING =========
load best model
eval best model on devtest
  accuracy: 0.8031903427462815
```

After adding the 6 features, with a window size of 0, the best taggging accuracy on DEV is 79.30% from epoch 5; this best model has a tagging accuracy of 80.32% on DEVTEST.
Compared to the baseline tagger above, when w = 0, the additional features generated a slight improvement of tagging performance.

w=1, all vocab

In [ ]:

```
# construct datasets
train_w1_allvocab_addfeat = POSDataset(dataset=twpos_train, dataset_or
ig=orig_train, word2idx=word2idx_all_vocab, tag2idx=le, w=1,
                                       feature_funcs=[len, special_cha
r, special_char_at, special_char_hash, RT, url])
dev_w1_allvocab_addfeat = POSDataset(dataset=twpos_dev, dataset_orig=o
rig_dev, word2idx=word2idx_all_vocab, tag2idx=le, w=1,
                                     feature_funcs=[len, special_char,
special_char_at, special_char_hash, RT, url])
devtest_w1_allvocab_addfeat = POSDataset(dataset=twpos_devtest, datase
t_orig=orig_devtest, word2idx=word2idx_all_vocab, tag2idx=le, w=1,
                                         feature_funcs=[len, special_c
har, special_char_at, special_char_hash, RT, url])
```

In [ ]:

```
# instantiate model: single hidden layer 128 with tanh nonlinearity, w
=0, all vocab (random init)
tagger_w1_addfeat = FeedForwardNN(w=1, vocab_size=len(all_vocab), emb_
dim=50, nfeatures=6,
                            layer_sizes=[128, len(all_tags)],  # last la
yer is the output layer
                            layer_acts=[nn.Tanh(), nn.Identity()],   # n
n.CrossEntropyLoss() already includes softmax transformation
                            pretrained_emb=None, emb_freeze=False)

# instantiate optimizer
sgd = optim.SGD(tagger_w1_addfeat.parameters(), lr=0.02)

# train and eval
epoch_losses, train_evals, dev_evals, devtest_eval = main_process(
                                                                  model=tagger
_w1_addfeat,
                                                                  name='tagger
_w1_addfeat',  # file name used for using checkpoint
                                                                  optimizer=sg
d,
                                                                  criterion=n
n.CrossEntropyLoss(),   # objective: log loss
                                                                  train_data=t
rain_w1_allvocab_addfeat,
                                                                  batch_size=
1,
                                                                  shuffle=Tru
e,
                                                                  val_data=dev
_w1_allvocab_addfeat,
                                                                  test_data=de
vtest_w1_allvocab_addfeat,
                                                                  max_epochs=2
0,
                                                                  early_stoppi
ng=3   # when dev eval doesn't improve for 3 consecutive epochs
                                                                 )
```

```
epoch 1
  epoch loss: 30580.26798160514
  accuracy: 0.6035610040863981
  accuracy: 0.6009126737191454
  best model from epoch 1
epoch 2
  epoch loss: 16539.012995059427
  accuracy: 0.7466433158201985
  accuracy: 0.7143746110765401
  best model from epoch 2
epoch 3
  epoch loss: 11679.177184734923
  accuracy: 0.8239929947460596
  accuracy: 0.7699647376063058
  best model from epoch 3
epoch 4
  epoch loss: 8833.067225562787
  accuracy: 0.9167542323409223
  accuracy: 0.8089607965152458
  best model from epoch 4
epoch 5
  epoch loss: 6721.740915585424
  accuracy: 0.9287799182720373
  accuracy: 0.800663762704833
  best model from epoch 4
epoch 6
  epoch loss: 5487.215312124229
  accuracy: 0.9408639813193228
  accuracy: 0.8066791122173823
  best model from epoch 4
epoch 7
  epoch loss: 4583.883293655882
  accuracy: 0.961004086398132
  accuracy: 0.8126944617299315
  best model from epoch 7
epoch 8
  epoch loss: 3765.8063204532154
  accuracy: 0.9622883829538821
  accuracy: 0.8187098112424808
  best model from epoch 8
epoch 9
  epoch loss: 3186.9526656757507
  accuracy: 0.9661412726211325
  accuracy: 0.8170504044803982
  best model from epoch 8
epoch 10
  epoch loss: 2596.1511684314914
  accuracy: 0.961004086398132
  accuracy: 0.8070939639079029
  best model from epoch 8
epoch 11
  epoch loss: 2214.217167658359
  accuracy: 0.978225335668418
  accuracy: 0.8187098112424808
  best model from epoch 8
========= EARLY STOPPING =========
load best model
eval best model on devtest
  accuracy: 0.8256089674498814
```

After adding the 6 features, with a window size of 1, the best taggging accuracy on DEV is 81.87% from epoch 8; this best model has a tagging accuracy of 82.56% on DEVTEST. Compared to the baseline tagger, when w=1, we also saw a slight improvement of tagging performance.

Additionally, I experimented with not including the binary feature of whether there is any special character in the center word with w=1; thus we have 5 additional features in total this time:

- the length of the center word
- whether it contains the specific special character "@"
- whether it contains the specific special character "#"
- whether it contains "RT"
- whether it contrains URL

Results:

In [ ]:
```
# construct datasets
train_w1_allvocab_addfeat = POSDataset(dataset=twpos_train, dataset_or
ig=orig_train, word2idx=word2idx_all_vocab, tag2idx=le, w=1,
                                       feature_funcs=[len, special_cha
r_at, special_char_hash, RT, url])
dev_w1_allvocab_addfeat = POSDataset(dataset=twpos_dev, dataset_orig=o
rig_dev, word2idx=word2idx_all_vocab, tag2idx=le, w=1,
                                     feature_funcs=[len, special_char_
at, special_char_hash, RT, url])
devtest_w1_allvocab_addfeat = POSDataset(dataset=twpos_devtest, datase
t_orig=orig_devtest, word2idx=word2idx_all_vocab, tag2idx=le, w=1,
                                         feature_funcs=[len, special_c
har_at, special_char_hash, RT, url])
```

In [ ]:

```
# instantiate model: single hidden layer 128 with tanh nonlinearity, w
=0, all vocab (random init)
tagger_w1_addfeat = FeedForwardNN(w=1, vocab_size=len(all_vocab), emb_
dim=50, nfeatures=5,
                            layer_sizes=[128, len(all_tags)],  # last la
yer is the output layer
                            layer_acts=[nn.Tanh(), nn.Identity()],   # n
n.CrossEntropyLoss() already includes softmax transformation
                            pretrained_emb=None, emb_freeze=False)

# instantiate optimizer
sgd = optim.SGD(tagger_w1_addfeat.parameters(), lr=0.02)

# train and eval
epoch_losses, train_evals, dev_evals, devtest_eval = main_process(
                                                                  model=tagger
_w1_addfeat,
                                                                  name='tagger
_w1_addfeat',  # file name used for using checkpoint
                                                                  optimizer=sg
d,
                                                                  criterion=n
n.CrossEntropyLoss(),   # objective: log loss
                                                                  train_data=t
rain_w1_allvocab_addfeat,
                                                                  batch_size=
1,
                                                                  shuffle=Tru
e,
                                                                  val_data=dev
_w1_allvocab_addfeat,
                                                                  test_data=de
vtest_w1_allvocab_addfeat,
                                                                  max_epochs=2
0,
                                                                  early_stoppi
ng=3   # when dev eval doesn't improve for 3 consecutive epochs
                                                                 )
```

```
epoch 1
  epoch loss: 33888.83757842542
  accuracy: 0.5296555750145943
  accuracy: 0.5200165940676208
  best model from epoch 1
epoch 2
  epoch loss: 19600.554554729548
  accuracy: 0.7464681844716871
  accuracy: 0.7102260941713338
  best model from epoch 2
epoch 3
  epoch loss: 12578.3315212979
  accuracy: 0.8306479859894921
  accuracy: 0.7629122588674548
  best model from epoch 3
epoch 4
  epoch loss: 9132.86184076062
  accuracy: 0.8803269118505546
  accuracy: 0.790292470441817
  best model from epoch 4
epoch 5
  epoch loss: 6857.575029499546
  accuracy: 0.9350262697022768
  accuracy: 0.8129018875751919
  best model from epoch 5
epoch 6
  epoch loss: 5403.954741335297
  accuracy: 0.9364856976065382
  accuracy: 0.7969300974901473
  best model from epoch 5
epoch 7
  epoch loss: 4395.515808695583
  accuracy: 0.9488032691185055
  accuracy: 0.8176726820161793
  best model from epoch 7
epoch 8
  epoch loss: 3736.41588832973
  accuracy: 0.9467600700525394
  accuracy: 0.8039825762289982
  best model from epoch 7
epoch 9
  epoch loss: 3005.7384754534537
  accuracy: 0.9437244600116754
  accuracy: 0.7931964322754615
  best model from epoch 7
epoch 10
  epoch loss: 2589.5889414228945
  accuracy: 0.9637478108581436
  accuracy: 0.8081310931342045
  best model from epoch 7
========= EARLY STOPPING =========
load best model
eval best model on devtest
  accuracy: 0.8303513688294891
```

As shown above, with 5 additional features and a window size of 1, the best taggging accuracy on DEV is 81.77% from epoch 7; this best model has a tagging accuracy of 83.04% on DEVTEST. The performance on DEV is similar to that with 6 additional features, but the performance on DEVTEST is slightly better than using 6 additional features.

In summary, we have seen an improvement of model performance with additional features for both window sizes 0 and 1.

1.3 pretrained embeddings

I used the embedding for "UUUNKKK" when encountering words not in the pretrained embeddings, and used the embedding for "<\/s>" for both "<\s>" and "<\/s>".

fine-tuning

w=0

In [ ]:

```
# instantiate model: single hidden layer 128 with tanh nonlinearity, w
=0, fine-tuned pretrained embedding
tagger_w0_tunedpretrained = FeedForwardNN(w=0, vocab_size=len(emb_pret
rained_vocab), emb_dim=50, nfeatures=0,
                          layer_sizes=[128, len(all_tags)],  # last la
yer is the output layer
                          layer_acts=[nn.Tanh(), nn.Identity()],   # n
n.CrossEntropyLoss() already includes softmax transformation
                          pretrained_emb=emb_pretrained, emb_freeze=Fa
lse)

# instantiate optimizer
sgd = optim.SGD(tagger_w0_tunedpretrained.parameters(), lr=0.02)

# train and eval
epoch_losses, train_evals, dev_evals, devtest_eval = main_process(
                                              model=tagger_w0_tunedpretrai
ned,
                                              name='tagger_w0_tunedpretrai
ned',  # file name used for using checkpoint
                                              optimizer=sgd,
                                              criterion=nn.CrossEntropyLos
s(),   # objective: log loss
                                              train_data=train_w0_30k,
                                              batch_size=1,
                                              shuffle=True,
                                              val_data=dev_w0_30k,
                                              test_data=devtest_w0_30k,
                                              max_epochs=20,
                                              early_stopping=3   # when de
v eval doesn't improve for 3 consecutive epochs
                                             )
```

```
epoch 1
  epoch loss: 17637.035919039045
  accuracy: 0.8551079976649153
  accuracy: 0.8294959551960174
  best model from epoch 1
epoch 2
  epoch loss: 8682.902948501578
  accuracy: 0.8724460011675423
  accuracy: 0.8286662518149761
  best model from epoch 1
epoch 3
  epoch loss: 7443.902273253319
  accuracy: 0.8785755983654407
  accuracy: 0.8261771416718523
  best model from epoch 1
epoch 4
  epoch loss: 6872.446804946703
  accuracy: 0.8856392294220665
  accuracy: 0.8290811035054968
  best model from epoch 1
========= EARLY STOPPING =========
load best model
eval best model on devtest
  accuracy: 0.8294891140331968
```

With a window size of 0, the best taggging accuracy on DEV is 82.95% from epoch 1 when fine-tuning pre-trained embeddings; this best model has a tagging accuracy of 82.95% on DEVTEST. Model performance improved compared to the baseline tagger.

w=1

In [ ]:

```
# instantiate model: single hidden layer 128 with tanh nonlinearity, w
=1, fine-tuned pretrained embedding
tagger_w1_tunedpretrained = FeedForwardNN(w=1, vocab_size=len(emb_pret
rained_vocab), emb_dim=50, nfeatures=0,
                          layer_sizes=[128, len(all_tags)],  # last la
yer is the output layer
                          layer_acts=[nn.Tanh(), nn.Identity()],   # n
n.CrossEntropyLoss() already includes softmax transformation
                          pretrained_emb=emb_pretrained, emb_freeze=Fa
lse)

# instantiate optimizer
sgd = optim.SGD(tagger_w1_tunedpretrained.parameters(), lr=0.02)

# train and eval
epoch_losses, train_evals, dev_evals, devtest_eval = main_process(
                                            model=tagger_w1_tunedpretrai
ned,
                                            name='tagger_w1_tunedpretrai
ned',  # file name used for using checkpoint
                                            optimizer=sgd,
                                            criterion=nn.CrossEntropyLos
s(),   # objective: log loss
                                            train_data=train_w1_30k,
                                            batch_size=1,
                                            shuffle=True,
                                            val_data=dev_w1_30k,
                                            test_data=devtest_w1_30k,
                                            max_epochs=20,
                                            early_stopping=3   # when de
v eval doesn't improve for 3 consecutive epochs
                                           )
```

```
epoch 1
  epoch loss: 12692.350564856668
  accuracy: 0.9016929363689433
  accuracy: 0.8626840904376686
  best model from epoch 1
epoch 2
  epoch loss: 6180.496532521094
  accuracy: 0.9288382953882078
  accuracy: 0.8691142916407384
  best model from epoch 2
epoch 3
  epoch loss: 4832.6801328056945
  accuracy: 0.9440747227086982
  accuracy: 0.8718108276291225
  best model from epoch 3
epoch 4
  epoch loss: 3995.835629587277
  accuracy: 0.9546993578517221
  accuracy: 0.8643434971997511
  best model from epoch 3
epoch 5
  epoch loss: 3339.6667113360572
  accuracy: 0.9584354932866317
  accuracy: 0.8680771624144369
  best model from epoch 3
epoch 6
  epoch loss: 2862.923839181902
  accuracy: 0.9572679509632224
  accuracy: 0.8581207218419415
  best model from epoch 3
========= EARLY STOPPING =========
load best model
eval best model on devtest
  accuracy: 0.881008838111662
```

With a window size of 1, the best taggging accuracy on DEV is 87.18% from epoch 3 when fine-tuning pre-trained embeddings; this best model has a tagging accuracy of 88.10% on DEVTEST. Model performance improved by a large degree compared to the baseline tagger.

Additionally, we see that our models reached a stable high performance with little to no training for both w=0 and w=1 when using pretrained word embeddings.

freeze, w=1

We have experimented with fine-tuning pretrained embeddings in the previous section; here, we experiment with freezing pretrained embeddings with w=1:

In [ ]:

```
# instantiate model: single hidden layer 128 with tanh nonlinearity, w
=1, fixed pretrained embedding
tagger_w1_fixedpretrained = FeedForwardNN(w=1, vocab_size=len(emb_pret
rained_vocab), emb_dim=50, nfeatures=0,
                          layer_sizes=[128, len(all_tags)],  # last la
yer is the output layer
                          layer_acts=[nn.Tanh(), nn.Identity()],   # n
n.CrossEntropyLoss() already includes softmax transformation
                          pretrained_emb=emb_pretrained, emb_freeze=Tr
ue)

# instantiate optimizer
sgd = optim.SGD(tagger_w1_fixedpretrained.parameters(), lr=0.02)

# train and eval
epoch_losses, train_evals, dev_evals, devtest_eval = main_process(
                                              model=tagger_w1_fixedpretrai
ned,
                                              name='tagger_w1_fixedpretrai
ned',  # file name used for using checkpoint
                                              optimizer=sgd,
                                              criterion=nn.CrossEntropyLos
s(),   # objective: log loss
                                              train_data=train_w1_30k,
                                              batch_size=1,
                                              shuffle=True,
                                              val_data=dev_w1_30k,
                                              test_data=devtest_w1_30k,
                                              max_epochs=20,
                                              early_stopping=3   # when de
v eval doesn't improve for 3 consecutive epochs
                                             )
```

```
epoch 1
  epoch loss: 9565.277853610502
  accuracy: 0.9389959136018681
  accuracy: 0.8651732005807924
  best model from epoch 1
epoch 2
  epoch loss: 3710.707942625414
  accuracy: 0.9475773496789258
  accuracy: 0.866210329807094
  best model from epoch 2
epoch 3
  epoch loss: 3267.152017366822
  accuracy: 0.9457676590776416
  accuracy: 0.8624766645924082
  best model from epoch 2
epoch 4
  epoch loss: 3070.9350762073364
  accuracy: 0.9568009340338587
  accuracy: 0.8680771624144369
  best model from epoch 4
epoch 5
  epoch loss: 2900.1935686099264
  accuracy: 0.9558669001751313
  accuracy: 0.8693217174859987
  best model from epoch 5
epoch 6
  epoch loss: 2761.0836894430704
  accuracy: 0.9591360186806772
  accuracy: 0.8684920141049575
  best model from epoch 5
epoch 7
  epoch loss: 2662.3412220583004
  accuracy: 0.9587273788674839
  accuracy: 0.8716034017838623
  best model from epoch 7
epoch 8
  epoch loss: 2544.4780451544757
  accuracy: 0.9604786923525978
  accuracy: 0.8686994399502178
  best model from epoch 7
epoch 9
  epoch loss: 2413.225081450424
  accuracy: 0.964681844716871
  accuracy: 0.8678697365691765
  best model from epoch 7
epoch 10
  epoch loss: 2349.3169193156086
  accuracy: 0.9654991243432575
  accuracy: 0.8720182534743829
  best model from epoch 10
epoch 11
  epoch loss: 2275.460520183103
  accuracy: 0.9647402218330414
  accuracy: 0.8697365691765194
  best model from epoch 10
epoch 12
  epoch loss: 2165.3676792832916
  accuracy: 0.966841798015178
  accuracy: 0.8674548848786559
  best model from epoch 10
epoch 13
```

```
  epoch loss: 2077.958917092606
  accuracy: 0.9709865732632808
  accuracy: 0.8672474590333955
  best model from epoch 10
========= EARLY STOPPING =========
load best model
eval best model on devtest
  accuracy: 0.8784220737227851
```

With a window size of 1, the best taggging accuracy on DEV is 87.20% from epoch 10 with fixed pre-trained embeddings; this best model has a tagging accuracy of 87.84% on DEVTEST. This model performance is comparable to fune-tuning pre-trained embeddings in general. However, when pre-trained embeddings are fixed, the model takes more epochs of training to reach a stable high performance than when fine-tuning the pretrained embeddings.

add features

We will use w=1, fine-tuned pretrianed embeddings, and 5 additional features:

- the length of the center word
- whether it contains the specific special character "@"
- whether it contains the specific special character "#"
- whether it contains "RT"
- whether it contrains URL

In [ ]:
```
# construct datasets
train_w1_30k_addfeat = POSDataset(dataset=twpos_train, dataset_orig=or
ig_train, word2idx=word2idx_emb_pretrained_vocab, tag2idx=le, w=1,
                                  feature_funcs=[len, special_char_at,
special_char_hash, RT, url])
dev_w1_30k_addfeat = POSDataset(dataset=twpos_dev, dataset_orig=orig_d
ev, word2idx=word2idx_emb_pretrained_vocab, tag2idx=le, w=1,
                                feature_funcs=[len, special_char_at, s
pecial_char_hash, RT, url])
devtest_w1_30k_addfeat = POSDataset(dataset=twpos_devtest, dataset_ori
g=orig_devtest, word2idx=word2idx_emb_pretrained_vocab, tag2idx=le, w=
1,
                                    feature_funcs=[len, special_char_a
t, special_char_hash, RT, url])
```

In [ ]:

```
# instantiate model: single hidden layer 128 with tanh nonlinearity, w
=1, fine-tuned pretrained embedding, additional features
tagger_w1_tunedpretrained_addfeat = FeedForwardNN(
    w=1, vocab_size=len(emb_pretrained_vocab), emb_dim=50, nfeatures=
5,
    layer_sizes=[128, len(all_tags)],  # last layer is the output laye
r
    layer_acts=[nn.Tanh(), nn.Identity()],   # nn.CrossEntropyLoss() a
lready includes softmax transformation
    pretrained_emb=emb_pretrained, emb_freeze=False
)

# instantiate optimizer
sgd = optim.SGD(tagger_w1_tunedpretrained_addfeat.parameters(), lr=0.0
2)

# train and eval
epoch_losses, train_evals, dev_evals, devtest_eval = main_process(
                                            model=tagger_w1_tunedpretrai
ned_addfeat,
                                            name='tagger_w1_tunedpretrai
ned_addfeat',  # file name used for using checkpoint
                                            optimizer=sgd,
                                            criterion=nn.CrossEntropyLos
s(),   # objective: log loss
                                            train_data=train_w1_30k_addf
eat,
                                            batch_size=1,
                                            shuffle=True,
                                            val_data=dev_w1_30k_addfeat,
                                            test_data=devtest_w1_30k_add
feat,
                                            max_epochs=20,
                                            early_stopping=3   # when de
v eval doesn't improve for 3 consecutive epochs
                                           )
```

```
epoch 1
  epoch loss: 10977.770401666316
  accuracy: 0.9254524226503211
  accuracy: 0.8574984443061605
  best model from epoch 1
epoch 2
  epoch loss: 5040.5131826071765
  accuracy: 0.9332165791009924
  accuracy: 0.8512756689483509
  best model from epoch 1
epoch 3
  epoch loss: 4190.599393943067
  accuracy: 0.9508464681844717
  accuracy: 0.8643434971997511
  best model from epoch 3
epoch 4
  epoch loss: 3579.412623844145
  accuracy: 0.9542323409223584
  accuracy: 0.8668326073428749
  best model from epoch 4
epoch 5
  epoch loss: 3131.7722008914984
  accuracy: 0.9611792177466433
  accuracy: 0.87015142086704
  best model from epoch 5
epoch 6
  epoch loss: 2862.515060129004
  accuracy: 0.9366608289550497
  accuracy: 0.830533084422319
  best model from epoch 5
epoch 7
  epoch loss: 2395.6492170088786
  accuracy: 0.95569176882662
  accuracy: 0.8562538892345987
  best model from epoch 5
epoch 8
  epoch loss: 2237.443852695683
  accuracy: 0.9688266199649738
  accuracy: 0.8589504252229828
  best model from epoch 5
========= EARLY STOPPING =========
load best model
eval best model on devtest
  accuracy: 0.8756197456348351
```

After adding features, with w=1, the best taggging accuracy on DEV is 87.02% from epoch 5 when fine-tuning pre-trained embeddings; this best model has a tagging accuracy of 87.56% on DEVTEST. Compared to the tagger with the same setup but without additional featues, the tagging performance decreased slightly, indicating that the features we developed in 1.2 is no longer helpful with fine-tuned pre-trained embeddings and w=1.

1.4 architecture engineering

We will use fine-tuned pretrianed embeddings configuration without additional features.

window size: w = 2

In [ ]:
```
# construct datasets: w = 2, pretrained 30k vocab encoding
train_w2_30k = POSDataset(dataset=twpos_train, dataset_orig=orig_trai
n, word2idx=word2idx_emb_pretrained_vocab, tag2idx=le, w=2, feature_fu
ncs=None)
dev_w2_30k = POSDataset(dataset=twpos_dev, dataset_orig=orig_dev, word
2idx=word2idx_emb_pretrained_vocab, tag2idx=le, w=2, feature_funcs=Non
e)
devtest_w2_30k = POSDataset(dataset=twpos_devtest, dataset_orig=orig_d
evtest, word2idx=word2idx_emb_pretrained_vocab, tag2idx=le, w=2, featu
re_funcs=None)
```

In [ ]:

```
# instantiate model
tagger_w2_tunedpretrained = FeedForwardNN(w=2, vocab_size=len(emb_pret
rained_vocab), emb_dim=50, nfeatures=0,
                                          layer_sizes=[128, len(all_ta
gs)],  # last layer is the output layer
                                          layer_acts=[nn.Tanh(), nn.Id
entity()],   # nn.CrossEntropyLoss() already includes softmax transfor
mation
                                          pretrained_emb=emb_pretraine
d, emb_freeze=False)

# instantiate optimizer
sgd = optim.SGD(tagger_w2_tunedpretrained.parameters(), lr=0.02)

# train and eval
epoch_losses, train_evals, dev_evals, devtest_eval = main_process(
                                          model=tagger_w2_tunedpretrai
ned,
                                          name='tagger_w2_tunedpretrai
ned',  # file name used for using checkpoint
                                          optimizer=sgd,
                                          criterion=nn.CrossEntropyLos
s(),   # objective: log loss
                                          train_data=train_w2_30k,
                                          batch_size=1,
                                          shuffle=True,
                                          val_data=dev_w2_30k,
                                          test_data=devtest_w2_30k,
                                          max_epochs=20,
                                          early_stopping=3   # when de
v eval doesn't improve for 3 consecutive epochs
                                         )
```

```
epoch 1
  epoch loss: 8421.208888161826
  accuracy: 0.9460595446584938
  accuracy: 0.8504459655673097
  best model from epoch 1
epoch 2
  epoch loss: 3063.2848283956723
  accuracy: 0.9695855224751897
  accuracy: 0.8606098319850654
  best model from epoch 2
epoch 3
  epoch loss: 2305.7429271449073
  accuracy: 0.969994162288383
  accuracy: 0.8533499274009542
  best model from epoch 2
epoch 4
  epoch loss: 1882.3449916254554
  accuracy: 0.9775248102743724
  accuracy: 0.855839037544078
  best model from epoch 2
epoch 5
  epoch loss: 1557.537823565544
  accuracy: 0.9859311150029189
  accuracy: 0.8535573532462145
  best model from epoch 2
========= EARLY STOPPING =========
load best model
eval best model on devtest
  accuracy: 0.8704462168570812
```

With a window size of 2, the best taggging accuracy on DEV is 86.06% from epoch 2 when fine-tuning pre-trained embeddings without additional features; this best model has a tagging accuracy of 87.04% on DEVTEST.

window size	hidden layer	hidden layer size	best DEV accuracy	best epoch	DEVTEST accuracy
0	1	128	82.95%	1	82.95%
1	1	128	87.18%	3	88.10%
2	1	128	86.06%	2	87.04%

The table above shows the comparison of tagging performance with three different window sizes (0, 1, 2). Overall, a window size of 1 performed the best on both DEV and DEVTEST sets. Therefore, in the experiments below, we will use w=1.

hidden layers

We experiment with the following combination of hyperparameters:

- number of hidden layers: [1, 2]
- layer widths: [128, 256, 512]

In [ ]:

```
devtest_evals_all = []

num_hiddens = [1, 2]
layer_widths = [128, 256, 512]


for num_hidden in num_hiddens:

    devtest_evals_widths = []

    for layer_width in layer_widths:

        print(f'num_hidden: {num_hidden}; layer_width: {layer_widt
h}.')

        # instantiate model
        tagger_w1_tunedpretrained_exp = FeedForwardNN(w=1, vocab_size=
len(emb_pretrained_vocab), emb_dim=50, nfeatures=0,
                                                      layer_sizes=[layer_w
idth for _ in range(num_hidden)]+[len(all_tags)],  # last layer is the
output layer
                                                      layer_acts=[nn.Tanh
() for _ in range(num_hidden)]+[nn.Identity()],   # nn.CrossEntropyLos
s() already includes softmax transformation
                                                      pretrained_emb=emb_p
retrained, emb_freeze=False)

        # instantiate optimizer
        sgd = optim.SGD(tagger_w1_tunedpretrained_exp.parameters(), lr
=0.02)

        # train and eval
        epoch_losses, train_evals, dev_evals, devtest_eval = main_proc
ess(
                                                      model=tagger_w1_tune
dpretrained_exp,
                                                      name='tagger_w1_tune
dpretrained_exp_hidden'+str(num_hidden)+'_width'+str(layer_width),  #
file name used for using checkpoint
                                                      optimizer=sgd,
                                                      criterion=nn.CrossEn
tropyLoss(),   # objective: log loss
                                                      train_data=train_w1_
30k,
                                                      batch_size=1,
                                                      shuffle=True,
                                                      val_data=dev_w1_30k,
                                                      test_data=devtest_w1
_30k,
                                                      max_epochs=20,
                                                      early_stopping=3   #
when dev eval doesn't improve for 3 consecutive epochs
                                                  )
        devtest_evals_widths.append(devtest_eval)

    devtest_evals_all.append(devtest_evals_widths)
```

```
num_hidden: 1; layer_width: 128.
epoch 1
  epoch loss: 8002.428549980132
  accuracy: 0.958610624635143
  accuracy: 0.8643434971997511
  best model from epoch 1
epoch 2
  epoch loss: 2808.7195028667065
  accuracy: 0.9612375948628138
  accuracy: 0.8572910184609003
  best model from epoch 1
epoch 3
  epoch loss: 2359.8608396045565
  accuracy: 0.9731465265615878
  accuracy: 0.8562538892345987
  best model from epoch 1
epoch 4
  epoch loss: 2066.111594126962
  accuracy: 0.9781669585522476
  accuracy: 0.8601949802945447
  best model from epoch 1
========= EARLY STOPPING =========
load best model
eval best model on devtest
  accuracy: 0.8689372709635698
num_hidden: 1; layer_width: 256.
epoch 1
  epoch loss: 7259.04301124204
  accuracy: 0.9603035610040864
  accuracy: 0.8635137938187099
  best model from epoch 1
epoch 2
  epoch loss: 2673.097251441446
  accuracy: 0.9698190309398715
  accuracy: 0.8601949802945447
  best model from epoch 1
epoch 3
  epoch loss: 2298.74026803753
  accuracy: 0.9685347343841214
  accuracy: 0.8548019083177764
  best model from epoch 1
epoch 4
  epoch loss: 2101.1089563524383
  accuracy: 0.9755983654407472
  accuracy: 0.8585355735324621
  best model from epoch 1
========= EARLY STOPPING =========
load best model
eval best model on devtest
  accuracy: 0.8682905798663505
num_hidden: 1; layer_width: 512.
epoch 1
  epoch loss: 6607.899181794818
  accuracy: 0.9553998832457676
  accuracy: 0.8583281476872018
  best model from epoch 1
epoch 2
  epoch loss: 2644.0715728866116
  accuracy: 0.9690017513134851
  accuracy: 0.8608172578303257
  best model from epoch 2
```

```
epoch 3
  epoch loss: 2248.402686002717
  accuracy: 0.9715703444249854
  accuracy: 0.8581207218419415
  best model from epoch 2
epoch 4
  epoch loss: 2070.2316175494902
  accuracy: 0.9730297723292469
  accuracy: 0.8539722049367351
  best model from epoch 2
epoch 5
  epoch loss: 1829.7429495568172
  accuracy: 0.9740805604203152
  accuracy: 0.8483717071147064
  best model from epoch 2
========= EARLY STOPPING =========
load best model
eval best model on devtest
  accuracy: 0.8706617805561544
num_hidden: 2; layer_width: 128.
epoch 1
  epoch loss: 16385.028287690613
  accuracy: 0.9357851722124927
  accuracy: 0.8276291225886746
  best model from epoch 1
epoch 2
  epoch loss: 3806.9403331644044
  accuracy: 0.9571511967308814
  accuracy: 0.8436009126737192
  best model from epoch 2
epoch 3
  epoch loss: 3045.5770183272175
  accuracy: 0.9650321074138938
  accuracy: 0.846090022816843
  best model from epoch 3
epoch 4
  epoch loss: 2486.232075191664
  accuracy: 0.9538237011091653
  accuracy: 0.8336444721012238
  best model from epoch 3
epoch 5
  epoch loss: 2251.5237577389566
  accuracy: 0.9711617046117922
  accuracy: 0.845467745281062
  best model from epoch 3
epoch 6
  epoch loss: 2060.1807890257332
  accuracy: 0.9758902510215995
  accuracy: 0.8485791329599668
  best model from epoch 6
epoch 7
  epoch loss: 1863.1823431983983
  accuracy: 0.978984238178634
  accuracy: 0.8477494295789255
  best model from epoch 6
epoch 8
  epoch loss: 1733.3648369738753
  accuracy: 0.9768826619964974
  accuracy: 0.8421489317568969
  best model from epoch 6
epoch 9
```

```
  epoch loss: 1603.3903273756837
  accuracy: 0.9812609457092819
  accuracy: 0.8465048745073637
  best model from epoch 6
========= EARLY STOPPING =========
load best model
eval best model on devtest
  accuracy: 0.8566501401164044
num_hidden: 2; layer_width: 256.
epoch 1
  epoch loss: 12924.387187344973
  accuracy: 0.9529480443666083
  accuracy: 0.835096453018046
  best model from epoch 1
epoch 2
  epoch loss: 3113.0720837184886
  accuracy: 0.9635143023934618
  accuracy: 0.8301182327317984
  best model from epoch 1
epoch 3
  epoch loss: 2592.9771767035795
  accuracy: 0.9604203152364273
  accuracy: 0.8346816013275254
  best model from epoch 1
epoch 4
  epoch loss: 2357.0435707251213
  accuracy: 0.9685347343841214
  accuracy: 0.8365484339348683
  best model from epoch 4
epoch 5
  epoch loss: 2176.8374313069844
  accuracy: 0.97594862813777
  accuracy: 0.845467745281062
  best model from epoch 5
epoch 6
  epoch loss: 2021.5170714379217
  accuracy: 0.9614711033274956
  accuracy: 0.8299108068865381
  best model from epoch 5
epoch 7
  epoch loss: 1885.018652090294
  accuracy: 0.9732632807939288
  accuracy: 0.8406969508400747
  best model from epoch 5
epoch 8
  epoch loss: 1772.400274686241
  accuracy: 0.9791009924109749
  accuracy: 0.8340593237917444
  best model from epoch 5
========= EARLY STOPPING =========
load best model
eval best model on devtest
  accuracy: 0.8532011209312352
num_hidden: 2; layer_width: 512.
epoch 1
  epoch loss: 9846.733110608813
  accuracy: 0.9598949211908931
  accuracy: 0.8319850653391413
  best model from epoch 1
epoch 2
  epoch loss: 2894.1136875614075
```

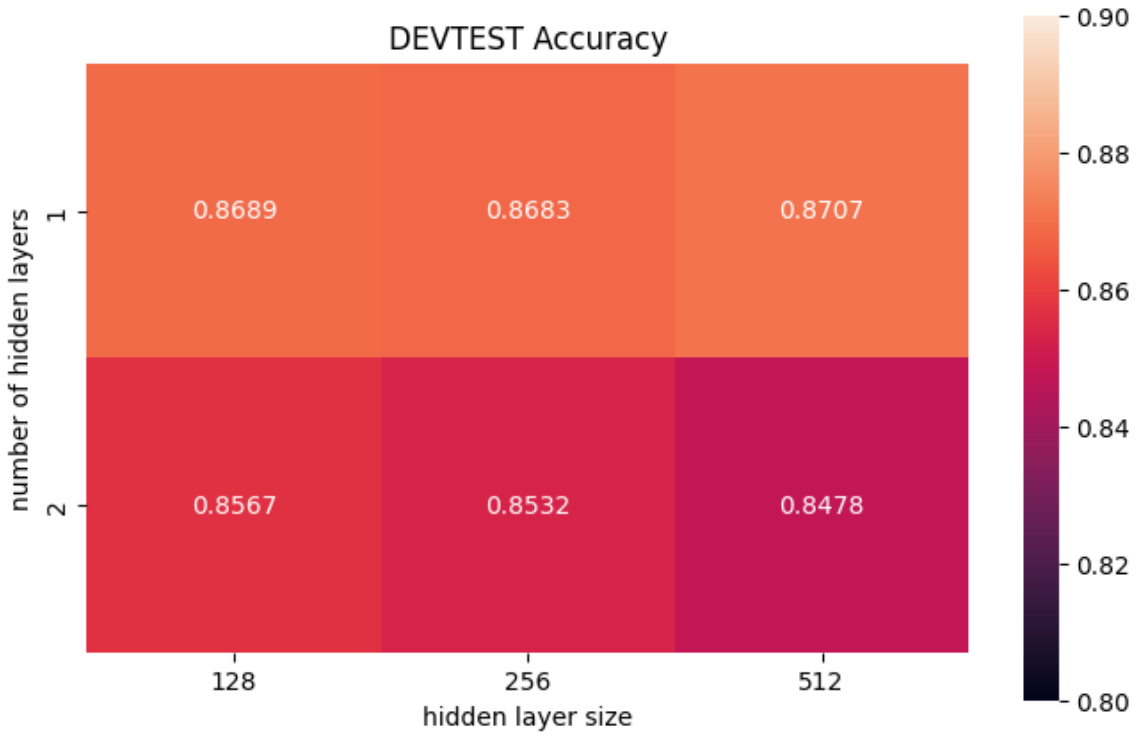
```
  accuracy: 0.9624635143023934
  accuracy: 0.8365484339348683
  best model from epoch 2
epoch 3
  epoch loss: 2410.43088908963
  accuracy: 0.9757734967892586
  accuracy: 0.8380004148516905
  best model from epoch 3
epoch 4
  epoch loss: 2318.0042265473544
  accuracy: 0.9647985989492119
  accuracy: 0.8321924911844015
  best model from epoch 3
epoch 5
  epoch loss: 2138.1631058182215
  accuracy: 0.9738470519556334
  accuracy: 0.8346816013275254
  best model from epoch 3
epoch 6
  epoch loss: 1964.7642538650161
  accuracy: 0.9654991243432575
  accuracy: 0.8392449699232525
  best model from epoch 6
epoch 7
  epoch loss: 1983.6526239437464
  accuracy: 0.9711617046117922
  accuracy: 0.8243103090645094
  best model from epoch 6
epoch 8
  epoch loss: 1817.647252127333
  accuracy: 0.9753648569760653
  accuracy: 0.8386226923874714
  best model from epoch 6
epoch 9
  epoch loss: 1697.2255046937057
  accuracy: 0.9785755983654407
  accuracy: 0.8411118025305954
  best model from epoch 9
epoch 10
  epoch loss: 1719.4924305945246
  accuracy: 0.9799182720373614
  accuracy: 0.8415266542211159
  best model from epoch 10
epoch 11
  epoch loss: 1644.3032142051181
  accuracy: 0.9719206071220081
  accuracy: 0.8367558597801286
  best model from epoch 10
epoch 12
  epoch loss: 1560.4351709833263
  accuracy: 0.9730297723292469
  accuracy: 0.8323999170296619
  best model from epoch 10
epoch 13
  epoch loss: 1595.0442613292123
  accuracy: 0.9778166958552248
  accuracy: 0.8377929890064302
  best model from epoch 10
========= EARLY STOPPING =========
load best model
```

```
eval best model on devtest
  accuracy: 0.8478120284544083
```

In [ ]:
```
fig, ax = plt.subplots(figsize=(8, 5))

sns.heatmap(devtest_evals_all,
            fmt='.4f',
            vmin=0.8,
            vmax=0.9,
            annot=True,
            square=True,
            xticklabels=['128', '256', '512'],
            yticklabels=['1', '2'],
            ax=ax)
ax.set_xlabel('hidden layer size')
ax.set_ylabel('number of hidden layers')
ax.set_title('DEVTEST Accuracy')
```

Out[ ]:
```
Text(0.5, 1.0, 'DEVTEST Accuracy')
```



As shown in the figure above, models with 1 hidden layer performed better than models with 2 hidden layers for all three layer size options.
For models with 1 hidden layer, model performance does not change much with the 3 different sizes of hidden layer, i.e., it stays at around 87%; although a layer size of 512 performed slightly better than layer sizes 128 and 256 in this iteration.
However, for models with 2 hidden layers, model performance decreased with the increase of hidden layer sizes.

nonlinearities

In this experiment, we will still use w=1, with one hidden layer of size 128, and fine-tuning pretrained embeddings.

In [ ]:

```
devtest_evals_nonlin_all = []

nonlin = {'identity': nn.Identity(), 'tanh': nn.Tanh(), 'ReLU': nn.ReL
U(), 'sigmoid': nn.Sigmoid()}


for nonlin_name, nonlin_func in nonlin.items():

    print(f'nonlinearity: {nonlin_name}')

    # instantiate model: single hidden layer 128, w=1, fine-tuned pret
rained embedding
    tagger_w1_tunedpretrained_exp = FeedForwardNN(w=1, vocab_size=len
(emb_pretrained_vocab), emb_dim=50, nfeatures=0,
                                        layer_sizes=[128, len(all_tags)],
# last layer is the output layer
                                        layer_acts=[nonlin_func, nn.Identi
ty()],   # nn.CrossEntropyLoss() already includes softmax transformati
on
                                        pretrained_emb=emb_pretrained, emb
_freeze=False)

    # instantiate optimizer
    sgd = optim.SGD(tagger_w1_tunedpretrained_exp.parameters(), lr=0.0
2)

    # train and eval
    epoch_losses, train_evals, dev_evals, devtest_eval = main_process(
                                                model=tagger_w1_tunedpre
trained_exp,
                                                name='tagger_w1_tunedpre
trained_'+nonlin_name,  # file name used for using checkpoint
                                                optimizer=sgd,
                                                criterion=nn.CrossEntrop
yLoss(),   # objective: log loss
                                                train_data=train_w1_30k,
                                                batch_size=1,
                                                shuffle=True,
                                                val_data=dev_w1_30k,
                                                test_data=devtest_w1_30
k,
                                                max_epochs=20,
                                                early_stopping=3   # whe
n dev eval doesn't improve for 3 consecutive epochs
                                               )
    devtest_evals_nonlin_all.append(devtest_eval)

    print(f'\n')
```

```
nonlinearity: identity
epoch 1
  epoch loss: 6145.1695303484885
  accuracy: 0.9746059544658494
  accuracy: 0.8450528935905414
  best model from epoch 1
epoch 2
  epoch loss: 1610.0183923154937
  accuracy: 0.9798598949211909
  accuracy: 0.8485791329599668
  best model from epoch 2
epoch 3
  epoch loss: 1442.9373112355054
  accuracy: 0.981903093987157
  accuracy: 0.840282099149554
  best model from epoch 2
epoch 4
  epoch loss: 1374.423037948443
  accuracy: 0.9800934033858727
  accuracy: 0.8344741754822651
  best model from epoch 2
epoch 5
  epoch loss: 1346.8888742349563
  accuracy: 0.9847635726795096
  accuracy: 0.8415266542211159
  best model from epoch 2
========= EARLY STOPPING =========
load best model
eval best model on devtest
  accuracy: 0.8486742832507006


nonlinearity: tanh
epoch 1
  epoch loss: 6033.632384385666
  accuracy: 0.973671920607122
  accuracy: 0.8458825969715826
  best model from epoch 1
epoch 2
  epoch loss: 1536.2270437387954
  accuracy: 0.9793345008756568
  accuracy: 0.8380004148516905
  best model from epoch 1
epoch 3
  epoch loss: 1357.537470244333
  accuracy: 0.9836544074722708
  accuracy: 0.844845467745281
  best model from epoch 1
epoch 4
  epoch loss: 1268.8026601945714
  accuracy: 0.9834792761237595
  accuracy: 0.8421489317568969
  best model from epoch 1
========= EARLY STOPPING =========
load best model
eval best model on devtest
  accuracy: 0.8538478120284544


nonlinearity: ReLU
epoch 1
```

```
  epoch loss: 6741.494326191054
  accuracy: 0.974430823117338
  accuracy: 0.8419415059116366
  best model from epoch 1
epoch 2
  epoch loss: 1575.1648056167203
  accuracy: 0.9809690601284297
  accuracy: 0.8452603194358017
  best model from epoch 2
epoch 3
  epoch loss: 1374.085061946933
  accuracy: 0.9845884413309982
  accuracy: 0.846090022816843
  best model from epoch 3
epoch 4
  epoch loss: 1204.8710040180742
  accuracy: 0.9863981319322825
  accuracy: 0.8465048745073637
  best model from epoch 4
epoch 5
  epoch loss: 1095.839107198723
  accuracy: 0.9859894921190894
  accuracy: 0.8475420037336652
  best model from epoch 5
epoch 6
  epoch loss: 1050.6857171055453
  accuracy: 0.9850554582603619
  accuracy: 0.8433934868284588
  best model from epoch 5
epoch 7
  epoch loss: 947.7163945235717
  accuracy: 0.9875072971395213
  accuracy: 0.8475420037336652
  best model from epoch 5
epoch 8
  epoch loss: 896.9190524930845
  accuracy: 0.9876824284880327
  accuracy: 0.8419415059116366
  best model from epoch 5
========= EARLY STOPPING =========
load best model
eval best model on devtest
  accuracy: 0.855356757921966


nonlinearity: sigmoid
epoch 1
  epoch loss: 17366.308353043416
  accuracy: 0.9366608289550497
  accuracy: 0.835096453018046
  best model from epoch 1
epoch 2
  epoch loss: 3035.6065769381325
  accuracy: 0.9716287215411559
  accuracy: 0.846090022816843
  best model from epoch 2
epoch 3
  epoch loss: 2004.5142706584897
  accuracy: 0.9780502043199066
  accuracy: 0.8438083385189795
  best model from epoch 2
```

```
epoch 4
  epoch loss: 1627.4104384902785
  accuracy: 0.9831873905429072
  accuracy: 0.8446380419000208
  best model from epoch 2
epoch 5
  epoch loss: 1412.255696289337
  accuracy: 0.9821949795680094
  accuracy: 0.8429786351379381
  best model from epoch 2
========= EARLY STOPPING =========
load best model
eval best model on devtest
  accuracy: 0.8570812675145506
```

The table below shows the comparison of tagging performance with 4 different nonlinearities (Identity, Tanh, ReLU, Sigmoid). They performed similarly on DEV set; but sigmoid performed slightly better on DEVTEST set, followed by ReLU and Tanh.

window size	hidden layer	hidden layer size	nonlinearity	best DEV accuracy	best epoch	DEVTEST accuracy
1	1	128	Identity	84.86%	2	84.87%
1	1	128	Tanh	84.59%	1	85.38%
1	1	128	ReLU	84.75%	5	85.54%
1	1	128	Sigmoid	84.61%	2	85.71%

1.5 RNN taggers

model architecture

In [ ]:
```
# references:
# - https://pytorch.org/docs/stable/generated/torch.nn.RNN.html
# - https://pytorch.org/docs/stable/generated/torch.nn.LSTM.html
# - https://pytorch.org/docs/stable/generated/torch.nn.GRU.html
# - https://blog.floydhub.com/a-beginners-guide-on-recurrent-neural-networks-with-pytorch/


class RNNTagger(nn.Module):


    def __init__(self, vocab_size, emb_dim, rnn_func, hidden_size, num_layers, bidirectional, out_dim, pretrained_emb=None, emb_freeze=False):

        # call parent constructor
        super(RNNTagger, self).__init__()

        # set initial embeddings
        if pretrained_emb is not None:
            self.emb = nn.Embedding.from_pretrained(pretrained_emb, fr
```

```
eeze=emb_freeze)
        else:   # randomly init embeddings
            self.emb = nn.Embedding(vocab_size, emb_dim)
            self.emb.weight.data.uniform_(-0.01, 0.01)

        # set embeddings' dimensionality
        self.emb_dim = self.emb.weight.shape[1]

        # RNN layer
        self.rnn = rnn_func(self.emb_dim, hidden_size=hidden_size, num
_layers=num_layers, batch_first=True, bidirectional=bidirectional)

        # set some params
        self.hidden_size = hidden_size
        self.num_layers = num_layers
        self.bidirectional = bidirectional

        # fully connected layer
        if bidirectional:
            self.fc = nn.Linear(2 * hidden_size, out_dim)
            self.fc.weight.data.uniform_(-0.01, 0.01)
            self.fc.bias.data.zero_()
        else:
            self.fc = nn.Linear(1 * hidden_size, out_dim)
            self.fc.weight.data.uniform_(-0.01, 0.01)
            self.fc.bias.data.zero_()


    def forward(self, x):

        # encode input word into emb
        x = self.emb(x)

        # init hidden state
        if self.bidirectional:
            hidden = torch.zeros(2 * self.num_layers, x.size(0), self.
hidden_size)
        else:
            hidden = torch.zeros(1 * self.num_layers, x.size(0), self.
hidden_size)

        # rnn
        out, hidden = self.rnn(x, hidden)

        # fc
        out = self.fc(out.contiguous().view(-1, out.size(2)))

        return out
```

Data

In [ ]:
```
# class POSSentenceDataset(Dataset):


#     def __init__(self, dataset:list, word2idx:dict, tag2idx:LabelEnc
oder()):


#         tweets_encoded = []
#         tags = []
#         tags_encoded = []


#         # encode context window and center word featuress
#         for tweet in dataset:

#             tweet_encoded = []
#             tags_curr = []
#             # tags_encoded_curr = []

#             # process every center word in each tweet
#             for i, (word, tag) in enumerate(tweet):

#                 # encode word
#                 try: tweet_encoded.append(word2idx[word])
#                 except: tweet_encoded.append(word2idx['UUUNKKK'])  #
use emb for unknown words

#                 # target of curr obs
#                 tags_curr.append(tag)

#             # encode all target tags
#             tags_encoded.append(tag2idx.transform(tags_curr))
#             tags.append(tags_curr)

#             # append tweet
#             tweets_encoded.append(tweet_encoded)


#         self.tweets_encoded = tweets_encoded
#         self.tags_encoded = tags_encoded
#         self.tags = tags


#     def __len__(self):
#         return len(self.tweets_encoded)


#     def __getitem__(self, idx):
#         return torch.tensor(self.tweets_encoded[idx]), torch.tensor
(self.tags_encoded[idx])
```

In [ ]:
```
# train_30k = POSSentenceDataset(dataset=twpos_train, word2idx=word2id
x_all_vocab, tag2idx=le)
# dev_30k = POSSentenceDataset(dataset=twpos_dev, word2idx=word2idx_al
l_vocab, tag2idx=le)
# devtest_30k = POSSentenceDataset(dataset=twpos_devtest, word2idx=wor
d2idx_all_vocab, tag2idx=le)
```

unidirectional

RNN

In [ ]:

```
# instantiate rnn model
tagger_rnn_uni_tunedpretrained = RNNTagger(
    vocab_size=len(emb_pretrained_vocab), emb_dim=50,
    rnn_func=nn.RNN, hidden_size=128, num_layers=1, bidirectional=Fals
e,
    out_dim=len(all_tags),
    pretrained_emb=emb_pretrained, emb_freeze=False
)

# instantiate optimizer
sgd = optim.SGD(tagger_rnn_uni_tunedpretrained.parameters(), lr=0.02)

# train and eval
epoch_losses, train_evals, dev_evals, devtest_eval = main_process(
                                                model=tagger_rnn_uni_tunedpr
etrained,
                                                name='tagger_rnn_uni_tunedpr
etrained',  # file name used for using checkpoint
                                                optimizer=sgd,
                                                criterion=nn.CrossEntropyLos
s(),   # objective: log loss
                                                train_data=train_w0_30k,
                                                batch_size=1,
                                                shuffle=True,
                                                val_data=dev_w0_30k,
                                                test_data=devtest_w0_30k,
                                                max_epochs=20,
                                                early_stopping=3   # when de
v eval doesn't improve for 3 consecutive epochs
                                            )
```

```
epoch 1
  epoch loss: 10224.533007556245
  accuracy: 0.8796847635726796
  accuracy: 0.8359261563990873
  best model from epoch 1
epoch 2
  epoch loss: 6967.228591184685
  accuracy: 0.8835376532399299
  accuracy: 0.8315702136486206
  best model from epoch 1
epoch 3
  epoch loss: 6594.675907360186
  accuracy: 0.8894337419731465
  accuracy: 0.8334370462559635
  best model from epoch 1
epoch 4
  epoch loss: 6332.950126517113
  accuracy: 0.8890834792761237
  accuracy: 0.840904376685335
  best model from epoch 4
epoch 5
  epoch loss: 6220.6918522919
  accuracy: 0.8907180385288966
  accuracy: 0.8388301182327318
  best model from epoch 4
epoch 6
  epoch loss: 6115.446526459738
  accuracy: 0.8927028604786924
  accuracy: 0.8334370462559635
  best model from epoch 4
epoch 7
  epoch loss: 6033.3319419802865
  accuracy: 0.8847635726795097
  accuracy: 0.825969715826592
  best model from epoch 4
========= EARLY STOPPING =========
load best model
eval best model on devtest
  accuracy: 0.8450097003664583
```

GRU

In [ ]:

```
# instantiate rnn model
tagger_gru_uni_tunedpretrained = RNNTagger(
    vocab_size=len(emb_pretrained_vocab), emb_dim=50,
    rnn_func=nn.GRU, hidden_size=128, num_layers=1, bidirectional=Fals
e,
    out_dim=len(all_tags),
    pretrained_emb=emb_pretrained, emb_freeze=False
)

# instantiate optimizer
sgd = optim.SGD(tagger_gru_uni_tunedpretrained.parameters(), lr=0.02)

# train and eval
epoch_losses, train_evals, dev_evals, devtest_eval = main_process(
                                                model=tagger_gru_uni_tunedpr
etrained,
                                                name='tagger_gru_uni_tunedpr
etrained',  # file name used for using checkpoint
                                                optimizer=sgd,
                                                criterion=nn.CrossEntropyLos
s(),   # objective: log loss
                                                train_data=train_w0_30k,
                                                batch_size=1,
                                                shuffle=True,
                                                val_data=dev_w0_30k,
                                                test_data=devtest_w0_30k,
                                                max_epochs=20,
                                                early_stopping=3   # when de
v eval doesn't improve for 3 consecutive epochs
                                               )
```

```
epoch 1
  epoch loss: 11735.981326428242
  accuracy: 0.8892586106246352
  accuracy: 0.8290811035054968
  best model from epoch 1
epoch 2
  epoch loss: 6254.4584421229665
  accuracy: 0.8944541739638062
  accuracy: 0.8406969508400747
  best model from epoch 2
epoch 3
  epoch loss: 5972.008260978968
  accuracy: 0.889492119089317
  accuracy: 0.8369632856253889
  best model from epoch 2
epoch 4
  epoch loss: 5797.793786261231
  accuracy: 0.8941622883829539
  accuracy: 0.8373781373159096
  best model from epoch 2
epoch 5
  epoch loss: 5694.19262394488
  accuracy: 0.892527729130181
  accuracy: 0.8348890271727857
  best model from epoch 2
========= EARLY STOPPING =========
load best model
eval best model on devtest
  accuracy: 0.8404828626859236
```

bidirectional

RNN

In [ ]:

```
# instantiate rnn model
tagger_rnn_bi_tunedpretrained = RNNTagger(
    vocab_size=len(emb_pretrained_vocab), emb_dim=50,
    rnn_func=nn.RNN, hidden_size=128, num_layers=1, bidirectional=Tru
e,
    out_dim=len(all_tags),
    pretrained_emb=emb_pretrained, emb_freeze=False
)

# instantiate optimizer
sgd = optim.SGD(tagger_rnn_bi_tunedpretrained.parameters(), lr=0.02)

# train and eval
epoch_losses, train_evals, dev_evals, devtest_eval = main_process(
                                                model=tagger_rnn_bi_tunedpre
trained,
                                                name='tagger_rnn_bi_tunedpre
trained',  # file name used for using checkpoint
                                                optimizer=sgd,
                                                criterion=nn.CrossEntropyLos
s(),   # objective: log loss
                                                train_data=train_w0_30k,
                                                batch_size=1,
                                                shuffle=True,
                                                val_data=dev_w0_30k,
                                                test_data=devtest_w0_30k,
                                                max_epochs=20,
                                                early_stopping=3   # when de
v eval doesn't improve for 3 consecutive epochs
                                            )
```

```
epoch 1
  epoch loss: 8302.125266798794
  accuracy: 0.8873321657910099
  accuracy: 0.8336444721012238
  best model from epoch 1
epoch 2
  epoch loss: 6257.6768378699635
  accuracy: 0.886456509048453
  accuracy: 0.8261771416718523
  best model from epoch 1
epoch 3
  epoch loss: 6039.686782999903
  accuracy: 0.8952714535901927
  accuracy: 0.8367558597801286
  best model from epoch 3
epoch 4
  epoch loss: 5970.646520831939
  accuracy: 0.8886164623467601
  accuracy: 0.8311553619581
  best model from epoch 3
epoch 5
  epoch loss: 5952.388773485949
  accuracy: 0.8892002335084647
  accuracy: 0.830533084422319
  best model from epoch 3
epoch 6
  epoch loss: 5894.658673944905
  accuracy: 0.8789258610624635
  accuracy: 0.8199543663140427
  best model from epoch 3
========= EARLY STOPPING =========
load best model
eval best model on devtest
  accuracy: 0.8381116619961199
```

GRU

In [ ]:

```
# instantiate rnn model
tagger_gru_bi_tunedpretrained = RNNTagger(
    vocab_size=len(emb_pretrained_vocab), emb_dim=50,
    rnn_func=nn.GRU, hidden_size=128, num_layers=1, bidirectional=Tru
e,
    out_dim=len(all_tags),
    pretrained_emb=emb_pretrained, emb_freeze=False
)

# instantiate optimizer
sgd = optim.SGD(tagger_gru_bi_tunedpretrained.parameters(), lr=0.02)

# train and eval
epoch_losses, train_evals, dev_evals, devtest_eval = main_process(
                                                model=tagger_gru_bi_tunedpre
trained,
                                                name='tagger_gru_bi_tunedpre
trained',  # file name used for using checkpoint
                                                optimizer=sgd,
                                                criterion=nn.CrossEntropyLos
s(),   # objective: log loss
                                                train_data=train_w0_30k,
                                                batch_size=1,
                                                shuffle=True,
                                                val_data=dev_w0_30k,
                                                test_data=devtest_w0_30k,
                                                max_epochs=20,
                                                early_stopping=3   # when de
v eval doesn't improve for 3 consecutive epochs
                                            )
```

```
epoch 1
  epoch loss: 10267.473302467493
  accuracy: 0.8913018096906012
  accuracy: 0.8334370462559635
  best model from epoch 1
epoch 2
  epoch loss: 5931.813815677611
  accuracy: 0.8947460595446585
  accuracy: 0.8344741754822651
  best model from epoch 2
epoch 3
  epoch loss: 5769.115994589658
  accuracy: 0.8953298307063631
  accuracy: 0.8394523957685127
  best model from epoch 3
epoch 4
  epoch loss: 5712.821717942381
  accuracy: 0.8943957968476357
  accuracy: 0.8373781373159096
  best model from epoch 3
epoch 5
  epoch loss: 5635.7459171848095
  accuracy: 0.8903093987157035
  accuracy: 0.8353038788633064
  best model from epoch 3
epoch 6
  epoch loss: 5615.526090011721
  accuracy: 0.8925861062463515
  accuracy: 0.8377929890064302
  best model from epoch 3
========= EARLY STOPPING =========
load best model
eval best model on devtest
  accuracy: 0.844363009269239
```

As shown above, RNN taggers have lower tagging performance compared to feedforward neural network tagger with w=1 and fine-tuned pretrained embeddings.

In [ ]:

2. Language Modeling

Dataset D:
<s> I am </s>
<s> am I </s>
<s> am am </s>

Bigram probabilities:

	Model U	**Model S**
P(I | <s>)	1/3	2/5
P(am | <s>)	2/3	3/5
P(</s> | <s>)	0	
P(I | I)	0/2 = 0	1/4
P(am | I)	1/2	2/4 = 1/2
P(</s> | I)	1/2	2/4 = 1/2
P(<s> | I)	0	
P(I | am)	1/4	2/6 = 1/3
P(am | am)	1/4	2/6 = 1/3
P(</s> | am)	2/4 = 1/2	3/6 = 1/2
P(<s> | am)	0	

Key sentence in D with higher probability in Model S than Model U: <s> I am </s>
Model U: P_{ModelU}(<s> I am </s>) = P_{ModelU} (I | <s>) * P_{ModelU} (am | I) * P_{ModelU} (</s> | am) = 1/3 * 1/2 * 1/2 = 1/12
Model S: P_{ModelS}(<s> I am </s>) = P_{ModelS} (I | <s>) * P_{ModelS} (am | I) * PModelS (</s> | am) = 2/5 * 1/2 * 1/2 = 1/10
Therefore, P_{ModelU}(<s> I am </s>) < P_{ModelS}(<s> I am </s>).