Homework 3 Release

February 28, 2022

Before you turn this problem in, make sure everything runs as expected. First, **restart the kernel** (in the menubar, select Kernel→Restart) and then **run all cells** (in the menubar, select Cell→Run All).

Make sure you fill in any place that says `YOUR CODE HERE` or "YOUR ANSWER HERE".

Also, please write how much time it took you to finish the homework. This will not affect your grade in any way and is used for statistical purposes.

[1]:
```
TIME_SPENT = "00h00m"
```

0.1 Homework 3

0.1.1 Word Embeddings

Welcome to Homework 3!

The homework contains several tasks. You can find the amount of points that you get for the correct solution in the task header. Maximum amount of points for each homework is *four*.

The **grading** for each task is the following: - correct answer - **full points** - insufficient solution or solution resulting in the incorrect output - **half points** - no answer or completely wrong solution - **no points**

Even if you don't know how to solve the task, we encourage you to write down your thoughts and progress and try to address the issues that stop you from completing the task.

When working on the written tasks, try to make your answers short and accurate. Most of the times, it is possible to answer the question in 1-3 sentences.

When writing code, make it readable. Choose appropriate names for your variables (`a = 'cat'` - not good, `word = 'cat'` - good). Avoid constructing lines of code longer than 100 characters (79 characters is ideal). If needed, provide the commentaries for your code, however, a good code should be easily readable without them :)

Finally, all your answers should be written only by yourself. If you copy them from other sources it will be considered as an academic fraud. You can discuss the tasks with your classmates but each solution must be individual.

Important!: before sending your solution, do the `Kernel -> Restart & Run All` to ensure that all your code works.

[2]:
```
from pprint import pprint

import fasttext
from datasets import load_dataset
from pathlib import Path
from sklearn.metrics import classification_report

from gensim.utils import simple_preprocess
from nltk.corpus import stopwords
```

You are going to work with the Large Movie Review Dataset again and use it to train a Fasttext classifier. To spare your time loading the dataset, we will do it using HuggingFace datasets library. We will also split train set into 20,000 samples for training and 5,000 samples for validation.

Working with `datasets` is easy. After loading and splitting the dataset, each split can be accessed by its index in the same order as defined while loading. For example, training set will be `imdb_dataset[0]`, validation set `imdb_dataset[1]`, test set `imdb_dataset[2]`.

Each individual sample can be also accessed with the corresponding index. For example, to take the first sample from the training set, we should write `imdb_dataset[0][0]`.

Each sample is a dictionary, that has `'text'` and `'label'`.

You can play around with the dataset to better see how it works.

[3]:
```
imdb_dataset = load_dataset('imdb', split=['train[:10000]+train[15000:]',␣
↪'train[10000:15000]', 'test'])
```

```
Reusing dataset imdb (/home/utlab/.cache/huggingface/datasets/imdb/plain_text/1.
0.0/2fdd8b9bcadd6e7055e742a706876ba43f19faee861df134affd7a3f60fc38a1)
```

```
  0%|          | 0/3 [00:00<?, ?it/s]
```

[4]:
```
imdb_dataset
```

[4]:
```
[Dataset({
    features: ['text', 'label'],
    num_rows: 20000
}),
Dataset({
    features: ['text', 'label'],
    num_rows: 5000
}),
Dataset({
    features: ['text', 'label'],
    num_rows: 25000
})]
```

0.2 Task 1. Preprocess the text (0.5 points)

To train a Fasttext classifier, the dataset should be written into text files, where each line corresponds to one example. Each line must also start with `__label__` prefix followed by the label, e.g. label `0` would be `__label__0` for Fasttext. This technical part was implemented for you. You can read more about it here.

You task is to preprocess the text. You can use any preprocessing tool that we've learned so far. The minimum preprocessing should do: - tokenize - lowercase - remove stopwords - remove punctuation

The preprocessed text should be represented by a string where each token is separated by the whitespace. For example: `someone actually gave movie stars high chance need immediate professional help`.

[5]:
```
def convert_to_fasttext(dataset, file_path):
    english_stop = stopwords.words('english')  # set of English stopwords
    file_path = Path(file_path)
    with open(file_path, 'w', encoding='utf-8') as f:  # open file for writing
        for item in dataset:
            label = item['label']  # label either 0 or 1
            text = item['text']  # movie review itself

            # transform text variable to contain preprocessed text
            # the label will be attached later

            ### YOUR CODE HERE
            from string import punctuation
            text = simple_preprocess(text)
            text = [word for word in text if word not in punctuation]
            text = [word for word in text if word not in english_stop]

            f.write(f"__label__{label} {text}\n")  # write into the file
```

[6]:
```
# Write train, validation and test sets
convert_to_fasttext(imdb_dataset[0], 'imdb.train')
convert_to_fasttext(imdb_dataset[1], 'imdb.valid')
convert_to_fasttext(imdb_dataset[2], 'imdb.test')
```

[7]:
```
# Read the validation and test files for prediction
valid_texts = [' '.join(line.split(' ')[1:]) for line in Path('imdb.valid').
 ↪read_text().split('\n')][:-1]
test_texts = [' '.join(line.split(' ')[1:]) for line in Path('imdb.test').
 ↪read_text().split('\n')][:-1]

# Validation and test labels for classification report
valid_trues = ['__label__' + str(item['label']) for item in imdb_dataset[1]]
test_trues = ['__label__' + str(item['label']) for item in imdb_dataset[2]]
```

0.3 Task 2.1 Train the classifier (1.5 points)

Read through the Fasttext classification tutorial. Try training different models with different parameters to achieve the highest score on the validation set. Then, see the results of the best model on the test set.

[8]:
```
# Put the parameters here
from sklearn.metrics import f1_score, precision_score, recall_score,␣
 ↪accuracy_score

lrs = [0.3, 0.5, 0.9, 1.0]
epochs = [10, 20, 30]
wordNgrams = [1, 2, 3]


def print_results(N, p, r):
    print("# of Sample:", N)
    print("Accuracy:", p)
    print("Recall:", r)


best_model = ''
best_acc = 0
tmp_acc = 0
for lr in lrs:
    for epoch in epochs:
        for ngram in wordNgrams:
            print(f'Parameter: [lr={lr}, epoch={epoch}, wordNgrams={ngram}]')
            model = fasttext.train_supervised(
                input='imdb.train',
                lr=lr,
                epoch=epoch,
                wordNgrams=ngram)
            result = model.test('imdb.valid')
            tmp_acc = result[1]
            if tmp_acc > best_acc:
                best_acc = tmp_acc
                best_model = f'[lr={lr}, epoch={epoch}, wordNgrams={ngram}]'

print(f'Best Accuracy: {best_acc} \n Best model:{best_model}')
print_results(*result)
```

```
Parameter: [lr=0.3, epoch=10, wordNgrams=1]

Read 2M words
Number of words:  74962
Number of labels: 2
Progress: 100.0% words/sec/thread: 1084055 lr:  0.000000 avg.loss:  0.139920
ETA:   0h 0m 0s
```

```
Parameter: [lr=0.3, epoch=10, wordNgrams=2]

Read 2M words
Number of words:  74962
Number of labels: 2
Progress:  99.4% words/sec/thread:  500329 lr:  0.006178 avg.loss:  0.098238
ETA:   0h 0m 0s 13.8% words/sec/thread:  547158 lr:  0.258696 avg.loss:
0.435947 ETA:   0h 0m 5s 53.5% words/sec/thread:  504017 lr:  0.139382 avg.loss:
0.168963 ETA:   0h 0m 3s 73.1% words/sec/thread:  502599 lr:  0.080812 avg.loss:
0.128290 ETA:   0h 0m 1s 88.5% words/sec/thread:  502123 lr:  0.034606 avg.loss:
0.107710 ETA:   0h 0m 0s words/sec/thread:  500520 lr:  0.001870 avg.loss:
0.096942 ETA:   0h 0m 0s

Parameter: [lr=0.3, epoch=10, wordNgrams=3]

Progress: 100.0% words/sec/thread:  496668 lr:  0.000000 avg.loss:  0.096360
ETA:   0h 0m 0s
Read 2M words
Number of words:  74962
Number of labels: 2
Progress: 100.0% words/sec/thread:  300535 lr:  0.000000 avg.loss:  0.115896
ETA:   0h 0m 0s 37.5% words/sec/thread:  304674 lr:  0.187548 avg.loss:
0.267986 ETA:   0h 0m 7s 46.6% words/sec/thread:  303323 lr:  0.160062 avg.loss:
0.223800 ETA:   0h 0m 6s 0.147563 avg.loss:  0.207887 ETA:   0h 0m 5s

Parameter: [lr=0.3, epoch=20, wordNgrams=1]

Read 2M words
Number of words:  74962
Number of labels: 2
Progress: 100.0% words/sec/thread:  794076 lr:  0.000000 avg.loss:  0.080060
ETA:   0h 0m 0s% words/sec/thread: 1092543 lr:  0.267916 avg.loss:  0.338101
ETA:   0h 0m 5s 16.5% words/sec/thread:  980958 lr:  0.250629 avg.loss:
0.280049 ETA:   0h 0m 6s 41.0% words/sec/thread:  792858 lr:  0.176861 avg.loss:
0.173083 ETA:   0h 0m 5s

Parameter: [lr=0.3, epoch=20, wordNgrams=2]

Read 2M words
Number of words:  74962
Number of labels: 2
Progress: 100.0% words/sec/thread:  449426 lr:  0.000000 avg.loss:  0.049969
ETA:   0h 0m 0sh 0m11s 75.7% words/sec/thread:  436378 lr:  0.072755 avg.loss:
0.063910 ETA:   0h 0m 3s

Parameter: [lr=0.3, epoch=20, wordNgrams=3]

Read 2M words
Number of words:  74962
Number of labels: 2
Progress: 100.0% words/sec/thread:  291835 lr:  0.000000 avg.loss:  0.057120
ETA:   0h 0m 0s 11.6% words/sec/thread:  249858 lr:  0.265346 avg.loss:
```

```
0.384922 ETA:   0h 0m25s 22.7% words/sec/thread:  274775 lr:  0.231913 avg.loss:
0.222304 ETA:   0h 0m20s 51.7% words/sec/thread:  290757 lr:  0.145029 avg.loss:
0.105222 ETA:   0h 0m11s 89.8% words/sec/thread:  290460 lr:  0.030644 avg.loss:
0.063004 ETA:   0h 0m 2s

Parameter: [lr=0.3, epoch=30, wordNgrams=1]

Read 2M words
Number of words:  74962
Number of labels: 2
Progress: 100.0% words/sec/thread:  723812 lr:  0.000000 avg.loss:  0.053011
ETA:   0h 0m 0s 27.6% words/sec/thread:  626967 lr:  0.217307 avg.loss:
0.171539 ETA:   0h 0m12s 54.2% words/sec/thread:  651547 lr:  0.137445 avg.loss:
0.095373 ETA:   0h 0m 7s

Parameter: [lr=0.3, epoch=30, wordNgrams=2]

Read 2M words
Number of words:  74962
Number of labels: 2
Progress: 100.0% words/sec/thread:  406101 lr:  0.000000 avg.loss:  0.032987
ETA:   0h 0m 0s 24.8% words/sec/thread:  379141 lr:  0.225689 avg.loss:
0.123449 ETA:   0h 0m21s 29.3% words/sec/thread:  378078 lr:  0.212144 avg.loss:
0.104854 ETA:   0h 0m20s words/sec/thread:  381372 lr:  0.206049 avg.loss:
0.098301 ETA:   0h 0m19s ETA:   0h 0m 7s

Parameter: [lr=0.3, epoch=30, wordNgrams=3]

Read 2M words
Number of words:  74962
Number of labels: 2
Progress: 100.0% words/sec/thread:  278481 lr:  0.000000 avg.loss:  0.038498
ETA:   0h 0m 0s 16.4% words/sec/thread:  283984 lr:  0.250770 avg.loss:
0.205928 ETA:   0h 0m31s 18.4% words/sec/thread:  282717 lr:  0.244661 avg.loss:
0.187322 ETA:   0h 0m31s 27.1% words/sec/thread:  284784 lr:  0.218770 avg.loss:
0.130498 ETA:   0h 0m27s 29.2% words/sec/thread:  284884 lr:  0.212347 avg.loss:
0.121501 ETA:   0h 0m26s 32.9% words/sec/thread:  284230 lr:  0.201415 avg.loss:
0.108997 ETA:   0h 0m25s 41.6% words/sec/thread:  280902 lr:  0.175066 avg.loss:
0.087222 ETA:   0h 0m22s 49.9% words/sec/thread:  280182 lr:  0.150310 avg.loss:
0.073276 ETA:   0h 0m19s 81.9% words/sec/thread:  277754 lr:  0.054383 avg.loss:
0.045985 ETA:   0h 0m 7s 90.3% words/sec/thread:  278245 lr:  0.029045 avg.loss:
0.042174 ETA:   0h 0m 3s% words/sec/thread:  278484 lr: -0.000002 avg.loss:
0.038498 ETA:   0h 0m 0s

Parameter: [lr=0.5, epoch=10, wordNgrams=1]

Read 2M words
Number of words:  74962
Number of labels: 2
Progress:  99.0% words/sec/thread:  698078 lr:  0.004781 avg.loss:  0.138131
ETA:   0h 0m 0s 17.4% words/sec/thread:  981773 lr:  0.413013 avg.loss:
0.335341 ETA:   0h 0m 3s
```

```
Parameter: [lr=0.5, epoch=10, wordNgrams=2]

Progress: 100.0% words/sec/thread:  691214 lr:  0.000000 avg.loss:  0.136907
ETA:   0h 0m 0s
Read 2M words
Number of words:  74962
Number of labels: 2
Progress: 100.0% words/sec/thread:  387600 lr:  0.000072 avg.loss:  0.073746
ETA:   0h 0m 0s  3.8% words/sec/thread:  451036 lr:  0.481022 avg.loss:
0.483605 ETA:   0h 0m 7s 95.5% words/sec/thread:  386919 lr:  0.022579 avg.loss:
0.076913 ETA:   0h 0m 0s

Parameter: [lr=0.5, epoch=10, wordNgrams=3]

Progress: 100.0% words/sec/thread:  383479 lr:  0.000000 avg.loss:  0.073735
ETA:   0h 0m 0s
Read 2M words
Number of words:  74962
Number of labels: 2
Progress: 100.0% words/sec/thread:  276995 lr:  0.000000 avg.loss:  0.079092
ETA:   0h 0m 0s 53.7% words/sec/thread:  277776 lr:  0.231700 avg.loss:
0.141576 ETA:   0h 0m 5s

Parameter: [lr=0.5, epoch=20, wordNgrams=1]

Read 2M words
Number of words:  74962
Number of labels: 2
Progress: 100.0% words/sec/thread:  714355 lr:  0.000000 avg.loss:  0.074232
ETA:   0h 0m 0s 12.4% words/sec/thread:  737512 lr:  0.438066 avg.loss:
0.296757 ETA:   0h 0m 8s  0h 0m 1s 91.1% words/sec/thread:  722976 lr:  0.044552
avg.loss:  0.081128 ETA:   0h 0m 0s

Parameter: [lr=0.5, epoch=20, wordNgrams=2]

Read 2M words
Number of words:  74962
Number of labels: 2
Progress:  99.3% words/sec/thread:  361655 lr:  0.003472 avg.loss:  0.039219
ETA:   0h 0m 0s 13.9% words/sec/thread:  341862 lr:  0.430652 avg.loss:
0.244428 ETA:   0h 0m18s 360709 lr:  0.366156 avg.loss:  0.135396 ETA:   0h
0m14s 41.7% words/sec/thread:  350509 lr:  0.291440 avg.loss:  0.088385 ETA:
0h 0m11s 42.9% words/sec/thread:  340723 lr:  0.285324 avg.loss:  0.085862 ETA:
0h 0m12s 0.073095 ETA:   0h 0m10s 68.8% words/sec/thread:  358683 lr:  0.155951
avg.loss:  0.055202 ETA:   0h 0m 6s

Parameter: [lr=0.5, epoch=20, wordNgrams=3]

Progress: 100.0% words/sec/thread:  362345 lr:  0.000000 avg.loss:  0.038970
ETA:   0h 0m 0s
Read 2M words
Number of words:  74962
```

```
Number of labels: 2
Progress: 100.0% words/sec/thread:  254260 lr:  0.000000 avg.loss:  0.040223
ETA:   0h 0m 0s 22.3% words/sec/thread:  241172 lr:  0.388466 avg.loss:
0.165998 ETA:   0h 0m23s 52.3% words/sec/thread:  255761 lr:  0.238623 avg.loss:
0.073786 ETA:   0h 0m13s 54.0% words/sec/thread:  255343 lr:  0.230116 avg.loss:
0.071702 ETA:   0h 0m12s 55.7% words/sec/thread:  255083 lr:  0.221478 avg.loss:
0.069835 ETA:   0h 0m12s  0h 0m12s 70.2% words/sec/thread:  253186 lr:  0.149188
avg.loss:  0.056245 ETA:   0h 0m 8s 94.4% words/sec/thread:  257458 lr:
0.028042 avg.loss:  0.042339 ETA:   0h 0m 1s

Parameter: [lr=0.5, epoch=30, wordNgrams=1]

Read 2M words
Number of words:  74962
Number of labels: 2
Progress: 100.0% words/sec/thread:  729067 lr: -0.000010 avg.loss:  0.055008
ETA:   0h 0m 0s 19.4% words/sec/thread:  717083 lr:  0.402901 avg.loss:
0.209459 ETA:   0h 0m12s lr:  0.370208 avg.loss:  0.179378 ETA:   0h 0m11s 65.7%
words/sec/thread:  725680 lr:  0.171538 avg.loss:  0.082089 ETA:   0h 0m 5s%
words/sec/thread:  732621 lr:  0.079587 avg.loss:  0.064938 ETA:   0h 0m 2s

Parameter: [lr=0.5, epoch=30, wordNgrams=2]

Progress: 100.0% words/sec/thread:  729051 lr:  0.000000 avg.loss:  0.055008
ETA:   0h 0m 0s
Read 2M words
Number of words:  74962
Number of labels: 2
Progress: 100.0% words/sec/thread:  383713 lr:  0.000000 avg.loss:  0.026334
ETA:   0h 0m 0s 14.0% words/sec/thread:  297821 lr:  0.430236 avg.loss:
0.168391 ETA:   0h 0m31s 309088 lr:  0.416073 avg.loss:  0.141687 ETA:   0h
0m28s 22.9% words/sec/thread:  331387 lr:  0.385258 avg.loss:  0.105476 ETA:
0h 0m25s 350178 lr:  0.337948 avg.loss:  0.075182 ETA:   0h 0m20s 65.0%
words/sec/thread:  378244 lr:  0.174938 avg.loss:  0.039533 ETA:   0h 0m 9s
71.0% words/sec/thread:  379934 lr:  0.145121 avg.loss:  0.036406 ETA:   0h 0m
8s 86.3% words/sec/thread:  384901 lr:  0.068534 avg.loss:  0.030161 ETA:   0h
0m 3s 87.1% words/sec/thread:  385129 lr:  0.064692 avg.loss:  0.029911 ETA:
0h 0m 3s

Parameter: [lr=0.5, epoch=30, wordNgrams=3]

Read 2M words
Number of words:  74962
Number of labels: 2
Progress: 100.0% words/sec/thread:  252175 lr: -0.000009 avg.loss:  0.027226
ETA:   0h 0m 0s avg.loss:  0.570133 ETA:   0h 0m45s  4.5% words/sec/thread:
243187 lr:  0.477309 avg.loss:  0.427098 ETA:   0h 0m42s 12.6% words/sec/thread:
255258 lr:  0.436899 avg.loss:  0.189652 ETA:   0h 0m36s 26.6% words/sec/thread:
259501 lr:  0.366904 avg.loss:  0.093704 ETA:   0h 0m30s 31.9% words/sec/thread:
255252 lr:  0.340478 avg.loss:  0.078765 ETA:   0h 0m28s 45.2% words/sec/thread:
257554 lr:  0.274160 avg.loss:  0.057151 ETA:   0h 0m22s 0.268425 avg.loss:
```

```
0.055949 ETA:   0h 0m22s 57.1% words/sec/thread:  256028 lr:  0.214639 avg.loss:
0.045763 ETA:   0h 0m18s 64.2% words/sec/thread:  257560 lr:  0.179230 avg.loss:
0.041117 ETA:   0h 0m14s 84.0% words/sec/thread:  254333 lr:  0.079986 avg.loss:
0.031942 ETA:   0h 0m 6s 85.0% words/sec/thread:  253778 lr:  0.074988 avg.loss:
0.031596 ETA:   0h 0m 6s 91.4% words/sec/thread:  251847 lr:  0.042955 avg.loss:
0.029617 ETA:   0h 0m 3s 95.4% words/sec/thread:  252384 lr:  0.023134 avg.loss:
0.028481 ETA:   0h 0m 1s

Parameter: [lr=0.9, epoch=10, wordNgrams=1]

Progress: 100.0% words/sec/thread:  252174 lr:  0.000000 avg.loss:  0.027226
ETA:   0h 0m 0s
Read 2M words
Number of words:  74962
Number of labels: 2
Progress: 100.0% words/sec/thread:  722250 lr:  0.000000 avg.loss:  0.132948
ETA:   0h 0m 0s

Parameter: [lr=0.9, epoch=10, wordNgrams=2]

Read 2M words
Number of words:  74962
Number of labels: 2
Progress: 100.0% words/sec/thread:  387861 lr:  0.000000 avg.loss:  0.066941
ETA:   0h 0m 0s 11.0% words/sec/thread:  326888 lr:  0.800640 avg.loss:
0.403033 ETA:   0h 0m 9s words/sec/thread:  330762 lr:  0.682115 avg.loss:
0.244917 ETA:   0h 0m 8s100.0% words/sec/thread:  387882 lr: -0.000036 avg.loss:
0.066941 ETA:   0h 0m 0s

Parameter: [lr=0.9, epoch=10, wordNgrams=3]

Read 2M words
Number of words:  74962
Number of labels: 2
Progress: 100.0% words/sec/thread:  270771 lr:  0.000000 avg.loss:  0.061856
ETA:   0h 0m 0s 57.7% words/sec/thread:  278937 lr:  0.380457 avg.loss:
0.104170 ETA:   0h 0m 5s 85.9% words/sec/thread:  274281 lr:  0.126616 avg.loss:
0.071493 ETA:   0h 0m 1s

Parameter: [lr=0.9, epoch=20, wordNgrams=1]

Read 2M words
Number of words:  74962
Number of labels: 2
Progress: 100.0% words/sec/thread:  728816 lr:  0.000000 avg.loss:  0.080713
ETA:   0h 0m 0s  8.3% words/sec/thread:  989215 lr:  0.825294 avg.loss:
0.334096 ETA:   0h 0m 6s 33.8% words/sec/thread:  731164 lr:  0.595795 avg.loss:
0.196335 ETA:   0h 0m 6s 46.9% words/sec/thread:  682476 lr:  0.478044 avg.loss:
0.161303 ETA:   0h 0m 5s 94.1% words/sec/thread:  730328 lr:  0.053264 avg.loss:
0.085528 ETA:   0h 0m 0s

Parameter: [lr=0.9, epoch=20, wordNgrams=2]
```

```
Read 2M words
Number of words:  74962
Number of labels: 2
Progress: 100.0% words/sec/thread:  410011 lr:  0.000000 avg.loss:  0.034481
ETA:   0h 0m 0s 17.8% words/sec/thread:  350060 lr:  0.739845 avg.loss:
0.179697 ETA:   0h 0m16s 42.7% words/sec/thread:  380065 lr:  0.515542 avg.loss:
0.077810 ETA:   0h 0m10s 45.8% words/sec/thread:  383547 lr:  0.487893 avg.loss:
0.072642 ETA:   0h 0m10s 62.5% words/sec/thread:  405154 lr:  0.337186 avg.loss:
0.054182 ETA:   0h 0m 6s% words/sec/thread:  413982 lr:  0.189474 avg.loss:
0.043114 ETA:   0h 0m 3s 85.3% words/sec/thread:  413781 lr:  0.132509 avg.loss:
0.040052 ETA:   0h 0m 2s

Parameter: [lr=0.9, epoch=20, wordNgrams=3]

Read 2M words
Number of words:  74962
Number of labels: 2
Progress: 100.0% words/sec/thread:  276944 lr:  0.000000 avg.loss:  0.032264
ETA:   0h 0m 0s  8.9% words/sec/thread:  289243 lr:  0.819730 avg.loss:
0.303548 ETA:   0h 0m22s 12.6% words/sec/thread:  271524 lr:  0.786911 avg.loss:
0.228083 ETA:   0h 0m23s 63.2% words/sec/thread:  273799 lr:  0.330765 avg.loss:
0.049617 ETA:   0h 0m 9s 75.0% words/sec/thread:  276179 lr:  0.225037 avg.loss:
0.042146 ETA:   0h 0m 6s

Parameter: [lr=0.9, epoch=30, wordNgrams=1]

Read 2M words
Number of words:  74962
Number of labels: 2
Progress: 100.0% words/sec/thread:  724724 lr:  0.000000 avg.loss:  0.056307
ETA:   0h 0m 0s 0.110228 ETA:   0h 0m 7s 70.8% words/sec/thread:  672025 lr:
0.262775 avg.loss:  0.078238 ETA:   0h 0m 4s

Parameter: [lr=0.9, epoch=30, wordNgrams=2]

Read 2M words
Number of words:  74962
Number of labels: 2
Progress: 100.0% words/sec/thread:  360395 lr:  0.000000 avg.loss:  0.023568
ETA:   0h 0m 0s 30.9% words/sec/thread:  340327 lr:  0.622249 avg.loss:
0.072277 ETA:   0h 0m21s 32.9% words/sec/thread:  334953 lr:  0.604130 avg.loss:
0.068100 ETA:   0h 0m21s 34.6% words/sec/thread:  336253 lr:  0.588877 avg.loss:
0.064775 ETA:   0h 0m20s 328919 lr:  0.534508 avg.loss:  0.055727 ETA:   0h
0m19s

Parameter: [lr=0.9, epoch=30, wordNgrams=3]

Read 2M words
Number of words:  74962
Number of labels: 2
Progress: 100.0% words/sec/thread:  258988 lr: -0.000020 avg.loss:  0.022039
ETA:   0h 0m 0s  8.8% words/sec/thread:  286499 lr:  0.820669 avg.loss:
```

```
0.217570 ETA:   0h 0m34s 17.2% words/sec/thread:  288760 lr:  0.744873 avg.loss:
0.116242 ETA:   0h 0m30s 41.2% words/sec/thread:  270827 lr:  0.529362 avg.loss:
0.050491 ETA:   0h 0m23s  0h 0m18s 55.7% words/sec/thread:  272826 lr:  0.398280
avg.loss:  0.037929 ETA:   0h 0m17s 62.6% words/sec/thread:  268596 lr:
0.336189 avg.loss:  0.033938 ETA:   0h 0m14s 70.9% words/sec/thread:  267465 lr:
0.262195 avg.loss:  0.030353 ETA:   0h 0m11s 99.1% words/sec/thread:  262379 lr:
0.008197 avg.loss:  0.022222 ETA:   0h 0m 0s 99.7% words/sec/thread:  260780 lr:
0.002673 avg.loss:  0.022093 ETA:   0h 0m 0s

Parameter: [lr=1.0, epoch=10, wordNgrams=1]

Progress: 100.0% words/sec/thread:  258986 lr:  0.000000 avg.loss:  0.022036
ETA:   0h 0m 0s
Read 2M words
Number of words:  74962
Number of labels: 2
Progress: 100.0% words/sec/thread:  850948 lr:  0.000000 avg.loss:  0.126245
ETA:   0h 0m 0s 23.4% words/sec/thread: 1046543 lr:  0.765946 avg.loss:
0.314872 ETA:   0h 0m 2s 36.6% words/sec/thread: 1007049 lr:  0.634014 avg.loss:
0.251299 ETA:   0h 0m 2s

Parameter: [lr=1.0, epoch=10, wordNgrams=2]

Read 2M words
Number of words:  74962
Number of labels: 2
Progress: 100.0% words/sec/thread:  429833 lr:  0.000000 avg.loss:  0.067118
ETA:   0h 0m 0s  8.1% words/sec/thread:  361169 lr:  0.919092 avg.loss:
0.441007 ETA:   0h 0m 9s 52.1% words/sec/thread:  394820 lr:  0.479158 avg.loss:
0.124993 ETA:   0h 0m 4s 73.7% words/sec/thread:  410906 lr:  0.262600 avg.loss:
0.089892 ETA:   0h 0m 2s

Parameter: [lr=1.0, epoch=10, wordNgrams=3]

Read 2M words
Number of words:  74962
Number of labels: 2
Progress: 100.0% words/sec/thread:  300038 lr:  0.000000 avg.loss:  0.059430
ETA:   0h 0m 0s 46.7% words/sec/thread:  315187 lr:  0.532876 avg.loss:
0.121539 ETA:   0h 0m 6s 77.2% words/sec/thread:  313882 lr:  0.227625 avg.loss:
0.076442 ETA:   0h 0m 2s 78.5% words/sec/thread:  312027 lr:  0.214629 avg.loss:
0.075122 ETA:   0h 0m 2s 84.0% words/sec/thread:  306016 lr:  0.159935 avg.loss:
0.070175 ETA:   0h 0m 1s100.0% words/sec/thread:  300041 lr: -0.000064 avg.loss:
0.059430 ETA:   0h 0m 0s

Parameter: [lr=1.0, epoch=20, wordNgrams=1]

Read 2M words
Number of words:  74962
Number of labels: 2
Progress: 100.0% words/sec/thread:  635658 lr:  0.000000 avg.loss:  0.079162
ETA:   0h 0m 0s 10.9% words/sec/thread:  866597 lr:  0.890934 avg.loss:
```

```
0.315219 ETA:   0h 0m 7s 0.365501 avg.loss:  0.123362 ETA:   0h 0m 3s 82.4%
words/sec/thread:  637423 lr:  0.176226 avg.loss:  0.095797 ETA:   0h 0m 1s
87.7% words/sec/thread:  643626 lr:  0.123212 avg.loss:  0.090079 ETA:   0h 0m
1s100.0% words/sec/thread:  635665 lr: -0.000023 avg.loss:  0.079162 ETA:   0h
0m 0s

Parameter: [lr=1.0, epoch=20, wordNgrams=2]

Read 2M words
Number of words:  74962
Number of labels: 2
Progress: 100.0% words/sec/thread:  348595 lr:  0.000000 avg.loss:  0.035289
ETA:   0h 0m 0s 30.8% words/sec/thread:  319645 lr:  0.692199 avg.loss:
0.109609 ETA:   0h 0m15s 38.6% words/sec/thread:  321504 lr:  0.613673 avg.loss:
0.087697 ETA:   0h 0m13s 54.9% words/sec/thread:  338346 lr:  0.451435 avg.loss:
0.062462 ETA:   0h 0m 9s

Parameter: [lr=1.0, epoch=20, wordNgrams=3]

Read 2M words
Number of words:  74962
Number of labels: 2
Progress:  99.7% words/sec/thread:  251397 lr:  0.002680 avg.loss:  0.031384
ETA:   0h 0m 0s  2.3% words/sec/thread:  266445 lr:  0.977465 avg.loss:
0.436884 ETA:   0h 0m26s 35.4% words/sec/thread:  237696 lr:  0.645864 avg.loss:
0.083949 ETA:   0h 0m19s 58.3% words/sec/thread:  238645 lr:  0.417089 avg.loss:
0.052100 ETA:   0h 0m12s 64.0% words/sec/thread:  242726 lr:  0.359623 avg.loss:
0.047701 ETA:   0h 0m10s 66.0% words/sec/thread:  243575 lr:  0.340291 avg.loss:
0.046350 ETA:   0h 0m10s 89.6% words/sec/thread:  248547 lr:  0.104360 avg.loss:
0.034589 ETA:   0h 0m 3s 96.3% words/sec/thread:  250591 lr:  0.037424 avg.loss:
0.032380 ETA:   0h 0m 1s

Parameter: [lr=1.0, epoch=30, wordNgrams=1]

Progress: 100.0% words/sec/thread:  251190 lr:  0.000000 avg.loss:  0.031317
ETA:   0h 0m 0s
Read 2M words
Number of words:  74962
Number of labels: 2
Progress: 100.0% words/sec/thread:  739501 lr:  0.000000 avg.loss:  0.056414
ETA:   0h 0m 0s 13.7% words/sec/thread:  863652 lr:  0.863215 avg.loss:
0.242129 ETA:   0h 0m10s 56.0% words/sec/thread:  723432 lr:  0.439892 avg.loss:
0.100055 ETA:   0h 0m 6s 59.0% words/sec/thread:  718382 lr:  0.410328 avg.loss:
0.095234 ETA:   0h 0m 6s

Parameter: [lr=1.0, epoch=30, wordNgrams=2]

Read 2M words
Number of words:  74962
Number of labels: 2
Progress: 100.0% words/sec/thread:  364454 lr:  0.000000 avg.loss:  0.024734
ETA:   0h 0m 0s 275328 lr:  0.992270 avg.loss:  0.453194 ETA:   0h 0m38s  9.1%
```

```
words/sec/thread:  250850 lr:  0.908505 avg.loss:  0.236472 ETA:   0h 0m38s
30.0% words/sec/thread:  317722 lr:  0.700229 avg.loss:  0.078752 ETA:   0h
0m23s 42.0% words/sec/thread:  328372 lr:  0.579853 avg.loss:  0.057025 ETA:
0h 0m18s 53.7% words/sec/thread:  342100 lr:  0.463450 avg.loss:  0.044768 ETA:
0h 0m14s 67.7% words/sec/thread:  351956 lr:  0.323190 avg.loss:  0.035691 ETA:
0h 0m 9s 78.1% words/sec/thread:  360567 lr:  0.219221 avg.loss:  0.031247 ETA:
0h 0m 6s 95.9% words/sec/thread:  365755 lr:  0.040859 avg.loss:  0.025738 ETA:
0h 0m 1s

Parameter: [lr=1.0, epoch=30, wordNgrams=3]

Read 2M words
Number of words:  74962
Number of labels: 2
Progress:  99.8% words/sec/thread:  237961 lr:  0.001756 avg.loss:  0.021511
ETA:   0h 0m 0s  0.8% words/sec/thread:  297627 lr:  0.991671 avg.loss:
0.536313 ETA:   0h 0m35s 14.5% words/sec/thread:  255202 lr:  0.854646 avg.loss:
0.134250 ETA:   0h 0m36s 18.1% words/sec/thread:  248999 lr:  0.818683 avg.loss:
0.108903 ETA:   0h 0m35s 22.8% words/sec/thread:  251705 lr:  0.771941 avg.loss:
0.087133 ETA:   0h 0m32s 45.7% words/sec/thread:  245643 lr:  0.543409 avg.loss:
0.044683 ETA:   0h 0m23s 81.0% words/sec/thread:  241550 lr:  0.189582 avg.loss:
0.026039 ETA:   0h 0m 8s 84.4% words/sec/thread:  238792 lr:  0.156487 avg.loss:
0.025122 ETA:   0h 0m 7s 85.5% words/sec/thread:  238829 lr:  0.145227 avg.loss:
0.024836 ETA:   0h 0m 6s 89.2% words/sec/thread:  237538 lr:  0.107658 avg.loss:
0.023895 ETA:   0h 0m 4s 94.1% words/sec/thread:  238689 lr:  0.058847 avg.loss:
0.022752 ETA:   0h 0m 2s

Best Accuracy: 0.8764
 Best model:[lr=0.3, epoch=20, wordNgrams=3]
# of Sample: 5000
Accuracy: 0.874
Recall: 0.874

Progress: 100.0% words/sec/thread:  237852 lr:  0.000000 avg.loss:  0.021480
ETA:   0h 0m 0s
```

[9]:
```
# Precision at one and recall at one scores
model.test('imdb.valid')
```

[9]:
```
(5000, 0.874, 0.874)
```

[10]:
```
# Sklearn classification report for validation
valid_preds = model.predict(valid_texts)[0]
print(classification_report(valid_trues, valid_preds, digits=4))
```

```
              precision    recall  f1-score   support

  __label__0     0.8740    0.8740    0.8740      2500
  __label__1     0.8740    0.8740    0.8740      2500
```

```
    accuracy                           0.8740      5000
   macro avg     0.8740    0.8740    0.8740      5000
weighted avg     0.8740    0.8740    0.8740      5000
```

[11]:
```
# Sklearn classification report for test
test_preds = model.predict(test_texts)[0]
print(classification_report(test_trues, test_preds, digits=4))
```

```
              precision    recall  f1-score   support

  __label__0     0.8792    0.8757    0.8774     12500
  __label__1     0.8762    0.8797    0.8779     12500

    accuracy                         0.8777     25000
   macro avg     0.8777    0.8777    0.8777     25000
weighted avg     0.8777    0.8777    0.8777     25000
```

0.4 Task 2.2 Describe the results (0.5 point)

- Briefly describe which parameters you tried and which parameters you ended up using for the final model.
- Which parameter had the most impact on the model's performance?
- What are the scores for your best model?

(A) : For optmizing fastText model, I try to fineturn `learning rate [0.3, 0.5, 0.9, 1.0]`,`epoch [10, 20, 30]`, and `number of ngrams [1, 2, 3]`.In the end, the model with `[lr=0.3, epoch=20, wordNgrams=3]` has best performence. In the other hands, `Learning Rate` had most impact on the model. In genernal, `0.874` is the best score in my model.

0.5 Task 3.1 Automatic hyperparameter optimization (0.5 points)

Read about Automatic hyperparameter optimization here.

Autotune your model on the validation set (this will take around five minutes to finish).

Look at the results on the validation and test sets.

[13]:
```
# Put the parameters here
auto_model = fasttext.train_supervised(input='imdb.train', lr=0.3, epoch=20,␣
 ↪wordNgrams=3)
```

```
Read 2M words
Number of words:  74962
Number of labels: 2
Progress: 100.0% words/sec/thread:  244408 lr:  0.000000 avg.loss:  0.057624
ETA:   0h 0m 0s 55.6% words/sec/thread:  236237 lr:  0.133061 avg.loss:
0.099612 ETA:   0h 0m13s 0.082079 ETA:   0h 0m 9s 76.3% words/sec/thread:
240828 lr:  0.071183 avg.loss:  0.074168 ETA:   0h 0m 7s
```

[14]:
```
# Sklearn classification report for validation
auto_valid_preds = auto_model.predict(valid_texts)[0]
print(classification_report(valid_trues, auto_valid_preds, digits=4))
```

```
              precision    recall  f1-score   support

   __label__0     0.8784    0.8728    0.8756      2500
   __label__1     0.8736    0.8792    0.8764      2500

    accuracy                         0.8760      5000
   macro avg     0.8760    0.8760    0.8760      5000
weighted avg     0.8760    0.8760    0.8760      5000
```

[15]:
```
# Sklearn classification report for test
auto_test_preds = auto_model.predict(test_texts)[0]
print(classification_report(test_trues, auto_test_preds, digits=4))
```

```
              precision    recall  f1-score   support

   __label__0     0.8822    0.8762    0.8791     12500
   __label__1     0.8770    0.8830    0.8800     12500

    accuracy                         0.8796     25000
   macro avg     0.8796    0.8796    0.8796     25000
weighted avg     0.8796    0.8796    0.8796     25000
```

0.6 Task 3.2 Look at the model's parameters (0.5 points)

Check the parameters of the autotuned model.

- Briefly decribe how different they are from your previous model
- What was the most surprising change?
- Did autotuned model perform better then your manually tuned model?

[16]:
```
print('Learning rate:', auto_model.lr)
print('Word vector dimention:', auto_model.dim)
print('Number of epochs:', auto_model.epoch)
print('Word n-grams:', auto_model.wordNgrams)
print('Minimum length of char n-gram:', auto_model.minn)
print('Maximum length of char n-gram:', auto_model.maxn)
print('Size of context window:', auto_model.ws)
```

```
Learning rate: 0.3
Word vector dimention: 100
Number of epochs: 20
Word n-grams: 3
Minimum length of char n-gram: 0
```

```
Maximum length of char n-gram: 0
Size of context window: 5
```

(A) : The result of model is not that different to mine, however it spend less time to mine.

0.7 Task 4. Play with word vectors (0.5 points)

Since the Fasttext classifier trained the word embeddings from scratch based on our dataset, you can explore them as any other word embeddings.

Read this tutorial on word representations, and try to explore a little the word vectors, for example try to see nearest neighbors of some words or get some word analogies. You can also try different movie related words to see how they are compared to more general words.

Probably, the quality of these word embeddings will not be very good since our dataset is relatively small, but it is still good idea to check yourself what did the model learn in the end.

[19]:
```
### YOUR CODE HERE
pprint(model.get_nearest_neighbors('dog'))
```

```
[(0.0, "'one',"),
 (0.0, "'movie',"),
 (0.0, '</s>'),
 (0.0, "'even',"),
 (0.0, "'like',"),
 (0.0, "'would',"),
 (0.0, "'time',"),
 (0.0, "'burg',"),
 (0.0, "'herts',"),
 (0.0, "['tough',")]
```

[21]:
```
pprint(model.get_analogies('pet', 'dog', 'humen'))
```

```
[(0.0, "'one',"),
 (0.0, "'movie',"),
 (0.0, '</s>'),
 (0.0, "'even',"),
 (0.0, "'like',"),
 (0.0, "'would',"),
 (0.0, "'time',"),
 (0.0, "'burg',"),
 (0.0, "'herts',"),
 (0.0, "['tough',")]
```

[ ]: