RTML-Final-2021

May 7, 2021

1 RTML Final 2021 - 2.5 Hrs (9.13 - 11:43)

In this exam, we'll have some practical exercises using RNNs and some short answer questions regarding the Transformer/attention and reinforcement learning.

Consider the AGNews text classification dataset:

[1]:
```
# !wget http://www.cs.ait.ac.th/~mdailey/data.zip
```

[2]:
```
from torchtext.datasets import AG_NEWS
from torchtext.data.utils import get_tokenizer
from collections import Counter
from torchtext.vocab import Vocab

train_iter = AG_NEWS(split='train')
tokenizer = get_tokenizer('basic_english')
counter = Counter()

def clean(line):
    line = line.replace('\\', ' ')
    return line

labels = {}
for (label, line) in train_iter:
    if label in labels:
        labels[label] += 1
    else:
        labels[label] = 1
    counter.update(tokenizer(clean(line)))

vocab = Vocab(counter, min_freq=1)

print('Label frequencies:', labels)
print('A few token frequencies:', vocab.freqs.most_common(5))
print('Label meanings: 1: World news, 2: Sports news, 3: Business news, 4: Sci/
 ↪Tech news')
```

```
Label frequencies: {3: 30000, 4: 30000, 2: 30000, 1: 30000}
A few token frequencies: [('.', 225971), ('the', 205040), (',', 165685), ('to',
```

```
119817), ('a', 110942)]
Label meanings: 1: World news, 2: Sports news, 3: Business news, 4: Sci/Tech
news
```

Here's how we can get a sequence of tokens for a sentence with the cleaner, tokenizer, and vocabulary:

[3]:
```
[vocab[token] for token in tokenizer(clean('Bangkok, or The Big Mango, is one␣
↪of the great cities of Asia'))]
```

[3]:
```
[4248, 4, 116, 3, 244, 46857, 4, 23, 62, 7, 3, 812, 2009, 7, 989]
```

Let's make pipelines for processing a news story and a label:

[4]:
```
text_pipeline = lambda x: [vocab[token] for token in tokenizer(clean(x))]
label_pipeline = lambda x: int(x) - 1
```

Here's how to create dataloaders for the training and test datasets:

[5]:
```
import torch
from torch.utils.data import DataLoader
from torch.nn.utils.rnn import pad_sequence
device = torch.device("cuda:1" if torch.cuda.is_available() else "cpu")

def collate_batch(batch):
    label_list, text_list, length_list = [], [], []
    for (_label, _text) in batch:
        label_list.append(label_pipeline(_label))
        processed_text = torch.tensor(text_pipeline(_text), dtype=torch.int64)
        length_list.append(processed_text.shape[0])
        text_list.append(processed_text)
    label_list = torch.tensor(label_list, dtype=torch.int64)
    text_list = pad_sequence(text_list, padding_value=0)
    length_list = torch.tensor(length_list, dtype=torch.int64)
    return label_list.to(device), text_list.to(device), length_list.to(device)

train_iter = AG_NEWS(split='train')
train_dataset = list(train_iter)
test_iter = AG_NEWS(split='test')
test_dataset = list(test_iter)
train_dataloader = DataLoader(train_dataset, batch_size=8, shuffle=True,␣
↪collate_fn=collate_batch)
test_dataloader = DataLoader(test_dataset, batch_size=8, shuffle=False,␣
↪collate_fn=collate_batch)
```

Here's how to get a batch from one of these dataloaders. The first entry is a 1D tensor of labels for the batch (8 values between 0 and 3), then a 2D tensor representing the stories with dimension T x B (number of tokens x batch size).

[6]:
```
batch = next(enumerate(train_dataloader))
print(batch)
```

```
(0, (tensor([2, 1, 2, 0, 3, 1, 1, 1], device='cuda:1'), tensor([[ 1470,   761,
4465,  1381,   840,  8809,   426, 26032],
        [  188,  8055,  3937,  3846,  2019,  1648,  1482,  9958],
        [ 2279,  1816,   728,  5084,   135,    20,  9688,  3566],
        [   11,  2217,     5,  1438, 10870,   629,  3473,  7800],
        [ 1564,    20,    45,  8168,   137,  1406,    12, 26032],
        [    4,    52,   284,     8,  6056,    13, 13021,     7],
        [ 1771,     2,    11,   497,    37,    10,  3121,     3],
        [ 4233,    10,    45,  5931,   840,   343,    90,    89],
        [   67,     2,    35,  3846,    40,   232,  1696,   160],
        [ 1569,  5497,  4447,  4769,  2019,   629,  1159,  6186],
        [  145,   283,  1604,  5021,   135,  1406,  2984, 18865],
        [ 1030,  2604,  1334,  2839,   456,  1648,  9688, 36067],
        [  247,   358,  1678,    11,  2010,   142,    34, 31190],
        [ 1470,    14,   426,    65,   137,   213,    38,     7],
        [    9,    28, 36611,    18,  6056,  1406,  3473,     3],
        [  221,    15,  3937,  4484,   394,  2843,    12,  3566],
        [    2,    16,    27,   840,    25,  6778,   276,  1806],
        [   13,   426,    26,    96,     3,  8809,    13,  2736],
        [   10,  9440,    36,  3531,  2010,    20,    10,  2736],
        [  188,  4471,  4465,     5,   418,    22, 12505,     5],
        [ 6654,  8055,     6,  1438,   495,   368,   175,   680],
        [   35,   378,  1248,   928,    14,   343,  2938,     3],
        [    3, 14315,  1604,     5, 12582,   232,    64,  5416],
        [  321,   582,    19,  3242,    15,    11, 13021,     7],
        [ 1549,  2217,   327,   147,     9,    65,  3121,     3],
        [ 1564,     8,    24,    12,  2453,     4,    11,  4012],
        [    9,     3,  1311,     3,    49, 10882,    58,  1157],
        [ 1771,   435,    86, 10152, 10870,   285,    35,   476],
        [ 5780,    17,    83,  1413,   226,    20, 13366,   101],
        [   67,    10,  1334,     8,  1302,     3,   111,     2],
        [    3,  8264,    25,    31,     2,   336,     6,     0],
        [  746,  5553,    24,  2368,     0,     5,  8802,     0],
        [    7,    11,  1311,  6333,     0,   171,   907,     0],
        [   22,    56,    13,   620,     0,     3,    11,     0],
        [ 2864,    20,    10,   388,     0,   471,    56,     0],
        [  349,     3,   238,  1114,     0,    45,     2,     0],
        [ 1569,    89,   156,     2,     0,    67,     0,     0],
        [    2,   160,    46,     0,     0,     6,     0,     0],
        [    0, 10866,  1297,     0,     0,    24,     0,     0],
        [    0,   119,    61,     0,     0,  2419,     0,     0],
        [    0,     4,  8160,     0,     0,     2,     0,     0],
        [    0,   276,     3,     0,     0,     0,     0,     0],
        [    0,     9,   527,     0,     0,     0,     0,     0],
```

```
        [    0,   195,     2,     0,     0,     0,     0,     0],
        [    0,     8,     0,     0,     0,     0,     0,     0],
        [    0,     3,     0,     0,     0,     0,     0,     0],
        [    0,  2837,     0,     0,     0,     0,     0,     0],
        [    0,  3048,     0,     0,     0,     0,     0,     0],
        [    0,    11,     0,     0,     0,     0,     0,     0],
        [    0,   133,     0,     0,     0,     0,     0,     0],
        [    0,    95,     0,     0,     0,     0,     0,     0],
        [    0,     7,     0,     0,     0,     0,     0,     0],
        [    0,     3,     0,     0,     0,     0,     0,     0],
        [    0,   315,     0,     0,     0,     0,     0,     0],
        [    0,   223,     0,     0,     0,     0,     0,     0],
        [    0,     2,     0,     0,     0,     0,     0,     0]],
       device='cuda:1'), tensor([38, 56, 44, 37, 31, 41, 36, 30],
device='cuda:1')))
```

1.1 Question 1, 10 points

The vocabulary currently is too large for a simple one-hot embedding. Let's reduce the vocabulary size so that we can use one-hot. First, add a step that removes tokens from a list of "stop words" to the `text_pipeline` function. You probably want to remove punctuation (':', ',', '-', etc.) and articles ("a", "the").

Once you've removed stop words, modify the vocabulary to include only the most frequent 1000 tokens (including 0 for an unknown/infrequent word).

Write your revised code in the cell below and output the 999 top words with their frequencies:

[70]:
```
# Place code for Question 1 here
from torchtext.datasets import AG_NEWS
from torchtext.data.utils import get_tokenizer
from collections import Counter
from torchtext.vocab import Vocab

train_iter = AG_NEWS(split='train')
tokenizer = get_tokenizer('basic_english')
counter = Counter()

def clean(line):
    line = line.replace('\\', ' ')
    ### Before these replace
    # [('.', 225971),('the', 205040),(',', 165685),('a', 110942),('s',␣
↪61915),('on',␣
↪57279),('for',50417),('#39',44316),('(',41106),(')',40787),('-',39212),("'",32235),('that',
↪25324),('at', 24999)]
    ### remove but keep format
    line = line.replace('.',' ').replace(',',' ').replace('-',' ')
    line = line.replace(' the ',' ').replace('The ',' ').replace(' a ',' ').
↪replace('A ',' ')
```

```
    ### After the replace
    # I keep 's' '#39' because I don't know what are they and '(' ')' "'"␣
 ↪because I think it contribute to the context.
    # [('to', 120680), ('of', 98652), ('in', 96422), ('and', 69670), ('s',␣
 ↪62116), ('on', 57667), ('for', 50674), ('#39', 44316), ('(', 41106), (')',␣
 ↪40787), ("'", 32235), ('that', 28169), ('with', 26812), ('as', 25381),␣
 ↪('at', 25234), ('its', 22123), ('is', 22106), ('new', 21393), ('by', 20942),␣
 ↪('it', 20537)]
    return line

labels = {}
for (label, line) in train_iter:
    if label in labels:
        labels[label] += 1
    else:
        labels[label] = 1
    counter.update(tokenizer(clean(line)))

# Original
# vocab = Vocab(counter, min_freq=1, max_size=None, specials=('<unk>', '<pad>'))
# Modified
vocab = Vocab(counter, min_freq=1, max_size=1000-1, specials=('<unk>',))

print('Label frequencies:', labels)
print('A few token frequencies:', vocab.freqs.most_common(5))
print('Label meanings: 1: World news, 2: Sports news, 3: Business news, 4: Sci/
 ↪Tech news')
```

```
Label frequencies: {3: 30000, 4: 30000, 2: 30000, 1: 30000}
A few token frequencies: [('to', 120680), ('of', 98652), ('in', 96422), ('and',
69670), ('s', 62116)]
Label meanings: 1: World news, 2: Sports news, 3: Business news, 4: Sci/Tech
news
```

[75]:
```
# Write your revised code in the cell below and output the 999 top words with␣
 ↪their frequencies:
print("Top 999 freq\n",vocab.freqs.most_common(999))
```

```
Top 999 freq
 [('to', 120680), ('of', 98652), ('in', 96422), ('and', 69670), ('s', 62116),
('on', 57667), ('for', 50674), ('#39', 44316), ('(', 41106), (')', 40787), ("'",
32235), ('that', 28169), ('with', 26812), ('as', 25381), ('at', 25234), ('its',
22123), ('is', 22106), ('new', 21393), ('by', 20942), ('it', 20537), ('said',
20267), ('reuters', 19322), ('has', 19024), ('from', 17828), ('an', 17002),
('ap', 16200), ('his', 14942), ('will', 14615), ('after', 14552), ('was',
13730), ('us', 13463), ('be', 11931), ('over', 11356), ('have', 11213), ('up',
10743), ('their', 10530), ('two', 10226), ('&lt', 10208), ('first', 9802),
```

('are', 9794), ('year', 9772), ('quot', 9596), ('but', 9184), ('more', 9149), ('he', 8942), ('world', 8618), ('u', 8443), ('this', 8251), ('one', 8109), ('company', 7656), ('monday', 7616), ('oil', 7564), ('out', 7556), ('wednesday', 7531), ('tuesday', 7455), ('thursday', 7345), ('not', 7061), ('against', 6900), ('friday', 6870), ('inc', 6853), ('than', 6733), ('1', 6714), ('into', 6677), ('last', 6548), ('they', 6453), ('about', 6428), ('2', 6402), ('iraq', 6335), ('york', 6240), ('yesterday', 6099), ('who', 6086), ('three', 6035), ('president', 5993), ('no', 5947), ('microsoft', 5931), ('were', 5811), ('game', 5774), ('million', 5760), ('?', 5699), ('week', 5653), ('t', 5619), ('been', 5534), ('time', 5494), ('says', 5345), ('had', 5279), ('corp', 5170), ('united', 5128), ('when', 5022), ('sunday', 4930), ('prices', 4907), ('government', 4862), ('could', 4854), ('would', 4821), ('security', 4790), ('years', 4719), ('group', 4710), ('today', 4705), ('people', 4676), ('off', 4662), ('which', 4606), ('may', 4584), ('second', 4545), ('afp', 4524), ('percent', 4514), ('back', 4491), ('software', 4480), ('all', 4451), ('next', 4390), ('3', 4353), ('season', 4345), ('team', 4305), ('day', 4270), ('win', 4264), ('third', 4245), ('internet', 4085), ('saturday', 4044), ('night', 4014), ('or', 4008), ('quarter', 3997), ('high', 3956), ('china', 3945), ('top', 3893), ('state', 3875), ('6', 3845), ('market', 3798), ('can', 3795), ('deal', 3793), ('some', 3764), ('sales', 3739), ('open', 3734), ('four', 3715), ('minister', 3714), ('bush', 3678), ('billion', 3598), ('record', 3592), ('4', 3588), ('down', 3571), ('business', 3560), ('end', 3542), ('2004', 3496), ('news', 3447), ('announced', 3445), ('n', 3440), ('international', 3425), ('most', 3385), ('former', 3380), ('washington', 3374), ('killed', 3353), ('10', 3294), ('profit', 3256), ('5', 3243), ('report', 3242), ('victory', 3225), ('officials', 3223), ('city', 3200), ('court', 3198), ('service', 3161), ('000', 3142), ('home', 3140), ('stocks', 3139), ('plans', 3134), ('month', 3110), ('set', 3087), ('if', 3077), ('other', 3063), ('states', 3052), ('you', 3030), ('european', 3022), ('before', 3008), ('chief', 2998), ('american', 2985), ('/b&gt', 2984), ('b&gt', 2983), ('com', 2960), ('technology', 2956), ('lead', 2933), ('7', 2901), ('search', 2898), ('computer', 2896), ('league', 2889), ('talks', 2877), ('cup', 2849), ('just', 2838), ('space', 2817), ('online', 2810), ('five', 2808), ('0', 2794), ('country', 2776), ('what', 2745), ('now', 2731), ('national', 2728), ('co', 2723), ('british', 2722), ('expected', 2713), ('largest', 2709), ('reported', 2706), ('take', 2698), ('shares', 2671), ('red', 2670), ('india', 2645), ('bank', 2639), ('japan', 2629), ('won', 2627), ('federal', 2623), ('google', 2611), ('prime', 2611), ('major', 2602), ('network', 2597), ('final', 2593), ('police', 2570), ('under', 2566), ('least', 2566), ('make', 2562), ('iraqi', 2545), ('ibm', 2540), ('election', 2539), ('web', 2533), ('hit', 2529), ('another', 2526), ('research', 2523), ('south', 2520), ('music', 2511), ('made', 2503), ('according', 2489), ('maker', 2479), ('long', 2477), ('bid', 2457), ('leader', 2456), ('companies', 2451), ('help', 2436), ('get', 2435), ('big', 2427), ('mobile', 2419), ('while', 2415), ('games', 2406), ('during', 2403), ('there', 2378), ('series', 2377), ('san', 2371), ('coach', 2365), ('between', 2362), ('london', 2358), ('still', 2357), ('plan', 2352), ('industry', 2342), ('say', 2342), ('way', 2305), ('baghdad', 2283), ('war', 2271), ('dollar', 2264), ('old', 2254), ('run', 2240), ('them', 2238), ('giant', 2237), ('based', 2215), ('growth', 2199), ('i', 2189),

('services', 2187), ('since', 2186), ('al', 2184), ('through', 2179), ('early', 2169), ('data', 2145), ('north', 2145), ('investor', 2130), ('her', 2126), ('nuclear', 2125), ('wireless', 2122), ('//www', 2120), ('sports', 2119), ('href=http', 2117), ('/a&gt', 2117), ('start', 2116), ('system', 2115), ('trade', 2112), ('higher', 2111), ('cut', 2107), ('phone', 2098), ('left', 2076), ('six', 2074), ('military', 2074), ('gold', 2065), ('earnings', 2057), ('test', 2053), ('union', 2049), ('buy', 2048), ('so', 2043), ('australia', 2042), ('being', 2032), ('john', 2026), ('only', 2020), ('un', 2006), ('move', 2000), ('stock', 1992), ('him', 1992), ('palestinian', 1978), ('loss', 1969), ('like', 1966), ('players', 1961), ('official', 1957), ('general', 1928), ('apple', 1923), ('amp', 1922), ('months', 1920), ('agreed', 1911), ('sox', 1905), ('go', 1902), ('days', 1898), ('half', 1896), ('man', 1882), ('because', 1880), ('air', 1878), ('oracle', 1875), ('attack', 1862), ('israeli', 1858), ('8', 1855), ('windows', 1849), ('many', 1846), ('ahead', 1841), ('latest', 1835), ('global', 1824), ('olympic', 1822), ('troops', 1815), ('com/fullquote', 1813), ('aspx', 1813), ('target=/stocks/quickinfo/fullquote&gt', 1813), ('price', 1808), ('england', 1806), ('intel', 1804), ('play', 1801), ('again', 1799), ('biggest', 1794), ('russia', 1790), ('free', 1790), ('firm', 1785), ('executive', 1784), ('west', 1782), ('rose', 1781), ('round', 1781), ('found', 1774), ('held', 1769), ('even', 1762), ('press', 1761), ('near', 1756), ('head', 1748), ('jobs', 1746), ('users', 1746), ('drug', 1745), ('including', 1738), ('pay', 1737), ('russian', 1735), ('boston', 1733), ('iran', 1731), ('rise', 1731), ('ago', 1723), ('reports', 1718), ('update', 1716), ('football', 1708), ('released', 1701), ('despite', 1699), ('economy', 1690), ('points', 1683), ('part', 1680), ('europe', 1679), ('car', 1670), ('much', 1670), ('peoplesoft', 1667), ('20', 1664), ('forces', 1662), ('athens', 1661), ('how', 1660), ('investors', 1656), ('past', 1655), ('e', 1649), ('economic', 1645), ('release', 1643), ('peace', 1643), ('canadian', 1640), ('power', 1631), ('gaza', 1628), ('street', 1619), ('key', 1618), ('o', 1617), ('pakistan', 1608), ('eu', 1601), ('offer', 1599), ('work', 1594), ('your', 1589), ('video', 1587), ('2005', 1586), ('beat', 1584), ('use', 1580), ('strong', 1580), ('11', 1577), ('uk', 1573), ('public', 1568), ('seven', 1565), ('case', 1564), ('fourth', 1558), ('nation', 1551), ('source', 1549), ('presidential', 1540), ('bomb', 1538), ('share', 1528), ('foreign', 1527), ('where', 1523), ('right', 1520), ('title', 1517), ('should', 1516), ('nations', 1514), ('weeks', 1514), ('nearly', 1511), ('pc', 1505), ('sun', 1503), ('close', 1502), ('do', 1501), ('around', 1494), ('called', 1492), ('12', 1490), ('workers', 1488), ('tokyo', 1486), ('america', 1483), ('israel', 1483), ('linux', 1481), ('9', 1480), ('lower', 1480), ('attacks', 1477), ('life', 1475), ('financial', 1473), ('systems', 1473), ('wins', 1471), ('support', 1463), ('media', 1463), ('french', 1462), ('australian', 1454), ('kerry', 1450), ('house', 1447), ('championship', 1445), ('face', 1439), ('korea', 1436), ('agency', 1435), ('launch', 1434), ('contract', 1432), ('best', 1426), ('wall', 1423), ('st', 1422), ('fall', 1421), ('leaders', 1415), ('digital', 1412), ('francisco', 1412), ('low', 1411), ('put', 1407), ('men', 1403), ('late', 1401), ('leading', 1400), ('player', 1395), ('number', 1394), ('30', 1392), ('fell', 1387), ('september', 1382), ('october', 1380), ('line', 1379), ('party', 1378), ('demand', 1378), ('crude', 1371), ('used', 1371), ('also', 1370), ('chicago',

1370), ('led', 1369), ('death', 1369), ('took', 1366), ('net', 1364),
('scientists', 1362), ('florida', 1357), ('hurricane', 1351), ('any', 1347),
('rival', 1345), ('following', 1345), ('good', 1344), ('job', 1343), ('race',
1342), ('consumer', 1340), ('killing', 1340), ('return', 1334), ('arafat',
1333), ('darfur', 1329), ('chip', 1329), ('15', 1327), ('nas', 1321), ('per',
1320), ('capital', 1319), ('army', 1318), ('program', 1317), ('vote', 1314),
('here', 1309), ('version', 1309), ('#36', 1307), ('charges', 1307), ('center',
1304), ('france', 1303), ('keep', 1301), ('japanese', 1296), ('commission',
1294), ('agreement', 1289), ('campaign', 1289), ('yankees', 1289), ('school',
1287), ('future', 1287), ('trial', 1287), ('bill', 1286), ('both', 1284),
('quote', 1278), ('site', 1266), ('well', 1263), ('making', 1262), ('star',
1261), ('products', 1256), ('strike', 1255), ('we', 1252), ('profile', 1252),
('dead', 1250), ('region', 1248), ('give', 1247), ('study', 1242), ('anti',
1242), ('n&lt', 1241), ('show', 1239), ('britain', 1238), ('board', 1237),
('17', 1235), ('women', 1234), ('management', 1226), ('customers', 1223),
('might', 1222), ('defense', 1220), ('results', 1220), ('tech', 1219),
('battle', 1218), ('away', 1217), ('conference', 1217), ('decision', 1215),
('several', 1214), ('energy', 1213), ('office', 1211), ('november', 1210),
('little', 1207), ('$1', 1205), ('14', 1203), ('eight', 1199), ('northern',
1195), ('sony', 1195), ('manager', 1194), ('costs', 1191), ('place', 1191),
('meeting', 1191), ('without', 1189), ('southern', 1186), ('political', 1186),
('militants', 1185), ('shot', 1185), ('sudan', 1183), ('hostage', 1183), ('18',
1181), ('running', 1179), ('sell', 1177), ('elections', 1173), ('gets', 1172),
('takes', 1171), ('baseball', 1165), ('champions', 1164), ('health', 1162),
('field', 1161), ('recent', 1160), ('judge', 1158), ('taking', 1157), ('re',
1156), ('cost', 1153), ('champion', 1153), ('cuts', 1152), ('due', 1151),
('real', 1150), ('canada', 1150), ('fight', 1149), ('tax', 1147), ('california',
1147), ('interest', 1145), ('winning', 1145), ('13', 1144), ('central', 1138),
('such', 1137), ('straight', 1137), ('match', 1136), ('quarterly', 1133),
('force', 1131), ('mail', 1127), ('fans', 1126), ('16', 1126), ('hopes', 1125),
('department', 1119), ('told', 1119), ('m', 1116), ('ceo', 1115), ('tv', 1115),
('claims', 1112), ('she', 1111), ('whether', 1105), ('los', 1105), ('better',
1104), ('michael', 1103), ('rate', 1093), ('using', 1091), ('times', 1089),
('giants', 1087), ('likely', 1083), ('scored', 1083), ('east', 1082),
('increase', 1081), ('grand', 1078), ('across', 1074), ('own', 1073),
('launched', 1072), ('violence', 1071), ('look', 1066), ('university', 1066),
('amid', 1065), ('small', 1065), ('mark', 1065), ('lost', 1063), ('saying',
1063), ('become', 1061), ('texas', 1061), ('club', 1059), ('angeles', 1058),
('action', 1056), ('25', 1055), ('history', 1055), ('rates', 1051), ('chairman',
1051), ('chinese', 1044), ('trading', 1044), ('unit', 1044), ('fire', 1041),
('houston', 1039), ('williams', 1036), ('stores', 1035), ('weekend', 1034),
('server', 1029), ('hits', 1025), ('among', 1021), ('too', 1019), ('control',
1019), ('germany', 1019), ('c', 1013), ('100', 1012), ('cash', 1006), ('same',
1006), ('morning', 1006), ('rebels', 1005), ('secretary', 1005), ('call', 1004),
('desktop', 1004), ('come', 1003), ('behind', 1003), ('information', 1002),
('cell', 998), ('oct', 998), ('accused', 996), ('change', 993), ('phones', 992),
('yahoo', 986), ('soldiers', 984), ('seen', 983), ('going', 983), ('possible',
983), ('german', 982), ('drop', 981), ('aid', 981), ('ipod', 979), ('injured',

979), ('profits', 978), ('afghanistan', 978), ('boost', 977), ('white', 976), ('operating', 975), ('makes', 975), ('hold', 975), ('meet', 974), ('law', 973), ('see', 973), ('august', 967), ('communications', 967), ('senior', 964), ('few', 964), ('ever', 964), ('exchange', 964), ('revenue', 959), ('almost', 958), ('24', 958), ('calls', 957), ('store', 957), ('less', 956), ('speed', 955), ('warned', 955), ('bankruptcy', 955), ('reach', 954), ('did', 954), ('nine', 954), ('radio', 948), ('barrel', 947), ('point', 946), ('thousands', 946), ('medal', 945), ('hard', 943), ('rally', 942), ('indian', 940), ('nov', 940), ('access', 937), ('paris', 937), ('full', 936), ('gas', 936), ('airlines', 936), ('takeover', 933), ('posted', 930), ('ready', 928), ('food', 928), ('19', 925), ('reserve', 924), ('showed', 923), ('money', 923), ('euro', 921), ('file', 920), ('signed', 918), ('engine', 916), ('human', 916), ('fund', 915), ('term', 915), ('those', 914), ('production', 913), ('coast', 912), ('television', 912), ('afghan', 912), ('opening', 911), ('forecast', 911), ('sign', 911), ('must', 911), ('stadium', 911), ('africa', 910), ('yet', 910), ('terror', 909), ('toronto', 905), ('earlier', 904), ('tour', 902), ('step', 900), ('21', 900), ('david', 899), ('nfl', 898), ('short', 897), ('fuel', 897), ('markets', 895), ('growing', 895), ('offering', 892), ('got', 890), ('stop', 890), ('wants', 889), ('spain', 889), ('division', 888), ('soon', 887), ('post', 884), ('opposition', 882), ('outlook', 880), ('find', 879), ('jones', 877), ('sale', 876), ('helped', 873), ('olympics', 872), ('blue', 872), ('performance', 871), ('begin', 868), ('far', 867), ('stake', 865), ('association', 863), ('probe', 862), ('americans', 861), ('rebel', 859), ('corporate', 859), ('began', 858), ('concerns', 856), ('sept', 856), ('blair', 855), ('african', 854), ('the', 853), ('product', 851), ('local', 851), ('green', 849), ('came', 849), ('ltd', 849), ('!', 846), ('gains', 846), ('insurance', 845), ('target', 844), ('p', 844), ('networks', 840), ('threat', 839), ('western', 838), ('flight', 838), ('ended', 837), ('efforts', 837), ('countries', 835), ('legal', 834), ('died', 833), ('administration', 832), ('members', 832), ('career', 832), ('once', 830), ('already', 829), ('station', 829), ('airline', 829), ('investment', 828), ('arrested', 826), ('fighting', 825), ('list', 823), ('analysts', 822), ('looking', 819), ('council', 818), ('spending', 817), ('miami', 817), ('trying', 816), ('d', 815), ('weapons', 814), ('hours', 814), ('personal', 813), ('great', 813), ('competition', 812), ('within', 812), ('need', 811), ('teams', 810), ('family', 809), ('arsenal', 809), ('drive', 807), ('holiday', 805), ('visit', 805), ('continue', 804), ('computers', 804), ('supply', 803), ('madrid', 802), ('heart', 801), ('others', 801), ('pm', 800), ('rules', 799), ('charged', 799), ('airways', 796), ('spam', 793), ('rights', 793), ('black', 789), ('don', 788), ('outside', 787), ('beijing', 787), ('retail', 786), ('george', 784), ('until', 782), ('sharon', 782), ('starting', 779), ('pressure', 778), ('crisis', 778), ('paul', 775), ('authorities', 774), ('selling', 774), ('enough', 774), ('gave', 770), ('coming', 770), ('dell', 768), ('50', 768), ('offers', 766), ('storm', 766), ('consumers', 766), ('development', 765), ('italian', 765), ('finally', 763), ('manchester', 763), ('effort', 761), ('seattle', 761), ('ruling', 761), ('toward', 760), ('asia', 760), ('bay', 760), ('yards', 759), ('executives', 758), ('reached', 758), ('quarterback', 757), ('injury', 756), ('suicide', 756), ('children', 755), ('fears', 755), ('ban', 753), ('popular', 752), ('each', 752), ('#151', 750),

```
('then', 750), ('bowl', 750), ('retailer', 748), ('labor', 747), ('sees', 746),
('policy', 746), ('devices', 746), ('getting', 743), ('side', 743), ('charge',
743), ('survey', 742), ('taken', 741), ('push', 740), ('cisco', 740), ('aimed',
739), ('nb', 739), ('electronics', 736), ('failed', 736), ('blast', 734),
('large', 731), ('rivals', 730), ('yukos', 730), ('researchers', 728),
('operations', 728), ('continued', 728), ('shows', 727), ('problems', 726),
('double', 725), ('private', 724), ('college', 724), ('main', 723), ('kill',
723), ('goal', 721), ('known', 720), ('further', 720), ('ivan', 720), ('turn',
718), ('powerful', 717), ('rising', 717), ('basketball', 716), ('fraud', 716),
('yasser', 716), ('hope', 713), ('want', 713), ('name', 713), ('working', 712),
('moscow', 712), ('fired', 711), ('soccer', 710), ('area', 708), ('missing',
707), ('risk', 706), ('palestinians', 705), ('raised', 705), ('named', 704),
('card', 704), ('focus', 703), ('davis', 703), ('filed', 703), ('calif', 701),
('asian', 699), ('chips', 699), ('above', 698), ('kills', 698), ('suspected',
697), ('merger', 697), ('level', 696), ('winter', 695), ('warning', 694),
('leave', 693), ('signs', 693), ('28', 692), ('faces', 689), ('firms', 689),
('defeat', 689), ('challenge', 687), ('terrorism', 687), ('mission', 684),
('debt', 683), ('never', 682), ('designed', 681), ('muslim', 680), ('2006',
678), ('annual', 677), ('planned', 677), ('park', 677), ('27', 677), ('chance',
675), ('businesses', 674), ('23', 674), ('satellite', 673), ('town', 671),
('issue', 671), ('investigation', 671), ('try', 670), ('storage', 670),
('fresh', 669), ('order', 667), ('dollars', 666), ('track', 666), ('allow',
665), ('expectations', 664)]
```

[76]:
```
print("1000 words in vocab\n",vocab.stoi)
```

```
1000 words in vocab
 defaultdict(<bound method Vocab._default_unk_index of <torchtext.vocab.Vocab
object at 0x7f098de0e8b0>>, {'<unk>': 0, 'to': 1, 'of': 2, 'in': 3, 'and': 4,
's': 5, 'on': 6, 'for': 7, '#39': 8, '(': 9, ')': 10, "'": 11, 'that': 12,
'with': 13, 'as': 14, 'at': 15, 'its': 16, 'is': 17, 'new': 18, 'by': 19, 'it':
20, 'said': 21, 'reuters': 22, 'has': 23, 'from': 24, 'an': 25, 'ap': 26, 'his':
27, 'will': 28, 'after': 29, 'was': 30, 'us': 31, 'be': 32, 'over': 33, 'have':
34, 'up': 35, 'their': 36, 'two': 37, '&lt': 38, 'first': 39, 'are': 40, 'year':
41, 'quot': 42, 'but': 43, 'more': 44, 'he': 45, 'world': 46, 'u': 47, 'this':
48, 'one': 49, 'company': 50, 'monday': 51, 'oil': 52, 'out': 53, 'wednesday':
54, 'tuesday': 55, 'thursday': 56, 'not': 57, 'against': 58, 'friday': 59,
'inc': 60, 'than': 61, '1': 62, 'into': 63, 'last': 64, 'they': 65, 'about': 66,
'2': 67, 'iraq': 68, 'york': 69, 'yesterday': 70, 'who': 71, 'three': 72,
'president': 73, 'no': 74, 'microsoft': 75, 'were': 76, 'game': 77, 'million':
78, '?': 79, 'week': 80, 't': 81, 'been': 82, 'time': 83, 'says': 84, 'had': 85,
'corp': 86, 'united': 87, 'when': 88, 'sunday': 89, 'prices': 90, 'government':
91, 'could': 92, 'would': 93, 'security': 94, 'years': 95, 'group': 96, 'today':
97, 'people': 98, 'off': 99, 'which': 100, 'may': 101, 'second': 102, 'afp':
103, 'percent': 104, 'back': 105, 'software': 106, 'all': 107, 'next': 108, '3':
109, 'season': 110, 'team': 111, 'day': 112, 'win': 113, 'third': 114,
'internet': 115, 'saturday': 116, 'night': 117, 'or': 118, 'quarter': 119,
'high': 120, 'china': 121, 'top': 122, 'state': 123, '6': 124, 'market': 125,
```

'can': 126, 'deal': 127, 'some': 128, 'sales': 129, 'open': 130, 'four': 131,
'minister': 132, 'bush': 133, 'billion': 134, 'record': 135, '4': 136, 'down':
137, 'business': 138, 'end': 139, '2004': 140, 'news': 141, 'announced': 142,
'n': 143, 'international': 144, 'most': 145, 'former': 146, 'washington': 147,
'killed': 148, '10': 149, 'profit': 150, '5': 151, 'report': 152, 'victory':
153, 'officials': 154, 'city': 155, 'court': 156, 'service': 157, '000': 158,
'home': 159, 'stocks': 160, 'plans': 161, 'month': 162, 'set': 163, 'if': 164,
'other': 165, 'states': 166, 'you': 167, 'european': 168, 'before': 169,
'chief': 170, 'american': 171, '/b&gt': 172, 'b&gt': 173, 'com': 174,
'technology': 175, 'lead': 176, '7': 177, 'search': 178, 'computer': 179,
'league': 180, 'talks': 181, 'cup': 182, 'just': 183, 'space': 184, 'online':
185, 'five': 186, '0': 187, 'country': 188, 'what': 189, 'now': 190, 'national':
191, 'co': 192, 'british': 193, 'expected': 194, 'largest': 195, 'reported':
196, 'take': 197, 'shares': 198, 'red': 199, 'india': 200, 'bank': 201, 'japan':
202, 'won': 203, 'federal': 204, 'google': 205, 'prime': 206, 'major': 207,
'network': 208, 'final': 209, 'police': 210, 'least': 211, 'under': 212, 'make':
213, 'iraqi': 214, 'ibm': 215, 'election': 216, 'web': 217, 'hit': 218,
'another': 219, 'research': 220, 'south': 221, 'music': 222, 'made': 223,
'according': 224, 'maker': 225, 'long': 226, 'bid': 227, 'leader': 228,
'companies': 229, 'help': 230, 'get': 231, 'big': 232, 'mobile': 233, 'while':
234, 'games': 235, 'during': 236, 'there': 237, 'series': 238, 'san': 239,
'coach': 240, 'between': 241, 'london': 242, 'still': 243, 'plan': 244,
'industry': 245, 'say': 246, 'way': 247, 'baghdad': 248, 'war': 249, 'dollar':
250, 'old': 251, 'run': 252, 'them': 253, 'giant': 254, 'based': 255, 'growth':
256, 'i': 257, 'services': 258, 'since': 259, 'al': 260, 'through': 261,
'early': 262, 'data': 263, 'north': 264, 'investor': 265, 'her': 266, 'nuclear':
267, 'wireless': 268, '//www': 269, 'sports': 270, '/a&gt': 271, 'href=http':
272, 'start': 273, 'system': 274, 'trade': 275, 'higher': 276, 'cut': 277,
'phone': 278, 'left': 279, 'military': 280, 'six': 281, 'gold': 282, 'earnings':
283, 'test': 284, 'union': 285, 'buy': 286, 'so': 287, 'australia': 288,
'being': 289, 'john': 290, 'only': 291, 'un': 292, 'move': 293, 'him': 294,
'stock': 295, 'palestinian': 296, 'loss': 297, 'like': 298, 'players': 299,
'official': 300, 'general': 301, 'apple': 302, 'amp': 303, 'months': 304,
'agreed': 305, 'sox': 306, 'go': 307, 'days': 308, 'half': 309, 'man': 310,
'because': 311, 'air': 312, 'oracle': 313, 'attack': 314, 'israeli': 315, '8':
316, 'windows': 317, 'many': 318, 'ahead': 319, 'latest': 320, 'global': 321,
'olympic': 322, 'troops': 323, 'aspx': 324, 'com/fullquote': 325,
'target=/stocks/quickinfo/fullquote&gt': 326, 'price': 327, 'england': 328,
'intel': 329, 'play': 330, 'again': 331, 'biggest': 332, 'free': 333, 'russia':
334, 'firm': 335, 'executive': 336, 'west': 337, 'rose': 338, 'round': 339,
'found': 340, 'held': 341, 'even': 342, 'press': 343, 'near': 344, 'head': 345,
'jobs': 346, 'users': 347, 'drug': 348, 'including': 349, 'pay': 350, 'russian':
351, 'boston': 352, 'iran': 353, 'rise': 354, 'ago': 355, 'reports': 356,
'update': 357, 'football': 358, 'released': 359, 'despite': 360, 'economy': 361,
'points': 362, 'part': 363, 'europe': 364, 'car': 365, 'much': 366,
'peoplesoft': 367, '20': 368, 'forces': 369, 'athens': 370, 'how': 371,
'investors': 372, 'past': 373, 'e': 374, 'economic': 375, 'peace': 376,
'release': 377, 'canadian': 378, 'power': 379, 'gaza': 380, 'street': 381,

'key': 382, 'o': 383, 'pakistan': 384, 'eu': 385, 'offer': 386, 'work': 387,
'your': 388, 'video': 389, '2005': 390, 'beat': 391, 'strong': 392, 'use': 393,
'11': 394, 'uk': 395, 'public': 396, 'seven': 397, 'case': 398, 'fourth': 399,
'nation': 400, 'source': 401, 'presidential': 402, 'bomb': 403, 'share': 404,
'foreign': 405, 'where': 406, 'right': 407, 'title': 408, 'should': 409,
'nations': 410, 'weeks': 411, 'nearly': 412, 'pc': 413, 'sun': 414, 'close':
415, 'do': 416, 'around': 417, 'called': 418, '12': 419, 'workers': 420,
'tokyo': 421, 'america': 422, 'israel': 423, 'linux': 424, '9': 425, 'lower':
426, 'attacks': 427, 'life': 428, 'financial': 429, 'systems': 430, 'wins': 431,
'media': 432, 'support': 433, 'french': 434, 'australian': 435, 'kerry': 436,
'house': 437, 'championship': 438, 'face': 439, 'korea': 440, 'agency': 441,
'launch': 442, 'contract': 443, 'best': 444, 'wall': 445, 'st': 446, 'fall':
447, 'leaders': 448, 'digital': 449, 'francisco': 450, 'low': 451, 'put': 452,
'men': 453, 'late': 454, 'leading': 455, 'player': 456, 'number': 457, '30':
458, 'fell': 459, 'september': 460, 'october': 461, 'line': 462, 'demand': 463,
'party': 464, 'crude': 465, 'used': 466, 'also': 467, 'chicago': 468, 'death':
469, 'led': 470, 'took': 471, 'net': 472, 'scientists': 473, 'florida': 474,
'hurricane': 475, 'any': 476, 'following': 477, 'rival': 478, 'good': 479,
'job': 480, 'race': 481, 'consumer': 482, 'killing': 483, 'return': 484,
'arafat': 485, 'chip': 486, 'darfur': 487, '15': 488, 'nas': 489, 'per': 490,
'capital': 491, 'army': 492, 'program': 493, 'vote': 494, 'here': 495,
'version': 496, '#36': 497, 'charges': 498, 'center': 499, 'france': 500,
'keep': 501, 'japanese': 502, 'commission': 503, 'agreement': 504, 'campaign':
505, 'yankees': 506, 'future': 507, 'school': 508, 'trial': 509, 'bill': 510,
'both': 511, 'quote': 512, 'site': 513, 'well': 514, 'making': 515, 'star': 516,
'products': 517, 'strike': 518, 'profile': 519, 'we': 520, 'dead': 521,
'region': 522, 'give': 523, 'anti': 524, 'study': 525, 'n&lt': 526, 'show': 527,
'britain': 528, 'board': 529, '17': 530, 'women': 531, 'management': 532,
'customers': 533, 'might': 534, 'defense': 535, 'results': 536, 'tech': 537,
'battle': 538, 'away': 539, 'conference': 540, 'decision': 541, 'several': 542,
'energy': 543, 'office': 544, 'november': 545, 'little': 546, '$1': 547, '14':
548, 'eight': 549, 'northern': 550, 'sony': 551, 'manager': 552, 'costs': 553,
'meeting': 554, 'place': 555, 'without': 556, 'political': 557, 'southern': 558,
'militants': 559, 'shot': 560, 'hostage': 561, 'sudan': 562, '18': 563,
'running': 564, 'sell': 565, 'elections': 566, 'gets': 567, 'takes': 568,
'baseball': 569, 'champions': 570, 'health': 571, 'field': 572, 'recent': 573,
'judge': 574, 'taking': 575, 're': 576, 'champion': 577, 'cost': 578, 'cuts':
579, 'due': 580, 'canada': 581, 'real': 582, 'fight': 583, 'california': 584,
'tax': 585, 'interest': 586, 'winning': 587, '13': 588, 'central': 589,
'straight': 590, 'such': 591, 'match': 592, 'quarterly': 593, 'force': 594,
'mail': 595, '16': 596, 'fans': 597, 'hopes': 598, 'department': 599, 'told':
600, 'm': 601, 'ceo': 602, 'tv': 603, 'claims': 604, 'she': 605, 'los': 606,
'whether': 607, 'better': 608, 'michael': 609, 'rate': 610, 'using': 611,
'times': 612, 'giants': 613, 'likely': 614, 'scored': 615, 'east': 616,
'increase': 617, 'grand': 618, 'across': 619, 'own': 620, 'launched': 621,
'violence': 622, 'look': 623, 'university': 624, 'amid': 625, 'mark': 626,
'small': 627, 'lost': 628, 'saying': 629, 'become': 630, 'texas': 631, 'club':
632, 'angeles': 633, 'action': 634, '25': 635, 'history': 636, 'chairman': 637,

'rates': 638, 'chinese': 639, 'trading': 640, 'unit': 641, 'fire': 642,
'houston': 643, 'williams': 644, 'stores': 645, 'weekend': 646, 'server': 647,
'hits': 648, 'among': 649, 'control': 650, 'germany': 651, 'too': 652, 'c': 653,
'100': 654, 'cash': 655, 'morning': 656, 'same': 657, 'rebels': 658,
'secretary': 659, 'call': 660, 'desktop': 661, 'behind': 662, 'come': 663,
'information': 664, 'cell': 665, 'oct': 666, 'accused': 667, 'change': 668,
'phones': 669, 'yahoo': 670, 'soldiers': 671, 'going': 672, 'possible': 673,
'seen': 674, 'german': 675, 'aid': 676, 'drop': 677, 'injured': 678, 'ipod':
679, 'afghanistan': 680, 'profits': 681, 'boost': 682, 'white': 683, 'hold':
684, 'makes': 685, 'operating': 686, 'meet': 687, 'law': 688, 'see': 689,
'august': 690, 'communications': 691, 'ever': 692, 'exchange': 693, 'few': 694,
'senior': 695, 'revenue': 696, '24': 697, 'almost': 698, 'calls': 699, 'store':
700, 'less': 701, 'bankruptcy': 702, 'speed': 703, 'warned': 704, 'did': 705,
'nine': 706, 'reach': 707, 'radio': 708, 'barrel': 709, 'point': 710,
'thousands': 711, 'medal': 712, 'hard': 713, 'rally': 714, 'indian': 715, 'nov':
716, 'access': 717, 'paris': 718, 'airlines': 719, 'full': 720, 'gas': 721,
'takeover': 722, 'posted': 723, 'food': 724, 'ready': 725, '19': 726, 'reserve':
727, 'money': 728, 'showed': 729, 'euro': 730, 'file': 731, 'signed': 732,
'engine': 733, 'human': 734, 'fund': 735, 'term': 736, 'those': 737,
'production': 738, 'afghan': 739, 'coast': 740, 'television': 741, 'forecast':
742, 'must': 743, 'opening': 744, 'sign': 745, 'stadium': 746, 'africa': 747,
'yet': 748, 'terror': 749, 'toronto': 750, 'earlier': 751, 'tour': 752, '21':
753, 'step': 754, 'david': 755, 'nfl': 756, 'fuel': 757, 'short': 758,
'growing': 759, 'markets': 760, 'offering': 761, 'got': 762, 'stop': 763,
'spain': 764, 'wants': 765, 'division': 766, 'soon': 767, 'post': 768,
'opposition': 769, 'outlook': 770, 'find': 771, 'jones': 772, 'sale': 773,
'helped': 774, 'blue': 775, 'olympics': 776, 'performance': 777, 'begin': 778,
'far': 779, 'stake': 780, 'association': 781, 'probe': 782, 'americans': 783,
'corporate': 784, 'rebel': 785, 'began': 786, 'concerns': 787, 'sept': 788,
'blair': 789, 'african': 790, 'the': 791, 'local': 792, 'product': 793, 'came':
794, 'green': 795, 'ltd': 796, '!': 797, 'gains': 798, 'insurance': 799, 'p':
800, 'target': 801, 'networks': 802, 'threat': 803, 'flight': 804, 'western':
805, 'efforts': 806, 'ended': 807, 'countries': 808, 'legal': 809, 'died': 810,
'administration': 811, 'career': 812, 'members': 813, 'once': 814, 'airline':
815, 'already': 816, 'station': 817, 'investment': 818, 'arrested': 819,
'fighting': 820, 'list': 821, 'analysts': 822, 'looking': 823, 'council': 824,
'miami': 825, 'spending': 826, 'trying': 827, 'd': 828, 'hours': 829, 'weapons':
830, 'great': 831, 'personal': 832, 'competition': 833, 'within': 834, 'need':
835, 'teams': 836, 'arsenal': 837, 'family': 838, 'drive': 839, 'holiday': 840,
'visit': 841, 'computers': 842, 'continue': 843, 'supply': 844, 'madrid': 845,
'heart': 846, 'others': 847, 'pm': 848, 'charged': 849, 'rules': 850, 'airways':
851, 'rights': 852, 'spam': 853, 'black': 854, 'don': 855, 'beijing': 856,
'outside': 857, 'retail': 858, 'george': 859, 'sharon': 860, 'until': 861,
'starting': 862, 'crisis': 863, 'pressure': 864, 'paul': 865, 'authorities':
866, 'enough': 867, 'selling': 868, 'coming': 869, 'gave': 870, '50': 871,
'dell': 872, 'consumers': 873, 'offers': 874, 'storm': 875, 'development': 876,
'italian': 877, 'finally': 878, 'manchester': 879, 'effort': 880, 'ruling': 881,
'seattle': 882, 'asia': 883, 'bay': 884, 'toward': 885, 'yards': 886,

```
'executives': 887, 'reached': 888, 'quarterback': 889, 'injury': 890, 'suicide':
891, 'children': 892, 'fears': 893, 'ban': 894, 'each': 895, 'popular': 896,
'#151': 897, 'bowl': 898, 'then': 899, 'retailer': 900, 'labor': 901, 'devices':
902, 'policy': 903, 'sees': 904, 'charge': 905, 'getting': 906, 'side': 907,
'survey': 908, 'taken': 909, 'cisco': 910, 'push': 911, 'aimed': 912, 'nb': 913,
'electronics': 914, 'failed': 915, 'blast': 916, 'large': 917, 'rivals': 918,
'yukos': 919, 'continued': 920, 'operations': 921, 'researchers': 922, 'shows':
923, 'problems': 924, 'double': 925, 'college': 926, 'private': 927, 'kill':
928, 'main': 929, 'goal': 930, 'further': 931, 'ivan': 932, 'known': 933,
'turn': 934, 'powerful': 935, 'rising': 936, 'basketball': 937, 'fraud': 938,
'yasser': 939, 'hope': 940, 'name': 941, 'want': 942, 'moscow': 943, 'working':
944, 'fired': 945, 'soccer': 946, 'area': 947, 'missing': 948, 'risk': 949,
'palestinians': 950, 'raised': 951, 'card': 952, 'named': 953, 'davis': 954,
'filed': 955, 'focus': 956, 'calif': 957, 'asian': 958, 'chips': 959, 'above':
960, 'kills': 961, 'merger': 962, 'suspected': 963, 'level': 964, 'winter': 965,
'warning': 966, 'leave': 967, 'signs': 968, '28': 969, 'defeat': 970, 'faces':
971, 'firms': 972, 'challenge': 973, 'terrorism': 974, 'mission': 975, 'debt':
976, 'never': 977, 'designed': 978, 'muslim': 979, '2006': 980, '27': 981,
'annual': 982, 'park': 983, 'planned': 984, 'chance': 985, '23': 986,
'businesses': 987, 'satellite': 988, 'investigation': 989, 'issue': 990, 'town':
991, 'storage': 992, 'try': 993, 'fresh': 994, 'order': 995, 'dollars': 996,
'track': 997, 'allow': 998, '500': 999})
```

[46]:
```
text_pipeline = lambda x: [vocab[token] for token in tokenizer(clean(x))]
label_pipeline = lambda x: int(x) - 1
```

1.2 Question 2, 30 points

Next, let's build a simple RNN for classification of the AGNews dataset. Use a one-hot embedding of the vocabulary entries and the basic RNN from Lab 10. Use the lengths tensor (the third element in the batch returned by the dataloaders) to determine which output to apply the loss to.

Place your training code below, and plot the training and test accuracy as a function of epoch. Finally, output a confusion matrix for the test set.

Do not spend a lot of time on the training! A few minutes is enough. The point is to show that the model is learning, not to get the best possible performance.

[140]:
```
# Make one hotter

def label2tensor(l):
    tensor = torch.zeros(len(l), 1, len(labels))
    for index, vector in enumerate(l):
        tensor[index][0][vector] = 1
    return tensor

def list2tensor(l):
    tensor = torch.zeros(len(l), 1, len(vocab.stoi))
```

```
    for index, vector in enumerate(l):
        tensor[index][0][vector] = 1
    return tensor

temp = list2tensor([0,1])
print(temp.shape, temp)

# This has lenght of 3 because we remove 'the' from the string.
temp = list2tensor(text_pipeline("Greating to the world"))
print(temp.shape, temp)

temp = label2tensor([0,1,2,3])
print(temp.shape, temp)
```

```
torch.Size([2, 1, 1000]) tensor([[[1., 0., 0.,  …, 0., 0., 0.]],

        [[0., 1., 0.,  …, 0., 0., 0.]]])
torch.Size([3, 1, 1000]) tensor([[[1., 0., 0.,  …, 0., 0., 0.]],

        [[0., 1., 0.,  …, 0., 0., 0.]],

        [[0., 0., 0.,  …, 0., 0., 0.]]])
torch.Size([4, 1, 4]) tensor([[[1., 0., 0., 0.]],

        [[0., 1., 0., 0.]],

        [[0., 0., 1., 0.]],

        [[0., 0., 0., 1.]]])
```

[199]:
```
# Copy from above
import torch
from torch.utils.data import DataLoader
from torch.nn.utils.rnn import pad_sequence
device = torch.device("cuda:1" if torch.cuda.is_available() else "cpu")

def collate_batch(batch):
    label_list, text_list, length_list = [], [], []
    for (_label, _text) in batch:
        label_list.append(label_pipeline(_label))
        # text -> one hot
        # processed_text = torch.tensor(text_pipeline(_text), dtype=torch.int64)
        processed_text = list2tensor(text_pipeline(_text))
        length_list.append(processed_text.shape[0])
        text_list.append(processed_text)

    # label_list -> one hot
```

```
    # label_list = label2tensor(label_list)
    label_list = torch.tensor(label_list, dtype=torch.int64)
    text_list = pad_sequence(text_list, padding_value=0)

    formatted_textlist = None
    for text in text_list:
        if(formatted_textlist == None):
            formatted_textlist = text
        else:
            formatted_textlist = torch.cat( [formatted_textlist, text], dim = 1␣
 ↪)
    # print(formatted_textlist.shape)

    length_list = torch.tensor(length_list, dtype=torch.int64)
    return label_list.to(device), formatted_textlist.to(device), length_list.
 ↪to(device)

train_iter = AG_NEWS(split='train')
train_dataset = list(train_iter)
test_iter = AG_NEWS(split='test')
test_dataset = list(test_iter)
train_dataloader = DataLoader(train_dataset, batch_size=8, shuffle=True,␣
 ↪collate_fn=collate_batch)
test_dataloader = DataLoader(test_dataset, batch_size=8, shuffle=False,␣
 ↪collate_fn=collate_batch)
```

[198]:
```
# Copy from above
batch = next(enumerate(train_dataloader))
# print(batch.shape)
print(batch)
```

```
(0, (tensor([[[1, 0, 0, 0]],

        [[0, 1, 0, 0]],

        [[1, 0, 0, 0]],

        [[1, 0, 0, 0]],

        [[0, 0, 1, 0]],

        [[0, 0, 1, 0]],

        [[0, 0, 0, 1]],

        [[0, 1, 0, 0]]], device='cuda:1'), tensor([[[0., 0., 0.,  …, 0., 0.,
0.],
```

```
 [0., 0., 0.,  …, 0., 0., 0.],
 [1., 0., 0.,  …, 0., 0., 0.],
 …,
 [0., 0., 0.,  …, 0., 0., 0.],
 [0., 0., 0.,  …, 0., 0., 0.],
 [0., 0., 0.,  …, 0., 0., 0.]],

[[0., 0., 0.,  …, 0., 0., 0.],
 [1., 0., 0.,  …, 0., 0., 0.],
 [0., 0., 0.,  …, 0., 0., 0.],
 …,
 [1., 0., 0.,  …, 0., 0., 0.],
 [0., 0., 0.,  …, 0., 0., 0.],
 [0., 0., 0.,  …, 0., 0., 0.]],

[[0., 0., 0.,  …, 0., 0., 0.],
 [0., 0., 0.,  …, 0., 0., 0.],
 [0., 1., 0.,  …, 0., 0., 0.],
 …,
 [0., 0., 0.,  …, 0., 0., 0.],
 [0., 0., 0.,  …, 0., 0., 0.],
 [0., 0., 0.,  …, 0., 0., 0.]],

…,

[[0., 0., 0.,  …, 0., 0., 0.],
 [1., 0., 0.,  …, 0., 0., 0.],
 [0., 0., 0.,  …, 0., 0., 0.],
 …,
 [0., 0., 0.,  …, 0., 0., 0.],
 [0., 0., 0.,  …, 0., 0., 0.],
 [0., 0., 0.,  …, 0., 0., 0.]],

[[1., 0., 0.,  …, 0., 0., 0.],
 [1., 0., 0.,  …, 0., 0., 0.],
 [0., 0., 1.,  …, 0., 0., 0.],
 …,
 [0., 0., 0.,  …, 0., 0., 0.],
 [0., 0., 0.,  …, 0., 0., 0.],
 [0., 0., 0.,  …, 0., 0., 0.]],

[[0., 0., 0.,  …, 0., 0., 0.],
 [0., 0., 0.,  …, 0., 0., 0.],
 [1., 0., 0.,  …, 0., 0., 0.],
 …,
 [0., 0., 0.,  …, 0., 0., 0.],
 [0., 0., 0.,  …, 0., 0., 0.],
 [0., 0., 0.,  …, 0., 0., 0.]]], device='cuda:1'), tensor([41, 45, 35,
```

```
36, 31, 40, 38, 31], device='cuda:1')))
```

[156]:
```python
# Place code for Question 2 here
### Copy the Batchtify RNN from my report lab10
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.h2o = nn.Linear(hidden_size, output_size)
        # A bit more efficient than normal Softmax
        self.softmax = nn.LogSoftmax(dim=1)

    def forward(self, input, hidden):
        # print(input.shape, hidden.shape)
        combined = torch.cat((input, hidden), 1)
        a = self.i2h(combined)
        hidden = torch.tanh(a)
        o = self.h2o(hidden)
        y_hat = self.softmax(o)
        return y_hat, hidden

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

[217]:
```python
# Use the lengths tensor (the third element in the batch returned by the␣
 ↪dataloaders) to determine which output to apply the loss to.
# Place your training code below, and plot the training and test accuracy as a␣
 ↪function of epoch. Finally, output a confusion matrix for the test set.
# Do not spend a lot of time on the training! A few minutes is enough. The␣
 ↪point is to show that the model is learning, not to get the best possible␣
 ↪performance.

# Training is copy from the lab10
import torch.nn as nn

criterion = nn.NLLLoss()
learning_rate = 0.005
n_hidden = 128
rnn = RNN(len(vocab.stoi), n_hidden, len(labels))
rnn = rnn.to('cuda:1')
epochs = 2
```

```
def train(category_tensor, line_tensor, lenght):
    hidden = rnn.initHidden(line_tensor.shape[1]).to('cuda:1')
    rnn.zero_grad()
    output_all = None
    # print("input shape:",line_tensor.shape)
    for i in range(line_tensor.size()[0]):
        output, hidden = rnn(line_tensor[i], hidden)
        # prit()
        if i == 0:
            output_all = output.unsqueeze(0)
            # print(output_all.shape)
        else:
            output_all = torch.cat( [output_all, output.unsqueeze(0)] , dim=0)
    # print("The output all shape:", output_all.shape)
    # slecting the output
    selected_output = None
    # print(output_all[1])
    for index,pos in enumerate(lenght):
        # print(index,pos)
        if selected_output == None:
            selected_output = output_all[index][pos-1].unsqueeze(0)
        else:
            selected_output = torch.cat([selected_output,␣
 ↪output_all[index][pos-1].unsqueeze(0) ], 0)
    # print(selected_output.shape)
    # loss = criterion(output, category_tensor)
    # print(selected_output.shape, category_tensor.argmax(dim=2).shape)
    loss = criterion(selected_output, category_tensor)
    loss.backward()

    # Add parameters' gradients to their values, multiplied by learning rate
    for p in rnn.parameters():
        p.data.add_(-learning_rate, p.grad.data)

    return selected_output,loss.item()

loss_hist = []
acc_hist = []
for epoch in range(epochs):
    print("="*20, f"{epoch+1}/{epochs}", "="*20)
    batch_loss = 0
    batch_acc = 0
    for iter, batch in enumerate(train_dataloader):
        label, input, lenght = batch
        # print(label)
        # print(input.shape)
        # print("lenght:",lenght)
```

```
        pred, loss = train(label,input,lenght)
        batch_loss += loss

        correct = sum(pred.argmax(dim=1) == label)
        acc = correct/len(pred)
        batch_acc += acc
        if(iter % 1000 == 0):
            print(acc,batch_loss, f"{iter}/{len(train_dataloader)}")

    print("   Batch loss:", batch_loss/len(train_dataloader), batch_acc/
↪len(train_dataloader))
    loss_hist.append(batch_loss/len(train_dataloader))
    acc_hist.append(batch_acc/len(train_dataloader))
    # break
```

```
==================== 1/2 ====================
tensor(0.3750, device='cuda:1') 1.3611924648284912 0/15000
tensor(0.5000, device='cuda:1') 1387.4968800544739 1000/15000
tensor(0.3750, device='cuda:1') 2772.602895259857 2000/15000
tensor(0.5000, device='cuda:1') 4157.1780606508255 3000/15000
tensor(0.5000, device='cuda:1') 5541.100346088409 4000/15000
tensor(0.3750, device='cuda:1') 6923.501227140427 5000/15000
tensor(0.1250, device='cuda:1') 8305.584011435509 6000/15000
tensor(0.2500, device='cuda:1') 9686.704478740692 7000/15000
tensor(0.5000, device='cuda:1') 11067.206221699715 8000/15000
tensor(0.2500, device='cuda:1') 12446.09828054905 9000/15000
tensor(0.3750, device='cuda:1') 13823.868894457817 10000/15000
tensor(0.3750, device='cuda:1') 15202.223905682564 11000/15000
tensor(0.2500, device='cuda:1') 16578.168561577797 12000/15000
tensor(0.2500, device='cuda:1') 17955.58872449398 13000/15000
tensor(0.2500, device='cuda:1') 19332.549511432648 14000/15000
   Batch loss: 1.3803311303695043 tensor(0.2981, device='cuda:1')
==================== 2/2 ====================
tensor(0.2500, device='cuda:1') 1.4146804809570312 0/15000
tensor(0.2500, device='cuda:1') 1373.9571205377579 1000/15000
tensor(0.1250, device='cuda:1') 2745.719015479088 2000/15000
tensor(0.5000, device='cuda:1') 4118.117331624031 3000/15000
tensor(0.3750, device='cuda:1') 5487.407497644424 4000/15000
tensor(0.3750, device='cuda:1') 6855.4224063158035 5000/15000
tensor(0.3750, device='cuda:1') 8222.662576675415 6000/15000
tensor(0.2500, device='cuda:1') 9588.018653273582 7000/15000
tensor(0.3750, device='cuda:1') 10953.163286447525 8000/15000
tensor(0.1250, device='cuda:1') 12314.927069306374 9000/15000
tensor(0.2500, device='cuda:1') 13676.397647738457 10000/15000
tensor(0.2500, device='cuda:1') 15039.464994549751 11000/15000
tensor(0.2500, device='cuda:1') 16399.261925816536 12000/15000
tensor(0.3750, device='cuda:1') 17757.605967640877 13000/15000
```

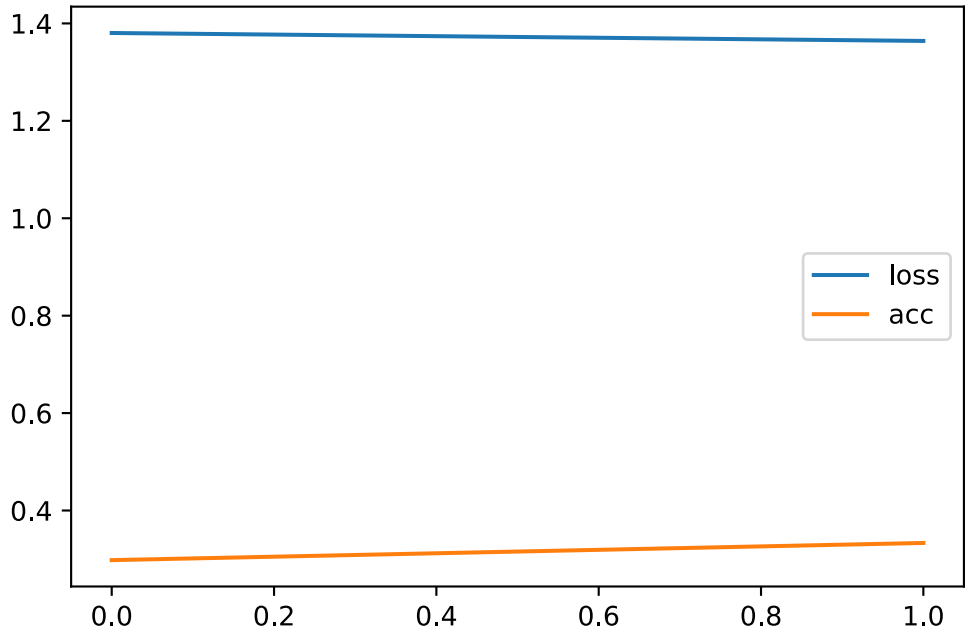
```
tensor(0.6250, device='cuda:1') 19110.65113890171 14000/15000
   Batch loss: 1.3640705216646194 tensor(0.3334, device='cuda:1')
```

[223]:
```python
import matplotlib.pyplot as plt

print_acc = []
for acc in acc_hist:
    print_acc.append(acc.cpu().numpy())

plt.plot(loss_hist,label='loss')
plt.plot(print_acc,label='acc')
plt.legend()
plt.show
```

[223]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



[ ]:
```python
# Confusion Matrix copy from lab

# Keep track of correct guesses in a confusion matrix
confusion = torch.zeros(len(labels), len(vocab.stoi))
n_confusion = 10000

# Just return an output given a line
def evaluate(line_tensor):
    hidden = rnn.initHidden().to('cuda:1')
```

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

    return output

# Go through a bunch of examples and record which are correctly guessed
for i in range(n_confusion):
    category, line, category_tensor, line_tensor = randomTrainingExample()
    output = evaluate(line_tensor.to('cuda:1'))
    guess, guess_i = categoryFromOutput(output)
    category_i = all_categories.index(category)
    confusion[category_i][guess_i] += 1

# Normalize by dividing every row by its sum
for i in range(n_categories):
    confusion[i] = confusion[i] / confusion[i].sum()

# Set up plot
fig = plt.figure()
ax = fig.add_subplot(111)
cax = ax.matshow(confusion.numpy())
fig.colorbar(cax)

# Set up axes
ax.set_xticklabels(['']  + all_categories, rotation=90)
ax.set_yticklabels(['']  + all_categories)

# Force label at every tick
ax.xaxis.set_major_locator(ticker.MultipleLocator(1))
ax.yaxis.set_major_locator(ticker.MultipleLocator(1))

# sphinx_gallery_thumbnail_number = 2
plt.show()
```

1.3 Question 3, 10 points

Next, replace the SRNN from Question 2 with a single-layer LSTM. Give the same output (training and testing accuracy as a function of epoch, as well as confusion matrix for the test set). Comment on the differences you observe between the two models.

[9]:
```
# Place code for Question 3 here
# Copy Naive LSTM from lab 11

import torch
from torch import nn

class NaiveCustomLSTM(nn.Module):
```

```
    def __init__(self, input_sz: int, hidden_sz: int):
        super().__init__()
        self.input_size = input_sz
        self.hidden_size = hidden_sz

        # Parameters for computing i_t
        self.U_i = nn.Parameter(torch.Tensor(input_sz, hidden_sz))
        self.V_i = nn.Parameter(torch.Tensor(hidden_sz, hidden_sz))
        self.b_i = nn.Parameter(torch.Tensor(hidden_sz))

        # Parameters for computing f_t
        self.U_f = nn.Parameter(torch.Tensor(input_sz, hidden_sz))
        self.V_f = nn.Parameter(torch.Tensor(hidden_sz, hidden_sz))
        self.b_f = nn.Parameter(torch.Tensor(hidden_sz))

        # Parameters for computing c_t
        self.U_c = nn.Parameter(torch.Tensor(input_sz, hidden_sz))
        self.V_c = nn.Parameter(torch.Tensor(hidden_sz, hidden_sz))
        self.b_c = nn.Parameter(torch.Tensor(hidden_sz))

        # Parameters for computing o_t
        self.U_o = nn.Parameter(torch.Tensor(input_sz, hidden_sz))
        self.V_o = nn.Parameter(torch.Tensor(hidden_sz, hidden_sz))
        self.b_o = nn.Parameter(torch.Tensor(hidden_sz))

        self.init_weights()

    def init_weights(self):
        stdv = 1.0 / math.sqrt(self.hidden_size)
        for weight in self.parameters():
            weight.data.uniform_(-stdv, stdv)

    def forward(self, x, init_states=None):
        """
        forward: Run input x through the cell. Assumes x.shape is (batch_size,␣
↪sequence_length, input_size)
        """
        bs, seq_sz, _ = x.size()
        hidden_seq = []

        if init_states is None:
            h_t, c_t = (
                torch.zeros(bs, self.hidden_size).to(x.device),
                torch.zeros(bs, self.hidden_size).to(x.device),
            )
        else:
```

```
            h_t, c_t = init_states

        for t in range(seq_sz):
            x_t = x[:, t, :]

            i_t = torch.sigmoid(x_t @ self.U_i + h_t @ self.V_i + self.b_i)
            f_t = torch.sigmoid(x_t @ self.U_f + h_t @ self.V_f + self.b_f)
            g_t = torch.tanh(x_t @ self.U_c + h_t @ self.V_c + self.b_c)
            o_t = torch.sigmoid(x_t @ self.U_o + h_t @ self.V_o + self.b_o)
            c_t = f_t * c_t + i_t * g_t
            h_t = o_t * torch.tanh(c_t)

            hidden_seq.append(h_t.unsqueeze(0))

        # Reshape hidden_seq tensor to (batch size, sequence length,␣
↪hidden_size)
        hidden_seq = torch.cat(hidden_seq, dim=0)
        hidden_seq = hidden_seq.transpose(0, 1).contiguous()

        return hidden_seq, (h_t, c_t)
```

[ ]:
```
# For result when have time
```

1.4 Question 4, 10 points

Explain how you could use the Transformer model to perform the same task you explored in Questions 2 and 3. How would attention be useful for this text classification task? Give a precise and detailed answer. Be sure to discuss what parts of the original Transformer you would use and what you would have to remove.

Write your answer here.

I believed the Transformer could be just use because we would like to feed the entire text sequence into the Transformer.

The output will be right after we put the input (becase we feed the entire sequence) (but we still need to fix the padding 0 issues if we want exactly apple to apple comparision).

Benefit: 1. It is faster due to entire text input (some say paralle) 2. It will have better relation among words because of the attension + the relation from current word to next (which can not happend in simple RNN and LSTM and also better than bi-RNN)

1.5 Question 5, 10 points

In Lab 13, you implemented a DQN model for tic-tac-toe. You method learned to play against a fairly dumb `expert_action` opponent, however. Also, DQN has proven to be less stable than other methods such as Double DQN, also discussed in Lab 13.

Explain below how you would apply double DQN and self-play to improve your tic-tac-toe agent. Provide pseudocode for the algorithm below.

Write your explanation and pseudocode here.

1.6 Question 6, 30 points

Based on your existing DQN implementation, implement the double DQN and self-play training method you just described. After some training (don't spend too much time on training – again, we just want to see that the model can learn), show the result you playing a game against your learned agent.

[10]:
```
# Code for training and playing goes here
```

[ ]:
```

```