In [1]:

```
import numpy as np

from tensorflow.keras.utils import to_categorical
from tensorflow.keras.datasets import imdb
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import SimpleRNN, Dense
from tensorflow.keras.optimizers import Adam

import matplotlib.pyplot as plt
```

In [2]:

```
(x_train, t_train), (x_test, t_test) = imdb.load_data(num_words = 1000)

print("x_train shape : ", x_train.shape)
print("t_train shape : ", t_train.shape)

print("x_test shape : ", x_test.shape)
print("t_test shape : ", t_test.shape)
```

```
x_train shape :  (25000,)
t_train shape :  (25000,)
x_test shape :  (25000,)
t_test shape :  (25000,)
```

In [3]:

```
x_train = x_train[:5000]
t_train = t_train[:5000]

x_test = x_test[:1000]
t_test = t_test[:1000]


print("개수 변경 후 x_train shape : ", x_train.shape)
print("개수 변경 후 t_train shape : ", t_train.shape)

print("개수 변경 후 x_test shape : ", x_test.shape)
print("개수 변경 후 t_test shape : ", t_test.shape)

num_classes=  len(set(t_train))
print("정답 레이블 종류 : {}".format(num_classes))
```

```
개수 변경 후 x_train shape :  (5000,)
개수 변경 후 t_train shape :  (5000,)
개수 변경 후 x_test shape :  (1000,)
개수 변경 후 t_test shape :  (1000,)
정답 레이블 종류 : 2
```

In [4]:

```
print("train용 1번째 리뷰 / 판별 : ", x_train[0], "/", t_train[0])
print(" ")
print("test용 1번째 리뷰 / 판별 : ", x_test[0], "/", t_test[0])
```

```
train용 1번째 리뷰 / 판별 :  [1, 14, 22, 16, 43, 530, 973, 2, 2, 65, 458, 2, 66, 2,
4, 173, 36, 256, 5, 25, 100, 43, 838, 112, 50, 670, 2, 9, 35, 480, 284, 5, 150, 4, 1
72, 112, 167, 2, 336, 385, 39, 4, 172, 2, 2, 17, 546, 38, 13, 447, 4, 192, 50, 16,
6, 147, 2, 19, 14, 22, 4, 2, 2, 469, 4, 22, 71, 87, 12, 16, 43, 530, 38, 76, 15, 13,
2, 4, 22, 17, 515, 17, 12, 16, 626, 18, 2, 5, 62, 386, 12, 8, 316, 8, 106, 5, 4, 2,
2, 16, 480, 66, 2, 33, 4, 130, 12, 16, 38, 619, 5, 25, 124, 51, 36, 135, 48, 25, 2,
33, 6, 22, 12, 215, 28, 77, 52, 5, 14, 407, 16, 82, 2, 8, 4, 107, 117, 2, 15, 256,
4, 2, 7, 2, 5, 723, 36, 71, 43, 530, 476, 26, 400, 317, 46, 7, 4, 2, 2, 13, 104, 88,
4, 381, 15, 297, 98, 32, 2, 56, 26, 141, 6, 194, 2, 18, 4, 226, 22, 21, 134, 476, 2
6, 480, 5, 144, 30, 2, 18, 51, 36, 28, 224, 92, 25, 104, 4, 226, 65, 16, 38, 2, 88,
12, 16, 283, 5, 16, 2, 113, 103, 32, 15, 16, 2, 19, 178, 32] / 1

test용 1번째 리뷰 / 판별 :  [1, 591, 202, 14, 31, 6, 717, 10, 10, 2, 2, 5, 4, 360,
7, 4, 177, 2, 394, 354, 4, 123, 9, 2, 2, 2, 10, 10, 13, 92, 124, 89, 488, 2, 100, 2
8, 2, 14, 31, 23, 27, 2, 29, 220, 468, 8, 124, 14, 286, 170, 8, 157, 46, 5, 27, 239,
16, 179, 2, 38, 32, 25, 2, 451, 202, 14, 6, 717] / 0
```

In [5]:

```
word_to_index = imdb.get_word_index()

print(word_to_index)
```

```
{'fawn': 34701, 'tsukino': 52006, 'nunnery': 52007, 'sonja': 16816, 'vani': 63951,
'woods': 1408, 'spiders': 16115, 'hanging': 2345, 'woody': 2289, 'trawling': 52008,
"hold's": 52009, 'comically': 11307, 'localized': 40830, 'disobeying': 30568, "'roya
le": 52010, "harpo's": 40831, 'canet': 52011, 'aileen': 19313, 'acurately': 52012,
"diplomat's": 52013, 'rickman': 25242, 'arranged': 6746, 'rumbustious': 52014, 'fami
liarness': 52015, "spider'": 52016, 'hahahah': 68804, "wood'": 52017, 'transvestis
m': 40833, "hangin'": 34702, 'bringing': 2338, 'seamier': 40834, 'wooded': 34703, 'b
ravora': 52018, 'grueling': 16817, 'wooden': 1636, 'wednesday': 16818, "'prix": 5201
9, 'altagracia': 34704, 'circuitry': 52020, 'crotch': 11585, 'busybody': 57766, "tar
t'n'tangy": 52021, 'burgade': 14129, 'thrace': 52023, "tom's": 11038, 'snuggles': 52
025, 'francesco': 29114, 'complainers': 52027, 'templarios': 52125, '272': 40835, '2
73': 52028, 'zaniacs': 52130, '275': 34706, 'consenting': 27631, 'snuggled': 40836,
'inanimate': 15492, 'uality': 52030, 'bronte': 11926, 'errors': 4010, 'dialogs': 323
0, "yomada's": 52031, "madman's": 34707, 'dialoge': 30585, 'usenet': 52033, 'videodr
ome': 40837, "kid'": 26338, 'pawed': 52034, "'girlfriend'": 30569, "'pleasure": 5203
5, "'reloaded'": 52036, "kazakos'": 40839, 'rocque': 52037, 'mailings': 52038, 'brai
nwashed': 11927, 'mcanally': 16819, "tom''": 52039, 'kurupt': 25243, 'affiliated': 2
1905, 'babaganoosh': 52040, "noe's": 40840, 'quart': 40841, 'kids': 359, 'upliftin
g': 5034, 'controversy': 7093, 'kida': 21906, 'kidd': 23379, "error'": 52041, 'neuro
```

In [6]:

```
index_to_word = dict([(value, key) for (key, value) in word_to_index.items()])
print(index_to_word)
```

```
{34701: 'fawn', 52006: 'tsukino', 52007: 'nunnery', 16816: 'sonja', 63951: 'vani', 1
408: 'woods', 16115: 'spiders', 2345: 'hanging', 2289: 'woody', 52008: 'trawling', 5
2009: "hold's", 11307: 'comically', 40830: 'localized', 30568: 'disobeying', 52010:
"'royale", 40831: "harpo's", 52011: 'canet', 19313: 'aileen', 52012: 'acurately', 52
013: "diplomat's", 25242: 'rickman', 6746: 'arranged', 52014: 'rumbustious', 52015:
'familiarness', 52016: "spider'", 68804: 'hahahah', 52017: "wood'", 40833: 'transves
tism', 34702: "hangin'", 2338: 'bringing', 40834: 'seamier', 34703: 'wooded', 52018:
'bravora', 16817: 'grueling', 1636: 'wooden', 16818: 'wednesday', 52019: "'prix", 34
704: 'altagracia', 52020: 'circuitry', 11585: 'crotch', 57766: 'busybody', 52021: "t
art'n'tangy", 14129: 'burgade', 52023: 'thrace', 11038: "tom's", 52025: 'snuggles',
29114: 'francesco', 52027: 'complainers', 52125: 'templarios', 40835: '272', 52028:
'273', 52130: 'zaniacs', 34706: '275', 27631: 'consenting', 40836: 'snuggled', 1549
2: 'inanimate', 52030: 'uality', 11926: 'bronte', 4010: 'errors', 3230: 'dialogs', 5
2031: "yomada's", 34707: "madman's", 30585: 'dialoge', 52033: 'usenet', 40837: 'vide
odrome', 26338: "kid'", 52034: 'pawed', 30569: "'girlfriend'", 52035: "'pleasure", 5
2036: "'reloaded'", 40839: "kazakos'", 52037: 'rocque', 52038: 'mailings', 11927: 'b
rainwashed', 16819: 'mcanally', 52039: "tom'", 25243: 'kurupt', 21905: 'affiliate
d', 52040: 'babaganoosh', 40840: "noe's", 40841: 'quart', 359: 'kids', 5034: 'uplift
ing', 7093: 'controversy', 21906: 'kida', 23379: 'kidd', 52041: "error'", 52042: 'ne
```

In [7]:

```
word_to_index.items()
```

Out[7]:

```
dict_items([('fawn', 34701), ('tsukino', 52006), ('nunnery', 52007), ('sonja', 1681
6), ('vani', 63951), ('woods', 1408), ('spiders', 16115), ('hanging', 2345), ('wood
y', 2289), ('trawling', 52008), ("hold's", 52009), ('comically', 11307), ('localize
d', 40830), ('disobeying', 30568), ("'royale", 52010), ("harpo's", 40831), ('canet',
52011), ('aileen', 19313), ('acurately', 52012), ("diplomat's", 52013), ('rickman',
25242), ('arranged', 6746), ('rumbustious', 52014), ('familiarness', 52015), ("spide
r'", 52016), ('hahahah', 68804), ("wood'", 52017), ('transvestism', 40833), ("hangi
n'", 34702), ('bringing', 2338), ('seamier', 40834), ('wooded', 34703), ('bravora',
52018), ('grueling', 16817), ('wooden', 1636), ('wednesday', 16818), ("'prix", 5201
9), ('altagracia', 34704), ('circuitry', 52020), ('crotch', 11585), ('busybody', 577
66), ("tart'n'tangy", 52021), ('burgade', 14129), ('thrace', 52023), ("tom's", 1103
8), ('snuggles', 52025), ('francesco', 29114), ('complainers', 52027), ('templario
s', 52125), ('272', 40835), ('273', 52028), ('zaniacs', 52130), ('275', 34706), ('co
nsenting', 27631), ('snuggled', 40836), ('inanimate', 15492), ('uality', 52030), ('b
ronte', 11926), ('errors', 4010), ('dialogs', 3230), ("yomada's", 52031), ("madma
n's", 34707), ('dialoge', 30585), ('usenet', 52033), ('videodrome', 40837), ("kid'",
26338), ('pawed', 52034), ("'girlfriend'", 30569), ("'pleasure", 52035), ("'reloade
```

In [8]:

```
print("빈도수 최상위 (1등) 단어 : {}".format(index_to_word[1]))
print("빈도수 상위 3938등 단어 : {}".format(index_to_word[3938]))
print("빈도수 최하위 (꼴등", str(len(word_to_index)), ")) 단어 : {}".format(index_to_word[len(word_t
```

```
빈도수 최상위 (1등) 단어 : the
빈도수 상위 3938등 단어 : suited
빈도수 최하위 (꼴등 88584 )) 단어 : 'l'
```

In [9]:

```
print(" ".join([index_to_word[index] for index in x_train[0]]))
```

```
the as you with out themselves powerful and and their becomes and had and of lot fro
m anyone to have after out atmosphere never more room and it so heart shows to years
of every never going and help moments or of every and and movie except her was sever
al of enough more with is now and film as you of and and unfortunately of you than h
im that with out themselves her get for was and of you movie sometimes movie that wi
th scary but and to story wonderful that in seeing in character to of and and with h
eart had and they of here that with her serious to have does when from why what have
and they is you that isn't one will very to as itself with other and in of seen over
and for anyone of and br and to whether from than out themselves history he name hal
f some br of and and was two most of mean for 1 any an and she he should is thought
and but of script you not while history he heart to real at and but when from one bi
t then have two of script their with her and most that with wasn't to with and actin
g watch an for with and film want an
```

In [10]:

```
index_to_word = {}

for key, value in word_to_index.items():
    index_to_word[value + 3] = key

print("빈도수 최상위 (1등) 단어 : {}".format(index_to_word[4]))
print("빈도수 상위 3938등 단어 : {}".format(index_to_word[3941]))
print("빈도수 최하위 (꼴등", str(len(word_to_index)), ")) 단어 : {}".format(index_to_word[len(word_t
```

```
빈도수 최상위 (1등) 단어 : the
빈도수 상위 3938등 단어 : suited
빈도수 최하위 (꼴등 88584 )) 단어 : 'l'
```

In [11]:

```
index_to_word
```

Out[11]:

```
{34704: 'fawn',
 52009: 'tsukino',
 52010: 'nunnery',
 16819: 'sonja',
 63954: 'vani',
 1411: 'woods',
 16118: 'spiders',
 2348: 'hanging',
 2292: 'woody',
 52011: 'trawling',
 52012: "hold's",
 11310: 'comically',
 40833: 'localized',
 30571: 'disobeying',
 52013: "'royale",
 40834: "harpo's",
 52014: 'canet',
 19316: 'aileen'.
```

In [12]:

```
for index, token in enumerate(("?", "?", "?", "?")):
    print("index, token : ", index, token)
    index_to_word[index] = token
```

```
index, token :  0 ?
index, token :  1 ?
index, token :  2 ?
index, token :  3 ?
```

In [13]:

```
for i in range(0, 4):
    print(str(i) + "번째 index_to_word : {}".format(index_to_word[i]))
```

```
0번째 index_to_word : ?
1번째 index_to_word : ?
2번째 index_to_word : ?
3번째 index_to_word : ?
```

In [14]:

```
print(" ".join([index_to_word[index] for index in x_train[0]]))
```

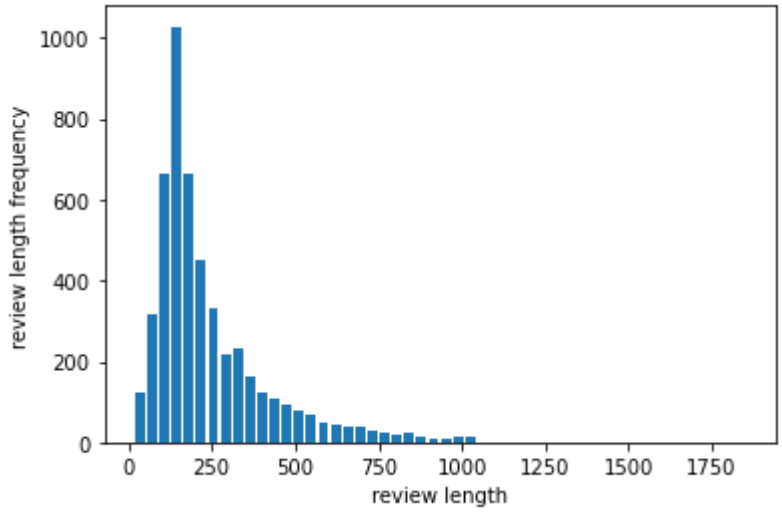
```
? this film was just brilliant casting ? ? story direction ? really ? the part they
played and you could just imagine being there robert ? is an amazing actor and now t
he same being director ? father came from the same ? ? as myself so i loved the fact
there was a real ? with this film the ? ? throughout the film were great it was just
brilliant so much that i ? the film as soon as it was released for ? and would recom
mend it to everyone to watch and the ? ? was amazing really ? at the end it was so s
ad and you know what they say if you ? at a film it must have been good and this def
initely was also ? to the two little ? that played the ? of ? and paul they were jus
t brilliant children are often left out of the ? ? i think because the stars that pl
ay them all ? up are such a big ? for the whole film but these children are amazing
and should be ? for what they have done don't you think the whole story was so ? bec
ause it was true and was ? life after all that was ? with us all
```

In [15]:

```
review_length = np.array([len(x) for x in x_train])
print("train용 리뷰 길이 : ", review_length)
print("리뷰의 최대 길이 : {}".format(np.max(review_length)))
print("리뷰의 최소 길이 : {}".format(np.min(review_length)))
print("리뷰의 평균 길이 : {}".format(np.mean(review_length)))

plt.hist(review_length, bins = 50, rwidth=0.8)
plt.xlabel("review length")
plt.ylabel("review length frequency")
plt.show()
```

```
train용 리뷰 길이 :  [218 189 141 ... 852 130 171]
리뷰의 최대 길이 : 1851
리뷰의 최소 길이 : 16
리뷰의 평균 길이 : 243.8212
```



In [16]:

```
print("x_train shape : ", x_train.shape)
print("x_test shape : ", x_test.shape)

train_seq = pad_sequences(x_train, maxlen = 100)
test_seq = pad_sequences(x_test, maxlen = 100)

print("train_seq shape : ", train_seq.shape)
print("test_seq shape : ", test_seq.shape)
```

```
x_train shape :  (5000,)
x_test shape :  (1000,)
train_seq shape :  (5000, 100)
test_seq shape :  (1000, 100)
```

In [17]:

```
x_train
```

Out[17]:

```
array([list([1, 14, 22, 16, 43, 530, 973, 2, 2, 65, 458, 2, 66, 2, 4, 173, 36, 256,
5, 25, 100, 43, 838, 112, 50, 670, 2, 9, 35, 480, 284, 5, 150, 4, 172, 112, 167, 2,
336, 385, 39, 4, 172, 2, 2, 17, 546, 38, 13, 447, 4, 192, 50, 16, 6, 147, 2, 19, 14,
22, 4, 2, 2, 469, 4, 22, 71, 87, 12, 16, 43, 530, 38, 76, 15, 13, 2, 4, 22, 17, 515,
17, 12, 16, 626, 18, 2, 5, 62, 386, 12, 8, 316, 8, 106, 5, 4, 2, 2, 16, 480, 66, 2,
33, 4, 130, 12, 16, 38, 619, 5, 25, 124, 51, 36, 135, 48, 25, 2, 33, 6, 22, 12, 215,
28, 77, 52, 5, 14, 407, 16, 82, 2, 8, 4, 107, 117, 2, 15, 256, 4, 2, 7, 2, 5, 723, 3
6, 71, 43, 530, 476, 26, 400, 317, 46, 7, 4, 2, 2, 13, 104, 88, 4, 381, 15, 297, 98,
32, 2, 56, 26, 141, 6, 194, 2, 18, 4, 226, 22, 21, 134, 476, 26, 480, 5, 144, 30, 2,
18, 51, 36, 28, 224, 92, 25, 104, 4, 226, 65, 16, 38, 2, 88, 12, 16, 283, 5, 16, 2,
113, 103, 32, 15, 16, 2, 19, 178, 32]),
       list([1, 194, 2, 194, 2, 78, 228, 5, 6, 2, 2, 2, 134, 26, 4, 715, 8, 118, 2,
14, 394, 20, 13, 119, 954, 189, 102, 5, 207, 110, 2, 21, 14, 69, 188, 8, 30, 23, 7,
4, 249, 126, 93, 4, 114, 9, 2, 2, 5, 647, 4, 116, 9, 35, 2, 4, 229, 9, 340, 2, 4, 11
8, 9, 4, 130, 2, 19, 4, 2, 5, 89, 29, 952, 46, 37, 4, 455, 9, 45, 43, 38, 2, 2, 398,
4, 2, 26, 2, 5, 163, 11, 2, 2, 4, 2, 9, 194, 775, 7, 2, 2, 349, 2, 148, 605, 2, 2, 1
5, 123, 125, 68, 2, 2, 15, 349, 165, 2, 98, 5, 4, 228, 9, 43, 2, 2, 15, 299, 120, 5,
120, 174, 11, 220, 175, 136, 50, 9, 2, 228, 2, 5, 2, 656, 245, 2, 5, 4, 2, 131, 152,
491, 18, 2, 32, 2, 2, 14, 9, 6, 371, 78, 22, 625, 64, 2, 9, 8, 168, 145, 23, 4, 2, 1
5, 16, 4, 2, 5, 28, 6, 52, 154, 462, 33, 89, 78, 285, 16, 145, 95]),
       list([1, 14, 47, 8, 30, 31, 7, 4, 249, 108, 7, 4, 2, 54, 61, 369, 13, 71, 14
9, 14, 22, 112, 4, 2, 311, 12, 16, 2, 33, 75, 43, 2, 296, 4, 86, 320, 35, 534, 19, 2
63, 2, 2, 4, 2, 33, 89, 78, 12, 66, 16, 4, 360, 7, 4, 58, 316, 334, 11, 4, 2, 43, 64
5, 662, 8, 257, 85, 2, 42, 2, 2, 83, 68, 2, 15, 36, 165, 2, 278, 36, 69, 2, 780, 8,
106, 14, 2, 2, 18, 6, 22, 12, 215, 28, 610, 40, 6, 87, 326, 23, 2, 21, 23, 22, 12, 2
72, 40, 57, 31, 11, 4, 22, 47, 6, 2, 51, 9, 170, 23, 595, 116, 595, 2, 13, 191, 79,
638, 89, 2, 14, 9, 8, 106, 607, 624, 35, 534, 6, 227, 7, 129, 113]),
       list([1, 14, 16, 691, 35, 589, 8, 81, 4, 172, 155, 19, 2, 15, 16, 224, 19, 2,
2, 11, 2, 2, 2, 18, 148, 7, 25, 37, 2, 12, 5, 902, 2, 2, 2, 368, 425, 2, 2, 2, 4, 96
6, 283, 651, 7, 4, 2, 299, 534, 2, 11, 479, 16, 6, 318, 39, 6, 171, 153, 145, 121, 2,
2, 2, 2, 367, 4, 96, 537, 71,
2, 19, 156, 2, 4, 204, 177, 5, 2, 12, 16, 55, 73, 224, 12, 16, 163, 73, 177, 5, 385,
638, 17, 6, 2, 2, 7, 4, 123, 10, 10, 994, 8, 4, 2, 4, 2, 7, 2, 5, 2, 9, 11, 6, 729,
405, 4, 796, 979, 7, 4, 479, 7, 4, 123, 4, 796, 2, 4, 831, 235, 26, 32, 4, 172, 803,
2, 9, 4, 2, 2, 15, 2, 562, 10, 10, 11, 994, 2, 2, 5, 2, 2, 199, 2, 2, 8, 6, 519, 12
3, 8, 63, 36, 71, 24, 981, 8, 30, 2, 103, 112, 2, 8, 789, 2, 5, 2, 2, 4, 2, 7, 4, 2,
7, 4, 123, 4, 2, 2, 2, 5, 2, 2, 8, 2, 103, 12, 533, 971, 98, 143, 2, 15, 2, 98, 8, 1
04, 44, 4, 479, 7, 4, 123, 14, 853, 832, 8, 4, 2, 7, 37, 2, 4, 2, 5, 138, 10, 10, 2,
8, 381, 14, 2, 165, 260, 6, 114, 9, 4, 2, 2, 7, 14, 123, 2, 2, 994, 2, 4, 529, 8, 79
4, 6, 329, 221, 769, 4, 2, 7, 4, 2, 305, 7, 2, 32, 94, 692, 23, 51, 4, 311, 62, 91,
30, 928, 14, 4, 479, 7, 4, 123, 12, 9, 4, 2, 2, 15, 157, 4, 118, 4, 973, 17, 11, 2,
9, 321, 718, 2, 2, 2, 5, 2, 2, 2, 2, 2, 4, 235, 7, 4, 156, 209, 267, 99, 76, 40, 98,
2, 2, 2, 11, 4, 984, 8, 27, 875, 21, 50, 600, 7, 4, 522, 2, 2, 64, 27, 2, 85, 2,
2, 26, 346, 5, 2, 19, 600, 2, 46, 17, 262, 52, 42, 78, 111, 7, 4, 537, 28, 77, 579,
159, 21, 36, 669, 297, 46, 2, 19, 64, 4, 2, 2, 2, 160, 389, 227, 39, 4, 2, 2, 16, 4,
361, 7, 2, 929, 7, 2, 2, 2, 2, 18, 4, 173, 7, 2, 4, 64, 273, 121, 4, 2, 2, 9, 54, 36,
2, 540, 93, 56, 114, 757, 8, 2, 4, 293, 65, 14, 2, 4, 2, 7, 4, 2, 2, 10, 10, 4, 293,
114, 48, 15, 9, 51, 75, 215, 683, 12, 9, 7, 265, 2, 14, 9, 24, 66, 6, 2, 11, 5, 7, 4
0, 21, 45, 43, 24, 2, 126, 73, 198, 8, 2, 66, 4, 2, 365, 354, 5, 52, 487, 70, 97, 5
6, 18, 6, 710, 114, 262, 11, 134, 2, 7, 183, 21, 75, 66, 79, 2, 133, 4, 354, 34, 2,
5, 2, 306, 643, 2, 60, 18, 98, 4, 807, 99, 2, 398, 18, 12, 8, 235, 2, 174, 13, 104,
4, 2, 8, 2, 2, 2, 70, 30, 110, 11, 15, 4, 807, 9, 669, 43, 50, 8, 270, 58, 6, 2, 11,
2, 198, 32, 12, 2, 8, 30, 11, 994, 12, 497, 8, 81, 2, 2, 5, 472, 400, 996, 2, 5, 2,
81, 73, 261, 13, 2, 19, 409, 15, 2, 69, 24, 2, 73, 5, 15, 2, 69, 5, 803, 2, 656, 8,
168, 15, 52, 21, 713, 348, 195, 8, 81, 174, 13, 104, 38, 32, 62, 28, 77, 128, 2, 34,
6, 53, 2, 2, 74, 4, 31, 2, 133, 10, 10, 160, 2, 213, 44, 994, 14, 318, 9, 44, 4, 2,
248, 201, 5, 94, 479, 246, 32, 4, 2, 617, 26, 39, 4, 2, 20, 60, 4, 2, 929, 9, 2, 20,
929, 48, 25, 124, 335, 2, 479, 95, 25, 124, 15, 4, 20, 16, 2, 2, 8, 30, 93, 86, 64,
8, 30, 2, 11, 2, 7, 4, 248, 123, 54, 2, 887, 8, 2, 58, 702, 38, 54, 2, 5, 2, 71, 2,
18, 4, 217, 12, 16, 18, 4, 20, 24, 4, 248, 123, 138, 994, 64, 2, 20, 929, 9, 2, 12,
```

In [18]:
```
print("x_train의 1번째 요소 type : ", type(x_train[0]))
print("train_seq의 1번째 요소 type : ", type(train_seq[0]))
```

```
x_train의 1번째 요소 type :  <class 'list'>
train_seq의 1번째 요소 type :  <class 'numpy.ndarray'>
```

In [19]:
```
train_seq
```

Out[19]:
```
array([[   2,   33,    6,   19,  178,   32],
       [ 163,   11,    2,   16,  145,   95],
       [   2,    4,    2,    7,  129,  113],
       ...,
       [  11,    4,  984,    8,   27,  875,   30,   21,   50],
       [   2,  115,  188,    2,    2,    2],
       [   2,   56,  345,    4,    2,    2]])
```

In [20]:
```
print(" ".join([index_to_word[index] for index in train_seq[0]]))
```

```
? at a film it must have been good and this definitely was also ? to the two little
? that played the ? of ? and paul they were just brilliant children are often left o
ut of the ? ? i think because the stars that play them all ? up are such a big ? for
the whole film but these children are amazing and should be ? for what they have don
e don't you think the whole story was so ? because it was true and was ? life after
all that was ? with us all
```

In [21]:
```
print(" ".join([index_to_word[index] for index in x_train[0]]))
```

```
? this film was just brilliant casting ? ? story direction ? really ? the part they played and you could just imagine being there robert ? is an amazing actor and now the same being director ? father came from the same ? ? as myself so i loved the fact there was a real ? with this film the ? ? throughout the film were great it was just brilliant so much that i ? the film as soon as it was released for ? and would recommend it to everyone to watch and the ? ? was amazing really ? at the end it was so sad and you know what they say if you ? at a film it must have been good and this definitely was also ? to the two little ? that played the ? of ? and paul they were just brilliant children are often left out of the ? ? i think because the stars that play them all ? up are such a big ? for the whole film but these children are amazing and should be ? for what they have done don ' t you think the whole story was so ? because it was true and was ? life after all that was ? with us all
```

In [22]:
```
train_seq = to_categorical(train_seq)
test_seq = to_categorical(test_seq)
print("One-hot Vector 적용 후 train_seq.shape : ", train_seq.shape)
print("One-hot Vector 적용 후 test_seq.shape : ", test_seq.shape)
```

```
One-hot Vector 적용 후 train_seq.shape :  (5000, 100, 1000)
One-hot Vector 적용 후 test_seq.shape :  (1000, 100, 1000)
```

In [23]:

```
cell_size = 128
timesteps = 100
feature = 1000

model = Sequential(name="IMDB_RNN")

model.add(SimpleRNN(cell_size, kernel_initializer="he_uniform",
                    input_shape = (timesteps, feature), activation = "relu",
                     return_sequences=True))

model.add(SimpleRNN(cell_size, kernel_initializer="he_uniform", activation = "relu"))

model.add(Dense(1, activation = "sigmoid"))

model.compile(loss="binary_crossentropy",
              optimizer=Adam(learning_rate=0.001, clipnorm=1.), metrics=["accuracy"])

model.summary()
```

```
Model: "IMDB_RNN"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 simple_rnn (SimpleRNN)      (None, 100, 128)          144512

 simple_rnn_1 (SimpleRNN)    (None, 128)               32896

 dense (Dense)               (None, 1)                 129

=================================================================
Total params: 177,537
Trainable params: 177,537
Non-trainable params: 0
_________________________________________________________________
```

In [24]:

```
cell_size = 128
timesteps = 100
feature = 1000

model = Sequential(name="Weight_test")

model.add(SimpleRNN(cell_size, kernel_initializer="he_uniform",
                    input_shape = (timesteps, feature),
                    activation = "relu", return_sequences=True))

print("모델의 가중치 (He Uniform 초기화 적용) : ", model.get_weights(), sep="₩n")
```

```
모델의 가중치 (He Uniform 초기화 적용) :
[array([[ 0.02528623, -0.00571372, -0.02847837, ..., -0.07433107,
        -0.04312996, -0.02664093],
       [-0.01441333,  0.02215694, -0.0539744 , ...,  0.03751545,
         0.05281314,  0.05248276],
       [-0.07483885,  0.04505942, -0.01550309, ...,  0.04304167,
         0.05684812, -0.01361191],
       ...,
       [ 0.06563636,  0.06770753, -0.04944444, ..., -0.00722434,
         0.0146812 , -0.03801624],
       [ 0.05810916,  0.04593533, -0.04771511, ...,  0.02450895,
         0.05802829,  0.0574582 ],
       [ 0.03365015, -0.03395734,  0.06340573, ..., -0.07446622,
        -0.01032228,  0.01575745]], dtype=float32), array([[ 0.02931273,  0.0432764
9, -0.03951617, ...,  0.02125496,
        -0.02520056, -0.02845988],
       [-0.04338985, -0.0005995 , -0.08761381, ..., -0.03104129,
        -0.09091025, -0.04551224],
       [ 0.03496369,  0.02908603, -0.0505414 , ...,  0.04909174,
         0.04677046, -0.17311096],
       ...,
       [-0.08172233,  0.05814296, -0.09281648, ...,  0.01193016,
         0.10840854, -0.04910937],
       [ 0.0537726 ,  0.03227289, -0.08151715, ...,  0.01133603,
        -0.16292465, -0.13646778],
       [ 0.03962686,  0.02357973, -0.01959816, ..., -0.04077387,
        -0.05138836,  0.05192256]], dtype=float32), array([0., 0., 0., 0., 0., 0.,
0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0.], dtype=float32)]
```

In [25]:

```
cell_size = 128
timesteps = 100
feature = 1000

model = Sequential(name="IMDB_RNN")

model.add(SimpleRNN(cell_size, kernel_initializer="he_uniform",
                    input_shape = (timesteps, feature), activation = "relu",
                     return_sequences=True))

model.add(SimpleRNN(cell_size, kernel_initializer="he_uniform", activation = "relu"))

model.add(Dense(1, activation = "sigmoid"))

model.compile(loss="binary_crossentropy",
              optimizer=Adam(learning_rate=0.001, clipnorm=1.), metrics=["accuracy"])

model.summary()
```

```
Model: "IMDB_RNN"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 simple_rnn_3 (SimpleRNN)    (None, 100, 128)          144512

 simple_rnn_4 (SimpleRNN)    (None, 128)               32896

 dense_1 (Dense)             (None, 1)                 129

=================================================================
Total params: 177,537
Trainable params: 177,537
Non-trainable params: 0
_________________________________________________________________
```

In [26]:

```
model.fit(train_seq, t_train, epochs=20, batch_size = 64)
```

```
Epoch 1/20
79/79 [==============================] - 22s 221ms/step - loss: 0.6850 - accuracy:
0.5404
Epoch 2/20
79/79 [==============================] - 16s 204ms/step - loss: 0.5843 - accuracy:
0.7074
Epoch 3/20
79/79 [==============================] - 16s 202ms/step - loss: 0.5460 - accuracy:
0.7806
Epoch 4/20
79/79 [==============================] - 16s 200ms/step - loss: 0.6525 - accuracy:
0.7624
Epoch 5/20
79/79 [==============================] - 17s 220ms/step - loss: 1.1107 - accuracy:
0.8010
Epoch 6/20
79/79 [==============================] - 16s 207ms/step - loss: 0.5616 - accuracy:
0.8010
Epoch 7/20
79/79 [==============================] - 17s 212ms/step - loss: 0.8111 - accuracy:
0.8144
Epoch 8/20
79/79 [==============================] - 20s 250ms/step - loss: 5.3154 - accuracy:
0.8238
Epoch 9/20
79/79 [==============================] - 22s 273ms/step - loss: 1.1088 - accuracy:
0.8696
Epoch 10/20
79/79 [==============================] - 20s 246ms/step - loss: 1.0905 - accuracy:
0.8968
Epoch 11/20
79/79 [==============================] - 24s 301ms/step - loss: 0.1971 - accuracy:
0.9394
Epoch 12/20
79/79 [==============================] - 25s 318ms/step - loss: 0.1638 - accuracy:
0.9574
Epoch 13/20
79/79 [==============================] - 20s 250ms/step - loss: 0.0920 - accuracy:
0.9728
Epoch 14/20
79/79 [==============================] - 19s 233ms/step - loss: 0.0719 - accuracy:
0.9784
Epoch 15/20
79/79 [==============================] - 18s 232ms/step - loss: 0.1614 - accuracy:
0.9768
Epoch 16/20
79/79 [==============================] - 20s 258ms/step - loss: 0.1721 - accuracy:
0.9870
Epoch 17/20
79/79 [==============================] - 21s 259ms/step - loss: 0.0374 - accuracy:
0.9906
Epoch 18/20
79/79 [==============================] - 19s 237ms/step - loss: 0.6445 - accuracy:
0.9812
Epoch 19/20
79/79 [==============================] - 18s 224ms/step - loss: 0.0214 - accuracy:
0.9946
Epoch 20/20
79/79 [==============================] - 19s 239ms/step - loss: 0.0369 - accuracy:
0.9888
```

Out[26]:

```
<keras.callbacks.History at 0x26bd0c53ee0>
```

In [27]:

```
timesteps = 100
feature = 1000

for i in range(5):
    test_review = " ".join([index_to_word[index] for index in x_test[i]])
    print("==========================", str(i + 1), "번째 리뷰==========================")
    print(test_review)
    print(" ")

    score = float(model.predict(test_seq[i].reshape(1, timesteps, feature)))

    if (score > 0.5):
        print(str(i + 1), "번째 리뷰는 {:.2f}% 확률로 긍정 리뷰입니다.".format(score * 100))
    else :
        print(str(i + 1), "번째 리뷰는 {:.2f}% 확률로 긍정 리뷰입니다.".format((1 - score) * 100))


    print(" ")
```

========== 1 번째 리뷰==========
? please give this one a miss br br ? ? and the rest of the cast ? terrible performa
nces the show is ? ? ? br br i don't know how michael ? could have ? this one on his
? he almost seemed to know this wasn't going to work out and his performance was qui
te ? so all you ? fans give this a miss

1/1 [==============================] - 1s 647ms/step
1 번째 리뷰는 99.69% 확률로 긍정 리뷰입니다.

========== 2 번째 리뷰==========
? this film ? a lot of ? because it ? on ? and character development the plot is ver
y simple and many of the scenes take place on the same set in ? ? the ? ? character
? but the film ? to a ? ? br br the characters create an atmosphere ? with sexual ?
and ? ? it's very interesting that robert ? directed this ? the style and ? of his o
ther films still the ? ? ? style is ? here and there i think what really makes this
film work is the brilliant performance by ? ? it's definitely one of her ? character
s but she plays it so perfectly and ? that it's scary michael ? does a good job as t
he ? young man ? ? ? michael ? has a small part the ? ? set ? the ? of the story ver
y well in short this movie is a powerful ? of ? sexual ? and ? be ? ? up the atmosph
ere and pay attention to the ? written script br br i ? robert ? this is one of his
many films that ? with ? ? subject matter this film is ? but it's ? and it's sure to
? a strong emotional ? from the viewer if you want to see an ? film some might even
say ? this is worth the time br br unfortunately it's very difficult to find in vide
o ? you may have to buy it off the ?

1/1 [==============================] - 0s 62ms/step
2 번째 리뷰는 99.77% 확률로 긍정 리뷰입니다.

========== 3 번째 리뷰==========
? many animation ? ? ? ? the great ? ? of one special ? of the art ? animation which
he ? almost single ? and as it happened almost ? as a young man ? was more intereste
d in ? than the cinema but his ? attempt to film two ? ? fighting ? to an ? ? in fil
m making when he ? he could ? ? by ? ? ? and ? them one ? at a time this ? ? to the
production of ? ? classic short the ? ? which he made in ? in ? at a time when ? pic
ture animation of all ? was in its ? br br the political ? of the ? ? ? ? to move to
? where one of his first ? ? was a dark political ? ? known as ? or the ? who wanted
a king a ? of black comedy can be found in almost all of films but here it is very d
ark indeed ? more at ? ? who can ? the ? ? than children who would most ? find the ?
? i'm middle ? and found it pretty ? myself and indeed ? of the film ? for english ?
viewers of the ? were given title ? ? with ? and ? in order to help ? the ? ? of the
? br br our tale is set in a ? the ? ? where the ? are ? with their ? and have calle
d a special ? to see what they can do to ? ? they ? to ? ? for a king the ? are ? ?
in this opening sequence it couldn't have been easy to make so many ? ? look ? ? whi
le ? for his part is ? as a ? white ? guy in the ? who looks like ? rather be taking
a ? when ? ? them a ? like god who ? them the ? ? that this is no ? and ? a differen
t king ? ? ? them a ? br br ? with this ? looking new king who ? above them the ? ?
him with a ? of ? ? ? the ? ? forward to hand him the ? to the ? as ? ? ? the ? to ?
horror the ? ? ? the ? and then goes on a ? ? ? ? at ? a title ? ? ? ? of the ? ? th
roughout the ? when the now ? ? once more ? ? for help he ? his ? and ? their ? with
? ? the ? of our story ? by a ? ? just before he is ? is let well enough alone br br
? the time period when this ? little film was made and ? the fact that it was made b
y a ? ? at the ? of that ? ? ? war it would be easy to see this as a ? about those e
vents ? may or may not have had ? ? in mind when he made ? but whatever ? his ? of m
aterial the film ? as a ? tale of ? ? ? could be the ? ? ? ? or ? in the ? or any co
untry of any era that ? its ? down and is ? by ? it's a ? film even a ? one in its ?
way but its message is no joke

1/1 [==============================] - 0s 53ms/step
3 번째 리뷰는 93.55% 확률로 긍정 리뷰입니다.

========== 4 번째 리뷰==========

```
? i ? love this type of movie however this time i found myself ? to ? the screen sin
ce i can't do that i will just ? about it this was absolutely ? the things that happ
en with the dead kids are very cool but the ? people are ? ? i am a ? man pretty big
and i can ? myself well however i would not do half the stuff the little girl does i
n this movie also the mother in this movie is ? with her children to the point of ?
i wish i wasn't so ? about her and her ? because i would have otherwise enjoyed the
flick what a number she was take my ? and fast forward through everything you see he
r do until the end also is anyone else getting ? of watching movies that are filmed
so dark ? one can hardly see what is being filmed as an audience we are ? involved w
ith the ? on the screen so then why the hell can't we have night ?

1/1 [==============================] - 0s 52ms/step
4 번째 리뷰는 99.86% 확률로 긍정 리뷰입니다.

============================ 5 번째 리뷰============================
? like some other people ? i'm a die hard ? fan and i loved this game br br this gam
e starts ? boring but ? me it's worth it as soon as you start your ? the ? are fun a
nd ? they will ? you ? your mind turns to ? i'm not ? this game is also ? and is ? d
one br br to keep this ? free i have to keep my ? ? about ? but please try this game
? be worth it br br story 9 9 action 10 1 it's that good ? 10 attention ? 10 average
10

1/1 [==============================] - 0s 48ms/step
5 번째 리뷰는 94.38% 확률로 긍정 리뷰입니다.
```

In [28]:

```
loss, accuracy = model.evaluate(test_seq, t_test, verbose=1)

print("test loss : ", round(loss, 6))
print("test accuracy : ", round(accuracy * 100, 3), "%")
```

```
32/32 [==============================] - 3s 61ms/step - loss: 1.2912 - accuracy: 0.6
600
test loss :  1.291165
test accuracy :  66.0 %
```