hw2(1)

October 5, 2021

[1]:
```python
import pandas as pd
import numpy as np
import copy
import re
```

[2]:
```python
import pandas as pd
test = pd.read_csv("amazon_reviews_us_Kitchen_v1_00.tsv", sep =␣
 ↪'\t',error_bad_lines = False)
test['label'] = -1
```

```
b'Skipping line 16148: expected 15 fields, saw 22\nSkipping line 20100: expected
15 fields, saw 22\nSkipping line 45178: expected 15 fields, saw 22\nSkipping
line 48700: expected 15 fields, saw 22\nSkipping line 63331: expected 15 fields,
saw 22\n'
b'Skipping line 86053: expected 15 fields, saw 22\nSkipping line 88858: expected
15 fields, saw 22\nSkipping line 115017: expected 15 fields, saw 22\n'
b'Skipping line 137366: expected 15 fields, saw 22\nSkipping line 139110:
expected 15 fields, saw 22\nSkipping line 165540: expected 15 fields, saw
22\nSkipping line 171813: expected 15 fields, saw 22\n'
b'Skipping line 203723: expected 15 fields, saw 22\nSkipping line 209366:
expected 15 fields, saw 22\nSkipping line 211310: expected 15 fields, saw
22\nSkipping line 246351: expected 15 fields, saw 22\nSkipping line 252364:
expected 15 fields, saw 22\n'
b'Skipping line 267003: expected 15 fields, saw 22\nSkipping line 268957:
expected 15 fields, saw 22\nSkipping line 303336: expected 15 fields, saw
22\nSkipping line 306021: expected 15 fields, saw 22\nSkipping line 311569:
expected 15 fields, saw 22\nSkipping line 316767: expected 15 fields, saw
22\nSkipping line 324009: expected 15 fields, saw 22\n'
b'Skipping line 359107: expected 15 fields, saw 22\nSkipping line 368367:
expected 15 fields, saw 22\nSkipping line 381180: expected 15 fields, saw
22\nSkipping line 390453: expected 15 fields, saw 22\n'
b'Skipping line 412243: expected 15 fields, saw 22\nSkipping line 419342:
expected 15 fields, saw 22\nSkipping line 457388: expected 15 fields, saw 22\n'
b'Skipping line 459935: expected 15 fields, saw 22\nSkipping line 460167:
expected 15 fields, saw 22\nSkipping line 466460: expected 15 fields, saw
22\nSkipping line 500314: expected 15 fields, saw 22\nSkipping line 500339:
expected 15 fields, saw 22\nSkipping line 505396: expected 15 fields, saw
```

```
22\nSkipping line 507760: expected 15 fields, saw 22\nSkipping line 513626:
expected 15 fields, saw 22\n'
b'Skipping line 527638: expected 15 fields, saw 22\nSkipping line 534209:
expected 15 fields, saw 22\nSkipping line 535687: expected 15 fields, saw
22\nSkipping line 547671: expected 15 fields, saw 22\nSkipping line 549054:
expected 15 fields, saw 22\n'
b'Skipping line 599929: expected 15 fields, saw 22\nSkipping line 604776:
expected 15 fields, saw 22\nSkipping line 609937: expected 15 fields, saw
22\nSkipping line 632059: expected 15 fields, saw 22\nSkipping line 638546:
expected 15 fields, saw 22\n'
b'Skipping line 665017: expected 15 fields, saw 22\nSkipping line 677680:
expected 15 fields, saw 22\nSkipping line 684370: expected 15 fields, saw
22\nSkipping line 720217: expected 15 fields, saw 29\n'
b'Skipping line 723240: expected 15 fields, saw 22\nSkipping line 723433:
expected 15 fields, saw 22\nSkipping line 763891: expected 15 fields, saw 22\n'
b'Skipping line 800288: expected 15 fields, saw 22\nSkipping line 802942:
expected 15 fields, saw 22\nSkipping line 803379: expected 15 fields, saw
22\nSkipping line 805122: expected 15 fields, saw 22\nSkipping line 821899:
expected 15 fields, saw 22\nSkipping line 831707: expected 15 fields, saw
22\nSkipping line 842829: expected 15 fields, saw 22\nSkipping line 843604:
expected 15 fields, saw 22\n'
b'Skipping line 863904: expected 15 fields, saw 22\nSkipping line 875655:
expected 15 fields, saw 22\nSkipping line 886796: expected 15 fields, saw
22\nSkipping line 892299: expected 15 fields, saw 22\nSkipping line 902518:
expected 15 fields, saw 22\nSkipping line 903079: expected 15 fields, saw
22\nSkipping line 912678: expected 15 fields, saw 22\n'
b'Skipping line 932953: expected 15 fields, saw 22\nSkipping line 936838:
expected 15 fields, saw 22\nSkipping line 937177: expected 15 fields, saw
22\nSkipping line 947695: expected 15 fields, saw 22\nSkipping line 960713:
expected 15 fields, saw 22\nSkipping line 965225: expected 15 fields, saw
22\nSkipping line 980776: expected 15 fields, saw 22\n'
b'Skipping line 999318: expected 15 fields, saw 22\nSkipping line 1007247:
expected 15 fields, saw 22\nSkipping line 1015987: expected 15 fields, saw
22\nSkipping line 1018984: expected 15 fields, saw 22\nSkipping line 1028671:
expected 15 fields, saw 22\n'
b'Skipping line 1063360: expected 15 fields, saw 22\nSkipping line 1066195:
expected 15 fields, saw 22\nSkipping line 1066578: expected 15 fields, saw
22\nSkipping line 1066869: expected 15 fields, saw 22\nSkipping line 1068809:
expected 15 fields, saw 22\nSkipping line 1069505: expected 15 fields, saw
22\nSkipping line 1087983: expected 15 fields, saw 22\nSkipping line 1108184:
expected 15 fields, saw 22\n'
b'Skipping line 1118137: expected 15 fields, saw 22\nSkipping line 1142723:
expected 15 fields, saw 22\nSkipping line 1152492: expected 15 fields, saw
22\nSkipping line 1156947: expected 15 fields, saw 22\nSkipping line 1172563:
expected 15 fields, saw 22\n'
b'Skipping line 1209254: expected 15 fields, saw 22\nSkipping line 1212966:
expected 15 fields, saw 22\nSkipping line 1236533: expected 15 fields, saw
22\nSkipping line 1237598: expected 15 fields, saw 22\n'
```

```
b'Skipping line 1273825: expected 15 fields, saw 22\nSkipping line 1277898:
expected 15 fields, saw 22\nSkipping line 1283654: expected 15 fields, saw
22\nSkipping line 1286023: expected 15 fields, saw 22\nSkipping line 1302038:
expected 15 fields, saw 22\nSkipping line 1305179: expected 15 fields, saw 22\n'
b'Skipping line 1326022: expected 15 fields, saw 22\nSkipping line 1338120:
expected 15 fields, saw 22\nSkipping line 1338503: expected 15 fields, saw
22\nSkipping line 1338849: expected 15 fields, saw 22\nSkipping line 1341513:
expected 15 fields, saw 22\nSkipping line 1346493: expected 15 fields, saw
22\nSkipping line 1373127: expected 15 fields, saw 22\n'
b'Skipping line 1389508: expected 15 fields, saw 22\nSkipping line 1413951:
expected 15 fields, saw 22\nSkipping line 1433626: expected 15 fields, saw 22\n'
b'Skipping line 1442698: expected 15 fields, saw 22\nSkipping line 1472982:
expected 15 fields, saw 22\nSkipping line 1482282: expected 15 fields, saw
22\nSkipping line 1487808: expected 15 fields, saw 22\nSkipping line 1500636:
expected 15 fields, saw 22\n'
b'Skipping line 1511479: expected 15 fields, saw 22\nSkipping line 1532302:
expected 15 fields, saw 22\nSkipping line 1537952: expected 15 fields, saw
22\nSkipping line 1539951: expected 15 fields, saw 22\nSkipping line 1541020:
expected 15 fields, saw 22\n'
b'Skipping line 1594217: expected 15 fields, saw 22\nSkipping line 1612264:
expected 15 fields, saw 22\nSkipping line 1615907: expected 15 fields, saw
22\nSkipping line 1621859: expected 15 fields, saw 22\n'
b'Skipping line 1653542: expected 15 fields, saw 22\nSkipping line 1671537:
expected 15 fields, saw 22\nSkipping line 1672879: expected 15 fields, saw
22\nSkipping line 1674523: expected 15 fields, saw 22\nSkipping line 1677355:
expected 15 fields, saw 22\nSkipping line 1703907: expected 15 fields, saw 22\n'
b'Skipping line 1713046: expected 15 fields, saw 22\nSkipping line 1722982:
expected 15 fields, saw 22\nSkipping line 1727290: expected 15 fields, saw
22\nSkipping line 1744482: expected 15 fields, saw 22\n'
b'Skipping line 1803858: expected 15 fields, saw 22\nSkipping line 1810069:
expected 15 fields, saw 22\nSkipping line 1829751: expected 15 fields, saw
22\nSkipping line 1831699: expected 15 fields, saw 22\n'
b'Skipping line 1863131: expected 15 fields, saw 22\nSkipping line 1867917:
expected 15 fields, saw 22\nSkipping line 1874790: expected 15 fields, saw
22\nSkipping line 1879952: expected 15 fields, saw 22\nSkipping line 1880501:
expected 15 fields, saw 22\nSkipping line 1886655: expected 15 fields, saw
22\nSkipping line 1887888: expected 15 fields, saw 22\nSkipping line 1894286:
expected 15 fields, saw 22\nSkipping line 1895400: expected 15 fields, saw 22\n'
b'Skipping line 1904040: expected 15 fields, saw 22\nSkipping line 1907604:
expected 15 fields, saw 22\nSkipping line 1915739: expected 15 fields, saw
22\nSkipping line 1921514: expected 15 fields, saw 22\nSkipping line 1939428:
expected 15 fields, saw 22\nSkipping line 1944342: expected 15 fields, saw
22\nSkipping line 1949699: expected 15 fields, saw 22\nSkipping line 1961872:
expected 15 fields, saw 22\n'
b'Skipping line 1968846: expected 15 fields, saw 22\nSkipping line 1999941:
expected 15 fields, saw 22\nSkipping line 2001492: expected 15 fields, saw
22\nSkipping line 2011204: expected 15 fields, saw 22\nSkipping line 2025295:
expected 15 fields, saw 22\n'
```

```
b'Skipping line 2041266: expected 15 fields, saw 22\nSkipping line 2073314:
expected 15 fields, saw 22\nSkipping line 2080133: expected 15 fields, saw
22\nSkipping line 2088521: expected 15 fields, saw 22\n'
b'Skipping line 2103490: expected 15 fields, saw 22\nSkipping line 2115278:
expected 15 fields, saw 22\nSkipping line 2153174: expected 15 fields, saw
22\nSkipping line 2161731: expected 15 fields, saw 22\n'
b'Skipping line 2165250: expected 15 fields, saw 22\nSkipping line 2175132:
expected 15 fields, saw 22\nSkipping line 2206817: expected 15 fields, saw
22\nSkipping line 2215848: expected 15 fields, saw 22\nSkipping line 2223811:
expected 15 fields, saw 22\n'
b'Skipping line 2257265: expected 15 fields, saw 22\nSkipping line 2259163:
expected 15 fields, saw 22\nSkipping line 2263291: expected 15 fields, saw 22\n'
b'Skipping line 2301943: expected 15 fields, saw 22\nSkipping line 2304371:
expected 15 fields, saw 22\nSkipping line 2306015: expected 15 fields, saw
22\nSkipping line 2312186: expected 15 fields, saw 22\nSkipping line 2314740:
expected 15 fields, saw 22\nSkipping line 2317754: expected 15 fields, saw 22\n'
b'Skipping line 2383514: expected 15 fields, saw 22\n'
b'Skipping line 2449763: expected 15 fields, saw 22\n'
b'Skipping line 2589323: expected 15 fields, saw 22\n'
b'Skipping line 2775036: expected 15 fields, saw 22\n'
b'Skipping line 2935174: expected 15 fields, saw 22\n'
b'Skipping line 3078830: expected 15 fields, saw 22\n'
b'Skipping line 3123091: expected 15 fields, saw 22\n'
b'Skipping line 3185533: expected 15 fields, saw 22\n'
b'Skipping line 4150395: expected 15 fields, saw 22\n'
b'Skipping line 4748401: expected 15 fields, saw 22\n'
```

[3]:
```
test
```

[3]:
```
        marketplace  customer_id       review_id  product_id  product_parent  \
0                US     37000337  R3DT59XH7HXR9K  B00303FI0G       529320574
1                US     15272914  R1LFS11BNASSU8  B00JCZKZN6       274237558
2                US     36137863  R296RT05AG0AF6  B00JLIKA5C       544675303
3                US     43311049  R3V37XDZ7ZCI3L  B000GBNB8G       491599489
4                US     13763148  R14GU232NQFYX2  B00VJ5KX9S       353790155
...             ...          ...             ...         ...             ...
4874885          US     51094108  R22DLC2P26MUMR  B00004SBGS       732420532
4874886          US     50562512  R1N6KLTENLQOMT  B00004SBIA       261705371
4874887          US     52469742  R10TW4QXDV8KJC  B00004SPEF       191184892
4874888          US     51865238   R41RL2U1FSQ4V  B00004RHR6       912491903
4874889          US     52900320  R1NHMPKSJG2E37  B0000021VO        41913389

                                             product_title product_category  \
0                            Arthur Court Paper Towel Holder          Kitchen
1        Olde Thompson Bavaria Glass Salt and Pepper Mi...          Kitchen
2        Progressive International PL8 Professional Man...          Kitchen
3                                  Zyliss Jumbo Garlic Press          Kitchen
4        1 X Premier Pizza Cutter - Stainless Steel 14"...          Kitchen
```

```
...                                                         ...          ...
4874885  Le Creuset Enameled Cast-Iron 6-3/4-Quart Oval...          Kitchen
4874886  Le Creuset Enameled Cast-Iron 2-Quart Heart Ca...          Kitchen
4874887           Krups 358-70 La Glaciere Ice Cream Maker          Kitchen
4874888        Hoffritz Stainless-Steel Manual Can Opener          Kitchen
4874889                                       Tammy Rogers          Kitchen

         star_rating  helpful_votes  total_votes vine verified_purchase  \
0                5.0            0.0          0.0    N                 Y
1                5.0            0.0          1.0    N                 Y
2                5.0            0.0          0.0    N                 Y
3                5.0            0.0          1.0    N                 Y
4                5.0            0.0          0.0    N                 Y
...              ...            ...          ...  ...               ...
4874885          4.0           30.0         41.0    N                 N
4874886          5.0           84.0         92.0    N                 N
4874887          4.0           55.0         60.0    N                 N
4874888          4.0           30.0         42.0    N                 N
4874889          5.0            5.0          5.0    N                 N

                                   review_headline  \
0                Beautiful. Looks great on counter
1                              Awesome & Self-ness
2                     Fabulous and worth every penny
3                                       Five Stars
4                                  Better than sex
...                                            ...
4874885              Not as sturdy as you'd think.
4874886                      A Sweetheart of A Pan
4874887                      Ice Cream Like a Dream
4874888              Opens anything and everything
4874889  The more you listen, the more you hear...

                                               review_body review_date  label
0                      Beautiful.  Looks great on counter.  2015-08-31     -1
1        I personally have 5 days sets and have also bo...  2015-08-31     -1
2        Fabulous and worth every penny. Used for clean...  2015-08-31     -1
3        A must if you love garlic on tomato marinara s...  2015-08-31     -1
4        Worth every penny! Buy one now and be a pizza ...  2015-08-31     -1
...                                                    ...         ...    ...
4874885  After a month of heavy use, primarily as a chi...  2000-04-28     -1
4874886  I've used my Le Creuset enameled cast iron coo...  2000-04-28     -1
4874887  According to my wife, this is \\"the best birt...  2000-04-28     -1
4874888  Hoffritz has a name of producing a trendy and ...  2000-04-24     -1
4874889  OK. I was late to snap to the Dead Reckoners. ...  2000-01-20     -1

[4874890 rows x 16 columns]
```

[3]:
```
a1 = test.loc[test['star_rating']==1].sample(50000)
a2 = test.loc[test['star_rating']==2].sample(50000)
a3 = test.loc[test['star_rating']==3].sample(50000)
a4 = test.loc[test['star_rating']==4].sample(50000)
a5 = test.loc[test['star_rating']==5].sample(50000)
```

[4]:
```
new_test = pd.concat([a1,a2,a3,a4,a5])
```

[5]:
```
new_test['lable'] = 1
```

[6]:
```
new_test.label[test.star_rating>3] = 1
new_test.label[test.star_rating<3] = 2
new_test.label[test.star_rating==3] = 3
```

```
/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:1:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  """Entry point for launching an IPython kernel.
/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:2:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:3:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  This is separate from the ipykernel package so we can avoid doing imports
until
```

[7]:
```
new_test = new_test[['label','review_body']]
```

[8]:
```
copy_test = copy.deepcopy(new_test)
```

[9]:
```
new_test.groupby('label').count()
```

[9]:
```
       review_body
label
1            99994
2            99994
3            49999
```

[10]:
```
import gensim.downloader as api
```

[17]:
```
wv = api.load('word2vec-google-news-300')
```

```
[==================================================] 100.0% 1662.8/1662.8MB
downloaded
```

[18]:
```
vec_king = wv['king']
```

[19]:
```
vec_man = wv['man']
```

[21]:
```
vec_woman = wv['woman']
```

[22]:
```
queen_test = vec_king - vec_man + vec_woman
```

[24]:
```
vec_queen = wv['queen']
```

[25]:
```
def cos_sim(vector_a, vector_b):
    vector_a = np.mat(vector_a)
    vector_b = np.mat(vector_b)
    num = float(vector_a * vector_b.T)
    denom = np.linalg.norm(vector_a) * np.linalg.norm(vector_b)
    cos = num / denom
    sim = 0.5 + 0.5 * cos
    return sim
```

[32]:
```
cos_sim(queen_test,vec_queen)
```

[32]:
```
0.8650258715231234
```

[ ]:
```

```

[33]:
```
vec_excellent = wv['excellent']
vec_out = wv['outstanding']
```

[34]:
```
cos_sim(vec_excellent,vec_out)
```

[34]:
```
0.7783743005874819
```

[11]:
```
from nltk import word_tokenize
```

[12]:
```
len(new_test)
```

[12]:
```
250000
```

[13]:
```
new_test['review_body'] = new_test['review_body'].apply(lambda x:
 ↪word_tokenize(str(x)))
```

[14]:
```
x_matrix = list(new_test['review_body'].values)
```

[15]:
```
from gensim.models import Word2Vec
```

[26]:
```
model = Word2Vec(x_matrix,sg=1,vector_size=300,window=11,min_count=10)
import tempfile
```

[17]:
```
import tempfile
```

[34]:
```
import tempfile
```

```
with tempfile.NamedTemporaryFile(prefix='gensim-model-', delete=False) as tmp:
    temporary_filepath = 'save'
    model.save(temporary_filepath)
```

[19]:
```
import gensim
```

[20]:
```
model = gensim.models.Word2Vec.load('save')
```

[21]:
```
n_king = model.wv['king']
n_man = model.wv['man']
n_woman = model.wv['woman']
n_queen = model.wv['queen']
```

[22]:
```
tt_queen = n_king-n_man+n_woman
```

[26]:
```
cos_sim(tt_queen,n_queen)
```

[26]:
```
0.7368551514727513
```

[35]:
```
cos_sim(model.wv['excellent'],model.wv['outstanding'])
```

[35]:
```
0.8221981926372779
```

[27]:
```
#data clearing
copy_test['review_body'] = copy_test['review_body'].str.lower()

def tag(x):
    return re.sub('<.*?>','',str(x))
copy_test['review_body'] = copy_test['review_body'].apply(lambda x:tag(x))

def url(x):
    return re.sub('(https?|ftp|file)://[-A-Za-z0-9+&@#/%?=~_|!:,.;
 ↪]+[-A-Za-z0-9+&@#/%=~_|]','',str(x))

copy_test['review_body'] = copy_test['review_body'].apply(lambda x:url(x))

import contractions
copy_test['review_body'] = copy_test['review_body'].apply(lambda x:contractions.
 ↪fix(x))

def non_alphabetical(x):
    return re.sub('[^a-zA-Z\s]','',str(x))
copy_test['review_body'] = copy_test['review_body'].apply(lambda x:
 ↪non_alphabetical(x))
def extra_space(x):
    return re.sub( ' +',' ',str(x))
copy_test['review_body'] = copy_test['review_body'].apply(lambda x:
 ↪extra_space(x))
```

[28]:
```
#Pre-processing
```

```
import nltk
from nltk.corpus import stopwords
nltk.download('stopwords')
stop_words_set = set(stopwords.words('english'))
from nltk import word_tokenize, pos_tag

def stop_words(x):
    word_tokens = word_tokenize(x)
    temp = []
    for i in word_tokens:
        if i not in stop_words_set:
            temp.append(i)
    return temp
copy_test['review_body'] = copy_test['review_body'].apply(lambda x:
 ↪stop_words(x))
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     /Users/chenlin/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

[97]:

```
```

[30]:

```
def vector_avg(x):
    i = 0
    n = 0
    a = np.zeros(300)
    while i<len(x):
        if x[i] in model.wv:
            a += model.wv[x[i]]
        else:
            n += 1
        i = i+1
    if n == 0: return a
    else:return a/(len(x)-n)
```

[ ]:

```
```

[271]:

```
copy_test['avg_vector'] = copy_test['review_body'].apply(lambda x:vector_avg(x))
```

```
/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:1:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  """Entry point for launching an IPython kernel.
```

[272]:
```
copy_test = copy_test.loc[copy_test['label'] <3]
```

[273]:
```
X = copy_test['avg_vector']
Y = copy_test['label']
```

[274]:
```
from sklearn.model_selection import train_test_split
```

[275]:
```
x_train, x_test, y_train, y_test = train_test_split(X, Y, random_state = 19,␣
 ↪test_size = 0.2)
```

[281]:
```
x_train = np.array(x_train.tolist())

y_train = np.array(y_train.tolist())

x_train = np.nan_to_num(x_train)
```

[221]:
```
from sklearn.datasets import load_digits
from sklearn.linear_model import Perceptron
clf = Perceptron(tol=1e-3, random_state=0)
clf.fit(x_train, y_train)
```

[221]:
```
Perceptron()
```

[284]:
```
x_test = np.array(x_test.tolist())

y_test = np.array(y_test.tolist())

x_test = np.nan_to_num(x_test)
```

[231]:
```
y_test_predict = clf.predict(x_test)
```

[234]:
```
y_test = np.array(y_test.tolist())
```

1 perceptron

[245]:
```
from sklearn.metrics import classification_report
mymodel_perceptron_report =␣
 ↪classification_report(y_test,y_test_predict,output_dict=True)
```

[247]:
```
mymodel_perceptron_report['accuracy']
```

[247]:
```
0.798275
```

2 svm

[286]:
```
from sklearn.svm import LinearSVC
svc = LinearSVC()
svc.fit(x_train, y_train)
```

```
/opt/anaconda3/lib/python3.7/site-packages/sklearn/svm/_base.py:986:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  "the number of iterations.", ConvergenceWarning)
```

[286]:
```
LinearSVC()
```

[287]:
```
y_test_predict = svc.predict(x_test)
```

[291]:
```
mymodel_svm_report =␣
 ↪classification_report(y_test,y_test_predict,output_dict=True)
```

[292]:
```
mymodel_svm_report['accuracy']
```

[292]:
```
0.771825
```

3 Logistic Regression

[290]:
```
from sklearn.linear_model import LogisticRegression
logistic = LogisticRegression(random_state=0,max_iter = 300).fit(x_train,␣
 ↪y_train)
```

[293]:
```
y_test_predict = logistic.predict(x_test)
```

[294]:
```
mymodel_log_report =␣
 ↪classification_report(y_test,y_test_predict,output_dict=True)
```

[295]:
```
mymodel_log_report['accuracy']
```

[295]:
```
0.795125
```

[299]:
```
#opm
```

[300]:
```
opm = api.load('word2vec-google-news-300')
```

[301]:
```
def opm_vector_avg(x):
    i = 0
    n = 0
    a = np.zeros(300)
    while i<len(x):
        if x[i] in opm:
            a +=opm[x[i]]
        else:
            n += 1
        i = i+1
    if n == 0: return a
    else:return a/(len(x)-n)
```

[302]:
```
copy_test['opm_avg_vector'] = copy_test['review_body'].apply(lambda x:
 ↪opm_vector_avg(x))
```

[303]:
```
opm_x_train, opm_x_test, opm_y_train, opm_y_test =␣
 ↪train_test_split(copy_test['opm_avg_vector'], copy_test['label'],␣
 ↪random_state = 19, test_size = 0.2)
```

[304]:
```
opm_x_train = np.array(opm_x_train.tolist())
opm_y_train = np.array(opm_y_train.tolist())
opm_x_test = np.array(opm_x_test.tolist())
opm_y_test = np.array(opm_y_test.tolist())
opm_x_test = np.nan_to_num(opm_x_test)
opm_x_train = np.nan_to_num(opm_x_train)
```

[305]:
```
opm_clf = Perceptron(tol=1e-3, random_state=0)
opm_clf.fit(opm_x_train, opm_y_train)
```

[305]:
```
Perceptron()
```

[306]:
```
opm_y_test_predict = opm_clf.predict(opm_x_test)
```

[307]:
```
opm_model_perceptron_report =␣
 ↪classification_report(opm_y_test,opm_y_test_predict,output_dict=True)
```

[308]:
```
opm_model_perceptron_report['accuracy']
```

[308]:
```
0.773875
```

[309]:
```
svc = LinearSVC()
svc.fit(opm_x_train, opm_y_train)
```

```
/opt/anaconda3/lib/python3.7/site-packages/sklearn/svm/_base.py:986:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  "the number of iterations.", ConvergenceWarning)
```

[309]:
```
LinearSVC()
```

[310]:
```
omg_y_test_predict = svc.predict(opm_x_test)
```

[311]:
```
opm_svc_report =␣
 ↪classification_report(opm_y_test,opm_y_test_predict,output_dict=True)
```

[312]:
```
opm_svc_report['accuracy']
```

[312]:
```
0.773875
```

[313]:
```
logistic = LogisticRegression(random_state=0,max_iter = 300).fit(opm_x_train,␣
 ↪opm_y_train)
```

[314]:
```
omp_y_test_predict = logistic.predict(opm_x_test)
```

[315]:
```
opm_log_report =␣
 ↪classification_report(opm_y_test,opm_y_test_predict,output_dict=True)
```

[316]:
```
opm_log_report['accuracy']
```

[316]:
```
0.773875
```

[317]:
```
#tf-idf
```

[319]:
```
new_p_n = copy_test
```

[321]:
```
new_p_n['review_str'] = new_p_n['review_body'].apply(lambda x:' '.join(x))
from sklearn.feature_extraction.text import TfidfVectorizer
X = new_p_n['review_str']
Y = new_p_n['label']
v = TfidfVectorizer()
x_tfidf = v.fit_transform(X)

from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(x_tfidf, Y, random_state =␣
 ↪19, test_size = 0.2)
```

[325]:
```
perceptron = Perceptron()
perceptron.fit(x_train, y_train)

from sklearn.metrics import precision_recall_fscore_support as score
from sklearn.metrics import classification_report


y_test_predict = perceptron.predict(x_test)
```

[326]:
```
report_test = classification_report(y_test,y_test_predict,output_dict=True)
```

[328]:
```
accuracy_tf_perceptron = report_test['accuracy']
```

[329]:
```
accuracy_tf_perceptron
```

[329]:
```
0.832825
```

[331]:
```
svc = LinearSVC()
svc.fit(x_train, y_train)
y_test_predict = svc.predict(x_test)
```

[332]:
```
report_test = classification_report(y_test,y_test_predict,output_dict=True)
```

[333]:
```
accuracy_tf_svc = report_test['accuracy']
```

[334]:
```
accuracy_tf_svc
```

[334]:
```
0.874675
```

[336]:
```
logistic = LogisticRegression(random_state=0,max_iter = 300).fit(x_train,␣
 ↪y_train)
y_test_predict = logistic.predict(x_test)
```

[337]:
```
report_test = classification_report(y_test,y_test_predict,output_dict=True)
```

[338]:
```
accuracy_tf_log = report_test['accuracy']
```

[339]:
```
accuracy_tf_log
```

[339]:
```
0.878375
```

[353]:
```
compare = {'perceptron':
 ↪[mymodel_perceptron_report['accuracy'],opm_model_perceptron_report['accuracy'],accuracy_tf_
          ,'svm':
 ↪[mymodel_svm_report['accuracy'],opm_svc_report['accuracy'],accuracy_tf_svc]\
          ,'logistic':
 ↪[mymodel_log_report['accuracy'],opm_log_report['accuracy'],accuracy_tf_log]}
```

[357]:
```
pd.DataFrame(compare,index=['my model accuracy','google news accuracy','TF-IDF␣
 ↪accuracy'])
```

[357]:
```
                      perceptron       svm  logistic
my model accuracy       0.771825  0.771825  0.795125
google news accuracy    0.773875  0.773875  0.773875
TF-IDF accuracy         0.832825  0.874675  0.878375
```

4 Conlusion: I think TF-IDF has the best accuracy among all the ML methods.

hw2(2)

October 5, 2021

[1]:
```
import pandas as pd
import numpy as np
import copy
import re
```

[2]:
```
import pandas as pd
test = pd.read_csv("amazon_reviews_us_Kitchen_v1_00.tsv", sep =␣
 ↪'\t',error_bad_lines = False)
test['label'] = -1
a1 = test.loc[test['star_rating']==1].sample(50000)
a2 = test.loc[test['star_rating']==2].sample(50000)
a3 = test.loc[test['star_rating']==3].sample(50000)
a4 = test.loc[test['star_rating']==4].sample(50000)
a5 = test.loc[test['star_rating']==5].sample(50000)
```

```
b'Skipping line 16148: expected 15 fields, saw 22\nSkipping line 20100: expected
15 fields, saw 22\nSkipping line 45178: expected 15 fields, saw 22\nSkipping
line 48700: expected 15 fields, saw 22\nSkipping line 63331: expected 15 fields,
saw 22\n'
b'Skipping line 86053: expected 15 fields, saw 22\nSkipping line 88858: expected
15 fields, saw 22\nSkipping line 115017: expected 15 fields, saw 22\n'
b'Skipping line 137366: expected 15 fields, saw 22\nSkipping line 139110:
expected 15 fields, saw 22\nSkipping line 165540: expected 15 fields, saw
22\nSkipping line 171813: expected 15 fields, saw 22\n'
b'Skipping line 203723: expected 15 fields, saw 22\nSkipping line 209366:
expected 15 fields, saw 22\nSkipping line 211310: expected 15 fields, saw
22\nSkipping line 246351: expected 15 fields, saw 22\nSkipping line 252364:
expected 15 fields, saw 22\n'
b'Skipping line 267003: expected 15 fields, saw 22\nSkipping line 268957:
expected 15 fields, saw 22\nSkipping line 303336: expected 15 fields, saw
22\nSkipping line 306021: expected 15 fields, saw 22\nSkipping line 311569:
expected 15 fields, saw 22\nSkipping line 316767: expected 15 fields, saw
22\nSkipping line 324009: expected 15 fields, saw 22\n'
b'Skipping line 359107: expected 15 fields, saw 22\nSkipping line 368367:
expected 15 fields, saw 22\nSkipping line 381180: expected 15 fields, saw
22\nSkipping line 390453: expected 15 fields, saw 22\n'
b'Skipping line 412243: expected 15 fields, saw 22\nSkipping line 419342:
```

```
expected 15 fields, saw 22\nSkipping line 457388: expected 15 fields, saw 22\n'
b'Skipping line 459935: expected 15 fields, saw 22\nSkipping line 460167:
expected 15 fields, saw 22\nSkipping line 466460: expected 15 fields, saw
22\nSkipping line 500314: expected 15 fields, saw 22\nSkipping line 500339:
expected 15 fields, saw 22\nSkipping line 505396: expected 15 fields, saw
22\nSkipping line 507760: expected 15 fields, saw 22\nSkipping line 513626:
expected 15 fields, saw 22\n'
b'Skipping line 527638: expected 15 fields, saw 22\nSkipping line 534209:
expected 15 fields, saw 22\nSkipping line 535687: expected 15 fields, saw
22\nSkipping line 547671: expected 15 fields, saw 22\nSkipping line 549054:
expected 15 fields, saw 22\n'
b'Skipping line 599929: expected 15 fields, saw 22\nSkipping line 604776:
expected 15 fields, saw 22\nSkipping line 609937: expected 15 fields, saw
22\nSkipping line 632059: expected 15 fields, saw 22\nSkipping line 638546:
expected 15 fields, saw 22\n'
b'Skipping line 665017: expected 15 fields, saw 22\nSkipping line 677680:
expected 15 fields, saw 22\nSkipping line 684370: expected 15 fields, saw
22\nSkipping line 720217: expected 15 fields, saw 29\n'
b'Skipping line 723240: expected 15 fields, saw 22\nSkipping line 723433:
expected 15 fields, saw 22\nSkipping line 763891: expected 15 fields, saw 22\n'
b'Skipping line 800288: expected 15 fields, saw 22\nSkipping line 802942:
expected 15 fields, saw 22\nSkipping line 803379: expected 15 fields, saw
22\nSkipping line 805122: expected 15 fields, saw 22\nSkipping line 821899:
expected 15 fields, saw 22\nSkipping line 831707: expected 15 fields, saw
22\nSkipping line 842829: expected 15 fields, saw 22\nSkipping line 843604:
expected 15 fields, saw 22\n'
b'Skipping line 863904: expected 15 fields, saw 22\nSkipping line 875655:
expected 15 fields, saw 22\nSkipping line 886796: expected 15 fields, saw
22\nSkipping line 892299: expected 15 fields, saw 22\nSkipping line 902518:
expected 15 fields, saw 22\nSkipping line 903079: expected 15 fields, saw
22\nSkipping line 912678: expected 15 fields, saw 22\n'
b'Skipping line 932953: expected 15 fields, saw 22\nSkipping line 936838:
expected 15 fields, saw 22\nSkipping line 937177: expected 15 fields, saw
22\nSkipping line 947695: expected 15 fields, saw 22\nSkipping line 960713:
expected 15 fields, saw 22\nSkipping line 965225: expected 15 fields, saw
22\nSkipping line 980776: expected 15 fields, saw 22\n'
b'Skipping line 999318: expected 15 fields, saw 22\nSkipping line 1007247:
expected 15 fields, saw 22\nSkipping line 1015987: expected 15 fields, saw
22\nSkipping line 1018984: expected 15 fields, saw 22\nSkipping line 1028671:
expected 15 fields, saw 22\n'
b'Skipping line 1063360: expected 15 fields, saw 22\nSkipping line 1066195:
expected 15 fields, saw 22\nSkipping line 1066578: expected 15 fields, saw
22\nSkipping line 1066869: expected 15 fields, saw 22\nSkipping line 1068809:
expected 15 fields, saw 22\nSkipping line 1069505: expected 15 fields, saw
22\nSkipping line 1087983: expected 15 fields, saw 22\nSkipping line 1108184:
expected 15 fields, saw 22\n'
b'Skipping line 1118137: expected 15 fields, saw 22\nSkipping line 1142723:
expected 15 fields, saw 22\nSkipping line 1152492: expected 15 fields, saw
```

```
22\nSkipping line 1156947: expected 15 fields, saw 22\nSkipping line 1172563:
expected 15 fields, saw 22\n'
b'Skipping line 1209254: expected 15 fields, saw 22\nSkipping line 1212966:
expected 15 fields, saw 22\nSkipping line 1236533: expected 15 fields, saw
22\nSkipping line 1237598: expected 15 fields, saw 22\n'
b'Skipping line 1273825: expected 15 fields, saw 22\nSkipping line 1277898:
expected 15 fields, saw 22\nSkipping line 1283654: expected 15 fields, saw
22\nSkipping line 1286023: expected 15 fields, saw 22\nSkipping line 1302038:
expected 15 fields, saw 22\nSkipping line 1305179: expected 15 fields, saw 22\n'
b'Skipping line 1326022: expected 15 fields, saw 22\nSkipping line 1338120:
expected 15 fields, saw 22\nSkipping line 1338503: expected 15 fields, saw
22\nSkipping line 1338849: expected 15 fields, saw 22\nSkipping line 1341513:
expected 15 fields, saw 22\nSkipping line 1346493: expected 15 fields, saw
22\nSkipping line 1373127: expected 15 fields, saw 22\n'
b'Skipping line 1389508: expected 15 fields, saw 22\nSkipping line 1413951:
expected 15 fields, saw 22\nSkipping line 1433626: expected 15 fields, saw 22\n'
b'Skipping line 1442698: expected 15 fields, saw 22\nSkipping line 1472982:
expected 15 fields, saw 22\nSkipping line 1482282: expected 15 fields, saw
22\nSkipping line 1487808: expected 15 fields, saw 22\nSkipping line 1500636:
expected 15 fields, saw 22\n'
b'Skipping line 1511479: expected 15 fields, saw 22\nSkipping line 1532302:
expected 15 fields, saw 22\nSkipping line 1537952: expected 15 fields, saw
22\nSkipping line 1539951: expected 15 fields, saw 22\nSkipping line 1541020:
expected 15 fields, saw 22\n'
b'Skipping line 1594217: expected 15 fields, saw 22\nSkipping line 1612264:
expected 15 fields, saw 22\nSkipping line 1615907: expected 15 fields, saw
22\nSkipping line 1621859: expected 15 fields, saw 22\n'
b'Skipping line 1653542: expected 15 fields, saw 22\nSkipping line 1671537:
expected 15 fields, saw 22\nSkipping line 1672879: expected 15 fields, saw
22\nSkipping line 1674523: expected 15 fields, saw 22\nSkipping line 1677355:
expected 15 fields, saw 22\nSkipping line 1703907: expected 15 fields, saw 22\n'
b'Skipping line 1713046: expected 15 fields, saw 22\nSkipping line 1722982:
expected 15 fields, saw 22\nSkipping line 1727290: expected 15 fields, saw
22\nSkipping line 1744482: expected 15 fields, saw 22\n'
b'Skipping line 1803858: expected 15 fields, saw 22\nSkipping line 1810069:
expected 15 fields, saw 22\nSkipping line 1829751: expected 15 fields, saw
22\nSkipping line 1831699: expected 15 fields, saw 22\n'
b'Skipping line 1863131: expected 15 fields, saw 22\nSkipping line 1867917:
expected 15 fields, saw 22\nSkipping line 1874790: expected 15 fields, saw
22\nSkipping line 1879952: expected 15 fields, saw 22\nSkipping line 1880501:
expected 15 fields, saw 22\nSkipping line 1886655: expected 15 fields, saw
22\nSkipping line 1887888: expected 15 fields, saw 22\nSkipping line 1894286:
expected 15 fields, saw 22\nSkipping line 1895400: expected 15 fields, saw 22\n'
b'Skipping line 1904040: expected 15 fields, saw 22\nSkipping line 1907604:
expected 15 fields, saw 22\nSkipping line 1915739: expected 15 fields, saw
22\nSkipping line 1921514: expected 15 fields, saw 22\nSkipping line 1939428:
expected 15 fields, saw 22\nSkipping line 1944342: expected 15 fields, saw
22\nSkipping line 1949699: expected 15 fields, saw 22\nSkipping line 1961872:
```

```
expected 15 fields, saw 22\n'
b'Skipping line 1968846: expected 15 fields, saw 22\nSkipping line 1999941:
expected 15 fields, saw 22\nSkipping line 2001492: expected 15 fields, saw
22\nSkipping line 2011204: expected 15 fields, saw 22\nSkipping line 2025295:
expected 15 fields, saw 22\n'
b'Skipping line 2041266: expected 15 fields, saw 22\nSkipping line 2073314:
expected 15 fields, saw 22\nSkipping line 2080133: expected 15 fields, saw
22\nSkipping line 2088521: expected 15 fields, saw 22\n'
b'Skipping line 2103490: expected 15 fields, saw 22\nSkipping line 2115278:
expected 15 fields, saw 22\nSkipping line 2153174: expected 15 fields, saw
22\nSkipping line 2161731: expected 15 fields, saw 22\n'
b'Skipping line 2165250: expected 15 fields, saw 22\nSkipping line 2175132:
expected 15 fields, saw 22\nSkipping line 2206817: expected 15 fields, saw
22\nSkipping line 2215848: expected 15 fields, saw 22\nSkipping line 2223811:
expected 15 fields, saw 22\n'
b'Skipping line 2257265: expected 15 fields, saw 22\nSkipping line 2259163:
expected 15 fields, saw 22\nSkipping line 2263291: expected 15 fields, saw 22\n'
b'Skipping line 2301943: expected 15 fields, saw 22\nSkipping line 2304371:
expected 15 fields, saw 22\nSkipping line 2306015: expected 15 fields, saw
22\nSkipping line 2312186: expected 15 fields, saw 22\nSkipping line 2314740:
expected 15 fields, saw 22\nSkipping line 2317754: expected 15 fields, saw 22\n'
b'Skipping line 2383514: expected 15 fields, saw 22\n'
b'Skipping line 2449763: expected 15 fields, saw 22\n'
b'Skipping line 2589323: expected 15 fields, saw 22\n'
b'Skipping line 2775036: expected 15 fields, saw 22\n'
b'Skipping line 2935174: expected 15 fields, saw 22\n'
b'Skipping line 3078830: expected 15 fields, saw 22\n'
b'Skipping line 3123091: expected 15 fields, saw 22\n'
b'Skipping line 3185533: expected 15 fields, saw 22\n'
b'Skipping line 4150395: expected 15 fields, saw 22\n'
b'Skipping line 4748401: expected 15 fields, saw 22\n'
```

[3]:
```python
new_test = pd.concat([a1,a2,a3,a4,a5])
```

[4]:
```python
new_test['lable'] = 1
```

[5]:
```python
new_test.label[test.star_rating>3] = 1
new_test.label[test.star_rating<3] = 2
new_test.label[test.star_rating==3] = 3
new_test = new_test[['label','review_body']]
```

```
/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:1:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  """Entry point for launching an IPython kernel.
```

```
/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:2:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:3:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  This is separate from the ipykernel package so we can avoid doing imports
until
```

[6]:
```
copy_test = copy.deepcopy(new_test)
```

[7]:
```
import gensim
model_1 = gensim.models.Word2Vec.load('save')
```

[8]:
```
#data clearing
copy_test['review_body'] = copy_test['review_body'].str.lower()

def tag(x):
    return re.sub('<.*?>','',str(x))
copy_test['review_body'] = copy_test['review_body'].apply(lambda x:tag(x))

def url(x):
    return re.sub('(https?|ftp|file)://[-A-Za-z0-9+&@#/%?=~_|!:,.;
 ↪]+[-A-Za-z0-9+&@#/%=~_|]','',str(x))

copy_test['review_body'] = copy_test['review_body'].apply(lambda x:url(x))

import contractions
copy_test['review_body'] = copy_test['review_body'].apply(lambda x:contractions.
 ↪fix(x))

def non_alphabetical(x):
    return re.sub('[^a-zA-Z\s]','',str(x))
copy_test['review_body'] = copy_test['review_body'].apply(lambda x:
 ↪non_alphabetical(x))
def extra_space(x):
    return re.sub( ' +',' ',str(x))
copy_test['review_body'] = copy_test['review_body'].apply(lambda x:
 ↪extra_space(x))
```

[9]:
```
#Pre-processing

import nltk
from nltk.corpus import stopwords
nltk.download('stopwords')
stop_words_set = set(stopwords.words('english'))
from nltk import word_tokenize, pos_tag

def stop_words(x):
    word_tokens = word_tokenize(x)
    temp = []
    for i in word_tokens:
        if i not in stop_words_set:
            temp.append(i)
    return temp
copy_test['review_body'] = copy_test['review_body'].apply(lambda x:
 ↪stop_words(x))
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     /Users/chenlin/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

[10]:
```
ternary_copy = copy_test
```

[11]:
```
ternary_copy
```

[11]:
```
         label                                        review_body
2901789      2  [griddle, works, finebut, plates, rust, cost, ...
4079589      2  [badthe, blade, sharp, enough, cut, tomato, fi...
461277       2  [products, like, actually, make, productionmar...
2583042      2  [second, time, washed, largest, container, dis...
3856377      2  [admit, love, keurig, coffee, brewer, said, as...
...        ...                                                ...
2397865      1  [oven, pantry, delivered, stated, specificatio...
519273       1                                             [like]
4557793      1  [ratings, already, given, product, reason, pur...
428144       1  [purchased, gold, igrind, could, happier, come...
963579       1                                             [huge]

[250000 rows x 2 columns]
```

[12]:
```
def vector_avg(x):
    i = 0
    n = 0
    a = np.zeros(300)
    while i<len(x):
        if x[i] in model_1.wv:
            a += model_1.wv[x[i]]
        else:
            n += 1
        i = i+1
    if n == 0: return a
    else:return a/(len(x)-n)
```

[13]:
```
copy_test['avg_vector'] = copy_test['review_body'].apply(lambda x:vector_avg(x))
```

[14]:
```
binary_copy = copy_test.loc[copy_test['label']!=3]
```

1 binary_copy

[15]:
```
binary_copy['y'] = 1
```

```
/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:1:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  """Entry point for launching an IPython kernel.
```

[16]:
```
binary_copy.y[binary_copy['label']==2] = 0
```

```
/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:1:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  """Entry point for launching an IPython kernel.
/opt/anaconda3/lib/python3.7/site-packages/pandas/core/series.py:992:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  self._where(~key, value, inplace=True)
```

[17]:
```
binary_copy
```

[17]:
```
         label                                        review_body  \
2901789      2  [griddle, works, finebut, plates, rust, cost, ...
4079589      2  [badthe, blade, sharp, enough, cut, tomato, fi...
461277       2  [products, like, actually, make, productionmar...
2583042      2  [second, time, washed, largest, container, dis...
3856377      2  [admit, love, keurig, coffee, brewer, said, as...
```

```
...         ...                                                   ...
2397865       1  [oven, pantry, delivered, stated, specificatio...
519273        1                                                [like]
4557793       1  [ratings, already, given, product, reason, pur...
428144        1  [purchased, gold, igrind, could, happier, come...
963579        1                                                [huge]

                                               avg_vector  y
2901789  [0.024641906699308984, 0.08881746848615316, 0...  0
4079589  [-0.016323742822610905, 0.05006689679742392, -...  0
461277   [0.06468200084054843, -0.010757067662780173, 0...  0
2583042  [0.05246899649500847, 1.5992026403546333, 0.13...  0
3856377  [0.06659146030120719, 0.03285078117067912, 0.0...  0
...                                                   ... ..
2397865  [0.8622930869460106, 1.1786859966814518, -0.08...  1
519273   [0.04800622910261154, 0.02565152198076248, 0.1...  1
4557793  [0.025994234447382757, 0.05975001984367483, -0...  1
428144   [0.017926203673899483, 0.167255596448538, -0.0...  1
963579   [-0.0970420092344284, 0.042717333883047104, 0...  1

[200000 rows x 4 columns]
```

[18]:
```
X = binary_copy['avg_vector']
Y = binary_copy['y']
```

[19]:
```
from sklearn.model_selection import train_test_split
```

[20]:
```
x_train, x_test, y_train, y_test = train_test_split(X, Y, random_state = 19,␣
 ↪test_size = 0.2)
```

[21]:
```
x_train = np.array(x_train.tolist())

y_train = np.array(y_train.tolist())

x_train = np.nan_to_num(x_train)
```

[22]:
```
x_test = np.array(x_test.tolist())

y_test = np.array(y_test.tolist())

x_test = np.nan_to_num(x_test)
```

[23]:
```
# import libraries
import torch
from torch.utils.data import DataLoader, Dataset
import torchvision
import torchvision.transforms as transforms
from torch.utils.data.sampler import SubsetRandomSampler
import matplotlib.pyplot as plt
import numpy as np
```

```python
import pandas as pd
```

[24]:
```python
from torch.utils.data import DataLoader, Dataset
from torch.utils.data import TensorDataset, DataLoader
```

[25]:
```python
# number of subprocesses to use for data loading
num_workers = 0
# how many samples per batch to load
batch_size = 20
# percentage of training set to use as validation
#valid_size = 0.2

DatasetTrain = TensorDataset(torch.from_numpy(x_train),torch.
 ↪from_numpy(y_train))

DatasetTest=TensorDataset(torch.from_numpy(x_test),torch.from_numpy(y_test))

train_loader=torch.utils.data.
 ↪DataLoader(DatasetTrain,batch_size=batch_size,shuffle=True,drop_last=True,␣
 ↪num_workers=0)

valid_loader=torch.utils.data.DataLoader(DatasetTest, batch_size=batch_size,␣
 ↪drop_last=True,num_workers=0)
```

[26]:
```python
import torch.nn as nn
import torch.nn.functional as F

# define the NN architecture
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        # number of hidden nodes in each layer
        hidden_1 = 50
        hidden_2 = 10
        # linear layer (300 -> hidden_1)
        self.fc1 = nn.Linear(300, hidden_1)
        # linear layer (n_hidden -> hidden_2)
        self.fc2 = nn.Linear(hidden_1, hidden_2)
        # linear layer (n_hidden -> 10)
        self.fc3 = nn.Linear(hidden_2, 2)
        # dropout layer (p=0.2)
        # dropout prevents overfitting of data
        self.dropout = nn.Dropout(0.2)

    def forward(self, x):
        # flatten image input
        x = x.view(-1, 300)
        # add hidden layer, with relu activation function
        x = F.relu(self.fc1(x))
        # add dropout layer
        x = self.dropout(x)
        # add hidden layer, with relu activation function
        x = F.relu(self.fc2(x))
        # add dropout layer
        x = self.dropout(x)
        # add output layer
        x = self.fc3(x)
        return x

# initialize the NN
model = Net()
print(model)
```

```
Net(
  (fc1): Linear(in_features=300, out_features=50, bias=True)
  (fc2): Linear(in_features=50, out_features=10, bias=True)
  (fc3): Linear(in_features=10, out_features=2, bias=True)
  (dropout): Dropout(p=0.2, inplace=False)
)
```

[27]:
```
# import libraries
import torch
## Specify loss and optimization functions

# specify loss function
criterion = nn.CrossEntropyLoss()

# specify optimizer
optimizer = torch.optim.SGD(model.parameters(), lr=0.01)
```

[28]:
```
train_loader
```

[28]:
```
<torch.utils.data.dataloader.DataLoader at 0x7ff13b70c550>
```

[29]:
```
# number of epochs to train the model
n_epochs = 20  # suggest training between 20-50 epochs

model.train() # prep model for training

for epoch in range(n_epochs):
    # monitor training loss
    train_loss = 0.0

    ###################
    # train the model #
    ###################
    for data, target in train_loader:
        # clear the gradients of all optimized variables
        optimizer.zero_grad()
        # forward pass: compute predicted outputs by passing inputs to the␣
↪model
        output = model(data.float())
        # calculate the loss
        loss = criterion(output, target)
        # backward pass: compute gradient of the loss with respect to model␣
↪parameters
        loss.backward()
        # perform a single optimization step (parameter update)
        optimizer.step()
        # update running training loss
        train_loss += loss.item()*data.size(0)

    # print training statistics
    # calculate average loss over an epoch
    train_loss = train_loss/len(train_loader.dataset)

    print('Epoch: {} \tTraining Loss: {:.6f}'.format(
        epoch+1,
        train_loss
        ))
```

```
Epoch: 1        Training Loss: 0.483913
Epoch: 2        Training Loss: 0.392670
Epoch: 3        Training Loss: 0.378242
Epoch: 4        Training Loss: 0.370848
Epoch: 5        Training Loss: 0.365118
Epoch: 6        Training Loss: 0.361285
Epoch: 7        Training Loss: 0.356872
Epoch: 8        Training Loss: 0.353547
Epoch: 9        Training Loss: 0.351196
Epoch: 10       Training Loss: 0.348159
Epoch: 11       Training Loss: 0.347070
Epoch: 12       Training Loss: 0.343552
Epoch: 13       Training Loss: 0.342804
Epoch: 14       Training Loss: 0.341586
Epoch: 15       Training Loss: 0.338453
Epoch: 16       Training Loss: 0.339810
Epoch: 17       Training Loss: 0.337566
Epoch: 18       Training Loss: 0.336264
Epoch: 19       Training Loss: 0.335728
Epoch: 20       Training Loss: 0.334147
```

[30]:
```
model
```

[30]:
```
Net(
  (fc1): Linear(in_features=300, out_features=50, bias=True)
  (fc2): Linear(in_features=50, out_features=10, bias=True)
  (fc3): Linear(in_features=10, out_features=2, bias=True)
  (dropout): Dropout(p=0.2, inplace=False)
)
```

[31]:
```
DatasetTest=TensorDataset(torch.from_numpy(x_test),torch.from_numpy(y_test))

total_num = 0
correct = 0
for x_temp,y_real in DatasetTest:
    res = model(x_temp.float())
    _, pred = torch.max(res,1)
    total_num += 1
    if int(pred) == int(y_real):
        correct += 1
```

[32]:
```
if 0 in copy_test['label'].values:
    print('hhhh')
```

[33]:
```
accuracy_mymodel_avg =  correct/total_num
```

[34]:
```
accuracy_mymodel_avg
```

[34]:
```
0.854175
```

[35]:
```
import copy
```

[36]:
```
import gensim
model_1 = gensim.models.Word2Vec.load('save')
```

[37]:
```
def vector_concat(x):
    i = 0
    n_x = 0
    length = len(x)
    if length == 0:
        return np.zeros(3000)
    a = np.zeros(300)

    while i<10:
        if n_x < length:
            if x[n_x] in model_1.wv :
                if i == 0:
                    ans = copy.deepcopy(model_1.wv[x[n_x]])
                    i += 1
                else:
                    ans = np.concatenate((ans,model_1.wv[x[n_x]]), axis=None)
                    i += 1
            n_x += 1
        else:
            if i == 0:
                ans = a
                i += 1
            else:
                ans = np.concatenate((ans,a), axis=None)
                i += 1
    return ans
```

[38]:
```
binary_copy['concat_vec'] = binary_copy['review_body'].apply(lambda x:
 ↪vector_concat(x))
```

```
/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:1:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  """Entry point for launching an IPython kernel.
```

[39]:
```
binary_copy
```

[39]:
```
         label                                        review_body  \
2901789      2  [griddle, works, finebut, plates, rust, cost, ...
4079589      2  [badthe, blade, sharp, enough, cut, tomato, fi...
461277       2  [products, like, actually, make, productionmar...
2583042      2  [second, time, washed, largest, container, dis...
3856377      2  [admit, love, keurig, coffee, brewer, said, as...
...        ...                                                ...
2397865      1  [oven, pantry, delivered, stated, specificatio...
519273       1                                             [like]
4557793      1  [ratings, already, given, product, reason, pur...
428144       1  [purchased, gold, igrind, could, happier, come...
963579       1                                             [huge]

                                               avg_vector  y  \
2901789  [0.024641906699308984, 0.08881746848615316, 0...  0
4079589  [-0.016323742822610905, 0.05006689679742392, -...  0
461277   [0.06468200084054843, -0.010757067662780173, 0...  0
2583042  [0.05246899649500847, 1.5992026403546333, 0.13...  0
3856377  [0.06659146030120719, 0.03285078117067912, 0.0...  0
...                                                   ... ..
2397865  [0.8622930869460106, 1.1786859966814518, -0.08...  1
519273   [0.04800622910261154, 0.02565152198076248, 0.1...  1
4557793  [0.025994234447382757, 0.05975001984367483, -0...  1
428144   [0.017926203673899483, 0.167255596448538, -0.0...  1
963579   [-0.0970420092344284, 0.042717333883047104, 0...  1
```

```
                                                 concat_vec
2901789   [-0.04459453, -0.07535303, -0.4262394, -0.0587...
4079589   [-0.020585256, 0.03667236, -0.26981658, 0.1782...
461277    [-0.013184801, -0.005327605, 0.17744249, 0.035...
2583042   [-0.069223836, 0.11872141, 0.029483773, 0.0597...
3856377   [-0.10600603, 0.029664135, -0.09290071, 0.0523...
...                                                     ...
2397865   [0.07905123, 0.053250413, -0.01902701, 0.03232...
519273    [0.04800622910261154, 0.02565152198076248, 0.1...
4557793   [0.22660527, -0.20551513, -0.020619687, 0.1293...
428144    [0.07608983, -0.057236966, 0.3409337, -0.27152...
963579    [-0.0970420092344284, 0.042717333883047104, 0...

[200000 rows x 5 columns]
```

[40]:
```
X = binary_copy['concat_vec']
Y = binary_copy['y']
```

[41]:
```
x_train, x_test, y_train, y_test = train_test_split(X, Y, random_state = 19,␣
 ↪test_size = 0.2)
```

[42]:
```
x_train = np.array(x_train.tolist())

y_train = np.array(y_train.tolist())

x_train = np.nan_to_num(x_train)
```

[43]:
```
x_test = np.array(x_test.tolist())

y_test = np.array(y_test.tolist())

x_test = np.nan_to_num(x_test)
```

[44]:
```
# number of subprocesses to use for data loading
num_workers = 0
# how many samples per batch to load
batch_size = 20
# percentage of training set to use as validation
#valid_size = 0.2

DatasetTrain = TensorDataset(torch.from_numpy(x_train),torch.
 ↪from_numpy(y_train))

DatasetTest=TensorDataset(torch.from_numpy(x_test),torch.from_numpy(y_test))

train_loader=torch.utils.data.
 ↪DataLoader(DatasetTrain,batch_size=batch_size,shuffle=True,drop_last=True,␣
 ↪num_workers=0)
```

```
valid_loader=torch.utils.data.DataLoader(DatasetTest, batch_size=batch_size,␣
↪drop_last=True,num_workers=0)
```

[45]:
```
y_train
```

[45]:
```
array([0, 1, 1, ..., 1, 0, 0])
```

[46]:
```
import torch.nn as nn
import torch.nn.functional as F

# define the NN architecture
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        # number of hidden nodes in each layer
        hidden_1 = 50
        hidden_2 = 10
        # linear layer (300 -> hidden_1)
        self.fc1 = nn.Linear(3000, hidden_1)
        # linear layer (n_hidden -> hidden_2)
        self.fc2 = nn.Linear(hidden_1, hidden_2)
        # linear layer (n_hidden -> 10)
        self.fc3 = nn.Linear(hidden_2, 2)
        # dropout layer (p=0.2)
        # dropout prevents overfitting of data
        self.dropout = nn.Dropout(0.2)

    def forward(self, x):
        # flatten image input
        x = x.view(-1, 3000)
        # add hidden layer, with relu activation function
        x = F.relu(self.fc1(x))
        # add dropout layer
        x = self.dropout(x)
        # add hidden layer, with relu activation function
        x = F.relu(self.fc2(x))
        # add dropout layer
        x = self.dropout(x)
        # add output layer
        x = self.fc3(x)
        return x

# initialize the NN
model = Net()
```

[47]:
```
print(model)
```

```
Net(
  (fc1): Linear(in_features=3000, out_features=50, bias=True)
  (fc2): Linear(in_features=50, out_features=10, bias=True)
  (fc3): Linear(in_features=10, out_features=2, bias=True)
  (dropout): Dropout(p=0.2, inplace=False)
)
```

[48]:
```
# specify loss function
criterion = nn.CrossEntropyLoss()

# specify optimizer
optimizer = torch.optim.SGD(model.parameters(), lr=0.01)
```

[49]:
```
# number of epochs to train the model
n_epochs = 20  # suggest training between 20-50 epochs

model.train() # prep model for training

for epoch in range(n_epochs):
    # monitor training loss
    train_loss = 0.0

    ###################
    # train the model #
    ###################
    for data, target in train_loader:
        # clear the gradients of all optimized variables
        optimizer.zero_grad()
        # forward pass: compute predicted outputs by passing inputs to the␣
 ↪model
        output = model(data.float())
        # calculate the loss
        loss = criterion(output, target)
        # backward pass: compute gradient of the loss with respect to model␣
 ↪parameters
        loss.backward()
        # perform a single optimization step (parameter update)
        optimizer.step()
        # update running training loss
        train_loss += loss.item()*data.size(0)

    # print training statistics
    # calculate average loss over an epoch
    train_loss = train_loss/len(train_loader.dataset)

    print('Epoch: {} \tTraining Loss: {:.6f}'.format(
        epoch+1,
        train_loss
        ))
```

```
Epoch: 1        Training Loss: 0.569580
Epoch: 2        Training Loss: 0.464485
Epoch: 3        Training Loss: 0.440153
Epoch: 4        Training Loss: 0.424263
Epoch: 5        Training Loss: 0.409070
Epoch: 6        Training Loss: 0.393365
Epoch: 7        Training Loss: 0.378543
Epoch: 8        Training Loss: 0.363366
Epoch: 9        Training Loss: 0.346821
Epoch: 10       Training Loss: 0.331147
Epoch: 11       Training Loss: 0.317480
Epoch: 12       Training Loss: 0.302490
Epoch: 13       Training Loss: 0.288299
Epoch: 14       Training Loss: 0.275590
Epoch: 15       Training Loss: 0.265235
Epoch: 16       Training Loss: 0.254121
Epoch: 17       Training Loss: 0.243234
Epoch: 18       Training Loss: 0.233545
Epoch: 19       Training Loss: 0.224100
Epoch: 20       Training Loss: 0.217472
```

[ ]:

```
[50]: DatasetTest=TensorDataset(torch.from_numpy(x_test),torch.from_numpy(y_test))

total_num = 0
correct = 0
for x_temp,y_real in DatasetTest:
    res = model(x_temp.float())
    _, pred = torch.max(res,1)
    total_num += 1
    if int(pred) == int(y_real):
        correct += 1
```

```
[51]: accuracy_mymodel_concat =  correct/total_num
```

```
[54]: accuracy_mymodel_concat
```

```
[54]: 0.77565
```

```
[52]: accuracy_mymodel_avg
```

```
[52]: 0.854175
```

2 conclusion:

the accuracy for mymodel using avg vector with binary label is 0.854175
the accuracy for mymodel using concat vector with binary label is 0.77565

hw2(3)

October 5, 2021

[1]:
```
import pandas as pd
import numpy as np
import copy
import re
```

[2]:
```
import pandas as pd
test = pd.read_csv("amazon_reviews_us_Kitchen_v1_00.tsv", sep =␣
 ↪'\t',error_bad_lines = False)
test['label'] = -1
a1 = test.loc[test['star_rating']==1].sample(50000)
a2 = test.loc[test['star_rating']==2].sample(50000)
a3 = test.loc[test['star_rating']==3].sample(50000)
a4 = test.loc[test['star_rating']==4].sample(50000)
a5 = test.loc[test['star_rating']==5].sample(50000)
```

```
b'Skipping line 16148: expected 15 fields, saw 22\nSkipping line 20100: expected
15 fields, saw 22\nSkipping line 45178: expected 15 fields, saw 22\nSkipping
line 48700: expected 15 fields, saw 22\nSkipping line 63331: expected 15 fields,
saw 22\n'
b'Skipping line 86053: expected 15 fields, saw 22\nSkipping line 88858: expected
15 fields, saw 22\nSkipping line 115017: expected 15 fields, saw 22\n'
b'Skipping line 137366: expected 15 fields, saw 22\nSkipping line 139110:
expected 15 fields, saw 22\nSkipping line 165540: expected 15 fields, saw
22\nSkipping line 171813: expected 15 fields, saw 22\n'
b'Skipping line 203723: expected 15 fields, saw 22\nSkipping line 209366:
expected 15 fields, saw 22\nSkipping line 211310: expected 15 fields, saw
22\nSkipping line 246351: expected 15 fields, saw 22\nSkipping line 252364:
expected 15 fields, saw 22\n'
b'Skipping line 267003: expected 15 fields, saw 22\nSkipping line 268957:
expected 15 fields, saw 22\nSkipping line 303336: expected 15 fields, saw
22\nSkipping line 306021: expected 15 fields, saw 22\nSkipping line 311569:
expected 15 fields, saw 22\nSkipping line 316767: expected 15 fields, saw
22\nSkipping line 324009: expected 15 fields, saw 22\n'
b'Skipping line 359107: expected 15 fields, saw 22\nSkipping line 368367:
expected 15 fields, saw 22\nSkipping line 381180: expected 15 fields, saw
22\nSkipping line 390453: expected 15 fields, saw 22\n'
b'Skipping line 412243: expected 15 fields, saw 22\nSkipping line 419342:
```

```
expected 15 fields, saw 22\nSkipping line 457388: expected 15 fields, saw 22\n'
b'Skipping line 459935: expected 15 fields, saw 22\nSkipping line 460167:
expected 15 fields, saw 22\nSkipping line 466460: expected 15 fields, saw
22\nSkipping line 500314: expected 15 fields, saw 22\nSkipping line 500339:
expected 15 fields, saw 22\nSkipping line 505396: expected 15 fields, saw
22\nSkipping line 507760: expected 15 fields, saw 22\nSkipping line 513626:
expected 15 fields, saw 22\n'
b'Skipping line 527638: expected 15 fields, saw 22\nSkipping line 534209:
expected 15 fields, saw 22\nSkipping line 535687: expected 15 fields, saw
22\nSkipping line 547671: expected 15 fields, saw 22\nSkipping line 549054:
expected 15 fields, saw 22\n'
b'Skipping line 599929: expected 15 fields, saw 22\nSkipping line 604776:
expected 15 fields, saw 22\nSkipping line 609937: expected 15 fields, saw
22\nSkipping line 632059: expected 15 fields, saw 22\nSkipping line 638546:
expected 15 fields, saw 22\n'
b'Skipping line 665017: expected 15 fields, saw 22\nSkipping line 677680:
expected 15 fields, saw 22\nSkipping line 684370: expected 15 fields, saw
22\nSkipping line 720217: expected 15 fields, saw 29\n'
b'Skipping line 723240: expected 15 fields, saw 22\nSkipping line 723433:
expected 15 fields, saw 22\nSkipping line 763891: expected 15 fields, saw 22\n'
b'Skipping line 800288: expected 15 fields, saw 22\nSkipping line 802942:
expected 15 fields, saw 22\nSkipping line 803379: expected 15 fields, saw
22\nSkipping line 805122: expected 15 fields, saw 22\nSkipping line 821899:
expected 15 fields, saw 22\nSkipping line 831707: expected 15 fields, saw
22\nSkipping line 842829: expected 15 fields, saw 22\nSkipping line 843604:
expected 15 fields, saw 22\n'
b'Skipping line 863904: expected 15 fields, saw 22\nSkipping line 875655:
expected 15 fields, saw 22\nSkipping line 886796: expected 15 fields, saw
22\nSkipping line 892299: expected 15 fields, saw 22\nSkipping line 902518:
expected 15 fields, saw 22\nSkipping line 903079: expected 15 fields, saw
22\nSkipping line 912678: expected 15 fields, saw 22\n'
b'Skipping line 932953: expected 15 fields, saw 22\nSkipping line 936838:
expected 15 fields, saw 22\nSkipping line 937177: expected 15 fields, saw
22\nSkipping line 947695: expected 15 fields, saw 22\nSkipping line 960713:
expected 15 fields, saw 22\nSkipping line 965225: expected 15 fields, saw
22\nSkipping line 980776: expected 15 fields, saw 22\n'
b'Skipping line 999318: expected 15 fields, saw 22\nSkipping line 1007247:
expected 15 fields, saw 22\nSkipping line 1015987: expected 15 fields, saw
22\nSkipping line 1018984: expected 15 fields, saw 22\nSkipping line 1028671:
expected 15 fields, saw 22\n'
b'Skipping line 1063360: expected 15 fields, saw 22\nSkipping line 1066195:
expected 15 fields, saw 22\nSkipping line 1066578: expected 15 fields, saw
22\nSkipping line 1066869: expected 15 fields, saw 22\nSkipping line 1068809:
expected 15 fields, saw 22\nSkipping line 1069505: expected 15 fields, saw
22\nSkipping line 1087983: expected 15 fields, saw 22\nSkipping line 1108184:
expected 15 fields, saw 22\n'
b'Skipping line 1118137: expected 15 fields, saw 22\nSkipping line 1142723:
expected 15 fields, saw 22\nSkipping line 1152492: expected 15 fields, saw
```

```
22\nSkipping line 1156947: expected 15 fields, saw 22\nSkipping line 1172563:
expected 15 fields, saw 22\n'
b'Skipping line 1209254: expected 15 fields, saw 22\nSkipping line 1212966:
expected 15 fields, saw 22\nSkipping line 1236533: expected 15 fields, saw
22\nSkipping line 1237598: expected 15 fields, saw 22\n'
b'Skipping line 1273825: expected 15 fields, saw 22\nSkipping line 1277898:
expected 15 fields, saw 22\nSkipping line 1283654: expected 15 fields, saw
22\nSkipping line 1286023: expected 15 fields, saw 22\nSkipping line 1302038:
expected 15 fields, saw 22\nSkipping line 1305179: expected 15 fields, saw 22\n'
b'Skipping line 1326022: expected 15 fields, saw 22\nSkipping line 1338120:
expected 15 fields, saw 22\nSkipping line 1338503: expected 15 fields, saw
22\nSkipping line 1338849: expected 15 fields, saw 22\nSkipping line 1341513:
expected 15 fields, saw 22\nSkipping line 1346493: expected 15 fields, saw
22\nSkipping line 1373127: expected 15 fields, saw 22\n'
b'Skipping line 1389508: expected 15 fields, saw 22\nSkipping line 1413951:
expected 15 fields, saw 22\nSkipping line 1433626: expected 15 fields, saw 22\n'
b'Skipping line 1442698: expected 15 fields, saw 22\nSkipping line 1472982:
expected 15 fields, saw 22\nSkipping line 1482282: expected 15 fields, saw
22\nSkipping line 1487808: expected 15 fields, saw 22\nSkipping line 1500636:
expected 15 fields, saw 22\n'
b'Skipping line 1511479: expected 15 fields, saw 22\nSkipping line 1532302:
expected 15 fields, saw 22\nSkipping line 1537952: expected 15 fields, saw
22\nSkipping line 1539951: expected 15 fields, saw 22\nSkipping line 1541020:
expected 15 fields, saw 22\n'
b'Skipping line 1594217: expected 15 fields, saw 22\nSkipping line 1612264:
expected 15 fields, saw 22\nSkipping line 1615907: expected 15 fields, saw
22\nSkipping line 1621859: expected 15 fields, saw 22\n'
b'Skipping line 1653542: expected 15 fields, saw 22\nSkipping line 1671537:
expected 15 fields, saw 22\nSkipping line 1672879: expected 15 fields, saw
22\nSkipping line 1674523: expected 15 fields, saw 22\nSkipping line 1677355:
expected 15 fields, saw 22\nSkipping line 1703907: expected 15 fields, saw 22\n'
b'Skipping line 1713046: expected 15 fields, saw 22\nSkipping line 1722982:
expected 15 fields, saw 22\nSkipping line 1727290: expected 15 fields, saw
22\nSkipping line 1744482: expected 15 fields, saw 22\n'
b'Skipping line 1803858: expected 15 fields, saw 22\nSkipping line 1810069:
expected 15 fields, saw 22\nSkipping line 1829751: expected 15 fields, saw
22\nSkipping line 1831699: expected 15 fields, saw 22\n'
b'Skipping line 1863131: expected 15 fields, saw 22\nSkipping line 1867917:
expected 15 fields, saw 22\nSkipping line 1874790: expected 15 fields, saw
22\nSkipping line 1879952: expected 15 fields, saw 22\nSkipping line 1880501:
expected 15 fields, saw 22\nSkipping line 1886655: expected 15 fields, saw
22\nSkipping line 1887888: expected 15 fields, saw 22\nSkipping line 1894286:
expected 15 fields, saw 22\nSkipping line 1895400: expected 15 fields, saw 22\n'
b'Skipping line 1904040: expected 15 fields, saw 22\nSkipping line 1907604:
expected 15 fields, saw 22\nSkipping line 1915739: expected 15 fields, saw
22\nSkipping line 1921514: expected 15 fields, saw 22\nSkipping line 1939428:
expected 15 fields, saw 22\nSkipping line 1944342: expected 15 fields, saw
22\nSkipping line 1949699: expected 15 fields, saw 22\nSkipping line 1961872:
```

```
expected 15 fields, saw 22\n'
b'Skipping line 1968846: expected 15 fields, saw 22\nSkipping line 1999941:
expected 15 fields, saw 22\nSkipping line 2001492: expected 15 fields, saw
22\nSkipping line 2011204: expected 15 fields, saw 22\nSkipping line 2025295:
expected 15 fields, saw 22\n'
b'Skipping line 2041266: expected 15 fields, saw 22\nSkipping line 2073314:
expected 15 fields, saw 22\nSkipping line 2080133: expected 15 fields, saw
22\nSkipping line 2088521: expected 15 fields, saw 22\n'
b'Skipping line 2103490: expected 15 fields, saw 22\nSkipping line 2115278:
expected 15 fields, saw 22\nSkipping line 2153174: expected 15 fields, saw
22\nSkipping line 2161731: expected 15 fields, saw 22\n'
b'Skipping line 2165250: expected 15 fields, saw 22\nSkipping line 2175132:
expected 15 fields, saw 22\nSkipping line 2206817: expected 15 fields, saw
22\nSkipping line 2215848: expected 15 fields, saw 22\nSkipping line 2223811:
expected 15 fields, saw 22\n'
b'Skipping line 2257265: expected 15 fields, saw 22\nSkipping line 2259163:
expected 15 fields, saw 22\nSkipping line 2263291: expected 15 fields, saw 22\n'
b'Skipping line 2301943: expected 15 fields, saw 22\nSkipping line 2304371:
expected 15 fields, saw 22\nSkipping line 2306015: expected 15 fields, saw
22\nSkipping line 2312186: expected 15 fields, saw 22\nSkipping line 2314740:
expected 15 fields, saw 22\nSkipping line 2317754: expected 15 fields, saw 22\n'
b'Skipping line 2383514: expected 15 fields, saw 22\n'
b'Skipping line 2449763: expected 15 fields, saw 22\n'
b'Skipping line 2589323: expected 15 fields, saw 22\n'
b'Skipping line 2775036: expected 15 fields, saw 22\n'
b'Skipping line 2935174: expected 15 fields, saw 22\n'
b'Skipping line 3078830: expected 15 fields, saw 22\n'
b'Skipping line 3123091: expected 15 fields, saw 22\n'
b'Skipping line 3185533: expected 15 fields, saw 22\n'
b'Skipping line 4150395: expected 15 fields, saw 22\n'
b'Skipping line 4748401: expected 15 fields, saw 22\n'
```

[3]:
```
new_test = pd.concat([a1,a2,a3,a4,a5])
```

[4]:
```
new_test['lable'] = 1
```

[5]:
```
new_test.label[test.star_rating>3] = 1
new_test.label[test.star_rating<3] = 2
new_test.label[test.star_rating==3] = 3
new_test = new_test[['label','review_body']]
```

```
/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:1:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  """Entry point for launching an IPython kernel.
```

```
/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:2:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:3:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  This is separate from the ipykernel package so we can avoid doing imports
until
```

[6]:
```
import gensim
model_1 = gensim.models.Word2Vec.load('save')
```

[12]:
```
#data clearing
new_test['review_body'] = new_test['review_body'].str.lower()

def tag(x):
    return re.sub('<.*?>','',str(x))
new_test['review_body'] = new_test['review_body'].apply(lambda x:tag(x))

def url(x):
    return re.sub('(https?|ftp|file)://[-A-Za-z0-9+&@#/%?=~_|!:,.;
 ↪]+[-A-Za-z0-9+&@#/%=~_|]','',str(x))

new_test['review_body'] = new_test['review_body'].apply(lambda x:url(x))

import contractions
new_test['review_body'] = new_test['review_body'].apply(lambda x:contractions.
 ↪fix(x))

def non_alphabetical(x):
    return re.sub('[^a-zA-Z\s]','',str(x))
new_test['review_body'] = new_test['review_body'].apply(lambda x:
 ↪non_alphabetical(x))
def extra_space(x):
    return re.sub( ' +',' ',str(x))
new_test['review_body'] = new_test['review_body'].apply(lambda x:extra_space(x))
```

[14]:
```
#Pre-processing

import nltk
from nltk.corpus import stopwords
```

```
nltk.download('stopwords')
stop_words_set = set(stopwords.words('english'))
from nltk import word_tokenize, pos_tag

def stop_words(x):
    word_tokens = word_tokenize(x)
    temp = []
    for i in word_tokens:
        if i not in stop_words_set:
            temp.append(i)
    return temp
new_test['review_body'] = new_test['review_body'].apply(lambda x:stop_words(x))
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     /Users/chenlin/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

[15]:
```
new_test
```

[15]:
```
         label                                         review_body
1261010      2           [victorinox, product, first, word, read]
149938       2  [match, size, color, tho, color, name, also, e...
4379464      2  [advertised, add, char, broil, cooker, purchas...
28297        2                                              [yuck]
4844336      2  [bought, toaster, black, decker, name, sorry, ...
...        ...                                                ...
3962262      1  [well, made, lid, stays, tight, else, ask, mug...
1501108      1    [love, dishes, added, coffee, cups, set, years]
2533959      1  [pug, lover, like, love, cool, fridge, magnet,...
48365        1  [bought, pan, make, cake, yo, celebrate, offic...
4145163      1  [person, gave, toaster, oven, bad, review, sai...

[250000 rows x 2 columns]
```

[16]:
```
def vector_avg(x):
    i = 0
    n = 0
    a = np.zeros(300)
    while i<len(x):
        if x[i] in model_1.wv:
            a += model_1.wv[x[i]]
        else:
            n += 1
        i = i+1
    if n == 0: return a
    else:return a/(len(x)-n)
```

[17]:
```
new_test['avg_vector'] = new_test['review_body'].apply(lambda x:vector_avg(x))
```

[32]:
```
new_test['y'] = -1
```

[34]:
```
new_test['y'][new_test['label'] == 1] =0
```

```
/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:1:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  """Entry point for launching an IPython kernel.
```

[35]:
```
new_test['y'][new_test['label'] == 2] =1
```

```
/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:1:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  """Entry point for launching an IPython kernel.
```

[36]:
```
new_test['y'][new_test['label'] == 3] =2
```

```
/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:1:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  """Entry point for launching an IPython kernel.
```

[37]:
```
new_test
```

[37]:
```
         label                                        review_body  \
1261010      2           [victorinox, product, first, word, read]
149938       2  [match, size, color, tho, color, name, also, e...
4379464      2  [advertised, add, char, broil, cooker, purchas...
28297        2                                             [yuck]
4844336      2  [bought, toaster, black, decker, name, sorry, ...
...        ...                                                ...
3962262      1  [well, made, lid, stays, tight, else, ask, mug...
1501108      1    [love, dishes, added, coffee, cups, set, years]
2533959      1  [pug, lover, like, love, cool, fridge, magnet,...
48365        1  [bought, pan, make, cake, yo, celebrate, offic...
4145163      1  [person, gave, toaster, oven, bad, review, sai...
```

```
                                              avg_vector  y
1261010  [0.3361993785947561, 0.3755444842390716, 0.205...  1
149938   [0.023530922830104828, 0.7736800406128168, 0.0...  1
4379464  [0.9732713364064693, 1.8002999844029546, 0.767...  1
28297    [-0.30345696210861206, 0.005608719307929277, -...  1
4844336  [0.05253013025219666, 0.0007164585744825805, 0...  1
...                                                  ... ..
3962262  [0.6963680814951658, 1.8281475193798542, -0.74...  0
1501108  [1.1396409068256617, 0.5796800553798676, 0.723...  0
2533959  [0.031026512011885644, 0.06857175541420778, -0...  0
48365    [0.04685636181134863, 0.04049028242713705, -0...  0
4145163  [0.06293770963466673, 0.027035569575536152, -0...  0

[250000 rows x 4 columns]
```

[42]:
```
X = new_test['avg_vector']
Y = new_test['y']

from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(X, Y, random_state = 19,␣
 ↪test_size = 0.2)

x_train = np.array(x_train.tolist())

y_train = np.array(y_train.tolist())

x_train = np.nan_to_num(x_train)

x_test = np.array(x_test.tolist())

y_test = np.array(y_test.tolist())

x_test = np.nan_to_num(x_test)
```

1 ternary

[43]:
```
# import libraries
import torch
from torch.utils.data import DataLoader, Dataset
import torchvision
import torchvision.transforms as transforms
from torch.utils.data.sampler import SubsetRandomSampler
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
```

[44]:
```
from torch.utils.data import DataLoader, Dataset
from torch.utils.data import TensorDataset, DataLoader
```

[45]:
```
# number of subprocesses to use for data loading
num_workers = 0
# how many samples per batch to load
batch_size = 20
# percentage of training set to use as validation
#valid_size = 0.2

DatasetTrain = TensorDataset(torch.from_numpy(x_train),torch.
 ↪from_numpy(y_train))

DatasetTest=TensorDataset(torch.from_numpy(x_test),torch.from_numpy(y_test))

train_loader=torch.utils.data.
 ↪DataLoader(DatasetTrain,batch_size=batch_size,shuffle=True,drop_last=True,␣
 ↪num_workers=0)

valid_loader=torch.utils.data.DataLoader(DatasetTest, batch_size=batch_size,␣
 ↪drop_last=True,num_workers=0)
```

[46]:
```
for xi ,yi in train_loader:
    print(xi,yi)
    break
```

```
tensor([[ 2.4691e+00,  2.8362e+00,  6.5291e-01,  ...,  1.4724e+00,
          4.6802e+00, -3.3048e+00],
        [ 8.3865e-01,  1.8521e-01,  1.7748e-01,  ...,  1.5910e+00,
          1.3063e+00, -8.6692e-01],
        [-2.9935e-01,  5.9200e-01,  2.7225e-01,  ...,  7.0387e-01,
          5.6117e-01,  5.3711e-01],
        ...,
        [ 7.1137e-01,  1.1082e+00,  9.6256e-01,  ...,  1.1243e+00,
          1.8219e+00, -1.9473e-02],
        [-9.6564e-01,  2.0634e+00,  4.8056e-01,  ...,  1.2328e+00,
          1.0309e+00, -1.7420e+00],
        [ 3.0430e-02,  6.6379e-02,  4.6200e-03,  ...,  1.9586e-02,
          7.6102e-02, -6.7758e-02]], dtype=torch.float64) tensor([1, 0, 1, 0, 0,
1, 2, 2, 1, 2, 1, 2, 1, 1, 2, 2, 1, 1, 1, 1])
```

[47]:
```
import torch.nn as nn
import torch.nn.functional as F

# define the NN architecture
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        # number of hidden nodes in each layer
        hidden_1 = 50
        hidden_2 = 10
        # linear layer (300 -> hidden_1)
        self.fc1 = nn.Linear(300, hidden_1)
        # linear layer (n_hidden -> hidden_2)
        self.fc2 = nn.Linear(hidden_1, hidden_2)
        # linear layer (n_hidden -> 10)
        self.fc3 = nn.Linear(hidden_2, 3)
        # dropout layer (p=0.2)
        # dropout prevents overfitting of data
        self.dropout = nn.Dropout(0.2)

    def forward(self, x):
        # flatten image input
        x = x.view(-1, 300)
        # add hidden layer, with relu activation function
        x = F.relu(self.fc1(x))
        # add dropout layer
        x = self.dropout(x)
        # add hidden layer, with relu activation function
        x = F.relu(self.fc2(x))
        # add dropout layer
        x = self.dropout(x)
        # add output layer
        x = self.fc3(x)
        return x

# initialize the NN
model = Net()
print(model)
```

```
Net(
  (fc1): Linear(in_features=300, out_features=50, bias=True)
  (fc2): Linear(in_features=50, out_features=10, bias=True)
  (fc3): Linear(in_features=10, out_features=3, bias=True)
  (dropout): Dropout(p=0.2, inplace=False)
)
```

[48]:
```
# import libraries
import torch
## Specify loss and optimization functions

# specify loss function
criterion = nn.CrossEntropyLoss()

# specify optimizer
```

```
optimizer = torch.optim.SGD(model.parameters(), lr=0.01)
```

[50]:
```
# number of epochs to train the model
n_epochs = 20  # suggest training between 20-50 epochs

model.train() # prep model for training

for epoch in range(n_epochs):
    # monitor training loss
    train_loss = 0.0

    ###################
    # train the model #
    ###################
    for data, target in train_loader:
        # clear the gradients of all optimized variables
        optimizer.zero_grad()
        # forward pass: compute predicted outputs by passing inputs to the␣
 ↪model
        output = model(data.float())
        # calculate the loss
        loss = criterion(output, target)
        # backward pass: compute gradient of the loss with respect to model␣
 ↪parameters
        loss.backward()
        # perform a single optimization step (parameter update)
        optimizer.step()
        # update running training loss
        train_loss += loss.item()*data.size(0)

    # print training statistics
    # calculate average loss over an epoch
    train_loss = train_loss/len(train_loader.dataset)

    print('Epoch: {} \tTraining Loss: {:.6f}'.format(
        epoch+1,
        train_loss
        ))
```

```
Epoch: 1        Training Loss: 0.830280
Epoch: 2        Training Loss: 0.778383
Epoch: 3        Training Loss: 0.760423
Epoch: 4        Training Loss: 0.752586
Epoch: 5        Training Loss: 0.747508
Epoch: 6        Training Loss: 0.743031
Epoch: 7        Training Loss: 0.739228
Epoch: 8        Training Loss: 0.735913
Epoch: 9        Training Loss: 0.733647
```

```
Epoch: 10       Training Loss: 0.731600
Epoch: 11       Training Loss: 0.729493
Epoch: 12       Training Loss: 0.727222
Epoch: 13       Training Loss: 0.725659
Epoch: 14       Training Loss: 0.726512
Epoch: 15       Training Loss: 0.723097
Epoch: 16       Training Loss: 0.722527
Epoch: 17       Training Loss: 0.721406
Epoch: 18       Training Loss: 0.719671
Epoch: 19       Training Loss: 0.717970
Epoch: 20       Training Loss: 0.717478
```

[51]:
```
DatasetTest=TensorDataset(torch.from_numpy(x_test),torch.from_numpy(y_test))

total_num = 0
correct = 0
for x_temp,y_real in DatasetTest:
    res = model(x_temp.float())
    _, pred = torch.max(res,1)
    total_num += 1
    if int(pred) == int(y_real):
        correct += 1
```

[52]:
```
accuracy_mymodel_avg =  correct/total_num
```

[56]:
```
print('The accuracy for ternary label from my model using average vector is ',␣
 ↪accuracy_mymodel_avg)
```

```
The accuracy for ternary label from my model using average vector is  0.6855
```

[54]:
```
def vector_concat(x):
    i = 0
    n_x = 0
    length = len(x)
    if length == 0:
        return np.zeros(3000)
    a = np.zeros(300)

    while i<10:
        if n_x < length:
            if x[n_x] in model_1.wv :
                if i == 0:
                    ans = copy.deepcopy(model_1.wv[x[n_x]])
                    i += 1
                else:
                    ans = np.concatenate((ans,model_1.wv[x[n_x]]), axis=None)
                    i += 1
            n_x += 1
        else:
            if i == 0:
                ans = a
                i += 1
            else:
                ans = np.concatenate((ans,a), axis=None)
                i += 1
    return ans
```

[57]:
```
new_test['concat_vec'] = new_test['review_body'].apply(lambda x:
↪vector_concat(x))
```

[63]:
```
new_test
```

[63]:
```
         label                                       review_body  \
1261010      2           [victorinox, product, first, word, read]
149938       2  [match, size, color, tho, color, name, also, e...
4379464      2  [advertised, add, char, broil, cooker, purchas...
28297        2                                            [yuck]
4844336      2  [bought, toaster, black, decker, name, sorry, ...
...        ...                                               ...
3962262      1  [well, made, lid, stays, tight, else, ask, mug...
1501108      1    [love, dishes, added, coffee, cups, set, years]
2533959      1  [pug, lover, like, love, cool, fridge, magnet,...
48365        1  [bought, pan, make, cake, yo, celebrate, offic...
4145163      1  [person, gave, toaster, oven, bad, review, sai...

                                                avg_vector  y  \
1261010  [0.3361993785947561, 0.3755444842390716, 0.205...  1
149938   [0.023530922830104828, 0.7736800406128168, 0.0...  1
4379464  [0.9732713364064693, 1.8002999844029546, 0.767...  1
28297    [-0.30345696210861206, 0.005608719307929277, -...  1
4844336  [0.05253013025219666, 0.0007164585744825805, 0...  1
...                                                    ... ..
3962262  [0.6963680814951658, 1.8281475193798542, -0.74...  0
1501108  [1.1396409068256617, 0.5796800553798676, 0.723...  0
2533959  [0.031026512011885644, 0.06857175541420778, -0...  0
48365    [0.04685636181134863, 0.04049028242713705, -0...  0
4145163  [0.06293770963466673, 0.027035569575536152, -0...  0

                                                concat_vec
1261010  [0.11119204014539719, -0.04113255813717842, 0...
149938   [-0.15528874, 0.082478374, 0.16331409, 0.11810...
4379464  [0.0612764, 0.18021023, -0.078093216, -0.21873...
28297    [-0.30345696210861206, 0.005608719307929277, -...
4844336  [0.06608908, -0.18412489, 0.31485054, -0.13780...
...                                                    ...
3962262  [-0.023777505, 0.19637382, 0.094618395, 0.0448...
```

```
1501108  [0.22806349396705627, 0.019244728609919548, 0...
2533959  [0.41755006, 0.32110247, 0.17386933, 0.1215716...
48365    [0.06608908, -0.18412489, 0.31485054, -0.13780...
4145163  [0.16218725, 0.10329889, -0.3650812, -0.031553...

[250000 rows x 5 columns]
```

[65]:
```
X = new_test['concat_vec']
Y = new_test['y']

from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(X, Y, random_state = 19,␣
↪test_size = 0.2)

x_train = np.array(x_train.tolist())

y_train = np.array(y_train.tolist())

x_train = np.nan_to_num(x_train)

x_test = np.array(x_test.tolist())

y_test = np.array(y_test.tolist())

x_test = np.nan_to_num(x_test)
```

[68]:
```
len(x_train[0])
```

[68]:
```
3000
```

[66]:
```
# number of subprocesses to use for data loading
num_workers = 0
# how many samples per batch to load
batch_size = 20
# percentage of training set to use as validation
#valid_size = 0.2

DatasetTrain = TensorDataset(torch.from_numpy(x_train),torch.
↪from_numpy(y_train))

DatasetTest=TensorDataset(torch.from_numpy(x_test),torch.from_numpy(y_test))

train_loader=torch.utils.data.
↪DataLoader(DatasetTrain,batch_size=batch_size,shuffle=True,drop_last=True,␣
↪num_workers=0)

valid_loader=torch.utils.data.DataLoader(DatasetTest, batch_size=batch_size,␣
↪drop_last=True,num_workers=0)
```

[69]:
```python
import torch.nn as nn
import torch.nn.functional as F

# define the NN architecture
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        # number of hidden nodes in each layer
        hidden_1 = 50
        hidden_2 = 10
        # linear layer (300 -> hidden_1)
        self.fc1 = nn.Linear(3000, hidden_1)
        # linear layer (n_hidden -> hidden_2)
        self.fc2 = nn.Linear(hidden_1, hidden_2)
        # linear layer (n_hidden -> 10)
        self.fc3 = nn.Linear(hidden_2, 3)
        # dropout layer (p=0.2)
        # dropout prevents overfitting of data
        self.dropout = nn.Dropout(0.2)

    def forward(self, x):
        # flatten image input
        x = x.view(-1, 3000)
        # add hidden layer, with relu activation function
        x = F.relu(self.fc1(x))
        # add dropout layer
        x = self.dropout(x)
        # add hidden layer, with relu activation function
        x = F.relu(self.fc2(x))
        # add dropout layer
        x = self.dropout(x)
        # add output layer
        x = self.fc3(x)
        return x

# initialize the NN
model = Net()
print(model)
```

```
Net(
  (fc1): Linear(in_features=3000, out_features=50, bias=True)
  (fc2): Linear(in_features=50, out_features=10, bias=True)
  (fc3): Linear(in_features=10, out_features=3, bias=True)
  (dropout): Dropout(p=0.2, inplace=False)
)
```

[70]:
```
# import libraries
import torch
## Specify loss and optimization functions

# specify loss function
criterion = nn.CrossEntropyLoss()

# specify optimizer
optimizer = torch.optim.SGD(model.parameters(), lr=0.01)
```

[71]:
```
# number of epochs to train the model
n_epochs = 20  # suggest training between 20-50 epochs

model.train() # prep model for training

for epoch in range(n_epochs):
    # monitor training loss
    train_loss = 0.0

    ###################
    # train the model #
    ###################
    for data, target in train_loader:
        # clear the gradients of all optimized variables
        optimizer.zero_grad()
        # forward pass: compute predicted outputs by passing inputs to the␣
 ↪model
        output = model(data.float())
        # calculate the loss
        loss = criterion(output, target)
        # backward pass: compute gradient of the loss with respect to model␣
 ↪parameters
        loss.backward()
        # perform a single optimization step (parameter update)
        optimizer.step()
        # update running training loss
        train_loss += loss.item()*data.size(0)

    # print training statistics
    # calculate average loss over an epoch
    train_loss = train_loss/len(train_loader.dataset)

    print('Epoch: {} \tTraining Loss: {:.6f}'.format(
        epoch+1,
        train_loss
        ))
```

```
Epoch: 1        Training Loss: 0.919045
```

```
Epoch: 2        Training Loss: 0.838901
Epoch: 3        Training Loss: 0.816031
Epoch: 4        Training Loss: 0.799537
Epoch: 5        Training Loss: 0.785014
Epoch: 6        Training Loss: 0.770246
Epoch: 7        Training Loss: 0.757134
Epoch: 8        Training Loss: 0.744034
Epoch: 9        Training Loss: 0.730689
Epoch: 10       Training Loss: 0.717832
Epoch: 11       Training Loss: 0.704430
Epoch: 12       Training Loss: 0.692234
Epoch: 13       Training Loss: 0.681463
Epoch: 14       Training Loss: 0.670074
Epoch: 15       Training Loss: 0.660044
Epoch: 16       Training Loss: 0.648021
Epoch: 17       Training Loss: 0.640992
Epoch: 18       Training Loss: 0.630216
Epoch: 19       Training Loss: 0.623205
Epoch: 20       Training Loss: 0.614358
```

[73]:
```
DatasetTest=TensorDataset(torch.from_numpy(x_test),torch.from_numpy(y_test))

total_num = 0
correct = 0
for x_temp,y_real in DatasetTest:
    res = model(x_temp.float())
    _, pred = torch.max(res,1)
    total_num += 1
    if int(pred) == int(y_real):
        correct += 1
```

[74]:
```
accuracy_mymodel_concat_ternary =  correct/total_num
```

[75]:
```
print('The accuracy for ternary label from my model using concat vector is ',␣
 ↪accuracy_mymodel_concat_ternary)
```

```
The accuracy for ternary label from my model using concat vector is  0.61106
```

2 google news 300 for ternary label

3 average vector

[77]:
```
import gensim.downloader as api
wv = api.load('word2vec-google-news-300')
```

[81]:
```
googel_300 = new_test[['label','review_body']]
```

[83]:
```
def vector_concat(x):
    i = 0
    n_x = 0
    length = len(x)
    if length == 0:
        return np.zeros(3000)
    a = np.zeros(300)

    while i<10:
        if n_x < length:
            if x[n_x] in wv :
                if i == 0:
                    ans = wv[x[n_x]]
                    i += 1
                else:
                    ans = np.concatenate((ans,wv[x[n_x]]), axis=None)
                    i += 1
            n_x += 1
        else:
            if i == 0:
                ans = a
                i += 1
            else:
                ans = np.concatenate((ans,a), axis=None)
                i += 1
    return ans
```

[84]:
```
googel_300['concat_vec'] = googel_300['review_body'].apply(lambda x:
 ↪vector_concat(x))
```

```
/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:1:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  """Entry point for launching an IPython kernel.
```

[85]:
```
def vector_avg(x):
    i = 0
    n = 0
    a = np.zeros(300)
    while i<len(x):
        if x[i] in wv:
            a += wv[x[i]]
        else:
            n += 1
        i = i+1
    if n == 0: return a
    else:return a/(len(x)-n)
```

[86]:
```
googel_300['avg_vector'] = googel_300['review_body'].apply(lambda x:
 ↪vector_avg(x))
```

```
/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:1:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  """Entry point for launching an IPython kernel.
```

[96]:
```
googel_300['y'] = -1
googel_300['y'] [googel_300['label']== 1] = 0
googel_300['y'] [googel_300['label']== 2] = 1
googel_300['y'] [googel_300['label']== 3] = 2
```

```
/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:2:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:3:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  This is separate from the ipykernel package so we can avoid doing imports
until
/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:4:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  after removing the cwd from sys.path.
```

[97]:
```
googel_300
```

[97]:
```
         label                                         review_body  \
1261010      2          [victorinox, product, first, word, read]
```

```
149938       2  [match, size, color, tho, color, name, also, e...
4379464      2  [advertised, add, char, broil, cooker, purchas...
28297        2                                             [yuck]
4844336      2  [bought, toaster, black, decker, name, sorry, ...
...        ...                                                ...
3962262      1  [well, made, lid, stays, tight, else, ask, mug...
1501108      1    [love, dishes, added, coffee, cups, set, years]
2533959      1  [pug, lover, like, love, cool, fridge, magnet,...
48365        1  [bought, pan, make, cake, yo, celebrate, offic...
4145163      1  [person, gave, toaster, oven, bad, review, sai...

                                                concat_vec  \
1261010  [-0.0615234375, 0.09521484375, 0.1337890625, 0...
149938   [-0.15527344, 0.025024414, 0.064941406, -0.124...
4379464  [-0.044433594, -0.017456055, -0.10986328, -0.1...
28297    [-0.09765625, 0.1728515625, 0.25, 0.3515625, -...
4844336  [0.16699219, -0.05419922, -0.087402344, 0.0196...
...                                                    ...
3962262  [-0.08251953, 0.022460938, -0.14941406, 0.0991...
1501108  [0.10302734375, -0.15234375, 0.02587890625, 0...
2533959  [-0.07128906, -0.013244629, -0.16503906, 0.197...
48365    [0.16699219, -0.05419922, -0.087402344, 0.0196...
4145163  [0.27539062, -0.24707031, 0.017211914, 0.16796...

                                                avg_vector  y
1261010  [0.09716796875, -0.015625, 0.033599853515625, ...  1
149938   [-0.073455810546875, 0.9627685546875, -0.23001...  1
4379464  [0.16021728515625, 0.9654693603515625, -0.6411...  1
28297    [-0.09765625, 0.1728515625, 0.25, 0.3515625, -...  1
4844336  [3.5324554443359375, 0.277191162109375, 2.2111...  1
...                                                    ... ..
3962262  [0.4085693359375, 1.667144775390625, -0.440200...  0
1501108  [-0.83380126953125, 0.60302734375, -0.04724121...  0
2533959  [0.7264404296875, 0.86065673828125, -0.7277069...  0
48365    [-0.018688746861049106, 0.016660417829241072, ...  0
4145163  [-0.03807124385127315, 0.061185483579282406, 0...  0

[250000 rows x 5 columns]
```

[98]:
```
X = googel_300['avg_vector']
Y = googel_300['y']

from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(X, Y, random_state = 19,␣
↪test_size = 0.2)
```

```
x_train = np.array(x_train.tolist())

y_train = np.array(y_train.tolist())

x_train = np.nan_to_num(x_train)

x_test = np.array(x_test.tolist())

y_test = np.array(y_test.tolist())

x_test = np.nan_to_num(x_test)
```

[99]:
```
# number of subprocesses to use for data loading
num_workers = 0
# how many samples per batch to load
batch_size = 20
# percentage of training set to use as validation
#valid_size = 0.2

DatasetTrain = TensorDataset(torch.from_numpy(x_train),torch.
 ↪from_numpy(y_train))

DatasetTest=TensorDataset(torch.from_numpy(x_test),torch.from_numpy(y_test))

train_loader=torch.utils.data.
 ↪DataLoader(DatasetTrain,batch_size=batch_size,shuffle=True,drop_last=True,␣
 ↪num_workers=0)

valid_loader=torch.utils.data.DataLoader(DatasetTest, batch_size=batch_size,␣
 ↪drop_last=True,num_workers=0)
```

[100]:
```
import torch.nn as nn
import torch.nn.functional as F

# define the NN architecture
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        # number of hidden nodes in each layer
        hidden_1 = 50
        hidden_2 = 10
        # linear layer (300 -> hidden_1)
        self.fc1 = nn.Linear(300, hidden_1)
        # linear layer (n_hidden -> hidden_2)
        self.fc2 = nn.Linear(hidden_1, hidden_2)
        # linear layer (n_hidden -> 10)
        self.fc3 = nn.Linear(hidden_2, 3)
        # dropout layer (p=0.2)
        # dropout prevents overfitting of data
        self.dropout = nn.Dropout(0.2)

    def forward(self, x):
        # flatten image input
        x = x.view(-1, 300)
        # add hidden layer, with relu activation function
        x = F.relu(self.fc1(x))
        # add dropout layer
        x = self.dropout(x)
        # add hidden layer, with relu activation function
        x = F.relu(self.fc2(x))
        # add dropout layer
        x = self.dropout(x)
        # add output layer
        x = self.fc3(x)
        return x

# initialize the NN
model = Net()
print(model)
```

```
Net(
  (fc1): Linear(in_features=300, out_features=50, bias=True)
  (fc2): Linear(in_features=50, out_features=10, bias=True)
  (fc3): Linear(in_features=10, out_features=3, bias=True)
  (dropout): Dropout(p=0.2, inplace=False)
)
```

[101]:
```
# specify loss function
criterion = nn.CrossEntropyLoss()

# specify optimizer
optimizer = torch.optim.SGD(model.parameters(), lr=0.01)
```

[102]:
```
# number of epochs to train the model
n_epochs = 20  # suggest training between 20-50 epochs

model.train() # prep model for training

for epoch in range(n_epochs):
    # monitor training loss
    train_loss = 0.0

    ###################
    # train the model #
    ###################
    for data, target in train_loader:
        # clear the gradients of all optimized variables
        optimizer.zero_grad()
        # forward pass: compute predicted outputs by passing inputs to the␣
↪model
        output = model(data.float())
        # calculate the loss
        loss = criterion(output, target)
        # backward pass: compute gradient of the loss with respect to model␣
↪parameters
        loss.backward()
        # perform a single optimization step (parameter update)
        optimizer.step()
        # update running training loss
        train_loss += loss.item()*data.size(0)

    # print training statistics
    # calculate average loss over an epoch
    train_loss = train_loss/len(train_loader.dataset)

    print('Epoch: {} \tTraining Loss: {:.6f}'.format(
        epoch+1,
        train_loss
        ))
```

```
Epoch: 1        Training Loss: 0.888198
Epoch: 2        Training Loss: 0.819735
Epoch: 3        Training Loss: 0.803842
Epoch: 4        Training Loss: 0.794879
Epoch: 5        Training Loss: 0.789147
Epoch: 6        Training Loss: 0.784515
Epoch: 7        Training Loss: 0.780191
Epoch: 8        Training Loss: 0.778002
Epoch: 9        Training Loss: 0.774452
Epoch: 10       Training Loss: 0.772460
Epoch: 11       Training Loss: 0.769765
Epoch: 12       Training Loss: 0.767997
Epoch: 13       Training Loss: 0.765558
Epoch: 14       Training Loss: 0.765464
Epoch: 15       Training Loss: 0.762639
Epoch: 16       Training Loss: 0.760900
Epoch: 17       Training Loss: 0.759239
Epoch: 18       Training Loss: 0.757018
Epoch: 19       Training Loss: 0.755700
Epoch: 20       Training Loss: 0.754198
```

[103]:
```
DatasetTest=TensorDataset(torch.from_numpy(x_test),torch.from_numpy(y_test))
```

```
total_num = 0
correct = 0
for x_temp,y_real in DatasetTest:
    res = model(x_temp.float())
    _, pred = torch.max(res,1)
    total_num += 1
    if int(pred) == int(y_real):
        correct += 1
```

[104]:
```
accuracy_googel_avg_ternary =  correct/total_num
```

[105]:
```
print('The accuracy for ternary label from googel 300 model using average␣
 ↪vector is ', accuracy_googel_avg_ternary)
```

```
The accuracy for ternary label from googel 300 model using average vector is
0.66946
```

4 google news 300 for ternary label

5 concat vecter

[107]:
```
X = googel_300['concat_vec']
Y = googel_300['y']

from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(X, Y, random_state = 19,␣
 ↪test_size = 0.2)

x_train = np.array(x_train.tolist())

y_train = np.array(y_train.tolist())

x_train = np.nan_to_num(x_train)

x_test = np.array(x_test.tolist())

y_test = np.array(y_test.tolist())

x_test = np.nan_to_num(x_test)
```

[108]:
```
# number of subprocesses to use for data loading
num_workers = 0
# how many samples per batch to load
batch_size = 20
# percentage of training set to use as validation
#valid_size = 0.2
```

```
DatasetTrain = TensorDataset(torch.from_numpy(x_train),torch.
 ↪from_numpy(y_train))

DatasetTest=TensorDataset(torch.from_numpy(x_test),torch.from_numpy(y_test))

train_loader=torch.utils.data.
 ↪DataLoader(DatasetTrain,batch_size=batch_size,shuffle=True,drop_last=True,␣
 ↪num_workers=0)

valid_loader=torch.utils.data.DataLoader(DatasetTest, batch_size=batch_size,␣
 ↪drop_last=True,num_workers=0)
```

[109]:
```
# define the NN architecture
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        # number of hidden nodes in each layer
        hidden_1 = 50
        hidden_2 = 10
        # linear layer (300 -> hidden_1)
        self.fc1 = nn.Linear(3000, hidden_1)
        # linear layer (n_hidden -> hidden_2)
        self.fc2 = nn.Linear(hidden_1, hidden_2)
        # linear layer (n_hidden -> 10)
        self.fc3 = nn.Linear(hidden_2, 3)
        # dropout layer (p=0.2)
        # dropout prevents overfitting of data
        self.dropout = nn.Dropout(0.2)

    def forward(self, x):
        # flatten image input
        x = x.view(-1, 3000)
        # add hidden layer, with relu activation function
        x = F.relu(self.fc1(x))
        # add dropout layer
        x = self.dropout(x)
        # add hidden layer, with relu activation function
        x = F.relu(self.fc2(x))
        # add dropout layer
        x = self.dropout(x)
        # add output layer
        x = self.fc3(x)
        return x

# initialize the NN
model = Net()
print(model)
```

```
Net(
  (fc1): Linear(in_features=3000, out_features=50, bias=True)
  (fc2): Linear(in_features=50, out_features=10, bias=True)
  (fc3): Linear(in_features=10, out_features=3, bias=True)
  (dropout): Dropout(p=0.2, inplace=False)
)
```

[110]:
```
# specify loss function
criterion = nn.CrossEntropyLoss()

# specify optimizer
optimizer = torch.optim.SGD(model.parameters(), lr=0.01)
```

[111]:
```
# number of epochs to train the model
n_epochs = 20  # suggest training between 20-50 epochs

model.train() # prep model for training

for epoch in range(n_epochs):
    # monitor training loss
    train_loss = 0.0

    ###################
    # train the model #
    ###################
    for data, target in train_loader:
        # clear the gradients of all optimized variables
        optimizer.zero_grad()
        # forward pass: compute predicted outputs by passing inputs to the␣
 ↪model
        output = model(data.float())
        # calculate the loss
        loss = criterion(output, target)
        # backward pass: compute gradient of the loss with respect to model␣
 ↪parameters
        loss.backward()
        # perform a single optimization step (parameter update)
        optimizer.step()
        # update running training loss
        train_loss += loss.item()*data.size(0)

    # print training statistics
    # calculate average loss over an epoch
    train_loss = train_loss/len(train_loader.dataset)

    print('Epoch: {} \tTraining Loss: {:.6f}'.format(
        epoch+1,
        train_loss
        ))
```

```
Epoch: 1        Training Loss: 0.949798
Epoch: 2        Training Loss: 0.872999
Epoch: 3        Training Loss: 0.850158
Epoch: 4        Training Loss: 0.833774
Epoch: 5        Training Loss: 0.818439
Epoch: 6        Training Loss: 0.805019
Epoch: 7        Training Loss: 0.789864
Epoch: 8        Training Loss: 0.773915
Epoch: 9        Training Loss: 0.758746
Epoch: 10       Training Loss: 0.745903
Epoch: 11       Training Loss: 0.731850
Epoch: 12       Training Loss: 0.718900
Epoch: 13       Training Loss: 0.705711
Epoch: 14       Training Loss: 0.692805
Epoch: 15       Training Loss: 0.682410
Epoch: 16       Training Loss: 0.672709
Epoch: 17       Training Loss: 0.661839
Epoch: 18       Training Loss: 0.652909
Epoch: 19       Training Loss: 0.642177
Epoch: 20       Training Loss: 0.636413
```

[122]:
```
total_num = 0
correct = 0
for x_temp,y_real in DatasetTest:
    res = model(x_temp.float())
    _, pred = torch.max(res,1)
    total_num += 1
    if int(pred) == int(y_real):
        correct += 1
```

[123]:
```
accuracy_google_concat_ternary =  correct/total_num
```

[124]:
```
print('The accuracy for ternary label from google 300 model using concat vector␣
 ↪is ', accuracy_google_concat_ternary)
```

```
The accuracy for ternary label from google 300 model using concat vector is
0.59616
```

6 google news 300 for binary label

7 average vector

[126]:
```
googel_300_binary = googel_300.loc[googel_300['label']!=3]
```

[133]:
```
X = googel_300_binary['avg_vector']
Y = googel_300_binary['y']

from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(X, Y, random_state = 19,␣
 ↪test_size = 0.2)

x_train = np.array(x_train.tolist())

y_train = np.array(y_train.tolist())

x_train = np.nan_to_num(x_train)

x_test = np.array(x_test.tolist())

y_test = np.array(y_test.tolist())

x_test = np.nan_to_num(x_test)
```

[134]:
```
# number of subprocesses to use for data loading
num_workers = 0
# how many samples per batch to load
batch_size = 20
# percentage of training set to use as validation
#valid_size = 0.2

DatasetTrain = TensorDataset(torch.from_numpy(x_train),torch.
 ↪from_numpy(y_train))

DatasetTest=TensorDataset(torch.from_numpy(x_test),torch.from_numpy(y_test))

train_loader=torch.utils.data.
 ↪DataLoader(DatasetTrain,batch_size=batch_size,shuffle=True,drop_last=True,␣
 ↪num_workers=0)

valid_loader=torch.utils.data.DataLoader(DatasetTest, batch_size=batch_size,␣
 ↪drop_last=True,num_workers=0)
```

[135]:
```
# define the NN architecture
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        # number of hidden nodes in each layer
        hidden_1 = 50
        hidden_2 = 10
        # linear layer (300 -> hidden_1)
        self.fc1 = nn.Linear(300, hidden_1)
        # linear layer (n_hidden -> hidden_2)
        self.fc2 = nn.Linear(hidden_1, hidden_2)
        # linear layer (n_hidden -> 10)
        self.fc3 = nn.Linear(hidden_2, 2)
        # dropout layer (p=0.2)
        # dropout prevents overfitting of data
        self.dropout = nn.Dropout(0.2)

    def forward(self, x):
        # flatten image input
        x = x.view(-1, 300)
        # add hidden layer, with relu activation function
        x = F.relu(self.fc1(x))
        # add dropout layer
        x = self.dropout(x)
        # add hidden layer, with relu activation function
        x = F.relu(self.fc2(x))
        # add dropout layer
        x = self.dropout(x)
        # add output layer
        x = self.fc3(x)
        return x

# initialize the NN
model = Net()
print(model)
```

```
Net(
  (fc1): Linear(in_features=300, out_features=50, bias=True)
  (fc2): Linear(in_features=50, out_features=10, bias=True)
  (fc3): Linear(in_features=10, out_features=2, bias=True)
  (dropout): Dropout(p=0.2, inplace=False)
)
```

[136]:
```
# specify loss function
criterion = nn.CrossEntropyLoss()

# specify optimizer
optimizer = torch.optim.SGD(model.parameters(), lr=0.01)
```

[137]:
```
# number of epochs to train the model
n_epochs = 20  # suggest training between 20-50 epochs

model.train() # prep model for training

for epoch in range(n_epochs):
    # monitor training loss
    train_loss = 0.0
    ###################
    # train the model #
    ###################
    for data, target in train_loader:
        # clear the gradients of all optimized variables
        optimizer.zero_grad()
        # forward pass: compute predicted outputs by passing inputs to the␣
↪model
        output = model(data.float())
        # calculate the loss
        loss = criterion(output, target)
        # backward pass: compute gradient of the loss with respect to model␣
↪parameters
        loss.backward()
        # perform a single optimization step (parameter update)
        optimizer.step()
        # update running training loss
        train_loss += loss.item()*data.size(0)

    # print training statistics
    # calculate average loss over an epoch
    train_loss = train_loss/len(train_loader.dataset)

    print('Epoch: {} \tTraining Loss: {:.6f}'.format(
        epoch+1,
        train_loss
        ))
```

```
Epoch: 1 	Training Loss: 0.516591
Epoch: 2 	Training Loss: 0.436113
Epoch: 3 	Training Loss: 0.418995
Epoch: 4 	Training Loss: 0.408833
Epoch: 5 	Training Loss: 0.401627
Epoch: 6 	Training Loss: 0.398218
Epoch: 7 	Training Loss: 0.394187
Epoch: 8 	Training Loss: 0.390971
Epoch: 9 	Training Loss: 0.386998
Epoch: 10 	Training Loss: 0.383374
Epoch: 11 	Training Loss: 0.381110
Epoch: 12 	Training Loss: 0.378476
Epoch: 13 	Training Loss: 0.376614
Epoch: 14 	Training Loss: 0.374721
Epoch: 15 	Training Loss: 0.373161
Epoch: 16 	Training Loss: 0.370029
Epoch: 17 	Training Loss: 0.369939
Epoch: 18 	Training Loss: 0.366963
```

```
Epoch: 19       Training Loss: 0.366842
Epoch: 20       Training Loss: 0.364092
```

[138]:
```
total_num = 0
correct = 0
for x_temp,y_real in DatasetTest:
    res = model(x_temp.float())
    _, pred = torch.max(res,1)
    total_num += 1
    if int(pred) == int(y_real):
        correct += 1
```

[139]:
```
accuracy_googel_avg_binary =  correct/total_num
```

[140]:
```
print('The accuracy for binary label from google 300 model using avg vector is␣
 ↪', accuracy_googel_avg_binary)
```

```
The accuracy for binary label from google 300 model using avg vector is
0.831475
```

8 google news 300 for binary label

concat vector

9 concat vector

[142]:
```
X = googel_300_binary['concat_vec']
Y = googel_300_binary['y']

from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(X, Y, random_state = 19,␣
 ↪test_size = 0.2)

x_train = np.array(x_train.tolist())

y_train = np.array(y_train.tolist())

x_train = np.nan_to_num(x_train)

x_test = np.array(x_test.tolist())

y_test = np.array(y_test.tolist())

x_test = np.nan_to_num(x_test)
```

[143]:
```
# number of subprocesses to use for data loading
num_workers = 0
# how many samples per batch to load
batch_size = 20
# percentage of training set to use as validation
#valid_size = 0.2

DatasetTrain = TensorDataset(torch.from_numpy(x_train),torch.
 ↪from_numpy(y_train))

DatasetTest=TensorDataset(torch.from_numpy(x_test),torch.from_numpy(y_test))

train_loader=torch.utils.data.
 ↪DataLoader(DatasetTrain,batch_size=batch_size,shuffle=True,drop_last=True,␣
 ↪num_workers=0)

valid_loader=torch.utils.data.DataLoader(DatasetTest, batch_size=batch_size,␣
 ↪drop_last=True,num_workers=0)
```

[144]:
```
# define the NN architecture
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        # number of hidden nodes in each layer
        hidden_1 = 50
        hidden_2 = 10
        # linear layer (300 -> hidden_1)
        self.fc1 = nn.Linear(3000, hidden_1)
        # linear layer (n_hidden -> hidden_2)
        self.fc2 = nn.Linear(hidden_1, hidden_2)
        # linear layer (n_hidden -> 10)
        self.fc3 = nn.Linear(hidden_2, 2)
        # dropout layer (p=0.2)
        # dropout prevents overfitting of data
        self.dropout = nn.Dropout(0.2)

    def forward(self, x):
        # flatten image input
        x = x.view(-1, 3000)
        # add hidden layer, with relu activation function
        x = F.relu(self.fc1(x))
        # add dropout layer
        x = self.dropout(x)
        # add hidden layer, with relu activation function
        x = F.relu(self.fc2(x))
        # add dropout layer
        x = self.dropout(x)
        # add output layer
        x = self.fc3(x)
        return x

# initialize the NN
model = Net()
print(model)
```

```
Net(
  (fc1): Linear(in_features=3000, out_features=50, bias=True)
  (fc2): Linear(in_features=50, out_features=10, bias=True)
  (fc3): Linear(in_features=10, out_features=2, bias=True)
  (dropout): Dropout(p=0.2, inplace=False)
)
```

[145]:
```
# specify loss function
criterion = nn.CrossEntropyLoss()

# specify optimizer
optimizer = torch.optim.SGD(model.parameters(), lr=0.01)
```

[146]:
```
# number of epochs to train the model
n_epochs = 20  # suggest training between 20-50 epochs

model.train() # prep model for training

for epoch in range(n_epochs):
    # monitor training loss
    train_loss = 0.0

    ###################
    # train the model #
    ###################
    for data, target in train_loader:
        # clear the gradients of all optimized variables
        optimizer.zero_grad()
        # forward pass: compute predicted outputs by passing inputs to the␣
 ↪model
        output = model(data.float())
        # calculate the loss
        loss = criterion(output, target)
        # backward pass: compute gradient of the loss with respect to model␣
 ↪parameters
        loss.backward()
        # perform a single optimization step (parameter update)
        optimizer.step()
        # update running training loss
        train_loss += loss.item()*data.size(0)
    # print training statistics
    # calculate average loss over an epoch
    train_loss = train_loss/len(train_loader.dataset)

    print('Epoch: {} \tTraining Loss: {:.6f}'.format(
        epoch+1,
        train_loss
        ))
```

```
Epoch: 1        Training Loss: 0.555557
Epoch: 2        Training Loss: 0.485170
Epoch: 3        Training Loss: 0.467148
Epoch: 4        Training Loss: 0.450466
Epoch: 5        Training Loss: 0.433529
Epoch: 6        Training Loss: 0.415610
Epoch: 7        Training Loss: 0.398995
Epoch: 8        Training Loss: 0.381185
Epoch: 9        Training Loss: 0.362999
Epoch: 10       Training Loss: 0.346290
Epoch: 11       Training Loss: 0.331080
Epoch: 12       Training Loss: 0.314752
Epoch: 13       Training Loss: 0.301338
Epoch: 14       Training Loss: 0.288892
Epoch: 15       Training Loss: 0.276443
Epoch: 16       Training Loss: 0.264201
Epoch: 17       Training Loss: 0.254010
Epoch: 18       Training Loss: 0.243546
Epoch: 19       Training Loss: 0.236213
Epoch: 20       Training Loss: 0.229487
```

[147]:
```
total_num = 0
correct = 0
for x_temp,y_real in DatasetTest:
    res = model(x_temp.float())
    _, pred = torch.max(res,1)
    total_num += 1
    if int(pred) == int(y_real):
        correct += 1
```

[148]:
```
accuracy_googel_concat_binary =  correct/total_num
```

[149]:
```
print('The accuracy for binary label from google 300 model using concat vector␣
 ↪is ', accuracy_googel_avg_binary)
```

```
The accuracy for binary label from google 300 model using concat vector is
0.831475
```

9.1 conclusion

The accuracy for binary label from google 300 model using avg vector is 0.831475
The accuracy for binary label from google 300 model using concat vector is 0.831475
The accuracy for ternary label from google 300 model using concat vector is 0.59616
The accuracy for ternary label from googel 300 model using average vector is 0.66946
The accuracy for ternary label from my model using concat vector is 0.61106
The accuracy for ternary label from my model using average vector is 0.6855

[ ]:

```

```

hw2(4)

October 5, 2021

0.1 RNN

0.2 binary with my model

[68]:
```
import pandas as pd
import numpy as np
import copy
import re
import torch
from sklearn.model_selection import train_test_split
import random
```

[83]:
```
import time
```

[2]:
```
import pandas as pd
test = pd.read_csv("amazon_reviews_us_Kitchen_v1_00.tsv", sep =␣
 ↪'\t',error_bad_lines = False)
test['label'] = -1
a1 = test.loc[test['star_rating']==1].sample(50000)
a2 = test.loc[test['star_rating']==2].sample(50000)
a3 = test.loc[test['star_rating']==3].sample(50000)
a4 = test.loc[test['star_rating']==4].sample(50000)
a5 = test.loc[test['star_rating']==5].sample(50000)
```

```
b'Skipping line 16148: expected 15 fields, saw 22\nSkipping line 20100: expected
15 fields, saw 22\nSkipping line 45178: expected 15 fields, saw 22\nSkipping
line 48700: expected 15 fields, saw 22\nSkipping line 63331: expected 15 fields,
saw 22\n'
b'Skipping line 86053: expected 15 fields, saw 22\nSkipping line 88858: expected
15 fields, saw 22\nSkipping line 115017: expected 15 fields, saw 22\n'
b'Skipping line 137366: expected 15 fields, saw 22\nSkipping line 139110:
expected 15 fields, saw 22\nSkipping line 165540: expected 15 fields, saw
22\nSkipping line 171813: expected 15 fields, saw 22\n'
b'Skipping line 203723: expected 15 fields, saw 22\nSkipping line 209366:
expected 15 fields, saw 22\nSkipping line 211310: expected 15 fields, saw
22\nSkipping line 246351: expected 15 fields, saw 22\nSkipping line 252364:
expected 15 fields, saw 22\n'
b'Skipping line 267003: expected 15 fields, saw 22\nSkipping line 268957:
```

```
expected 15 fields, saw 22\nSkipping line 303336: expected 15 fields, saw
22\nSkipping line 306021: expected 15 fields, saw 22\nSkipping line 311569:
expected 15 fields, saw 22\nSkipping line 316767: expected 15 fields, saw
22\nSkipping line 324009: expected 15 fields, saw 22\n'
b'Skipping line 359107: expected 15 fields, saw 22\nSkipping line 368367:
expected 15 fields, saw 22\nSkipping line 381180: expected 15 fields, saw
22\nSkipping line 390453: expected 15 fields, saw 22\n'
b'Skipping line 412243: expected 15 fields, saw 22\nSkipping line 419342:
expected 15 fields, saw 22\nSkipping line 457388: expected 15 fields, saw 22\n'
b'Skipping line 459935: expected 15 fields, saw 22\nSkipping line 460167:
expected 15 fields, saw 22\nSkipping line 466460: expected 15 fields, saw
22\nSkipping line 500314: expected 15 fields, saw 22\nSkipping line 500339:
expected 15 fields, saw 22\nSkipping line 505396: expected 15 fields, saw
22\nSkipping line 507760: expected 15 fields, saw 22\nSkipping line 513626:
expected 15 fields, saw 22\n'
b'Skipping line 527638: expected 15 fields, saw 22\nSkipping line 534209:
expected 15 fields, saw 22\nSkipping line 535687: expected 15 fields, saw
22\nSkipping line 547671: expected 15 fields, saw 22\nSkipping line 549054:
expected 15 fields, saw 22\n'
b'Skipping line 599929: expected 15 fields, saw 22\nSkipping line 604776:
expected 15 fields, saw 22\nSkipping line 609937: expected 15 fields, saw
22\nSkipping line 632059: expected 15 fields, saw 22\nSkipping line 638546:
expected 15 fields, saw 22\n'
b'Skipping line 665017: expected 15 fields, saw 22\nSkipping line 677680:
expected 15 fields, saw 22\nSkipping line 684370: expected 15 fields, saw
22\nSkipping line 720217: expected 15 fields, saw 29\n'
b'Skipping line 723240: expected 15 fields, saw 22\nSkipping line 723433:
expected 15 fields, saw 22\nSkipping line 763891: expected 15 fields, saw 22\n'
b'Skipping line 800288: expected 15 fields, saw 22\nSkipping line 802942:
expected 15 fields, saw 22\nSkipping line 803379: expected 15 fields, saw
22\nSkipping line 805122: expected 15 fields, saw 22\nSkipping line 821899:
expected 15 fields, saw 22\nSkipping line 831707: expected 15 fields, saw
22\nSkipping line 842829: expected 15 fields, saw 22\nSkipping line 843604:
expected 15 fields, saw 22\n'
b'Skipping line 863904: expected 15 fields, saw 22\nSkipping line 875655:
expected 15 fields, saw 22\nSkipping line 886796: expected 15 fields, saw
22\nSkipping line 892299: expected 15 fields, saw 22\nSkipping line 902518:
expected 15 fields, saw 22\nSkipping line 903079: expected 15 fields, saw
22\nSkipping line 912678: expected 15 fields, saw 22\n'
b'Skipping line 932953: expected 15 fields, saw 22\nSkipping line 936838:
expected 15 fields, saw 22\nSkipping line 937177: expected 15 fields, saw
22\nSkipping line 947695: expected 15 fields, saw 22\nSkipping line 960713:
expected 15 fields, saw 22\nSkipping line 965225: expected 15 fields, saw
22\nSkipping line 980776: expected 15 fields, saw 22\n'
b'Skipping line 999318: expected 15 fields, saw 22\nSkipping line 1007247:
expected 15 fields, saw 22\nSkipping line 1015987: expected 15 fields, saw
22\nSkipping line 1018984: expected 15 fields, saw 22\nSkipping line 1028671:
expected 15 fields, saw 22\n'
```

```
b'Skipping line 1063360: expected 15 fields, saw 22\nSkipping line 1066195:
expected 15 fields, saw 22\nSkipping line 1066578: expected 15 fields, saw
22\nSkipping line 1066869: expected 15 fields, saw 22\nSkipping line 1068809:
expected 15 fields, saw 22\nSkipping line 1069505: expected 15 fields, saw
22\nSkipping line 1087983: expected 15 fields, saw 22\nSkipping line 1108184:
expected 15 fields, saw 22\n'
b'Skipping line 1118137: expected 15 fields, saw 22\nSkipping line 1142723:
expected 15 fields, saw 22\nSkipping line 1152492: expected 15 fields, saw
22\nSkipping line 1156947: expected 15 fields, saw 22\nSkipping line 1172563:
expected 15 fields, saw 22\n'
b'Skipping line 1209254: expected 15 fields, saw 22\nSkipping line 1212966:
expected 15 fields, saw 22\nSkipping line 1236533: expected 15 fields, saw
22\nSkipping line 1237598: expected 15 fields, saw 22\n'
b'Skipping line 1273825: expected 15 fields, saw 22\nSkipping line 1277898:
expected 15 fields, saw 22\nSkipping line 1283654: expected 15 fields, saw
22\nSkipping line 1286023: expected 15 fields, saw 22\nSkipping line 1302038:
expected 15 fields, saw 22\nSkipping line 1305179: expected 15 fields, saw 22\n'
b'Skipping line 1326022: expected 15 fields, saw 22\nSkipping line 1338120:
expected 15 fields, saw 22\nSkipping line 1338503: expected 15 fields, saw
22\nSkipping line 1338849: expected 15 fields, saw 22\nSkipping line 1341513:
expected 15 fields, saw 22\nSkipping line 1346493: expected 15 fields, saw
22\nSkipping line 1373127: expected 15 fields, saw 22\n'
b'Skipping line 1389508: expected 15 fields, saw 22\nSkipping line 1413951:
expected 15 fields, saw 22\nSkipping line 1433626: expected 15 fields, saw 22\n'
b'Skipping line 1442698: expected 15 fields, saw 22\nSkipping line 1472982:
expected 15 fields, saw 22\nSkipping line 1482282: expected 15 fields, saw
22\nSkipping line 1487808: expected 15 fields, saw 22\nSkipping line 1500636:
expected 15 fields, saw 22\n'
b'Skipping line 1511479: expected 15 fields, saw 22\nSkipping line 1532302:
expected 15 fields, saw 22\nSkipping line 1537952: expected 15 fields, saw
22\nSkipping line 1539951: expected 15 fields, saw 22\nSkipping line 1541020:
expected 15 fields, saw 22\n'
b'Skipping line 1594217: expected 15 fields, saw 22\nSkipping line 1612264:
expected 15 fields, saw 22\nSkipping line 1615907: expected 15 fields, saw
22\nSkipping line 1621859: expected 15 fields, saw 22\n'
b'Skipping line 1653542: expected 15 fields, saw 22\nSkipping line 1671537:
expected 15 fields, saw 22\nSkipping line 1672879: expected 15 fields, saw
22\nSkipping line 1674523: expected 15 fields, saw 22\nSkipping line 1677355:
expected 15 fields, saw 22\nSkipping line 1703907: expected 15 fields, saw 22\n'
b'Skipping line 1713046: expected 15 fields, saw 22\nSkipping line 1722982:
expected 15 fields, saw 22\nSkipping line 1727290: expected 15 fields, saw
22\nSkipping line 1744482: expected 15 fields, saw 22\n'
b'Skipping line 1803858: expected 15 fields, saw 22\nSkipping line 1810069:
expected 15 fields, saw 22\nSkipping line 1829751: expected 15 fields, saw
22\nSkipping line 1831699: expected 15 fields, saw 22\n'
b'Skipping line 1863131: expected 15 fields, saw 22\nSkipping line 1867917:
expected 15 fields, saw 22\nSkipping line 1874790: expected 15 fields, saw
22\nSkipping line 1879952: expected 15 fields, saw 22\nSkipping line 1880501:
```

```
expected 15 fields, saw 22\nSkipping line 1886655: expected 15 fields, saw
22\nSkipping line 1887888: expected 15 fields, saw 22\nSkipping line 1894286:
expected 15 fields, saw 22\nSkipping line 1895400: expected 15 fields, saw 22\n'
b'Skipping line 1904040: expected 15 fields, saw 22\nSkipping line 1907604:
expected 15 fields, saw 22\nSkipping line 1915739: expected 15 fields, saw
22\nSkipping line 1921514: expected 15 fields, saw 22\nSkipping line 1939428:
expected 15 fields, saw 22\nSkipping line 1944342: expected 15 fields, saw
22\nSkipping line 1949699: expected 15 fields, saw 22\nSkipping line 1961872:
expected 15 fields, saw 22\n'
b'Skipping line 1968846: expected 15 fields, saw 22\nSkipping line 1999941:
expected 15 fields, saw 22\nSkipping line 2001492: expected 15 fields, saw
22\nSkipping line 2011204: expected 15 fields, saw 22\nSkipping line 2025295:
expected 15 fields, saw 22\n'
b'Skipping line 2041266: expected 15 fields, saw 22\nSkipping line 2073314:
expected 15 fields, saw 22\nSkipping line 2080133: expected 15 fields, saw
22\nSkipping line 2088521: expected 15 fields, saw 22\n'
b'Skipping line 2103490: expected 15 fields, saw 22\nSkipping line 2115278:
expected 15 fields, saw 22\nSkipping line 2153174: expected 15 fields, saw
22\nSkipping line 2161731: expected 15 fields, saw 22\n'
b'Skipping line 2165250: expected 15 fields, saw 22\nSkipping line 2175132:
expected 15 fields, saw 22\nSkipping line 2206817: expected 15 fields, saw
22\nSkipping line 2215848: expected 15 fields, saw 22\nSkipping line 2223811:
expected 15 fields, saw 22\n'
b'Skipping line 2257265: expected 15 fields, saw 22\nSkipping line 2259163:
expected 15 fields, saw 22\nSkipping line 2263291: expected 15 fields, saw 22\n'
b'Skipping line 2301943: expected 15 fields, saw 22\nSkipping line 2304371:
expected 15 fields, saw 22\nSkipping line 2306015: expected 15 fields, saw
22\nSkipping line 2312186: expected 15 fields, saw 22\nSkipping line 2314740:
expected 15 fields, saw 22\nSkipping line 2317754: expected 15 fields, saw 22\n'
b'Skipping line 2383514: expected 15 fields, saw 22\n'
b'Skipping line 2449763: expected 15 fields, saw 22\n'
b'Skipping line 2589323: expected 15 fields, saw 22\n'
b'Skipping line 2775036: expected 15 fields, saw 22\n'
b'Skipping line 2935174: expected 15 fields, saw 22\n'
b'Skipping line 3078830: expected 15 fields, saw 22\n'
b'Skipping line 3123091: expected 15 fields, saw 22\n'
b'Skipping line 3185533: expected 15 fields, saw 22\n'
b'Skipping line 4150395: expected 15 fields, saw 22\n'
b'Skipping line 4748401: expected 15 fields, saw 22\n'
```

[3]:
```
new_test = pd.concat([a1,a2,a3,a4,a5])
```

[4]:
```
new_test['lable'] = 1
```

[5]:
```
new_test.label[test.star_rating>3] = 1
new_test.label[test.star_rating<3] = 2
new_test.label[test.star_rating==3] = 3
new_test = new_test[['label','review_body']]
```

```
/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:1:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  """Entry point for launching an IPython kernel.
/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:2:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:3:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  This is separate from the ipykernel package so we can avoid doing imports
until
```

[6]:
```
import gensim
model_1 = gensim.models.Word2Vec.load('save')
```

[7]:
```
#data clearing
new_test['review_body'] = new_test['review_body'].str.lower()

def tag(x):
    return re.sub('<.*?>','',str(x))
new_test['review_body'] = new_test['review_body'].apply(lambda x:tag(x))

def url(x):
    return re.sub('(https?|ftp|file)://[-A-Za-z0-9+&@#/%?=~_|!:,.;
↪]+[-A-Za-z0-9+&@#/%=~_|]','',str(x))

new_test['review_body'] = new_test['review_body'].apply(lambda x:url(x))

import contractions
new_test['review_body'] = new_test['review_body'].apply(lambda x:contractions.
↪fix(x))

def non_alphabetical(x):
    return re.sub('[^a-zA-Z\s]','',str(x))
new_test['review_body'] = new_test['review_body'].apply(lambda x:
↪non_alphabetical(x))
def extra_space(x):
    return re.sub( ' +',' ',str(x))
new_test['review_body'] = new_test['review_body'].apply(lambda x:extra_space(x))
```

[8]:
```
#Pre-processing

import nltk
from nltk.corpus import stopwords
nltk.download('stopwords')
stop_words_set = set(stopwords.words('english'))
from nltk import word_tokenize, pos_tag

def stop_words(x):
    word_tokens = word_tokenize(x)
    temp = []
    for i in word_tokens:
        if i not in stop_words_set:
            temp.append(i)
    return temp
new_test['review_body'] = new_test['review_body'].apply(lambda x:stop_words(x))
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     /Users/chenlin/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

[10]:
```
new_test['y'] = -1
```

[11]:
```
new_test['y'][new_test['label'] == 1] =0
```

```
/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:1:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  """Entry point for launching an IPython kernel.
```

[12]:
```
new_test['y'][new_test['label'] == 2] =1
```

```
/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:1:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  """Entry point for launching an IPython kernel.
```

[13]:
```
new_test['y'][new_test['label'] == 3] =2
```

```
/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:1:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  """Entry point for launching an IPython kernel.
```

[14]:
```
binary = new_test.loc[new_test['y']!=2]
```

[15]:
```
binary
```

[15]:
```
         label                                        review_body  y
1674978      2                           [wrong, sizehad, return]  1
3055998      2  [bought, replace, old, mandolins, slicer, cut,...  1
4165773      2  [looking, forward, getting, handy, new, kitche...  1
2168375      2  [item, cheap, could, made, paper, inside, croo...  1
2627653      2  [got, year, old, boyfriend, awesome, girlfrien...  1
...        ...                                                ... ..
4171764      1  [made, amazing, popcorn, hot, air, corn, poppe...  0
4272127      1  [bought, bagel, slicer, kids, would, cut, slic...  0
1302570      1                                     [great, deal]  0
2914686      1  [original, quality, quart, bowl, kitchenaid, r...  0
3063914      1  [straws, great, little, kids, keep, chewing, c...  0

[200000 rows x 3 columns]
```

[210]:
```
def mymodel_reviewtensor(x):
    max_review = 50
    ### create a matrix
    review = torch.zeros(max_review,1,300)
    pad = torch.zeros(300)
    index = 0
    for i in x:
        if i in model_1.wv and index < 50:
            vector = model_1.wv [i]
            review[index][0] = torch.from_numpy(vector)
            index += 1
        else:
            continue
    if index < 50:
        review[index][0] = pad
    review = torch.nan_to_num(review)
    return review
```

[213]:
```
X = binary['review_body']
Y = binary['y']

from sklearn.model_selection import train_test_split
```

```
x_train, x_test, y_train, y_test = train_test_split(X, Y, random_state = 19,␣
 ↪test_size = 0.2)

x_train = np.array(x_train.tolist())

y_train = np.array(y_train.tolist())

x_train = np.nan_to_num(x_train)

x_test = np.array(x_test.tolist())

y_test = np.array(y_test.tolist())

x_test = np.nan_to_num(x_test)
```

```
/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:8:
VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences
(which is a list-or-tuple of lists-or-tuples-or ndarrays with different lengths
or shapes) is deprecated. If you meant to do this, you must specify
'dtype=object' when creating the ndarray

/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:14:
VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences
(which is a list-or-tuple of lists-or-tuples-or ndarrays with different lengths
or shapes) is deprecated. If you meant to do this, you must specify
'dtype=object' when creating the ndarray
```

[214]:
```
from torch.utils.data import DataLoader, Dataset
from torch.utils.data import TensorDataset, DataLoader
```

[215]:
```
import torch.nn as nn

class RNN(nn.Module):
    def __init__(self, input_size, hidden_size, output_size):
        super(RNN, self).__init__()

        self.hidden_size = hidden_size

        self.i2h = nn.Linear(input_size + hidden_size, hidden_size)
        self.i2o = nn.Linear(input_size + hidden_size, output_size)
        self.softmax = nn.LogSoftmax(dim=1)

    def forward(self, input, hidden):
        combined = torch.cat((input, hidden), 1)
        hidden = self.i2h(combined)
        output = self.i2o(combined)
        output = self.softmax(output)
        return output, hidden

    def initHidden(self):
        return torch.zeros(1, self.hidden_size)

rnn = RNN(300, 50, 2)
```

[222]:
```
learning_rate = 0.005 # If you set this too high, it might explode. If too low,␣
↪it might not learn

def train(category_tensor, line_tensor):
    hidden = rnn.initHidden()

    rnn.zero_grad()

    for i in range(line_tensor.size()[0]):
        output, hidden = rnn(line_tensor[i], hidden)

    loss = criterion(output, category_tensor)
    loss.backward()
    torch.nn.utils.clip_grad_norm_(rnn.parameters(),0.5)
    # Add parameters' gradients to their values, multiplied by learning rate
    for p in rnn.parameters():
        p.data.add_(p.grad.data, alpha=-learning_rate)

    return output, loss.item()
```

[216]:
```
input = mymodel_reviewtensor(binary['review_body'].iloc[4])
hidden = torch.zeros(1, 50)
```

[217]:
```
def output_predict_y(output):
    top_n, top_i = output.topk(1)
    y = top_i[0].item()
    return y
```

[218]:
```
criterion = nn.NLLLoss()
```

[219]:
```
main_data = list(zip(x_train,y_train))
```

[220]:
```
# Keep track of losses for plotting
current_loss = 0

def timeSince(since):
    now = time.time()
    s = now - since
    m = math.floor(s / 60)
    s -= m * 60
    return '%dm %ds' % (m, s)
```

```
start = time.time()
ii=0
correct = 0
for x_i,y_i in main_data:
    line_tensor = mymodel_reviewtensor(x_i)
    y_i = torch.tensor([y_i],dtype =torch.long)
    output, loss = train(y_i, line_tensor)
    current_loss += loss
    ii += 1
    if ii %2500 == 0 :
        print(ii,"current avg lose is ",current_loss/ii,'time is ',time.time()␣
 ↪- start)
```

```
2500 current avg lose is  0.6935066653609275 time is  18.736644983291626
5000 current avg lose is  0.6938000483036041 time is  39.694331884384155
7500 current avg lose is  0.6940343046387036 time is  59.080775022506714
10000 current avg lose is  0.6943283890724182 time is  75.36496996879578
12500 current avg lose is  0.6985913027977944 time is  93.71864986419678
15000 current avg lose is  0.706857127382358 time is  113.98635816574097
17500 current avg lose is  0.7157444509387016 time is  133.35544085502625
20000 current avg lose is  0.719268296802044 time is  151.9071388244629
22500 current avg lose is  0.7231831370088789 time is  167.7384889125824
25000 current avg lose is  0.7257247288429737 time is  185.27695298194885
27500 current avg lose is  0.7280170371738347 time is  204.13538002967834
30000 current avg lose is  0.7308424241900444 time is  220.9835648536682
32500 current avg lose is  0.7335039394947199 time is  237.65777802467346
35000 current avg lose is  0.736566359599999 time is  254.8173429965973
37500 current avg lose is  0.7401764461016654 time is  273.89567017555237
40000 current avg lose is  0.7440658328901977 time is  291.83909583091736
42500 current avg lose is  0.7491485424865695 time is  311.85006403923035
45000 current avg lose is  0.7534946585579051 time is  328.0279891490936
47500 current avg lose is  0.7661504434654587 time is  350.39413690567017
50000 current avg lose is  0.8168986825359081 time is  373.1792631149292
52500 current avg lose is  0.8665489911493953 time is  389.9557740688324
55000 current avg lose is  0.9174279366305808 time is  406.3925199508667
57500 current avg lose is  0.966856176473939 time is  423.01318311691284
60000 current avg lose is  1.0062675636826433 time is  444.79478883743286
62500 current avg lose is  1.0448091871641731 time is  461.1006848812103
65000 current avg lose is  1.0810720270097345 time is  477.6718201637268
67500 current avg lose is  1.108496844244045 time is  493.89629912376404
70000 current avg lose is  1.1319343567918965 time is  510.42331409454346
72500 current avg lose is  1.153895557646119 time is  527.0030469894409
75000 current avg lose is  1.173600270377148 time is  543.5604491233826
77500 current avg lose is  1.1896666699030212 time is  560.0479960441589
80000 current avg lose is  1.2016800544987478 time is  577.5987119674683
82500 current avg lose is  1.2178468038481534 time is  594.5318260192871
85000 current avg lose is  1.230443255828 time is  612.9297821521759
```

```
87500 current avg lose is  1.242328551959492 time is  629.5187909603119
90000 current avg lose is  1.2475180744281704 time is  646.3995950222015
92500 current avg lose is  1.2558322203072503 time is  662.9913151264191
95000 current avg lose is  1.264708187838483 time is  679.7814490795135
97500 current avg lose is  1.2715983043432477 time is  696.2957010269165
100000 current avg lose is  1.278894634093908 time is  712.9258980751038
102500 current avg lose is  1.2852110577896705 time is  736.9921770095825
105000 current avg lose is  1.288830072370764 time is  757.801276922226
107500 current avg lose is  1.2934220022469416 time is  774.7577559947968
110000 current avg lose is  1.297304562862478 time is  791.1045119762421
112500 current avg lose is  1.3010337249147075 time is  806.8263080120087
115000 current avg lose is  1.305661886688165 time is  823.0702760219574
117500 current avg lose is  1.308220576642449 time is  839.2411432266235
120000 current avg lose is  1.309006782510335 time is  855.0096199512482
122500 current avg lose is  1.3090750498813901 time is  871.5667219161987
125000 current avg lose is  1.3101933612420882 time is  890.1921060085297
127500 current avg lose is  1.3123350381581371 time is  908.104542016983
130000 current avg lose is  1.3130587492290975 time is  926.8979659080505
132500 current avg lose is  1.3161465093867415 time is  946.0703160762787
135000 current avg lose is  1.3170001468154895 time is  965.9388539791107
137500 current avg lose is  1.3180350640610174 time is  984.0711009502411
140000 current avg lose is  1.3198708004461168 time is  1004.2074661254883
142500 current avg lose is  1.321299134416865 time is  1021.8995621204376
145000 current avg lose is  1.320986983209793 time is  1038.6785039901733
147500 current avg lose is  1.3219430335184217 time is  1054.1812980175018
150000 current avg lose is  1.3217860330345463 time is  1069.7205078601837
152500 current avg lose is  1.322496251604019 time is  1084.9681990146637
155000 current avg lose is  1.322955055061476 time is  1100.6275489330292
157500 current avg lose is  1.3227956603052677 time is  1115.9846510887146
160000 current avg lose is  1.3229543010432343 time is  1131.2285211086273
```

[223]:
```python
n_label = 2
confusion = torch.zeros(n_label, n_label)

def evaluate(line_tensor):
    hidden = rnn.initHidden()

    for i in range(line_tensor.size()[0]):
        output, hidden = rnn(line_tensor[i], hidden)



    return output

for x1,y1 in test_data:

    label_tensor = torch.tensor([y1],dtype =torch.long)
    review_tensor = mymodel_reviewtensor(x1)
    output = evaluate(review_tensor)

    label_pred = output_predict_y(output)
    confusion[y1][label_pred] += 1
```

[224]:
```
accuracy_my_model_binary = confusion.trace()/confusion.sum()
```

[225]:
```
print("the accuracy for my model with RNN binary is ",accuracy_my_model_binary)
```

```
the accuracy for my model with RNN binary is  tensor(0.8205)
```

0.3 binary RNN with google 300 model

[227]:
```
import gensim.downloader as api

opm = api.load("word2vec-google-news-300")
```

[228]:
```
def review2tensor(x):
    max_review = 50
    ### create a matrix
    review = torch.zeros(max_review,1,300)
    pad = torch.zeros(300)
    index = 0
    for i in x:
        if i in opm and index < 50:
            vector = opm [i]
            review[index][0] = torch.from_numpy(vector)
            index += 1
        else:
            continue
    if index < 50:
        review[index][0] = pad
    review = torch.nan_to_num(review)
    return review
```

[229]:
```
review2tensor('the').size()[0]
```

[229]:
```
50
```

[230]:
```
review2tensor(binary['review_body'].iloc[49])[10]
```

[230]:
```
tensor([[0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
0., 0., 0., 0., 0., 0.,
         0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
0., 0., 0., 0., 0., 0.,
         0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
0., 0., 0., 0., 0., 0.,
         0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
0., 0., 0., 0., 0., 0.,
         0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
0., 0., 0., 0., 0., 0.,
         0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
0., 0., 0., 0., 0., 0.,
         0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
0., 0., 0., 0., 0., 0.,
         0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
0., 0., 0., 0., 0., 0.,
         0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
0., 0., 0., 0., 0., 0.,
         0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
0., 0., 0., 0., 0., 0.,
         0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
0., 0., 0., 0., 0., 0.,
         0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
0., 0., 0., 0., 0., 0.,
         0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.]])
```

[231]:
```
learning_rate = 0.005 # If you set this too high, it might explode. If too low,␣
 ↪it might not learn

def train(category_tensor, line_tensor):
    hidden = rnn.initHidden()

    rnn.zero_grad()

    for i in range(line_tensor.size()[0]):
        output, hidden = rnn(line_tensor[i], hidden)

    loss = criterion(output, category_tensor)
    loss.backward()
    torch.nn.utils.clip_grad_norm_(rnn.parameters(),0.5)
    # Add parameters' gradients to their values, multiplied by learning rate
    for p in rnn.parameters():
        p.data.add_(p.grad.data, alpha=-learning_rate)

    return output, loss.item()
```

[232]:
```
X = binary['review_body']
Y = binary['y']

from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(X, Y, random_state = 19,␣
 ↪test_size = 0.2)

x_train = np.array(x_train.tolist())
```

```
y_train = np.array(y_train.tolist())

x_train = np.nan_to_num(x_train)

x_test = np.array(x_test.tolist())

y_test = np.array(y_test.tolist())

x_test = np.nan_to_num(x_test)
```

```
/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:8:
VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences
(which is a list-or-tuple of lists-or-tuples-or ndarrays with different lengths
or shapes) is deprecated. If you meant to do this, you must specify
'dtype=object' when creating the ndarray

/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:14:
VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences
(which is a list-or-tuple of lists-or-tuples-or ndarrays with different lengths
or shapes) is deprecated. If you meant to do this, you must specify
'dtype=object' when creating the ndarray
```

[233]:
```
from torch.utils.data import DataLoader, Dataset
from torch.utils.data import TensorDataset, DataLoader
```

[234]:
```
import torch.nn as nn

class RNN(nn.Module):
    def __init__(self, input_size, hidden_size, output_size):
        super(RNN, self).__init__()

        self.hidden_size = hidden_size

        self.i2h = nn.Linear(input_size + hidden_size, hidden_size)
        self.i2o = nn.Linear(input_size + hidden_size, output_size)
        self.softmax = nn.LogSoftmax(dim=1)

    def forward(self, input, hidden):
        combined = torch.cat((input, hidden), 1)
        hidden = self.i2h(combined)
        output = self.i2o(combined)
        output = self.softmax(output)
        return output, hidden

    def initHidden(self):
        return torch.zeros(1, self.hidden_size)
```

```
rnn = RNN(300, 50, 2)
```

[235]:
```
input = review2tensor(binary['review_body'].iloc[4])
hidden = torch.zeros(1, 50)
```

[236]:
```
output, next_hidden = rnn(input[0], hidden)
```

[237]:
```
def output_predict_y(output):
    top_n, top_i = output.topk(1)
    y = top_i[0].item()
    return y
```

[238]:
```
yfromdata(output)
```

[238]:
```
0
```

[239]:
```
criterion = nn.NLLLoss()
```

[240]:
```
learning_rate = 0.005 # If you set this too high, it might explode. If too low,␣
 ↪it might not learn

def train(category_tensor, line_tensor):
    hidden = rnn.initHidden()

    rnn.zero_grad()

    for i in range(line_tensor.size()[0]):
        output, hidden = rnn(line_tensor[i], hidden)

    loss = criterion(output, category_tensor)
    loss.backward()
    torch.nn.utils.clip_grad_norm_(rnn.parameters(),0.5)
    # Add parameters' gradients to their values, multiplied by learning rate
    for p in rnn.parameters():
        p.data.add_(p.grad.data, alpha=-learning_rate)

    return output, loss.item()
```

[241]:
```
from torch.utils.data import DataLoader, Dataset
from torch.utils.data import TensorDataset, DataLoader
```

[242]:
```
main_data = list(zip(x_train,y_train))
```

[ ]:

[243]:
```
# Keep track of losses for plotting
current_loss = 0

def timeSince(since):
    now = time.time()
    s = now - since
    m = math.floor(s / 60)
    s -= m * 60
    return '%dm %ds' % (m, s)

start = time.time()
ii=0
correct = 0
for x_i,y_i in main_data:
    line_tensor = review2tensor(x_i)
    y_i = torch.tensor([y_i],dtype =torch.long)
    output, loss = train(y_i, line_tensor)
    current_loss += loss
    ii += 1
    predict_y = yfromdata(output)
    if y_i == predict_y:
        correct +=1
    if ii %2500 == 0 :
        print(ii,"current avg lose is ",current_loss/ii,'time is ',time.time()␣
 ↪- start)
```

```
2500 current avg lose is  0.6946149057388306 time is  22.118892192840576
5000 current avg lose is  0.6957750918149949 time is  39.81695795059204
7500 current avg lose is  0.6966212011496226 time is  56.48860502243042
10000 current avg lose is  0.69695377741158 time is  73.53538584709167
12500 current avg lose is  0.7033569383120537 time is  93.37862300872803
15000 current avg lose is  0.7123965916832288 time is  114.35654783248901
17500 current avg lose is  0.72068262171575 time is  131.81934809684753
20000 current avg lose is  0.7233350892782211 time is  148.8105869293213
22500 current avg lose is  0.7266608888400925 time is  166.8097460269928
25000 current avg lose is  0.7288242967271805 time is  184.70361185073853
27500 current avg lose is  0.7300440458850427 time is  203.6105260848999
30000 current avg lose is  0.731794490335385 time is  221.02341198921204
32500 current avg lose is  0.7336278566360473 time is  238.65603375434875
35000 current avg lose is  0.7356945490990366 time is  257.80336809158325
37500 current avg lose is  0.7380929750283559 time is  274.521723985672
40000 current avg lose is  0.7402321053527295 time is  296.0276849269867
42500 current avg lose is  0.7424549901247025 time is  316.3202168941498
45000 current avg lose is  0.7431025891827212 time is  340.14389514923096
47500 current avg lose is  0.7456497730669223 time is  363.267657995224
50000 current avg lose is  0.7482634851193428 time is  385.98622012138367
52500 current avg lose is  0.749999521695716 time is  403.04541087150574
55000 current avg lose is  0.752858642465418 time is  423.7475640773773
57500 current avg lose is  0.7547070155126893 time is  442.65362906455994
60000 current avg lose is  0.7624538702889035 time is  458.9637677669525
62500 current avg lose is  0.8110369722896015 time is  475.06502199172974
65000 current avg lose is  0.8548702853062324 time is  491.0245599746704
```

```
67500 current avg lose is  0.89210491614323 time is  506.9926769733429
70000 current avg lose is  0.930156318563192 time is  524.7869830131531
72500 current avg lose is  0.9642878387305945 time is  541.1039109230042
75000 current avg lose is  0.9971683377809163 time is  557.5870459079742
77500 current avg lose is  1.0274707935839082 time is  573.6545059680939
80000 current avg lose is  1.0546235030927384 time is  590.199334859848
82500 current avg lose is  1.0773018713259848 time is  605.8982841968536
85000 current avg lose is  1.0952199241803813 time is  624.7830460071564
87500 current avg lose is  1.115322752236526 time is  641.2154159545898
90000 current avg lose is  1.1270416864774726 time is  657.0917508602142
92500 current avg lose is  1.1424993010932045 time is  676.2039058208466
95000 current avg lose is  1.1557575995815697 time is  706.4415879249573
97500 current avg lose is  1.1680131034833308 time is  732.3915359973907
100000 current avg lose is  1.179174193751666 time is  753.312472820282
102500 current avg lose is  1.188754206022267 time is  775.7434687614441
105000 current avg lose is  1.1963509835230297 time is  804.4075999259949
107500 current avg lose is  1.2049114987721077 time is  829.9771678447723
110000 current avg lose is  1.2120652044772853 time is  853.4426510334015
112500 current avg lose is  1.2200559295490918 time is  870.8683519363403
115000 current avg lose is  1.2276067807577682 time is  891.5767140388489
117500 current avg lose is  1.2346387695620553 time is  915.5201261043549
120000 current avg lose is  1.2397771903260764 time is  939.691565990448
122500 current avg lose is  1.2444561228367543 time is  958.2355720996857
125000 current avg lose is  1.2485308202114516 time is  974.4230170249939
127500 current avg lose is  1.2530430875246032 time is  990.3677740097046
130000 current avg lose is  1.256757725178107 time is  1006.3707869052887
132500 current avg lose is  1.2629742058953475 time is  1022.3463580608368
135000 current avg lose is  1.2670394796028586 time is  1039.089103937149
137500 current avg lose is  1.2707857760186643 time is  1059.609256029129
140000 current avg lose is  1.2759001230743308 time is  1081.2359149456024
142500 current avg lose is  1.2796641287027934 time is  1097.3742470741272
145000 current avg lose is  1.2841903255463907 time is  1113.232172012329
147500 current avg lose is  1.288557664455274 time is  1130.874839067459
150000 current avg lose is  1.292357830940174 time is  1147.2331640720367
152500 current avg lose is  1.295247374925193 time is  1163.402438879013
155000 current avg lose is  1.298593524060938 time is  1179.3178389072418
157500 current avg lose is  1.3025559112659642 time is  1195.6890890598297
160000 current avg lose is  1.3047873583760832 time is  1212.0009269714355
```

[244]:
```
loss
```

[244]:
```
0.0005920564290136099
```

[245]:
```
current_loss
```

[245]:
```
208765.9773401733
```

[246]:
```
test_data = list(zip(x_test,y_test))
```

[247]:
```
n_label = 2
confusion = torch.zeros(n_label, n_label)

def evaluate(line_tensor):
    hidden = rnn.initHidden()

    for i in range(line_tensor.size()[0]):
        output, hidden = rnn(line_tensor[i], hidden)



    return output

for x1,y1 in test_data:

    label_tensor = torch.tensor([y1],dtype =torch.long)
    review_tensor = review2tensor(x1)
    output = evaluate(review_tensor)

    label_pred = output_predict_y(output)
    confusion[y1][label_pred] += 1
```

[248]:
```
accuracy_googel_300_binary = confusion.trace()/confusion.sum()
```

[249]:
```
print("the accuracy for googel 300 model with RNN binary is␣
 ↪",accuracy_googel_300_binary)
```

```
the accuracy for googel 300 model with RNN binary is  tensor(0.7890)
```

0.4 trinary with my model RNN

[250]:
```
def mymodel_reviewtensor(x):
    max_review = 50
    ### create a matrix
    review = torch.zeros(max_review,1,300)
    pad = torch.zeros(300)
    index = 0
    for i in x:
        if i in model_1.wv and index < 50:
            vector = model_1.wv [i]
            review[index][0] = torch.from_numpy(vector)
            index += 1
        else:
            continue
    if index < 50:
        review[index][0] = pad
    review = torch.nan_to_num(review)
    return review
```

[252]:
```
X = new_test['review_body']
Y = new_test['y']

from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(X, Y, random_state = 19,␣
 ↪test_size = 0.2)

x_train = np.array(x_train.tolist())

y_train = np.array(y_train.tolist())

x_train = np.nan_to_num(x_train)

x_test = np.array(x_test.tolist())

y_test = np.array(y_test.tolist())

x_test = np.nan_to_num(x_test)
```

```
/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:8:
VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences
(which is a list-or-tuple of lists-or-tuples-or ndarrays with different lengths
or shapes) is deprecated. If you meant to do this, you must specify
'dtype=object' when creating the ndarray

/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:14:
VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences
(which is a list-or-tuple of lists-or-tuples-or ndarrays with different lengths
or shapes) is deprecated. If you meant to do this, you must specify
'dtype=object' when creating the ndarray
```

[254]:
```
import torch.nn as nn

class RNN(nn.Module):
    def __init__(self, input_size, hidden_size, output_size):
        super(RNN, self).__init__()

        self.hidden_size = hidden_size

        self.i2h = nn.Linear(input_size + hidden_size, hidden_size)
        self.i2o = nn.Linear(input_size + hidden_size, output_size)
        self.softmax = nn.LogSoftmax(dim=1)

    def forward(self, input, hidden):
        combined = torch.cat((input, hidden), 1)
        hidden = self.i2h(combined)
        output = self.i2o(combined)
        output = self.softmax(output)
        return output, hidden

    def initHidden(self):
        return torch.zeros(1, self.hidden_size)

rnn = RNN(300, 50, 3)
```

[255]:
```
input = review2tensor(binary['review_body'].iloc[4])
hidden = torch.zeros(1, 50)
```

[256]:
```
criterion = nn.NLLLoss()
```

[257]:
```
learning_rate = 0.005 # If you set this too high, it might explode. If too low,␣
 ↪it might not learn

def train(category_tensor, line_tensor):
    hidden = rnn.initHidden()

    rnn.zero_grad()

    for i in range(line_tensor.size()[0]):
        output, hidden = rnn(line_tensor[i], hidden)

    loss = criterion(output, category_tensor)
    loss.backward()
    torch.nn.utils.clip_grad_norm_(rnn.parameters(),0.5)
    # Add parameters' gradients to their values, multiplied by learning rate
    for p in rnn.parameters():
        p.data.add_(p.grad.data, alpha=-learning_rate)

    return output, loss.item()
```

[258]:
```
main_data = list(zip(x_train,y_train))
```

[259]:
```
# Keep track of losses for plotting
current_loss = 0

def timeSince(since):
    now = time.time()
    s = now - since
    m = math.floor(s / 60)
    s -= m * 60
    return '%dm %ds' % (m, s)

start = time.time()
ii=0
```

```
correct = 0
for x_i,y_i in main_data:
    line_tensor = review2tensor(x_i)
    y_i = torch.tensor([y_i],dtype =torch.long)
    output, loss = train(y_i, line_tensor)
    current_loss += loss
    ii += 1
    if ii %2500 == 0 :
        print(ii,"current avg lose is ",current_loss/ii,'time is ',time.time()␣
 ↪- start)
```

```
2500 current avg lose is  1.058883339238167 time is  32.93949484825134
5000 current avg lose is  1.0648978867650032 time is  69.53135681152344
7500 current avg lose is  1.0671727425018946 time is  101.4669828414917
10000 current avg lose is  1.070502868539095 time is  137.4072709083557
12500 current avg lose is  1.0674601861286164 time is  171.49279308319092
15000 current avg lose is  1.0676490481615066 time is  212.4065079689026
17500 current avg lose is  1.0673559479747499 time is  238.32458591461182
20000 current avg lose is  1.0682490519195795 time is  265.84134697914124
22500 current avg lose is  1.0685526640786065 time is  296.73723793029785
25000 current avg lose is  1.067924862034321 time is  340.1728708744049
27500 current avg lose is  1.0683107580510052 time is  377.68309903144836
30000 current avg lose is  1.0681639694203933 time is  402.9458739757538
32500 current avg lose is  1.06844231048639 time is  435.8811500072479
35000 current avg lose is  1.0687974279531411 time is  474.25870418548584
37500 current avg lose is  1.0680510608522098 time is  502.68456196784973
40000 current avg lose is  1.0679307772897184 time is  528.8641378879547
42500 current avg lose is  1.067544614909677 time is  549.3223950862885
45000 current avg lose is  1.0671093422227436 time is  574.4050459861755
47500 current avg lose is  1.0668882419184635 time is  593.3297369480133
50000 current avg lose is  1.0673958583831786 time is  611.2758169174194
52500 current avg lose is  1.0671651781110536 time is  629.3913021087646
55000 current avg lose is  1.0674287337704138 time is  647.611419916153
57500 current avg lose is  1.0673986786463987 time is  666.7348101139069
60000 current avg lose is  1.0674292934546867 time is  686.1097941398621
62500 current avg lose is  1.0676897953548432 time is  704.9740071296692
65000 current avg lose is  1.0675282738011618 time is  723.0551240444183
67500 current avg lose is  1.0679157680847027 time is  742.5105309486389
70000 current avg lose is  1.0681421054674047 time is  774.5150830745697
72500 current avg lose is  1.0681621937542125 time is  795.1100499629974
75000 current avg lose is  1.068906489068667 time is  816.1326999664307
77500 current avg lose is  1.0688752756503321 time is  840.0066690444946
80000 current avg lose is  1.0687999247074127 time is  859.6933438777924
82500 current avg lose is  1.0688307638681296 time is  879.3936541080475
85000 current avg lose is  1.0688828008555313 time is  898.4715600013733
87500 current avg lose is  1.0692061602795124 time is  916.1872630119324
90000 current avg lose is  1.069229385611746 time is  932.1801540851593
```

```
92500 current avg lose is  1.0693909089492786 time is  955.1374840736389
95000 current avg lose is  1.0696662023854884 time is  982.492262840271
97500 current avg lose is  1.0700385442831577 time is  1005.0275959968567
100000 current avg lose is  1.0703866867625713 time is  1031.2568221092224
102500 current avg lose is  1.070607228088815 time is  1053.1853449344635
105000 current avg lose is  1.0708351706147194 time is  1071.345862865448
107500 current avg lose is  1.0711136703724085 time is  1090.9855217933655
110000 current avg lose is  1.0715125057807022 time is  1110.938469171524
112500 current avg lose is  1.071382676229477 time is  1132.2656807899475
115000 current avg lose is  1.07182084618457 time is  1149.6435360908508
117500 current avg lose is  1.0721704680085182 time is  1168.021145105362
120000 current avg lose is  1.0723628629346689 time is  1185.6967658996582
122500 current avg lose is  1.072538261650533 time is  1201.6084110736847
125000 current avg lose is  1.0732937121067048 time is  1217.6509392261505
127500 current avg lose is  1.0736993805029813 time is  1233.139272928238
130000 current avg lose is  1.0738127784848213 time is  1248.815110206604
132500 current avg lose is  1.0744151264868818 time is  1264.4576029777527
135000 current avg lose is  1.0749497592378545 time is  1280.150857925415
137500 current avg lose is  1.0754431096032533 time is  1296.505954027176
140000 current avg lose is  1.0762516807617886 time is  1312.7235021591187
142500 current avg lose is  1.0767683569736648 time is  1328.55117893219
145000 current avg lose is  1.0778381568373276 time is  1344.4150059223175
147500 current avg lose is  1.0789071250371003 time is  1360.0745420455933
150000 current avg lose is  1.080071480565369 time is  1375.9857861995697
152500 current avg lose is  1.0812610080664276 time is  1393.290606021881
155000 current avg lose is  1.0820124134346363 time is  1413.3638708591461
157500 current avg lose is  1.08362149730812 time is  1430.3814189434052
160000 current avg lose is  1.0879190420290192 time is  1445.9106590747833
162500 current avg lose is  1.095478659599033 time is  1461.8841269016266
165000 current avg lose is  1.1105091630237296 time is  1478.5193021297455
167500 current avg lose is  1.1374214921888 time is  1494.359342098236
170000 current avg lose is  1.1600518989381232 time is  1510.375832080841
172500 current avg lose is  1.1817067828752488 time is  1526.18967294693
175000 current avg lose is  1.20119316435376 time is  1541.6961488723755
177500 current avg lose is  1.2194974404940115 time is  1557.3090360164642
180000 current avg lose is  1.2366499543420968 time is  1572.9594340324402
182500 current avg lose is  1.2567743561924472 time is  1588.428687095642
185000 current avg lose is  1.2738383172162429 time is  1603.9515030384064
187500 current avg lose is  1.2889522607763801 time is  1619.5939869880676
190000 current avg lose is  1.3030570407156976 time is  1635.0615639686584
192500 current avg lose is  1.3172570710113864 time is  1650.428759098053
195000 current avg lose is  1.3314729056965577 time is  1665.848848104477
197500 current avg lose is  1.3469677912197182 time is  1681.2876300811768
200000 current avg lose is  1.3612941409675718 time is  1696.6154670715332
```

[260]:
```
test_data = list(zip(x_test,y_test))
```

[261]:
```
n_label = 3
confusion = torch.zeros(n_label, n_label)

def evaluate(line_tensor):
    hidden = rnn.initHidden()

    for i in range(line_tensor.size()[0]):
        output, hidden = rnn(line_tensor[i], hidden)



    return output

for x1,y1 in test_data:

    label_tensor = torch.tensor([y1],dtype =torch.long)
    review_tensor = review2tensor(x1)
    output = evaluate(review_tensor)

    label_pred = output_predict_y(output)
    confusion[y1][label_pred] += 1
```

[262]:
```
accuracy_my_model_ternary = confusion.trace()/confusion.sum()
```

[263]:
```
accuracy_my_model_ternary
```

[263]:
```
tensor(0.6221)
```

[264]:
```
print("the accuracy for my model with RNN binary is ",accuracy_my_model_ternary)
```

```
the accuracy for my model with RNN binary is  tensor(0.6221)
```

0.5 trinary with google 300 model RNN

[268]:
```
def opm_reviewtensor(x):
    max_review = 50
    ### create a matrix
    review = torch.zeros(max_review,1,300)
    pad = torch.zeros(300)
    index = 0
    for i in x:
        if i in opm and index < 50:
            vector = opm [i]
            review[index][0] = torch.from_numpy(vector)
            index += 1
        else:
            continue
    if index < 50:
        review[index][0] = pad
    review = torch.nan_to_num(review)
    return review
```

[269]:
```
import torch.nn as nn

class RNN(nn.Module):
    def __init__(self, input_size, hidden_size, output_size):
        super(RNN, self).__init__()

        self.hidden_size = hidden_size

        self.i2h = nn.Linear(input_size + hidden_size, hidden_size)
        self.i2o = nn.Linear(input_size + hidden_size, output_size)
        self.softmax = nn.LogSoftmax(dim=1)

    def forward(self, input, hidden):
        combined = torch.cat((input, hidden), 1)
        hidden = self.i2h(combined)
        output = self.i2o(combined)
        output = self.softmax(output)
        return output, hidden

    def initHidden(self):
        return torch.zeros(1, self.hidden_size)

rnn = RNN(300, 50, 3)
```

[270]:
```
criterion = nn.NLLLoss()
```

[271]:
```
learning_rate = 0.005 # If you set this too high, it might explode. If too low,␣
 ↪it might not learn

def train(category_tensor, line_tensor):
    hidden = rnn.initHidden()

    rnn.zero_grad()

    for i in range(line_tensor.size()[0]):
        output, hidden = rnn(line_tensor[i], hidden)

    loss = criterion(output, category_tensor)
    loss.backward()
    torch.nn.utils.clip_grad_norm_(rnn.parameters(),0.5)
    # Add parameters' gradients to their values, multiplied by learning rate
    for p in rnn.parameters():
        p.data.add_(p.grad.data, alpha=-learning_rate)

    return output, loss.item()
```

[272]:
```
main_data = list(zip(x_train,y_train))
```

[273]:
```
# Keep track of losses for plotting
current_loss = 0

def timeSince(since):
    now = time.time()
    s = now - since
    m = math.floor(s / 60)
    s -= m * 60
    return '%dm %ds' % (m, s)

start = time.time()
ii=0
correct = 0
for x_i,y_i in main_data:
    line_tensor = opm_reviewtensor(x_i)
    y_i = torch.tensor([y_i],dtype =torch.long)
    output, loss = train(y_i, line_tensor)
    current_loss += loss
    ii += 1
    if ii %2500 == 0 :
        print(ii,"current avg lose is ",current_loss/ii,'time is ',time.time()␣
 ↪- start)
```

```
2500 current avg lose is  1.0594981743097305 time is  23.336733102798462
5000 current avg lose is  1.064748005270958 time is  46.70505118370056
7500 current avg lose is  1.0665473565498989 time is  63.84702515602112
10000 current avg lose is  1.069900180619955 time is  81.04995703697205
12500 current avg lose is  1.0668171494627 time is  98.48082613945007
15000 current avg lose is  1.0672467289964358 time is  116.14186596870422
17500 current avg lose is  1.066933060513224 time is  133.63144707679749
20000 current avg lose is  1.067680196505785 time is  150.6814329624176
22500 current avg lose is  1.0679751199086507 time is  167.5362868309021
25000 current avg lose is  1.067366377673149 time is  184.75301909446716
27500 current avg lose is  1.067827658234943 time is  201.7656970024109
30000 current avg lose is  1.0678597442567348 time is  218.99542498588562
32500 current avg lose is  1.0681703750610352 time is  236.04149508476257
35000 current avg lose is  1.0684610444409506 time is  253.11390805244446
37500 current avg lose is  1.0677561033082008 time is  273.18121314048767
40000 current avg lose is  1.0677225017450749 time is  290.8098340034485
42500 current avg lose is  1.0673887481205604 time is  307.96116185188293
45000 current avg lose is  1.0670583236687714 time is  325.34737396240234
47500 current avg lose is  1.0668896239531667 time is  342.1371760368347
50000 current avg lose is  1.0674965506768226 time is  359.2518639564514
52500 current avg lose is  1.0672683788594746 time is  380.49341201782227
55000 current avg lose is  1.0675473557634787 time is  402.39842104911804
57500 current avg lose is  1.06754001811069 time is  419.7279779911041
```

```
60000 current avg lose is  1.0675953443129858 time is  436.927148103714
62500 current avg lose is  1.0678798127880096 time is  454.3286690711975
65000 current avg lose is  1.0677606553531611 time is  471.0404679775238
67500 current avg lose is  1.0681976647487392 time is  487.91938495635986
70000 current avg lose is  1.068524079589333 time is  504.5579659938812
72500 current avg lose is  1.0685647462355679 time is  521.088385105133
75000 current avg lose is  1.0694102822200457 time is  541.2910070419312
77500 current avg lose is  1.0693500262260438 time is  558.7195630073547
80000 current avg lose is  1.0693192850824444 time is  575.1268022060394
82500 current avg lose is  1.0693833013919267 time is  594.3808679580688
85000 current avg lose is  1.0694949775364468 time is  611.840841293335
87500 current avg lose is  1.0698560204439505 time is  628.5954940319061
90000 current avg lose is  1.069898481336236 time is  645.9558482170105
92500 current avg lose is  1.0700374788427676 time is  664.6808271408081
95000 current avg lose is  1.070354220152372 time is  682.0515701770782
97500 current avg lose is  1.0707094923407603 time is  698.7866830825806
100000 current avg lose is  1.0710481071619689 time is  716.7230660915375
102500 current avg lose is  1.0712487298258921 time is  734.2472088336945
105000 current avg lose is  1.071468197950437 time is  751.8574929237366
107500 current avg lose is  1.0717564312385959 time is  768.8746490478516
110000 current avg lose is  1.0721288732883605 time is  785.8098080158234
112500 current avg lose is  1.0719723090370497 time is  804.3105058670044
115000 current avg lose is  1.0723668803686681 time is  823.2005460262299
117500 current avg lose is  1.072707923695128 time is  844.9211530685425
120000 current avg lose is  1.072853413661197 time is  861.6371009349823
122500 current avg lose is  1.0729630950057993 time is  877.2556829452515
125000 current avg lose is  1.073640887413144 time is  893.0236148834229
127500 current avg lose is  1.0740095382137627 time is  908.8859710693359
130000 current avg lose is  1.0740493706464767 time is  924.9225609302521
132500 current avg lose is  1.0745534437333637 time is  940.3724489212036
135000 current avg lose is  1.0749703991068733 time is  955.9236941337585
137500 current avg lose is  1.0753464512024142 time is  971.6035671234131
140000 current avg lose is  1.0759549313152474 time is  987.861456155777
142500 current avg lose is  1.0762306944811553 time is  1003.5206110477448
145000 current avg lose is  1.0768760815870144 time is  1019.2891991138458
147500 current avg lose is  1.0775488885355198 time is  1034.8012599945068
150000 current avg lose is  1.07824242566665 time is  1049.9733700752258
152500 current avg lose is  1.0788664567546766 time is  1065.3668999671936
155000 current avg lose is  1.0792265710586502 time is  1080.6259188652039
157500 current avg lose is  1.0798717930496684 time is  1096.1389381885529
160000 current avg lose is  1.0810140441008378 time is  1111.9292871952057
162500 current avg lose is  1.0829105518394708 time is  1127.6568629741669
165000 current avg lose is  1.086564545000401 time is  1142.9311730861664
167500 current avg lose is  1.092441711496133 time is  1158.6104590892792
170000 current avg lose is  1.1002550451258148 time is  1173.8612530231476
172500 current avg lose is  1.1275077948343504 time is  1195.6613211631775
175000 current avg lose is  1.1503944251237956 time is  1221.9788782596588
177500 current avg lose is  1.1701684366750869 time is  1239.8266541957855
```

```
180000 current avg lose is  1.1884959107016662 time is  1255.7306549549103
182500 current avg lose is  1.2083724974900445 time is  1271.3427398204803
185000 current avg lose is  1.2261250610874894 time is  1299.1251759529114
187500 current avg lose is  1.2426133169996108 time is  1326.855817079544
190000 current avg lose is  1.258071125571814 time is  1345.0316240787506
192500 current avg lose is  1.2726532343440726 time is  1362.7227640151978
195000 current avg lose is  1.2865889542886226 time is  1380.7086811065674
197500 current avg lose is  1.3023360601757314 time is  1398.571238040924
200000 current avg lose is  1.3172428433795291 time is  1416.3848860263824
```

[274]:
```
accuracy_opm_ternary = confusion.trace()/confusion.sum()
```

[275]:
```
print("the accuracy for google model with RNN binary is ",accuracy_opm_ternary)
```

```
the accuracy for google model with RNN binary is  tensor(0.6221)
```

#RNN
RNN : Binary Case with google model | The Test Accuracy is 0.7890
RNN : Ternary case with googel model | the Test Accuracy is 0.6221
RNN: Binary case with my own model | Test Accuracy is 0.8205
RNN: Ternary case with my own model | Test Accuracy is 0.6223

[ ]:

hw2(5b)

October 5, 2021

1 GRU

[6]:
```
import pandas as pd
import numpy as np
import copy
import re
import torch
from sklearn.model_selection import train_test_split
import random
```

[7]:
```
import pandas as pd
test = pd.read_csv("amazon_reviews_us_Kitchen_v1_00.tsv", sep =␣
 ↪'\t',error_bad_lines = False)
test['label'] = -1
a1 = test.loc[test['star_rating']==1].sample(50000)
a2 = test.loc[test['star_rating']==2].sample(50000)
a3 = test.loc[test['star_rating']==3].sample(50000)
a4 = test.loc[test['star_rating']==4].sample(50000)
a5 = test.loc[test['star_rating']==5].sample(50000)
```

```
b'Skipping line 16148: expected 15 fields, saw 22\nSkipping line 20100: expected
15 fields, saw 22\nSkipping line 45178: expected 15 fields, saw 22\nSkipping
line 48700: expected 15 fields, saw 22\nSkipping line 63331: expected 15 fields,
saw 22\n'
b'Skipping line 86053: expected 15 fields, saw 22\nSkipping line 88858: expected
15 fields, saw 22\nSkipping line 115017: expected 15 fields, saw 22\n'
b'Skipping line 137366: expected 15 fields, saw 22\nSkipping line 139110:
expected 15 fields, saw 22\nSkipping line 165540: expected 15 fields, saw
22\nSkipping line 171813: expected 15 fields, saw 22\n'
b'Skipping line 203723: expected 15 fields, saw 22\nSkipping line 209366:
expected 15 fields, saw 22\nSkipping line 211310: expected 15 fields, saw
22\nSkipping line 246351: expected 15 fields, saw 22\nSkipping line 252364:
expected 15 fields, saw 22\n'
b'Skipping line 267003: expected 15 fields, saw 22\nSkipping line 268957:
expected 15 fields, saw 22\nSkipping line 303336: expected 15 fields, saw
22\nSkipping line 306021: expected 15 fields, saw 22\nSkipping line 311569:
expected 15 fields, saw 22\nSkipping line 316767: expected 15 fields, saw
```

```
22\nSkipping line 324009: expected 15 fields, saw 22\n'
b'Skipping line 359107: expected 15 fields, saw 22\nSkipping line 368367:
expected 15 fields, saw 22\nSkipping line 381180: expected 15 fields, saw
22\nSkipping line 390453: expected 15 fields, saw 22\n'
b'Skipping line 412243: expected 15 fields, saw 22\nSkipping line 419342:
expected 15 fields, saw 22\nSkipping line 457388: expected 15 fields, saw 22\n'
b'Skipping line 459935: expected 15 fields, saw 22\nSkipping line 460167:
expected 15 fields, saw 22\nSkipping line 466460: expected 15 fields, saw
22\nSkipping line 500314: expected 15 fields, saw 22\nSkipping line 500339:
expected 15 fields, saw 22\nSkipping line 505396: expected 15 fields, saw
22\nSkipping line 507760: expected 15 fields, saw 22\nSkipping line 513626:
expected 15 fields, saw 22\n'
b'Skipping line 527638: expected 15 fields, saw 22\nSkipping line 534209:
expected 15 fields, saw 22\nSkipping line 535687: expected 15 fields, saw
22\nSkipping line 547671: expected 15 fields, saw 22\nSkipping line 549054:
expected 15 fields, saw 22\n'
b'Skipping line 599929: expected 15 fields, saw 22\nSkipping line 604776:
expected 15 fields, saw 22\nSkipping line 609937: expected 15 fields, saw
22\nSkipping line 632059: expected 15 fields, saw 22\nSkipping line 638546:
expected 15 fields, saw 22\n'
b'Skipping line 665017: expected 15 fields, saw 22\nSkipping line 677680:
expected 15 fields, saw 22\nSkipping line 684370: expected 15 fields, saw
22\nSkipping line 720217: expected 15 fields, saw 29\n'
b'Skipping line 723240: expected 15 fields, saw 22\nSkipping line 723433:
expected 15 fields, saw 22\nSkipping line 763891: expected 15 fields, saw 22\n'
b'Skipping line 800288: expected 15 fields, saw 22\nSkipping line 802942:
expected 15 fields, saw 22\nSkipping line 803379: expected 15 fields, saw
22\nSkipping line 805122: expected 15 fields, saw 22\nSkipping line 821899:
expected 15 fields, saw 22\nSkipping line 831707: expected 15 fields, saw
22\nSkipping line 842829: expected 15 fields, saw 22\nSkipping line 843604:
expected 15 fields, saw 22\n'
b'Skipping line 863904: expected 15 fields, saw 22\nSkipping line 875655:
expected 15 fields, saw 22\nSkipping line 886796: expected 15 fields, saw
22\nSkipping line 892299: expected 15 fields, saw 22\nSkipping line 902518:
expected 15 fields, saw 22\nSkipping line 903079: expected 15 fields, saw
22\nSkipping line 912678: expected 15 fields, saw 22\n'
b'Skipping line 932953: expected 15 fields, saw 22\nSkipping line 936838:
expected 15 fields, saw 22\nSkipping line 937177: expected 15 fields, saw
22\nSkipping line 947695: expected 15 fields, saw 22\nSkipping line 960713:
expected 15 fields, saw 22\nSkipping line 965225: expected 15 fields, saw
22\nSkipping line 980776: expected 15 fields, saw 22\n'
b'Skipping line 999318: expected 15 fields, saw 22\nSkipping line 1007247:
expected 15 fields, saw 22\nSkipping line 1015987: expected 15 fields, saw
22\nSkipping line 1018984: expected 15 fields, saw 22\nSkipping line 1028671:
expected 15 fields, saw 22\n'
b'Skipping line 1063360: expected 15 fields, saw 22\nSkipping line 1066195:
expected 15 fields, saw 22\nSkipping line 1066578: expected 15 fields, saw
22\nSkipping line 1066869: expected 15 fields, saw 22\nSkipping line 1068809:
```

```
expected 15 fields, saw 22\nSkipping line 1069505: expected 15 fields, saw
22\nSkipping line 1087983: expected 15 fields, saw 22\nSkipping line 1108184:
expected 15 fields, saw 22\n'
b'Skipping line 1118137: expected 15 fields, saw 22\nSkipping line 1142723:
expected 15 fields, saw 22\nSkipping line 1152492: expected 15 fields, saw
22\nSkipping line 1156947: expected 15 fields, saw 22\nSkipping line 1172563:
expected 15 fields, saw 22\n'
b'Skipping line 1209254: expected 15 fields, saw 22\nSkipping line 1212966:
expected 15 fields, saw 22\nSkipping line 1236533: expected 15 fields, saw
22\nSkipping line 1237598: expected 15 fields, saw 22\n'
b'Skipping line 1273825: expected 15 fields, saw 22\nSkipping line 1277898:
expected 15 fields, saw 22\nSkipping line 1283654: expected 15 fields, saw
22\nSkipping line 1286023: expected 15 fields, saw 22\nSkipping line 1302038:
expected 15 fields, saw 22\nSkipping line 1305179: expected 15 fields, saw 22\n'
b'Skipping line 1326022: expected 15 fields, saw 22\nSkipping line 1338120:
expected 15 fields, saw 22\nSkipping line 1338503: expected 15 fields, saw
22\nSkipping line 1338849: expected 15 fields, saw 22\nSkipping line 1341513:
expected 15 fields, saw 22\nSkipping line 1346493: expected 15 fields, saw
22\nSkipping line 1373127: expected 15 fields, saw 22\n'
b'Skipping line 1389508: expected 15 fields, saw 22\nSkipping line 1413951:
expected 15 fields, saw 22\nSkipping line 1433626: expected 15 fields, saw 22\n'
b'Skipping line 1442698: expected 15 fields, saw 22\nSkipping line 1472982:
expected 15 fields, saw 22\nSkipping line 1482282: expected 15 fields, saw
22\nSkipping line 1487808: expected 15 fields, saw 22\nSkipping line 1500636:
expected 15 fields, saw 22\n'
b'Skipping line 1511479: expected 15 fields, saw 22\nSkipping line 1532302:
expected 15 fields, saw 22\nSkipping line 1537952: expected 15 fields, saw
22\nSkipping line 1539951: expected 15 fields, saw 22\nSkipping line 1541020:
expected 15 fields, saw 22\n'
b'Skipping line 1594217: expected 15 fields, saw 22\nSkipping line 1612264:
expected 15 fields, saw 22\nSkipping line 1615907: expected 15 fields, saw
22\nSkipping line 1621859: expected 15 fields, saw 22\n'
b'Skipping line 1653542: expected 15 fields, saw 22\nSkipping line 1671537:
expected 15 fields, saw 22\nSkipping line 1672879: expected 15 fields, saw
22\nSkipping line 1674523: expected 15 fields, saw 22\nSkipping line 1677355:
expected 15 fields, saw 22\nSkipping line 1703907: expected 15 fields, saw 22\n'
b'Skipping line 1713046: expected 15 fields, saw 22\nSkipping line 1722982:
expected 15 fields, saw 22\nSkipping line 1727290: expected 15 fields, saw
22\nSkipping line 1744482: expected 15 fields, saw 22\n'
b'Skipping line 1803858: expected 15 fields, saw 22\nSkipping line 1810069:
expected 15 fields, saw 22\nSkipping line 1829751: expected 15 fields, saw
22\nSkipping line 1831699: expected 15 fields, saw 22\n'
b'Skipping line 1863131: expected 15 fields, saw 22\nSkipping line 1867917:
expected 15 fields, saw 22\nSkipping line 1874790: expected 15 fields, saw
22\nSkipping line 1879952: expected 15 fields, saw 22\nSkipping line 1880501:
expected 15 fields, saw 22\nSkipping line 1886655: expected 15 fields, saw
22\nSkipping line 1887888: expected 15 fields, saw 22\nSkipping line 1894286:
expected 15 fields, saw 22\nSkipping line 1895400: expected 15 fields, saw 22\n'
```

```
b'Skipping line 1904040: expected 15 fields, saw 22\nSkipping line 1907604:
expected 15 fields, saw 22\nSkipping line 1915739: expected 15 fields, saw
22\nSkipping line 1921514: expected 15 fields, saw 22\nSkipping line 1939428:
expected 15 fields, saw 22\nSkipping line 1944342: expected 15 fields, saw
22\nSkipping line 1949699: expected 15 fields, saw 22\nSkipping line 1961872:
expected 15 fields, saw 22\n'
b'Skipping line 1968846: expected 15 fields, saw 22\nSkipping line 1999941:
expected 15 fields, saw 22\nSkipping line 2001492: expected 15 fields, saw
22\nSkipping line 2011204: expected 15 fields, saw 22\nSkipping line 2025295:
expected 15 fields, saw 22\n'
b'Skipping line 2041266: expected 15 fields, saw 22\nSkipping line 2073314:
expected 15 fields, saw 22\nSkipping line 2080133: expected 15 fields, saw
22\nSkipping line 2088521: expected 15 fields, saw 22\n'
b'Skipping line 2103490: expected 15 fields, saw 22\nSkipping line 2115278:
expected 15 fields, saw 22\nSkipping line 2153174: expected 15 fields, saw
22\nSkipping line 2161731: expected 15 fields, saw 22\n'
b'Skipping line 2165250: expected 15 fields, saw 22\nSkipping line 2175132:
expected 15 fields, saw 22\nSkipping line 2206817: expected 15 fields, saw
22\nSkipping line 2215848: expected 15 fields, saw 22\nSkipping line 2223811:
expected 15 fields, saw 22\n'
b'Skipping line 2257265: expected 15 fields, saw 22\nSkipping line 2259163:
expected 15 fields, saw 22\nSkipping line 2263291: expected 15 fields, saw 22\n'
b'Skipping line 2301943: expected 15 fields, saw 22\nSkipping line 2304371:
expected 15 fields, saw 22\nSkipping line 2306015: expected 15 fields, saw
22\nSkipping line 2312186: expected 15 fields, saw 22\nSkipping line 2314740:
expected 15 fields, saw 22\nSkipping line 2317754: expected 15 fields, saw 22\n'
b'Skipping line 2383514: expected 15 fields, saw 22\n'
b'Skipping line 2449763: expected 15 fields, saw 22\n'
b'Skipping line 2589323: expected 15 fields, saw 22\n'
b'Skipping line 2775036: expected 15 fields, saw 22\n'
b'Skipping line 2935174: expected 15 fields, saw 22\n'
b'Skipping line 3078830: expected 15 fields, saw 22\n'
b'Skipping line 3123091: expected 15 fields, saw 22\n'
b'Skipping line 3185533: expected 15 fields, saw 22\n'
b'Skipping line 4150395: expected 15 fields, saw 22\n'
b'Skipping line 4748401: expected 15 fields, saw 22\n'
```

[8]:
```
new_test = pd.concat([a1,a2,a3,a4,a5])
```

[9]:
```
new_test['lable'] = 1
```

[10]:
```
new_test.label[test.star_rating>3] = 1
new_test.label[test.star_rating<3] = 2
new_test.label[test.star_rating==3] = 3
new_test = new_test[['label','review_body']]
```

[11]:
```
import gensim
model_1 = gensim.models.Word2Vec.load('save')
```

[12]:
```
#data clearing
new_test['review_body'] = new_test['review_body'].str.lower()

def tag(x):
    return re.sub('<.*?>','',str(x))
new_test['review_body'] = new_test['review_body'].apply(lambda x:tag(x))

def url(x):
    return re.sub('(https?|ftp|file)://[-A-Za-z0-9+&@#/%?=~_|!:,.;
↪]+[-A-Za-z0-9+&@#/%=~_|]','',str(x))

new_test['review_body'] = new_test['review_body'].apply(lambda x:url(x))

import contractions
new_test['review_body'] = new_test['review_body'].apply(lambda x:contractions.
↪fix(x))

def non_alphabetical(x):
    return re.sub('[^a-zA-Z\s]','',str(x))
new_test['review_body'] = new_test['review_body'].apply(lambda x:
↪non_alphabetical(x))
def extra_space(x):
    return re.sub( ' +',' ',str(x))
new_test['review_body'] = new_test['review_body'].apply(lambda x:extra_space(x))
```

[13]:
```
#Pre-processing

import nltk
from nltk.corpus import stopwords
nltk.download('stopwords')
stop_words_set = set(stopwords.words('english'))
from nltk import word_tokenize, pos_tag

def stop_words(x):
    word_tokens = word_tokenize(x)
    temp = []
    for i in word_tokens:
        if i not in stop_words_set:
            temp.append(i)
    return temp
new_test['review_body'] = new_test['review_body'].apply(lambda x:stop_words(x))
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     /Users/chenlin/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

[14]:
```
new_test['y'] = -1
```

[15]:
```
new_test['y'][new_test['label'] == 1] =0
new_test['y'][new_test['label'] == 2] =1
new_test['y'][new_test['label'] == 3] =2
```

[16]:
```
binary = new_test.loc[new_test['y']!=2]
```

[17]:
```
def mymodel_reviewtensor(x):
    max_review = 50
    review = torch.zeros(max_review,1,300)
    pad = torch.zeros(300)
    index = 0
    for i in x:
        if i in model_1.wv and index < 50:
            vector = model_1.wv [i]
            review[index][0] = torch.from_numpy(vector)
            index += 1
        else:
            continue
    if index < 50:
        review[index][0] = pad
    review = torch.nan_to_num(review)
    return review
```

[21]:
```
import gensim
model_1 = gensim.models.Word2Vec.load('save')
```

[22]:
```
import gensim.downloader as api

opm = api.load("word2vec-google-news-300")
```

[18]:
```
def opm_reviewtensor(x):
    max_review = 50
    ### create a matrix
    review = torch.zeros(max_review,1,300)
    pad = torch.zeros(300)
    index = 0
    for i in x:
        if i in opm and index < 50:
            vector = opm [i]
            review[index][0] = torch.from_numpy(vector)
            index += 1
        else:
            continue
    if index < 50:
        review[index][0] = pad
    review = torch.nan_to_num(review)
    return review
```

[19]:
```
def data_vector(dataset,opm):
    res = []
    if opm == True:
        for x,y in dataset:
            x_vector = opm_reviewtensor(x)
            res.append((x_vector,y))
    if opm == False:
        for x,y in dataset:
            x_vector = mymodel_reviewtensor(x)
            res.append((x_vector,y))
    return res
```

[23]:
```
binary
```

[23]:
```
         label                                        review_body  y
3840037      2  [one, highly, disappointed, product, years, wa...  1
3791453      2  [bought, month, ago, bbb, times, use, make, si...  1
2539564      2  [bought, brewer, jan, within, weeks, carafe, e...  1
3239783      2  [bought, couple, clean, bottom, beta, fish, bo...  1
2740932      2  [love, productbut, old, one, broke, wanted, ge...  1
...        ...                                                ... ..
2183853      1               [blast, family, making, taking, pics]  0
1020136      1        [great, purchase, please, came, time, loved]  0
1758315      1                            [perfect, glasses, love]  0
4175724      1  [pans, first, quality, mediumheavy, gauge, sta...  0
4038384      1  [cup, amazing, shipped, accurately, says, cano...  0

[200000 rows x 3 columns]
```

1.1 binary GRU my model

[26]:
```
binary_data = list(zip(binary.review_body.values,binary.y.values))
```

[29]:
```
train_set, test_set = train_test_split(binary_data, random_state = 19,␣
 ↪test_size = 0.2)
```

[30]:
```
test = data_vector(test_set, opm=False )

train = data_vector(train_set, opm=False)
```

[36]:
```
import torch.nn as nn
```

[34]:
```
# number of subprocesses to use for data loading
num_workers = 2
# how many samples per batch to load
batch_size = 100
# percentage of training set to use as validation
#valid_size = 0.2
```

```
train_loader=torch.utils.data.
↪DataLoader(train,batch_size=batch_size,shuffle=True,drop_last=True,␣
↪num_workers=0)

test_loader=torch.utils.data.DataLoader(test, batch_size=batch_size,␣
↪drop_last=True,num_workers=0)
```

[49]:
```
class GRU(nn.Module):
    def __init__(self, input_size, hidden_size, num_layers, num_classes):
        super(GRU, self).__init__()
        self.num_layers = num_layers
        self.hidden_size = hidden_size

        self.gru = nn.GRU(input_size, hidden_size, num_layers, batch_first=True)
        self.fc = nn.Linear(hidden_size, num_classes)

    def forward(self, x):
        h0 = torch.zeros(self.num_layers, x.size(0), self.hidden_size).
↪to(device)


        out, _ = self.gru(x, h0)
        out = out[:, -1, :]
        out = self.fc(out)
        return out

GRU_model = GRU(300, 50, 2, 2).to(device)
```

[50]:
```
criterion = nn.CrossEntropyLoss()
optimizer = torch.optim.Adam(GRU_model.parameters(), lr=0.002)
```

[51]:
```
# Train the model
num_epochs = 1
n_total_steps = len(train_loader)
for epoch in range(num_epochs):
    for i, (x_i,y_i) in enumerate(train_loader):
        # origin shape: [N, 1, 28, 28]
        # resized: [N, 28, 28]
        x_i = x_i.reshape(-1, 50, 300).to(device)
        y_i = y_i.to(device)

        # Forward pass
        outputs = GRU_model(x_i)
        loss = criterion(outputs, y_i)

        # Backward and optimize
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()

        if (i+1) % 100 == 0:
            print (f'Epoch [{epoch+1}/{num_epochs}], Step [{i+1}/
↪{n_total_steps}], Loss: {loss.item():.4f}')
```

```
Epoch [1/1], Step [100/1600], Loss: 0.5400
Epoch [1/1], Step [200/1600], Loss: 0.4260
Epoch [1/1], Step [300/1600], Loss: 0.2985
Epoch [1/1], Step [400/1600], Loss: 0.2918
Epoch [1/1], Step [500/1600], Loss: 0.3736
Epoch [1/1], Step [600/1600], Loss: 0.3736
Epoch [1/1], Step [700/1600], Loss: 0.2612
Epoch [1/1], Step [800/1600], Loss: 0.3358
Epoch [1/1], Step [900/1600], Loss: 0.2936
Epoch [1/1], Step [1000/1600], Loss: 0.3211
Epoch [1/1], Step [1100/1600], Loss: 0.3775
Epoch [1/1], Step [1200/1600], Loss: 0.3397
Epoch [1/1], Step [1300/1600], Loss: 0.4303
Epoch [1/1], Step [1400/1600], Loss: 0.2620
Epoch [1/1], Step [1500/1600], Loss: 0.3403
Epoch [1/1], Step [1600/1600], Loss: 0.2838
```

[54]:
```
with torch.no_grad():
    n_correct = 0
    n_samples = 0
    for images, labels in test_loader:
        images = images.reshape(-1, 50, 300).to(device)
        labels = labels.to(device)
        outputs = GRU_model(images)
        # max returns (value ,index)
        _, predicted = torch.max(outputs.data, 1)
        n_samples += labels.size(0)
        n_correct += (predicted == labels).sum().item()

    acc = 100.0 * n_correct / n_samples
    print(f'Accuracy of binary GRU my model: {acc} %')
```

```
Accuracy of the network on the 10000 test images: 87.6475 %
```

[55]:
```
binary_GRU_my_model = acc
```

[56]:
```
print(f'Accuracy of binary GRU my model: {acc} %')
```

```
Accuracy of binary GRU my model: 87.6475 %
```

2 binary GRU googel model

[57]:
```
test = data_vector(test_set, opm=True )

train = data_vector(train_set, opm=True)
```

[58]:
```
# number of subprocesses to use for data loading
num_workers = 2
# how many samples per batch to load
batch_size = 100
# percentage of training set to use as validation
#valid_size = 0.2
train_loader=torch.utils.data.
 ↪DataLoader(train,batch_size=batch_size,shuffle=True,drop_last=True,␣
 ↪num_workers=0)

test_loader=torch.utils.data.DataLoader(test, batch_size=batch_size,␣
 ↪drop_last=True,num_workers=0)
```

[59]:
```
class GRU(nn.Module):
    def __init__(self, input_size, hidden_size, num_layers, num_classes):
        super(GRU, self).__init__()
        self.num_layers = num_layers
        self.hidden_size = hidden_size

        self.gru = nn.GRU(input_size, hidden_size, num_layers, batch_first=True)
        self.fc = nn.Linear(hidden_size, num_classes)

    def forward(self, x):
        h0 = torch.zeros(self.num_layers, x.size(0), self.hidden_size).
 ↪to(device)


        out, _ = self.gru(x, h0)
        out = out[:, -1, :]
        out = self.fc(out)
        return out

GRU_model = GRU(300, 50, 2, 2).to(device)
```

[60]:
```
criterion = nn.CrossEntropyLoss()
optimizer = torch.optim.Adam(GRU_model.parameters(), lr=0.002)
```

[61]:
```
# Train the model
num_epochs = 1
n_total_steps = len(train_loader)
for epoch in range(num_epochs):
    for i, (x_i,y_i) in enumerate(train_loader):
        # origin shape: [N, 1, 28, 28]
        # resized: [N, 28, 28]
        x_i = x_i.reshape(-1, 50, 300).to(device)
        y_i = y_i.to(device)

        # Forward pass
        outputs = GRU_model(x_i)
        loss = criterion(outputs, y_i)

        # Backward and optimize
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()

        if (i+1) % 100 == 0:
            print (f'Epoch [{epoch+1}/{num_epochs}], Step [{i+1}/
↪{n_total_steps}], Loss: {loss.item():.4f}')
```

```
Epoch [1/1], Step [100/1600], Loss: 0.6452
Epoch [1/1], Step [200/1600], Loss: 0.5023
Epoch [1/1], Step [300/1600], Loss: 0.4549
Epoch [1/1], Step [400/1600], Loss: 0.3687
Epoch [1/1], Step [500/1600], Loss: 0.3416
Epoch [1/1], Step [600/1600], Loss: 0.3899
Epoch [1/1], Step [700/1600], Loss: 0.3783
Epoch [1/1], Step [800/1600], Loss: 0.4301
Epoch [1/1], Step [900/1600], Loss: 0.3203
Epoch [1/1], Step [1000/1600], Loss: 0.3960
Epoch [1/1], Step [1100/1600], Loss: 0.3546
Epoch [1/1], Step [1200/1600], Loss: 0.3367
Epoch [1/1], Step [1300/1600], Loss: 0.3432
Epoch [1/1], Step [1400/1600], Loss: 0.3284
Epoch [1/1], Step [1500/1600], Loss: 0.3906
Epoch [1/1], Step [1600/1600], Loss: 0.3222
```

[62]:
```
with torch.no_grad():
    n_correct = 0
    n_samples = 0
    for images, labels in test_loader:
        images = images.reshape(-1, 50, 300).to(device)
        labels = labels.to(device)
        outputs = GRU_model(images)
        # max returns (value ,index)
        _, predicted = torch.max(outputs.data, 1)
        n_samples += labels.size(0)
        n_correct += (predicted == labels).sum().item()

    acc = 100.0 * n_correct / n_samples
    print(f'Accuracy of binary GRU google model: {acc} %')
```

```
Accuracy of binary GRU google model: 87.3825 %
```

[63]:
```
binary_GRU_opm = acc
```

3 Ternary GRU my model

[70]:
```
ternary_data = list(zip(new_test.review_body.values,new_test.y.values))

train_set, test_set = train_test_split(ternary_data, random_state = 19,␣
 ↪test_size = 0.2)

test = data_vector(test_set, opm=False )

train = data_vector(train_set, opm=False)


# number of subprocesses to use for data loading
num_workers = 2
# how many samples per batch to load
batch_size = 100
# percentage of training set to use as validation
#valid_size = 0.2
train_loader=torch.utils.data.
 ↪DataLoader(train,batch_size=batch_size,shuffle=True,drop_last=True,␣
 ↪num_workers=0)

test_loader=torch.utils.data.DataLoader(test, batch_size=batch_size,␣
 ↪drop_last=True,num_workers=0)

class GRU(nn.Module):
    def __init__(self, input_size, hidden_size, num_layers, num_classes):
        super(GRU, self).__init__()
        self.num_layers = num_layers
        self.hidden_size = hidden_size

        self.gru = nn.GRU(input_size, hidden_size, num_layers, batch_first=True)
        self.fc = nn.Linear(hidden_size, num_classes)

    def forward(self, x):
        h0 = torch.zeros(self.num_layers, x.size(0), self.hidden_size).
 ↪to(device)


        out, _ = self.gru(x, h0)
        out = out[:, -1, :]
        out = self.fc(out)
        return out

GRU_model = GRU(300, 50, 2, 3).to(device)

criterion = nn.CrossEntropyLoss()
optimizer = torch.optim.Adam(GRU_model.parameters(), lr=0.002)

# Train the model
num_epochs = 1
n_total_steps = len(train_loader)
for epoch in range(num_epochs):
    for i, (x_i,y_i) in enumerate(train_loader):
        # origin shape: [N, 1, 28, 28]
        # resized: [N, 28, 28]
        x_i = x_i.reshape(-1, 50, 300).to(device)
        y_i = y_i.to(device)

        # Forward pass
        outputs = GRU_model(x_i)
        loss = criterion(outputs, y_i)

        # Backward and optimize
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()

        if (i+1) % 100 == 0:
            print (f'Epoch [{epoch+1}/{num_epochs}], Step [{i+1}/
 ↪{n_total_steps}], Loss: {loss.item():.4f}')
```

```
Epoch [1/1], Step [100/2000], Loss: 0.9219
Epoch [1/1], Step [200/2000], Loss: 0.7806
Epoch [1/1], Step [300/2000], Loss: 0.6303
Epoch [1/1], Step [400/2000], Loss: 0.6929
Epoch [1/1], Step [500/2000], Loss: 0.7986
Epoch [1/1], Step [600/2000], Loss: 0.7141
Epoch [1/1], Step [700/2000], Loss: 0.6560
Epoch [1/1], Step [800/2000], Loss: 0.6516
Epoch [1/1], Step [900/2000], Loss: 0.6664
Epoch [1/1], Step [1000/2000], Loss: 0.7144
Epoch [1/1], Step [1100/2000], Loss: 0.7327
Epoch [1/1], Step [1200/2000], Loss: 0.6977
Epoch [1/1], Step [1300/2000], Loss: 0.7030
Epoch [1/1], Step [1400/2000], Loss: 0.6500
Epoch [1/1], Step [1500/2000], Loss: 0.6327
Epoch [1/1], Step [1600/2000], Loss: 0.6348
Epoch [1/1], Step [1700/2000], Loss: 0.7318
Epoch [1/1], Step [1800/2000], Loss: 0.7805
Epoch [1/1], Step [1900/2000], Loss: 0.6831
Epoch [1/1], Step [2000/2000], Loss: 0.6341
```

[75]:
```
with torch.no_grad():
    n_correct = 0
    n_samples = 0
    for images, labels in test_loader:
        images = images.reshape(-1, 50, 300).to(device)
        labels = labels.to(device)
        outputs = GRU_model(images)
        # max returns (value ,index)
        _, predicted = torch.max(outputs.data, 1)
        n_samples += labels.size(0)
        n_correct += (predicted == labels).sum().item()

    acc = 100.0 * n_correct / n_samples
    print(f'Accuracy of ternary GRU my model: {acc} %')
```

```
Accuracy of ternary GRU my model: 72.188 %
```

[76]:
```
binary_GRU_my_model = acc
```

[77]:
```
binary_GRU_my_model
```

[77]:
```
72.188
```

4 Ternary GRU google model

[78]:
```
ternary_data = list(zip(new_test.review_body.values,new_test.y.values))

train_set, test_set = train_test_split(ternary_data, random_state = 19,␣
 ↪test_size = 0.2)

test = data_vector(test_set, opm=True )

train = data_vector(train_set, opm=True)


# number of subprocesses to use for data loading
num_workers = 2
# how many samples per batch to load
batch_size = 100
# percentage of training set to use as validation
#valid_size = 0.2
```

```
train_loader=torch.utils.data.
↪DataLoader(train,batch_size=batch_size,shuffle=True,drop_last=True,␣
↪num_workers=0)

test_loader=torch.utils.data.DataLoader(test, batch_size=batch_size,␣
↪drop_last=True,num_workers=0)

class GRU(nn.Module):
    def __init__(self, input_size, hidden_size, num_layers, num_classes):
        super(GRU, self).__init__()
        self.num_layers = num_layers
        self.hidden_size = hidden_size

        self.gru = nn.GRU(input_size, hidden_size, num_layers, batch_first=True)
        self.fc = nn.Linear(hidden_size, num_classes)

    def forward(self, x):
        h0 = torch.zeros(self.num_layers, x.size(0), self.hidden_size).
↪to(device)


        out, _ = self.gru(x, h0)
        out = out[:, -1, :]
        out = self.fc(out)
        return out

GRU_model = GRU(300, 50, 2, 3).to(device)

criterion = nn.CrossEntropyLoss()
optimizer = torch.optim.Adam(GRU_model.parameters(), lr=0.002)

# Train the model
num_epochs = 1
n_total_steps = len(train_loader)
for epoch in range(num_epochs):
    for i, (x_i,y_i) in enumerate(train_loader):
        # origin shape: [N, 1, 28, 28]
        # resized: [N, 28, 28]
        x_i = x_i.reshape(-1, 50, 300).to(device)
        y_i = y_i.to(device)

        # Forward pass
        outputs = GRU_model(x_i)
        loss = criterion(outputs, y_i)

        # Backward and optimize
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()

        if (i+1) % 100 == 0:
            print (f'Epoch [{epoch+1}/{num_epochs}], Step [{i+1}/
↪{n_total_steps}], Loss: {loss.item():.4f}')
```

```
Epoch [1/1], Step [100/2000], Loss: 0.9518
Epoch [1/1], Step [200/2000], Loss: 0.8481
Epoch [1/1], Step [300/2000], Loss: 0.7337
Epoch [1/1], Step [400/2000], Loss: 0.8770
Epoch [1/1], Step [500/2000], Loss: 0.7802
Epoch [1/1], Step [600/2000], Loss: 0.7466
Epoch [1/1], Step [700/2000], Loss: 0.8227
Epoch [1/1], Step [800/2000], Loss: 0.7063
Epoch [1/1], Step [900/2000], Loss: 0.7319
Epoch [1/1], Step [1000/2000], Loss: 0.6627
Epoch [1/1], Step [1100/2000], Loss: 0.6182
Epoch [1/1], Step [1200/2000], Loss: 0.7480
Epoch [1/1], Step [1300/2000], Loss: 0.6651
Epoch [1/1], Step [1400/2000], Loss: 0.6778
Epoch [1/1], Step [1500/2000], Loss: 0.6995
Epoch [1/1], Step [1600/2000], Loss: 0.6357
Epoch [1/1], Step [1700/2000], Loss: 0.5874
Epoch [1/1], Step [1800/2000], Loss: 0.6396
Epoch [1/1], Step [1900/2000], Loss: 0.6808
Epoch [1/1], Step [2000/2000], Loss: 0.7171
```

[79]:
```
with torch.no_grad():
    n_correct = 0
    n_samples = 0
    for images, labels in test_loader:
        images = images.reshape(-1, 50, 300).to(device)
        labels = labels.to(device)
        outputs = GRU_model(images)
        # max returns (value ,index)
        _, predicted = torch.max(outputs.data, 1)
        n_samples += labels.size(0)
        n_correct += (predicted == labels).sum().item()

    acc = 100.0 * n_correct / n_samples
    print(f'Accuracy of ternary GRU google model: {acc} %')
```

```
Accuracy of ternary GRU google model: 71.422 %
```

4.1 GRU conclusion

Accuracy of ternary GRU google model: 71.422 %

```
Accuracy of ternary GRU my model: 72.188 %
Accuracy of binary GRU google model: 87.3825 %
Accuracy of binary GRU my model: 87.6475 %
```