In [ ]:
```
import os
import re
import nltk
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from nltk.tokenize import word_tokenize
from nltk import pos_tag
from nltk.corpus import stopwords, wordnet
from nltk.stem import WordNetLemmatizer
from sklearn.datasets import fetch_20newsgroups
from tensorflow.keras import layers
from tensorflow.keras.models import Sequential
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
import os
import nltk
import subprocess

# List of resources to download
resources = ["punkt", "averaged_perceptron_tagger", "wordnet", "stopwords", "omw

# Download and unzip resources if necessary
for resource in resources:
    try:
        nltk.data.find(f'{resource}.zip')
    except:
        nltk.download(resource, download_dir='/kaggle/working/')
        command = f"unzip /kaggle/working/corpora/{resource}.zip -d /kaggle/work
        subprocess.run(command.split())
        nltk.data.path.append('/kaggle/working/')

# Now you can import the NLTK resources as usual
from nltk.tokenize import word_tokenize
from nltk import pos_tag
from nltk.corpus import wordnet, stopwords
from nltk.stem import WordNetLemmatizer
```

```
[nltk_data] Downloading package punkt to /kaggle/working/...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /kaggle/working/...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
[nltk_data] Downloading package wordnet to /kaggle/working/...
[nltk_data]   Package wordnet is already up-to-date!
Archive:  /kaggle/working/corpora/wordnet.zip
[nltk_data] Downloading package stopwords to /kaggle/working/...
[nltk_data]   Package stopwords is already up-to-date!
Archive:  /kaggle/working/corpora/stopwords.zip
[nltk_data] Downloading package omw-1.4 to /kaggle/working/...
[nltk_data]   Package omw-1.4 is already up-to-date!
Archive:  /kaggle/working/corpora/omw-1.4.zip
```

```
unzip:  cannot find or open /kaggle/working/corpora/punkt.zip, /kaggle/working/co
rpora/punkt.zip.zip or /kaggle/working/corpora/punkt.zip.ZIP.
unzip:  cannot find or open /kaggle/working/corpora/averaged_perceptron_tagger.zi
p, /kaggle/working/corpora/averaged_perceptron_tagger.zip.zip or /kaggle/working/
corpora/averaged_perceptron_tagger.zip.ZIP.
replace /kaggle/working/corpora/wordnet/lexnames? [y]es, [n]o, [A]ll, [N]one, [r]
ename:  NULL
(EOF or read error, treating as "[N]one" ...)
replace /kaggle/working/corpora/stopwords/dutch? [y]es, [n]o, [A]ll, [N]one, [r]e
name:  NULL
(EOF or read error, treating as "[N]one" ...)
replace /kaggle/working/corpora/omw-1.4/fin/LICENSE? [y]es, [n]o, [A]ll, [N]one,
[r]ename:  NULL
(EOF or read error, treating as "[N]one" ...)
```

In [ ]:
```
newsgroup_train = fetch_20newsgroups(subset='train', shuffle=True)
newsgroup_test = fetch_20newsgroups(subset='test', shuffle=True)
print(newsgroup_train.target_names)
```

```
['alt.atheism', 'comp.graphics', 'comp.os.ms-windows.misc', 'comp.sys.ibm.pc.hard
ware', 'comp.sys.mac.hardware', 'comp.windows.x', 'misc.forsale', 'rec.autos', 'r
ec.motorcycles', 'rec.sport.baseball', 'rec.sport.hockey', 'sci.crypt', 'sci.elec
tronics', 'sci.med', 'sci.space', 'soc.religion.christian', 'talk.politics.guns',
'talk.politics.mideast', 'talk.politics.misc', 'talk.religion.misc']
```

In [ ]:
```
df_train = pd.DataFrame({'article': newsgroup_train.data, 'label': newsgroup_tra
df_train.head()
```

Out[ ]:

	article	label
0	From: lerxst@wam.umd.edu (where's my thing)\nS...	7
1	From: guykuo@carson.u.washington.edu (Guy Kuo)...	4
2	From: twillis@ec.ecn.purdue.edu (Thomas E Will...	4
3	From: jgreen@amber (Joe Green)\nSubject: Re: W...	1
4	From: jcm@head-cfa.harvard.edu (Jonathan McDow...	14

In [ ]:
```
df_test = pd.DataFrame({'article': newsgroup_test.data, 'label': newsgroup_test.
df_test.head()
```

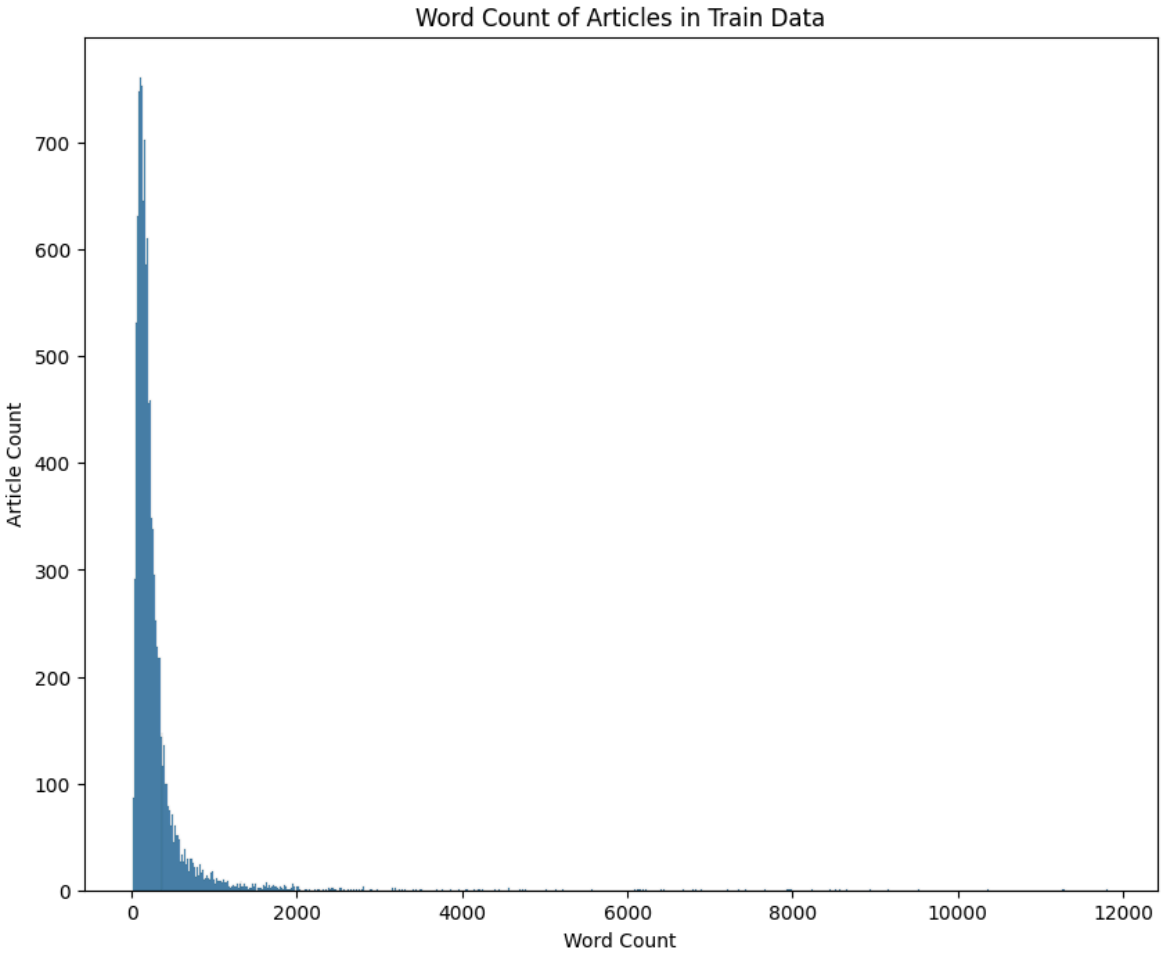
Out[ ]:

	article	label
0	From: v064mb9k@ubvmsd.cc.buffalo.edu (NEIL B. ...	7
1	From: Rick Miller <rick@ee.uwm.edu>\nSubject: ...	5
2	From: mathew <mathew@mantis.co.uk>\nSubject: R...	0
3	From: bakken@cs.arizona.edu (Dave Bakken)\nSub...	17
4	From: livesey@solntze.wpd.sgi.com (Jon Livesey...	19

In [ ]:
```
df_train['word_count'] = df_train['article'].apply(lambda x: len(str(x).split()))
plt.figure(figsize=(10,8))
sns.histplot(data=df_train, x='word_count')
plt.title('Word Count of Articles in Train Data')
plt.xlabel('Word Count')
```

```
plt.ylabel('Article Count')
plt.show()
```

```
/opt/conda/lib/python3.10/site-packages/seaborn/_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version. Conve
rt inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

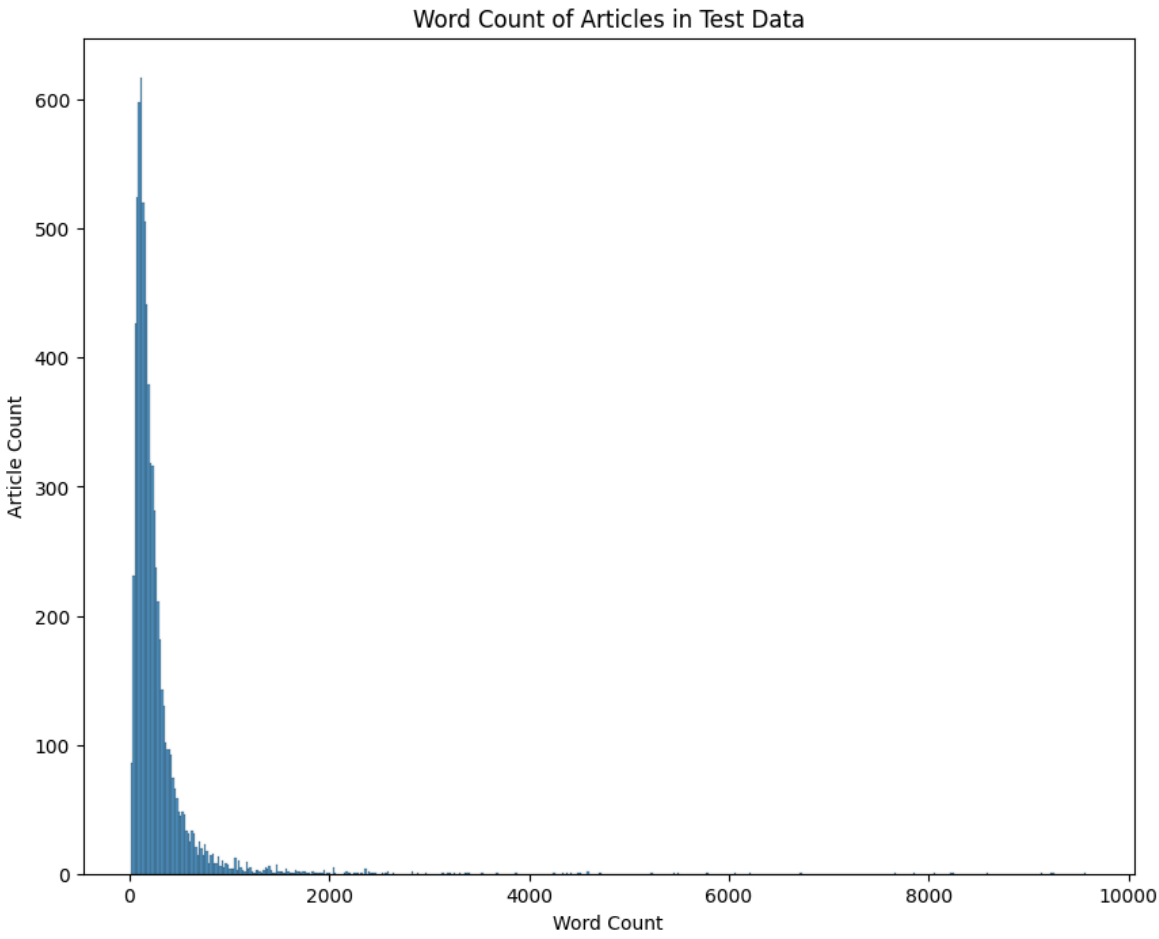


In [ ]:
```
train_articles = (sum(df_train['word_count'] < 1000)/df_train.shape[0])*100
print('Percentage of Training Articles having less than 1000 Words:{:.2f}%'.form
```

```
Percentage of Training Articles having less than 1000 Words:96.80%
```

In [ ]:
```
df_test['word_count'] = df_test['article'].apply(lambda x: len(str(x).split()))
plt.figure(figsize=(10,8))
sns.histplot(data=df_test, x='word_count')
plt.title('Word Count of Articles in Test Data')
plt.xlabel('Word Count')
plt.ylabel('Article Count')
plt.show()
```

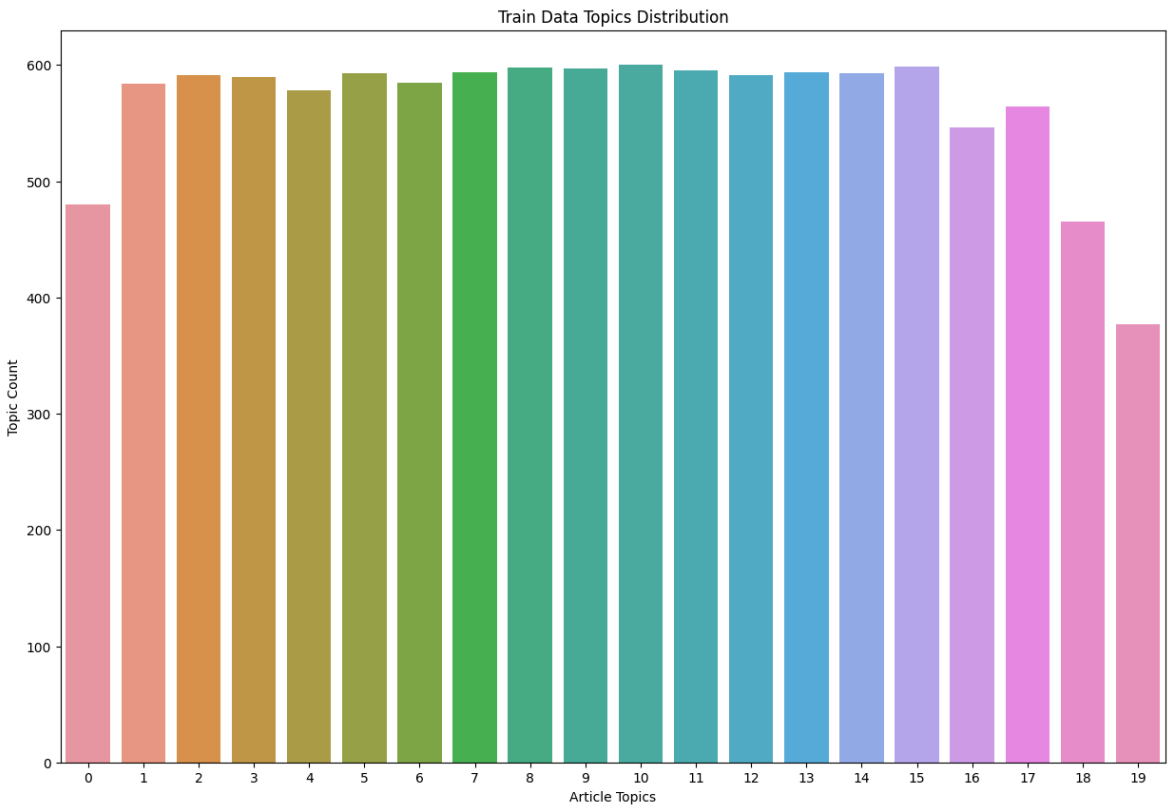
```
/opt/conda/lib/python3.10/site-packages/seaborn/_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version. Conve
rt inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



In [ ]:
```
test_articles = (sum(df_test['word_count'] < 1000)/df_test.shape[0])*100
print('Percentage of Test Articles having less than 1000 Words:{:.2f}%'.format(t
```

```
Percentage of Test Articles having less than 1000 Words:97.09%
```

In [ ]:
```
plt.figure(figsize=(15,10))
sns.countplot(data=df_train, x='label')
plt.title('Train Data Topics Distribution')
plt.xlabel('Article Topics')
plt.ylabel('Topic Count')
plt.show()
```



In [ ]:
```
def get_wordnet_pos (tag):
    if tag.startswith('J'):
        return wordnet.ADJ
    elif tag.startswith('V'):
        return wordnet.VERB
    elif tag.startswith('N'):
        return wordnet.NOUN
    elif tag.startswith('R'):
        return wordnet.ADV
    else:
        return wordnet.NOUN
def lemmatize (word_list):
    wl = WordNetLemmatizer()
    word_pos_tags = pos_tag(word_list)
    lemmatized_list = []
    for tag in word_pos_tags:
        lemmatize_word = wl.lemmatize(tag[0],get_wordnet_pos(tag[1]))
        lemmatized_list.append(lemmatize_word)
    return " ".join(lemmatized_list)
def clean_text (text):
    # Remove Pre and Post Spaces
    text = str(text).strip()

    # Lower case the entire text
    text = str(text).lower()

    # Substitute New Line Characters with spaces
    text = re.sub(r"\n", r" ", text)

    # Tokenize the sentence
    word_tokens = word_tokenize(text)

    # Remove the punctuation and  special characters from each individual word
    cleaned_text = []
    for word in word_tokens:
```

```
        cleaned_text.append("".join([char for char in word if char.isalnum()]))

    # Specify the stop words list
    stop_words = stopwords.words('english')

    # Remove the stopwords and words containing less then 2 characters
    text_tokens = [word for word in cleaned_text if (len(word) > 2) and (word no

    #Lemmatize each word in the word list
    text = lemmatize (text_tokens)

    return text
```

In [ ]:
```
df_train['article'][0]
```

Out[ ]:
```
"From: lerxst@wam.umd.edu (where's my thing)\nSubject: WHAT car is this!?\nNntp
-Posting-Host: rac3.wam.umd.edu\nOrganization: University of Maryland, College
Park\nLines: 15\n\n I was wondering if anyone out there could enlighten me on t
his car I saw\nthe other day. It was a 2-door sports car, looked to be from the
late 60s/\nearly 70s. It was called a Bricklin. The doors were really small. In
addition,\nthe front bumper was separate from the rest of the body. This is \na
ll I know. If anyone can tellme a model name, engine specs, years\nof productio
n, where this car is made, history, or whatever info you\nhave on this funky lo
oking car, please e-mail.\n\nThanks,\n- IL\n   ---- brought to you by your neig
hborhood Lerxst ----\n\n\n\n\n"
```

In [ ]:
```
clean_text (df_train['article'][0])
```

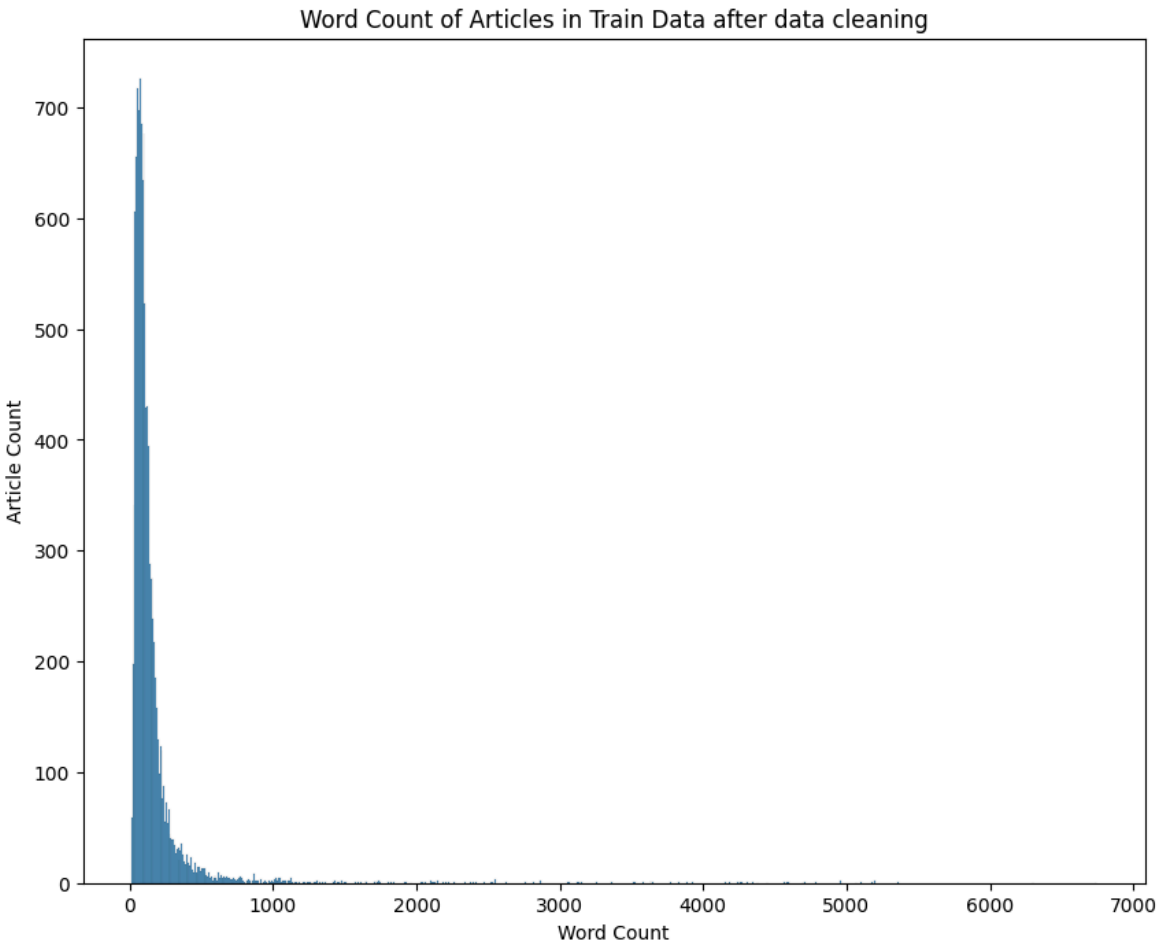
Out[ ]:
```
'lerxst wamumdedu thing subject car nntppostinghost rac3wamumdedu organization
university maryland college park line wonder anyone could enlighten car saw day
2door sport car look late 60 early 70 call bricklin door really small addition
front bumper separate rest body know anyone tellme model name engine spec year
production car make history whatever info funky look car please email thanks br
ing neighborhood lerxst'
```

In [ ]:
```
df_train['article'] = df_train['article'].apply(lambda x: clean_text(x))
```

In [ ]:
```
df_test['article'] = df_test['article'].apply(lambda x: clean_text(x))
```

In [ ]:
```
df_train['word_count'] = df_train['article'].apply(lambda x: len(str(x).split()))
plt.figure(figsize=(10,8))
sns.histplot(data=df_train, x='word_count')
plt.title('Word Count of Articles in Train Data after data cleaning')
plt.xlabel('Word Count')
plt.ylabel('Article Count')
plt.show()
```

```
/opt/conda/lib/python3.10/site-packages/seaborn/_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version. Conve
rt inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

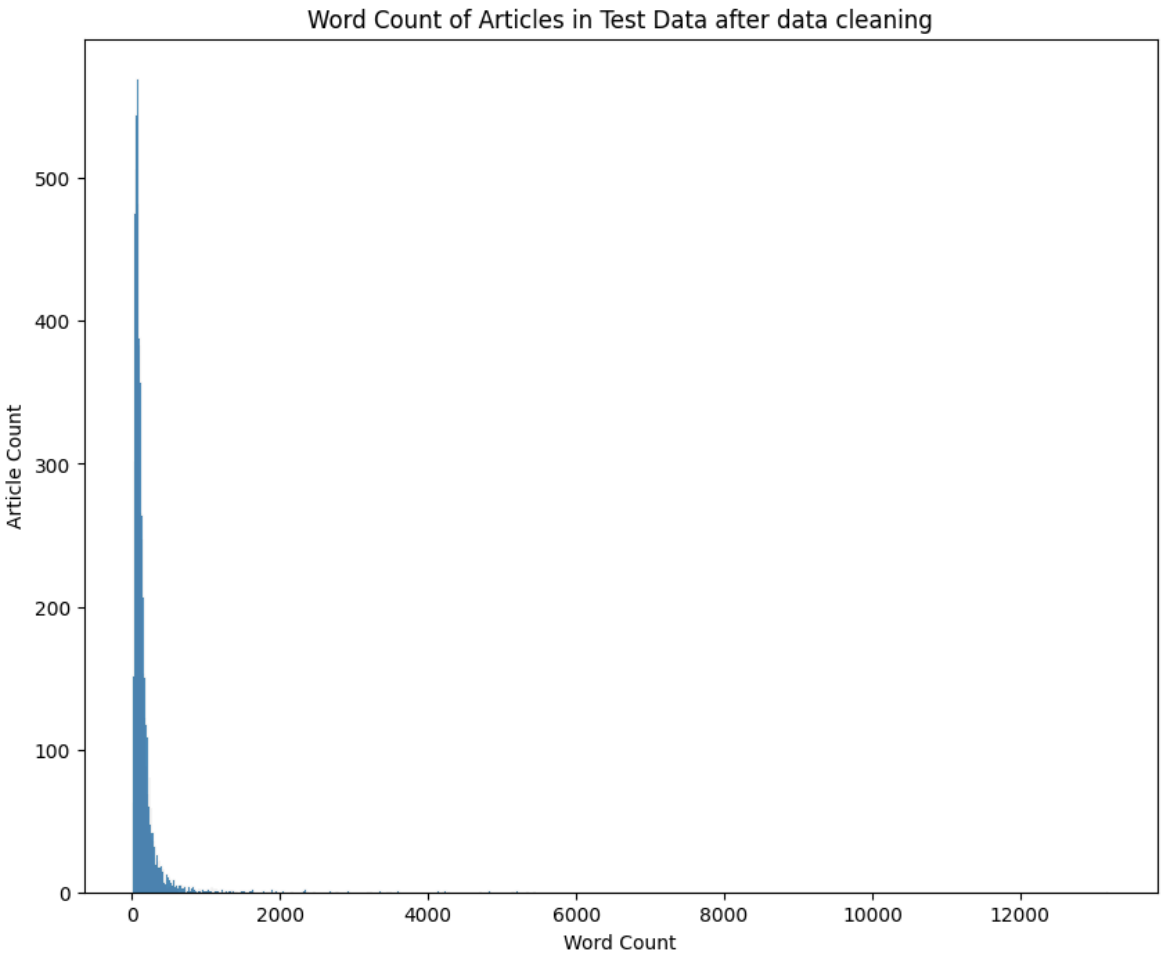


In [ ]:
```
train_articles = (sum(df_train['word_count'] < 300)/df_train.shape[0])*100
print('Percentage of Training Articles having less than 300 Words:{:.2f}%'.forma
```

```
Percentage of Training Articles having less than 300 Words:92.05%
```

In [ ]:
```
df_test['word_count'] = df_test['article'].apply(lambda x: len(str(x).split()))
plt.figure(figsize=(10,8))
sns.histplot(data=df_test, x='word_count')
plt.title('Word Count of Articles in Test Data after data cleaning')
plt.xlabel('Word Count')
plt.ylabel('Article Count')
plt.show()
```

```
/opt/conda/lib/python3.10/site-packages/seaborn/_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version. Conve
rt inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



In [ ]:
```
test_articles = (sum(df_test['word_count'] < 300)/df_test.shape[0])*100
print('Percentage of Test Articles having less than 300 Words:{:.2f}%'.format(te
```

```
Percentage of Test Articles having less than 300 Words:92.37%
```

In [ ]:
```
X_train = df_train['article']
y_train = df_train['label']
X_test = df_test['article']
y_test = df_test['label']
print("X_train:", X_train.shape)
print("X_test:", X_test.shape)
print("y_train:", y_train.shape)
print("y_test:", y_test.shape)
```

```
X_train: (11314,)
X_test: (7532,)
y_train: (11314,)
y_test: (7532,)
```

In [ ]:
```
tokenizer = Tokenizer(num_words=100000)
tokenizer.fit_on_texts(X_train)
tokenizer.index_word
```

Out[ ]:
```
{1: 'line',
 2: 'subject',
 3: 'organization',
 4: 'would',
 5: 'one',
 6: 'write',
 7: 'use',
 8: 'get',
 9: 'say',
 10: 'article',
 11: 'know',
 12: 'people',
 13: 'like',
 14: 'make',
 15: 'think',
 16: 'university',
 17: 'time',
 18: 'nntppostinghost',
 19: 'max',
 20: 'well',
 21: 'good',
 22: 'also',
 23: 'see',
 24: 'new',
 25: 'work',
 26: 'system',
 27: 'could',
 28: 'take',
 29: 'year',
 30: 'want',
 31: 'go',
 32: 'right',
 33: 'need',
 34: 'come',
 35: 'even',
 36: 'thing',
 37: 'problem',
 38: 'way',
 39: 'may',
 40: 'look',
 41: 'give',
 42: 'god',
 43: 'file',
 44: 'find',
 45: 'many',
 46: 'state',
 47: 'first',
 48: 'two',
 49: 'much',
 50: 'question',
 51: 'distribution',
 52: 'try',
 53: 'call',
 54: 'point',
 55: 'program',
 56: 'run',
 57: 'world',
 58: 'anyone',
 59: 'post',
 60: 'drive',
```

```
61: 'believe',
62: 'tell',
63: 'mean',
64: 'seem',
65: 'number',
66: 'computer',
67: 'help',
68: 'please',
69: 'something',
70: 'window',
71: 'really',
72: 'include',
73: 'read',
74: 'back',
75: 'since',
76: 'day',
77: 'case',
78: 'email',
79: 'still',
80: 'information',
81: 'game',
82: 'key',
83: 'law',
84: 'government',
85: 'part',
86: 'start',
87: 'last',
88: 'must',
89: 'group',
90: 'thanks',
91: 'usa',
92: 'never',
93: 'let',
94: 'ask',
95: 'might',
96: 'replyto',
97: 'car',
98: 'support',
99: 'another',
100: 'sure',
101: 'without',
102: 'follow',
103: 'space',
104: 'version',
105: 'set',
106: 'name',
107: 'david',
108: 'etc',
109: 'keep',
110: 'long',
111: 'power',
112: 'put',
113: 'fact',
114: 'data',
115: 'science',
116: 'someone',
117: 'great',
118: 'available',
119: 'do',
120: 'reason',
```

```
121: 'list',
122: 'card',
123: 'send',
124: 'team',
125: 'lot',
126: 'show',
127: 'change',
128: 'high',
129: 'christian',
130: 'gun',
131: 'little',
132: 'john',
133: 'chip',
134: 'bad',
135: 'place',
136: 'however',
137: 'play',
138: 'software',
139: 'opinion',
140: 'anything',
141: 'around',
142: 'every',
143: 'probably',
144: 'course',
145: 'leave',
146: 'best',
147: 'true',
148: 'word',
149: 'consider',
150: 'book',
151: 'happen',
152: 'end',
153: 'life',
154: 'old',
155: 'public',
156: 'technology',
157: 'least',
158: 'second',
159: 'different',
160: 'kill',
161: 'talk',
162: 'bit',
163: 'claim',
164: 'live',
165: 'enough',
166: 'order',
167: 'note',
168: 'center',
169: 'research',
170: 'provide',
171: 'image',
172: 'base',
173: 'writes',
174: 'buy',
175: 'jesus',
176: 'control',
177: '1993',
178: 'idea',
179: 'message',
180: 'hard',
```

```
181: 'source',
182: 'service',
183: 'issue',
184: 'far',
185: 'armenian',
186: 'possible',
187: 'actually',
188: 'example',
189: 'either',
190: 'though',
191: 'big',
192: 'inc',
193: 'real',
194: 'answer',
195: 'cause',
196: 'person',
197: 'b8f',
198: 'child',
199: 'rather',
200: 'nothing',
201: 'mail',
202: 'next',
203: 'mark',
204: 'driver',
205: 'internet',
206: 'else',
207: 'machine',
208: 'american',
209: 'wrong',
210: 'standard',
211: 'free',
212: 'access',
213: 'man',
214: 'address',
215: 'exist',
216: 'phone',
217: 'large',
218: 'build',
219: 'a86',
220: 'allow',
221: 'yes',
222: 'human',
223: 'disk',
224: 'maybe',
225: 'win',
226: 'bill',
227: 'national',
228: 'player',
229: 'code',
230: 'able',
231: 'user',
232: 'others',
233: 'always',
234: 'hand',
235: 'turn',
236: 'report',
237: 'hear',
238: 'price',
239: 'info',
240: 'type',
```

```
241: 'keywords',
242: 'require',
243: 'kind',
244: 'several',
245: 'today',
246: 'general',
247: 'israel',
248: 'small',
249: 'home',
250: 'area',
251: 'yet',
252: 'sound',
253: 'less',
254: 'view',
255: 'quite',
256: 'ever',
257: 'sale',
258: '145',
259: 'three',
260: 'pay',
261: 'result',
262: 'cost',
263: 'sell',
264: 'become',
265: 'away',
266: 'open',
267: 'application',
268: 'week',
269: 'test',
270: 'remember',
271: 'speed',
272: 'check',
273: 'move',
274: 'news',
275: 'company',
276: 'create',
277: 'study',
278: 'color',
279: 'president',
280: 'hold',
281: 'country',
282: 'whether',
283: 'current',
284: 'steve',
285: 'mac',
286: 'side',
287: 'feel',
288: 'design',
289: 'encryption',
290: 'agree',
291: 'already',
292: 'money',
293: 'michael',
294: 'war',
295: 'understand',
296: 'department',
297: 'evidence',
298: 'netcomcom',
299: 'value',
300: 'force',
```

```
301: 'display',
302: 'institute',
303: 'rule',
304: 'argument',
305: 'graphic',
306: 'assume',
307: 'matter',
308: 'lead',
309: 'love',
310: 'stop',
311: 'box',
312: 'offer',
313: 'local',
314: 'ago',
315: 'jew',
316: 'apr',
317: 'low',
318: 'mention',
319: 'city',
320: 'bible',
321: 'server',
322: 'add',
323: 'perhaps',
324: 'copy',
325: 'memory',
326: 'experience',
327: 'house',
328: 'robert',
329: 'woman',
330: 'clipper',
331: 'act',
332: 'fax',
333: 'hope',
334: 'package',
335: 'guy',
336: 'difference',
337: 'care',
338: 'mind',
339: 'whole',
340: 'close',
341: 'pretty',
342: 'lose',
343: 'april',
344: 'stuff',
345: 'interest',
346: 'mike',
347: 'return',
348: 'attack',
349: 'paul',
350: 'begin',
351: 'network',
352: 'job',
353: 'communication',
354: 'die',
355: 'expect',
356: 'member',
357: 'jim',
358: 'church',
359: 'deal',
360: 'carry',
```

```
361: 'israeli',
362: 'turkish',
363: 'contact',
364: 'interested',
365: 'device',
366: 'religion',
367: 'appear',
368: 'head',
369: 'sun',
370: 'death',
371: 'bike',
372: 'save',
373: 'canada',
374: 'model',
375: 'everything',
376: 'product',
377: 'important',
378: 'month',
379: 'comment',
380: 'accept',
381: 'school',
382: 'fire',
383: 'everyone',
384: 'error',
385: 'fast',
386: 'hit',
387: 'rate',
388: 'level',
389: 'original',
390: 'light',
391: 'easy',
392: 'action',
393: 'truth',
394: 'guess',
395: 'often',
396: 'white',
397: 'almost',
398: 'monitor',
399: 'sort',
400: 'effect',
401: 'scsi',
402: 'articleid',
403: 'advance',
404: 'reference',
405: 'form',
406: 'simply',
407: '1d9',
408: 'friend',
409: 'format',
410: 'weapon',
411: 'speak',
412: 'full',
413: 'video',
414: 'body',
415: 'board',
416: 'engineering',
417: 'dept',
418: 'statement',
419: 'wonder',
420: 'bring',
```

```
421: 'cover',
422: 'season',
423: 'arm',
424: 'position',
425: 'size',
426: 'instead',
427: 'although',
428: 'certainly',
429: 'history',
430: 'division',
431: 'california',
432: 'plan',
433: 'anybody',
434: 'regard',
435: 'couple',
436: 'single',
437: 'ground',
438: 'anyway',
439: 'xnewsreader',
440: 'discussion',
441: 'college',
442: 'summary',
443: 'men',
444: 'later',
445: 'hell',
446: 'output',
447: 'suggest',
448: 'mode',
449: 'correct',
450: 'receive',
451: 'press',
452: 'event',
453: 'ftp',
454: 'explain',
455: 'sense',
456: 'project',
457: 'crime',
458: 'unless',
459: 'security',
460: 'black',
461: 'present',
462: 'drug',
463: 'break',
464: 'top',
465: 'appreciate',
466: 'function',
467: 'hockey',
468: '100',
469: 'process',
470: 'situation',
471: 'entry',
472: 'clinton',
473: 'release',
474: 'major',
475: 'similar',
476: 'reply',
477: 'site',
478: 'certain',
479: 'faith',
480: 'apple',
```

```
481: 'continue',
482: 'san',
483: 'unix',
484: 'earth',
485: 'net',
486: 'individual',
487: 'term',
488: 'purpose',
489: 'face',
490: 'clear',
491: 'period',
492: 'within',
493: 'request',
494: 'quote',
495: 'likely',
496: 'private',
497: 'road',
498: 'late',
499: 'police',
500: 'policy',
501: 'goal',
502: 'suppose',
503: 'figure',
504: 'jewish',
505: 'record',
506: 'learn',
507: 'office',
508: 'stand',
509: 'nice',
510: 'land',
511: 'date',
512: 'decide',
513: 'christ',
514: 'simple',
515: 'via',
516: 'faq',
517: 'usually',
518: 'screen',
519: 'hardware',
520: 'atheist',
521: 'protect',
522: 'strong',
523: 'exactly',
524: 'saw',
525: 'except',
526: 'involve',
527: 'young',
528: 'especially',
529: 'windows',
530: 'dave',
531: 'early',
532: 'heard',
533: 'response',
534: 'fan',
535: 'mine',
536: 'washington',
537: 'section',
538: 'sorry',
539: 'keith',
540: 'nasa',
```

```
541: 'york',
542: 'wait',
543: 'text',
544: 'detail',
545: 'tax',
546: 'per',
547: 'gmt',
548: 'society',
549: 'widget',
550: 'million',
551: 'pick',
552: 'short',
553: 'health',
554: 'corporation',
555: 'watch',
556: 'tin',
557: 'bank',
558: 'fine',
559: 'dod',
560: 'common',
561: 'pittsburgh',
562: 'limit',
563: 'page',
564: 'western',
565: 'business',
566: 'league',
567: 'thus',
568: 'night',
569: 'dead',
570: 'cut',
571: 'launch',
572: 'condition',
573: 'attempt',
574: 'radio',
575: 'story',
576: 'food',
577: 'increase',
578: 'particular',
579: 'bob',
580: 'brian',
581: 'manager',
582: 'cheap',
583: 'apply',
584: 'rest',
585: 'produce',
586: 'port',
587: 'among',
588: 'bus',
589: 'option',
590: 'ibm',
591: 'pass',
592: 'belief',
593: 'air',
594: 'political',
595: 'score',
596: 'james',
597: 'concern',
598: 'contain',
599: 'water',
600: 'red',
```

```
601: 'mouse',
602: 'express',
603: 'handle',
604: 'fail',
605: 'command',
606: 'court',
607: 'define',
608: 'therefore',
609: 'chance',
610: 'moral',
611: 'method',
612: 'third',
613: 'tape',
614: 'accord',
615: 'future',
616: 'field',
617: 'whatever',
618: 'draw',
619: 'compare',
620: 'switch',
621: 'past',
622: 'military',
623: 'controller',
624: 'toronto',
625: 'smith',
626: 'paper',
627: 'unit',
628: 'due',
629: 'authority',
630: 'wire',
631: 'theory',
632: 'texas',
633: 'author',
634: 'king',
635: 'anonymous',
636: 'develop',
637: 'miss',
638: 'front',
639: 'personal',
640: 'shot',
641: 'directory',
642: 'total',
643: 'engine',
644: 'tool',
645: 'object',
646: 'solution',
647: 'andrew',
648: 'four',
649: 'criminal',
650: 'library',
651: 'peter',
652: 'final',
653: 'frank',
654: 'sometimes',
655: 'special',
656: 'flame',
657: 'upon',
658: 'family',
659: 'medium',
660: 'specific',
```

```
661: 'murder',
662: 'voice',
663: 'ram',
664: 'bear',
665: 'federal',
666: 'tom',
667: 'recently',
668: 'chicago',
669: 'fall',
670: 'algorithm',
671: 'sign',
672: 'agency',
673: 'worth',
674: 'series',
675: 'describe',
676: 'trade',
677: 'resource',
678: 'soon',
679: 'baseball',
680: 'behind',
681: 'greek',
682: 'near',
683: 'secret',
684: 'judge',
685: 'richard',
686: 'letter',
687: 'class',
688: 'along',
689: 'together',
690: 'choose',
691: 'international',
692: 'motif',
693: 'plus',
694: 'complete',
695: 'wish',
696: 'scott',
697: 'muslim',
698: 'interface',
699: 'font',
700: 'party',
701: 'technical',
702: 'religious',
703: 'feature',
704: 'official',
705: 'share',
706: 'station',
707: 'citizen',
708: 'lie',
709: 'amount',
710: 'peace',
711: 'previous',
712: 'firearm',
713: 'account',
714: 'delete',
715: '1992',
716: 'doubt',
717: 'meet',
718: 'prove',
719: 'father',
720: 'legal',
```

```
721: 'administration',
722: 'russian',
723: 'picture',
724: 'market',
725: 'approach',
726: 'various',
727: 'laboratory',
728: 'arab',
729: 'privacy',
730: 'necessary',
731: 'compute',
732: 'knowledge',
733: 'block',
734: 'occur',
735: 'development',
736: 'manual',
737: 'minute',
738: 'disclaimer',
739: 'medical',
740: 'currently',
741: 'choice',
742: 'nhl',
743: 'performance',
744: 'average',
745: 'slow',
746: 'sin',
747: 'printer',
748: 'notice',
749: 'thought',
750: 'fix',
751: 'age',
752: 'chris',
753: 'cable',
754: 'avoid',
755: 'otherwise',
756: 'population',
757: 'north',
758: 'thank',
759: 'insurance',
760: 'forget',
761: 'supply',
762: 'quality',
763: 'defense',
764: 'replace',
765: 'burn',
766: 'title',
767: 'remove',
768: 'thomas',
769: 'germany',
770: 'none',
771: 'spend',
772: 'outside',
773: 'univ',
774: 'operation',
775: 'hour',
776: 'owner',
777: 'effort',
778: 'clearly',
779: 'ide',
780: 'fight',
```

```
781: 'fit',
782: 'charge',
783: 'son',
784: 'community',
785: 'doctor',
786: 'freedom',
787: 'christianity',
788: 'shall',
789: 'remain',
790: 'eric',
791: 'united',
792: 'language',
793: 'input',
794: 'objective',
795: 'stay',
796: 'serial',
797: 'modem',
798: 'purchase',
799: 'sit',
800: 'pat',
801: 'vote',
802: 'document',
803: 'activity',
804: 'online',
805: 'serious',
806: 'fbi',
807: 'realize',
808: 'load',
809: 'america',
810: 'publish',
811: 'print',
812: 'search',
813: 'practice',
814: 'prevent',
815: 'basic',
816: 'main',
817: 'convert',
818: 'newsgroup',
819: 'digital',
820: 'refer',
821: 'eye',
822: 'george',
823: 'morality',
824: 'willing',
825: 'commercial',
826: 'keyboard',
827: 'gas',
828: 'count',
829: 'street',
830: 'gary',
831: 'kid',
832: 'completely',
833: 'armenia',
834: 'blue',
835: 'gordon',
836: 'student',
837: 'drop',
838: 'jon',
839: 'inside',
840: 'ship',
```

```
841: 'turkey',
842: 'boston',
843: 'half',
844: 'safety',
845: 'depend',
846: 'satellite',
847: 'orbit',
848: 'serve',
849: 'grant',
850: 'nature',
851: 'decision',
852: 'lack',
853: 'existence',
854: 'respond',
855: 'material',
856: 'suggestion',
857: 'normal',
858: 'tim',
859: 'determine',
860: 'secure',
861: 'mass',
862: 'south',
863: 'dan',
864: 'argue',
865: 'disease',
866: 'reach',
867: 'beat',
868: 'stephanopoulos',
869: 'corp',
870: 'lab',
871: 'scientific',
872: 'transfer',
873: 'mile',
874: 'trust',
875: 'thousand',
876: 'range',
877: 'connect',
878: 'fund',
879: 'indeed',
880: 'congress',
881: 'finally',
882: 'obtain',
883: 'adam',
884: 'archive',
885: 'dealer',
886: 'uunet',
887: 'room',
888: 'lord',
889: 'useful',
890: 'throw',
891: 'star',
892: 'mission',
893: 'turk',
894: 'easily',
895: 'matthew',
896: 'door',
897: 'inreplyto',
898: 'msg',
899: 'definition',
900: 'reasonable',
```

```
901: 'west',
902: 'rid',
903: 'generally',
904: 'advice',
905: 'happy',
906: 'obviously',
907: 'moon',
908: 'intend',
909: 'raise',
910: 'internal',
911: 'usenet',
912: 'amendment',
913: 'directly',
914: 'ten',
915: 'nation',
916: 'discuss',
917: 'difficult',
918: 'education',
919: 'stupid',
920: 'wing',
921: '550',
922: 'addition',
923: 'necessarily',
924: 'illinois',
925: 'respect',
926: 'conference',
927: 'doug',
928: 'magazine',
929: 'reserve',
930: 'character',
931: 'shoot',
932: 'unfortunately',
933: 'direct',
934: 'giz',
935: 'instal',
936: 'vehicle',
937: 'license',
938: 'los',
939: 'blood',
940: 'enforcement',
941: 'imagine',
942: 'basis',
943: 'henry',
944: 'floppy',
945: 'store',
946: 'joe',
947: 'trouble',
948: 'obvious',
949: 'entire',
950: 'playoff',
951: 'somebody',
952: 'reduce',
953: 'signal',
954: 'roger',
955: 'measure',
956: 'oil',
957: 'conclusion',
958: 'east',
959: 'circuit',
960: 'wife',
```

```
961: 'electronic',
962: 'folk',
963: 'neither',
964: 'item',
965: 'evil',
966: 'associate',
967: 'pull',
968: 'heart',
969: 'colorado',
970: 'trial',
971: 'excellent',
972: 'apparently',
973: 'aid',
974: 'risk',
975: 'hole',
976: 'link',
977: 'recent',
978: 'park',
979: 'stick',
980: 'suspect',
981: 'ride',
982: 'client',
983: 'dog',
984: 'van',
985: 'alone',
986: 'upgrade',
987: 'round',
988: 'step',
989: 'originator',
990: 'suffer',
991: 'environment',
992: 'appropriate',
993: 'whose',
994: 'ron',
995: 'soldier',
996: 'ability',
997: 'commit',
998: 'ken',
999: 'listen',
1000: 'btw',
...}
```

In [ ]:
```
vocab_size = len(tokenizer.index_word) + 1
print('Vocab Size:', vocab_size)
```

```
Vocab Size: 150641
```

In [ ]:
```
X_train_token = tokenizer.texts_to_sequences(X_train)
X_test_token = tokenizer.texts_to_sequences(X_test)
```

In [ ]:
```
print("First Intance Text:\n")
print(X_train[0])
print("\nFirst Intance Total Words:", len(str(X_train[0]).split()))
```

```
First Intance Text:

lerxst wamumdedu thing subject car nntppostinghost rac3wamumdedu organization uni
versity maryland college park line wonder anyone could enlighten car saw day 2doo
r sport car look late 60 early 70 call bricklin door really small addition front
bumper separate rest body know anyone tellme model name engine spec year producti
on car make history whatever info funky look car please email thanks bring neighb
orhood lerxst

First Intance Total Words: 62
```

In [ ]:
```
print("First Intance Text Sequence:\n")
print(X_train_token[0])
print("\nFirst Intance Text Sequence Length:", len(X_train_token[0]))
```

```
First Intance Text Sequence:

[26797, 4580, 36, 2, 97, 18, 18381, 3, 16, 2160, 441, 978, 1, 419, 58, 27, 5471,
97, 524, 76, 18382, 1039, 97, 40, 498, 9294, 531, 7168, 53, 26798, 896, 71, 248,
922, 638, 5270, 1124, 584, 414, 11, 58, 41507, 374, 106, 643, 1919, 29, 1950, 97,
14, 429, 617, 239, 18383, 40, 97, 68, 78, 90, 420, 4068, 26797]

First Intance Text Sequence Length: 62
```

In [ ]:
```
print("Second Intance Text:\n")
print(X_train[1])
print("\nSecond Intance Total Words:", len(str(X_train[1]).split()))
```

```
Second Intance Text:

guykuo carsonuwashingtonedu guy kuo subject clock poll final call summary final c
all clock report keywords acceleration clock upgrade articleid shelley1qvfo9innc3
s organization university washington line nntppostinghost carsonuwashingtonedu fa
ir number brave soul upgrade clock oscillator share experience poll please send b
rief message detail experience procedure top speed attain cpu rat speed add card
adapter heat sink hour usage per day floppy disk functionality 800 floppy especia
lly request summarize next two day please add network knowledge base do clock upg
rade answer poll thanks guy kuo guykuo uwashingtonedu

Second Intance Total Words: 84
```

In [ ]:
```
print("Second Intance Text Sequence:\n")
print(X_train_token[1])
print("\nSecond Intance Text Sequence Length:", len(X_train_token[1]))
```

```
Second Intance Text Sequence:

[10658, 3058, 335, 7841, 2, 1004, 3089, 652, 53, 442, 652, 53, 1004, 236, 241, 35
65, 1004, 986, 402, 62688, 3, 16, 536, 1, 18, 3058, 1258, 65, 1330, 1331, 986, 10
04, 5967, 705, 326, 3089, 68, 123, 2076, 179, 544, 326, 1819, 464, 271, 7842, 125
2, 2217, 271, 322, 122, 1837, 1617, 4186, 775, 2317, 546, 76, 944, 223, 3947, 164
9, 944, 528, 493, 3910, 202, 48, 76, 68, 322, 351, 732, 172, 119, 1004, 986, 194,
3089, 90, 335, 7841, 10658, 4430]

Second Intance Text Sequence Length: 84
```

In [ ]:
```
sequence_len = 300
X_train_token = pad_sequences(X_train_token, padding='post', maxlen=sequence_len
X_test_token = pad_sequences(X_test_token, padding='post', maxlen=sequence_len)
```

In [ ]:
```
print("First Intance Text Sequence:\n")
print(X_train_token[0])
print("\nFirst Intance Text Sequence Length:", len(X_train_token[0]))
```

```
First Intance Text Sequence:

[26797  4580    36     2    97    18 18381     3    16  2160   441   978
     1   419    58    27  5471    97   524    76 18382  1039    97    40
   498  9294   531  7168    53 26798   896    71   248   922   638  5270
  1124   584   414    11    58 41507   374   106   643  1919    29  1950
    97    14   429   617   239 18383    40    97    68    78    90   420
  4068 26797     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0]

First Intance Text Sequence Length: 300
```

In [ ]:
```
print("Second Intance Text Sequence:\n")
print(X_train_token[1])
print("\nSecond Intance Text Sequence Length:", len(X_train_token[1]))
```

```
Second Intance Text Sequence:

[10658  3058   335  7841     2  1004  3089   652    53   442   652    53
  1004   236   241  3565  1004   986   402 62688     3    16   536     1
    18  3058  1258    65  1330  1331   986  1004  5967   705   326  3089
    68   123  2076   179   544   326  1819   464   271  7842  1252  2217
   271   322   122  1837  1617  4186   775  2317   546    76   944   223
  3947  1649   944   528   493  3910   202    48    76    68   322   351
   732   172   119  1004   986   194  3089    90   335  7841 10658  4430
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0]

Second Intance Text Sequence Length: 300
```

In [ ]:
```
home = os.path.expanduser('~')
glove_embedding_filepath = os.path.join(home, "/kaggle/input/glove-6b-100dim/glo
```

In [ ]:
```
def create_embedding_matrix (filepath, word_index, embedding_dim):
    vocab_size = len(word_index) + 1
    embedding_matrix = np.zeros((vocab_size, embedding_dim))

    with open(filepath) as file:
        for line in file:
            word, *vector = line.split()
            if word in word_index:
                idx = word_index[word]
                embedding_matrix[idx] = np.array(vector, dtype=np.float32)[:embe

    return embedding_matrix
```

In [ ]:
```
embedding_dim = 100
embedding_matrix = create_embedding_matrix(glove_embedding_filepath, tokenizer.w
```

****without glove****

In [ ]:
```
#Without Glove
model = Sequential()
model.add(layers.Embedding(input_dim=vocab_size, output_dim=embedding_dim, input
model.add(layers.Conv1D(filters=128, kernel_size=5, activation='relu'))
# model.add(layers.GlobalMaxPool1D())
model.add(layers.Bidirectional(layers.LSTM(units=200, dropout=0.25, recurrent_dr
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(32, activation='relu'))
```

```
model.add(layers.Dense(20, activation='softmax'))
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=
model.summary()
```

Model: "sequential_11"

Layer (type)	Output Shape	Param #
embedding_10 (Embedding)	(None, 300, 100)	15,064,100
conv1d_6 (Conv1D)	(None, 296, 128)	64,128
bidirectional_6 (Bidirectional)	(None, 400)	526,400
dense_12 (Dense)	(None, 64)	25,664
dense_13 (Dense)	(None, 32)	2,080
dense_14 (Dense)	(None, 20)	660

Total params: 15,683,032 (59.83 MB)

Trainable params: 15,683,032 (59.83 MB)

Non-trainable params: 0 (0.00 B)

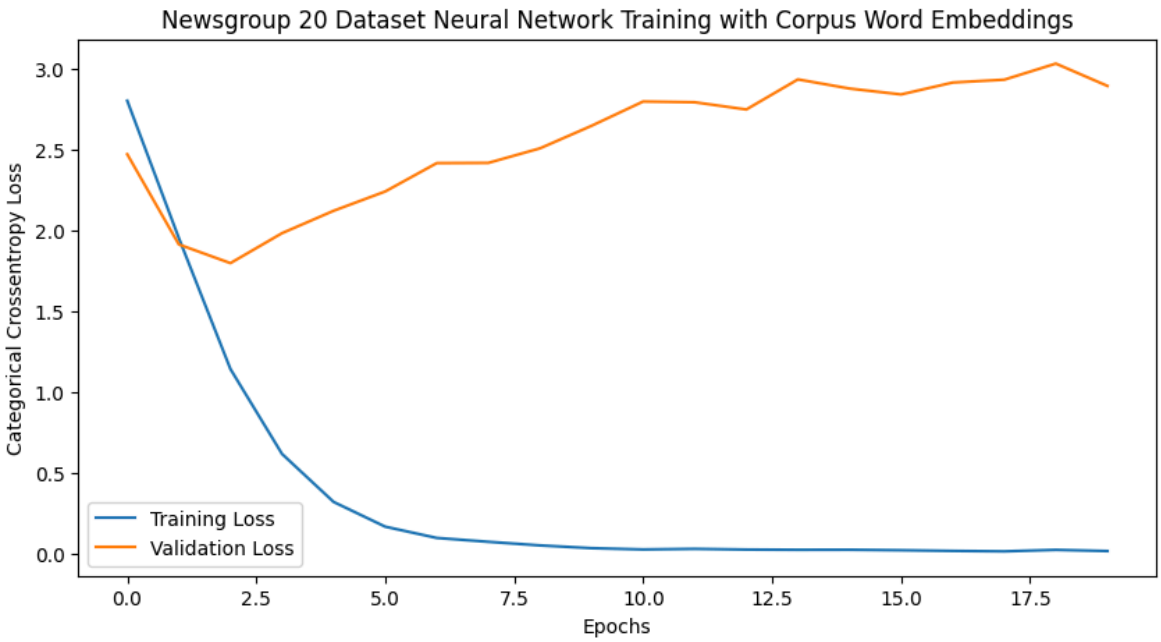
In [ ]:
```
history = model.fit(X_train_token, y_train, epochs=20, validation_data=(X_test_t
```

```
Epoch 1/20
89/89 ━━━━━━━━━━━━━━━━━━━━ 73s 776ms/step - accuracy: 0.0633 - loss: 2.9393 - val
_accuracy: 0.1673 - val_loss: 2.4704
Epoch 2/20
89/89 ━━━━━━━━━━━━━━━━━━━━ 68s 769ms/step - accuracy: 0.2526 - loss: 2.1336 - val
_accuracy: 0.3798 - val_loss: 1.9123
Epoch 3/20
89/89 ━━━━━━━━━━━━━━━━━━━━ 69s 773ms/step - accuracy: 0.5641 - loss: 1.2510 - val
_accuracy: 0.4888 - val_loss: 1.7966
Epoch 4/20
89/89 ━━━━━━━━━━━━━━━━━━━━ 68s 770ms/step - accuracy: 0.7789 - loss: 0.6609 - val
_accuracy: 0.5584 - val_loss: 1.9819
Epoch 5/20
89/89 ━━━━━━━━━━━━━━━━━━━━ 69s 771ms/step - accuracy: 0.8943 - loss: 0.3455 - val
_accuracy: 0.6018 - val_loss: 2.1205
Epoch 6/20
89/89 ━━━━━━━━━━━━━━━━━━━━ 68s 770ms/step - accuracy: 0.9533 - loss: 0.1586 - val
_accuracy: 0.6194 - val_loss: 2.2399
Epoch 7/20
89/89 ━━━━━━━━━━━━━━━━━━━━ 68s 765ms/step - accuracy: 0.9713 - loss: 0.0962 - val
_accuracy: 0.6244 - val_loss: 2.4156
Epoch 8/20
89/89 ━━━━━━━━━━━━━━━━━━━━ 68s 764ms/step - accuracy: 0.9826 - loss: 0.0654 - val
_accuracy: 0.6272 - val_loss: 2.4169
Epoch 9/20
89/89 ━━━━━━━━━━━━━━━━━━━━ 82s 763ms/step - accuracy: 0.9872 - loss: 0.0458 - val
_accuracy: 0.6357 - val_loss: 2.5068
Epoch 10/20
89/89 ━━━━━━━━━━━━━━━━━━━━ 83s 772ms/step - accuracy: 0.9923 - loss: 0.0330 - val
_accuracy: 0.6326 - val_loss: 2.6464
Epoch 11/20
89/89 ━━━━━━━━━━━━━━━━━━━━ 69s 781ms/step - accuracy: 0.9929 - loss: 0.0285 - val
_accuracy: 0.6308 - val_loss: 2.7966
Epoch 12/20
89/89 ━━━━━━━━━━━━━━━━━━━━ 69s 777ms/step - accuracy: 0.9917 - loss: 0.0319 - val
_accuracy: 0.6288 - val_loss: 2.7919
Epoch 13/20
89/89 ━━━━━━━━━━━━━━━━━━━━ 69s 778ms/step - accuracy: 0.9932 - loss: 0.0219 - val
_accuracy: 0.6378 - val_loss: 2.7473
Epoch 14/20
89/89 ━━━━━━━━━━━━━━━━━━━━ 69s 776ms/step - accuracy: 0.9944 - loss: 0.0232 - val
_accuracy: 0.6308 - val_loss: 2.9335
Epoch 15/20
89/89 ━━━━━━━━━━━━━━━━━━━━ 82s 776ms/step - accuracy: 0.9910 - loss: 0.0294 - val
_accuracy: 0.6451 - val_loss: 2.8768
Epoch 16/20
89/89 ━━━━━━━━━━━━━━━━━━━━ 69s 774ms/step - accuracy: 0.9942 - loss: 0.0246 - val
_accuracy: 0.6401 - val_loss: 2.8404
Epoch 17/20
89/89 ━━━━━━━━━━━━━━━━━━━━ 69s 775ms/step - accuracy: 0.9951 - loss: 0.0161 - val
_accuracy: 0.6419 - val_loss: 2.9140
Epoch 18/20
89/89 ━━━━━━━━━━━━━━━━━━━━ 69s 772ms/step - accuracy: 0.9973 - loss: 0.0119 - val
_accuracy: 0.6406 - val_loss: 2.9317
Epoch 19/20
89/89 ━━━━━━━━━━━━━━━━━━━━ 68s 769ms/step - accuracy: 0.9930 - loss: 0.0243 - val
_accuracy: 0.6333 - val_loss: 3.0312
Epoch 20/20
89/89 ━━━━━━━━━━━━━━━━━━━━ 69s 776ms/step - accuracy: 0.9945 - loss: 0.0161 - val
_accuracy: 0.6475 - val_loss: 2.8930
```

In [ ]:
```
metrics_df = pd.DataFrame(history.history)
print(metrics_df)
```

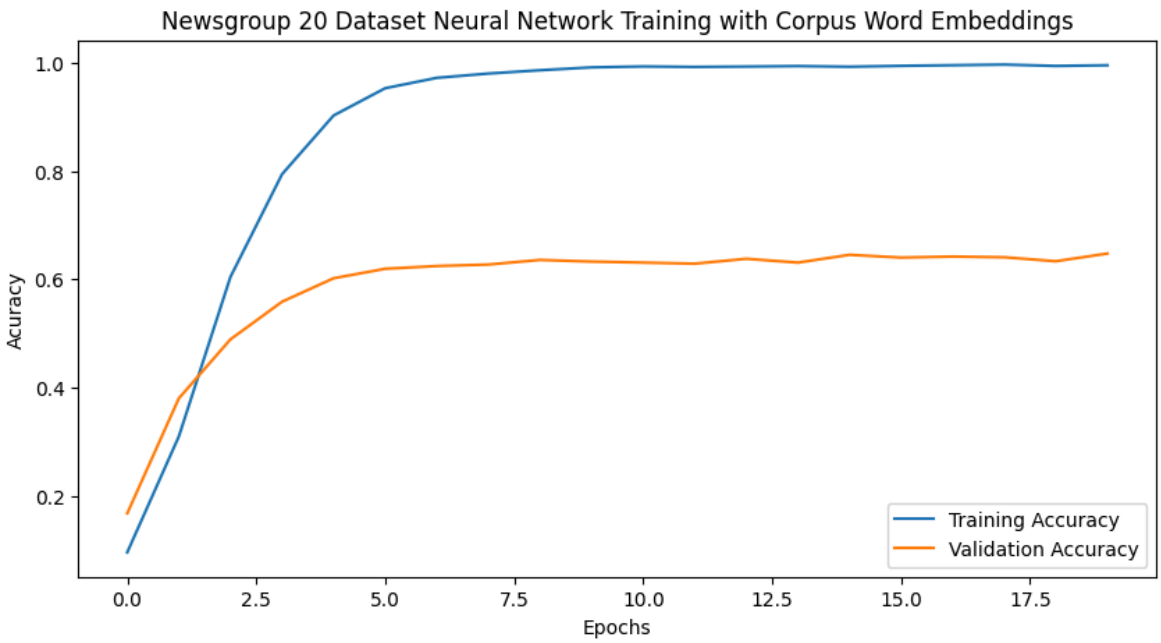
```
    accuracy      loss  val_accuracy  val_loss
0   0.095192  2.801353      0.167286  2.470394
1   0.309440  1.951918      0.379846  1.912320
2   0.604207  1.141521      0.488848  1.796633
3   0.794060  0.616264      0.558417  1.981927
4   0.902687  0.319255      0.601832  2.120490
5   0.953067  0.166178      0.619357  2.239902
6   0.972247  0.096420      0.624403  2.415577
7   0.980202  0.072512      0.627191  2.416868
8   0.986300  0.050239      0.635688  2.506788
9   0.991603  0.033082      0.632634  2.646400
10  0.993283  0.025210      0.630775  2.796607
11  0.992399  0.028891      0.628784  2.791950
12  0.993106  0.024535      0.637812  2.747289
13  0.993990  0.023071      0.630775  2.933472
14  0.992752  0.023407      0.645114  2.876846
15  0.994432  0.020360      0.640069  2.840378
16  0.995581  0.016195      0.641928  2.914028
17  0.996730  0.013409      0.640600  2.931664
18  0.994255  0.022739      0.633298  3.031223
19  0.995492  0.015326      0.647504  2.892998
```

In [ ]:
```
plt.figure(figsize=(10,5))
plt.plot(metrics_df.index, metrics_df.loss)
plt.plot(metrics_df.index, metrics_df.val_loss)
plt.title('Newsgroup 20 Dataset Neural Network Training with Corpus Word Embeddi
plt.xlabel('Epochs')
plt.ylabel('Categorical Crossentropy Loss')
plt.legend(['Training Loss', 'Validation Loss'])
plt.show()
```



In [ ]:
```
plt.figure(figsize=(10,5))
plt.plot(metrics_df.index, metrics_df.accuracy)
plt.plot(metrics_df.index, metrics_df.val_accuracy)
plt.title('Newsgroup 20 Dataset Neural Network Training with Corpus Word Embeddi
plt.xlabel('Epochs')
plt.ylabel('Acuracy')
```

```
plt.legend(['Training Accuracy', 'Validation Accuracy'])
plt.show()
```



With Glove

In [ ]:
```
model = Sequential()
model.add(layers.Embedding(input_dim=vocab_size, output_dim=embedding_dim, input
model.add(layers.Conv1D(filters=128, kernel_size=5, activation='relu'))
model.add(layers.Bidirectional(layers.LSTM(units=200, dropout=0.25, recurrent_dr
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(32, activation='relu'))
model.add(layers.Dense(20, activation='softmax'))
model.layers[0].set_weights([embedding_matrix])
model.layers[0].trainable = True
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=
model.summary()
```

Model: "sequential_21"

Layer (type)	Output Shape	Param #
embedding_20 (Embedding)	(None, 300, 100)	15,064,100
conv1d_9 (Conv1D)	(None, 296, 128)	64,128
bidirectional_9 (Bidirectional)	(None, 400)	526,400
dense_15 (Dense)	(None, 64)	25,664
dense_16 (Dense)	(None, 32)	2,080
dense_17 (Dense)	(None, 20)	660

Total params: 15,683,032 (59.83 MB)

Trainable params: 15,683,032 (59.83 MB)

Non-trainable params: 0 (0.00 B)

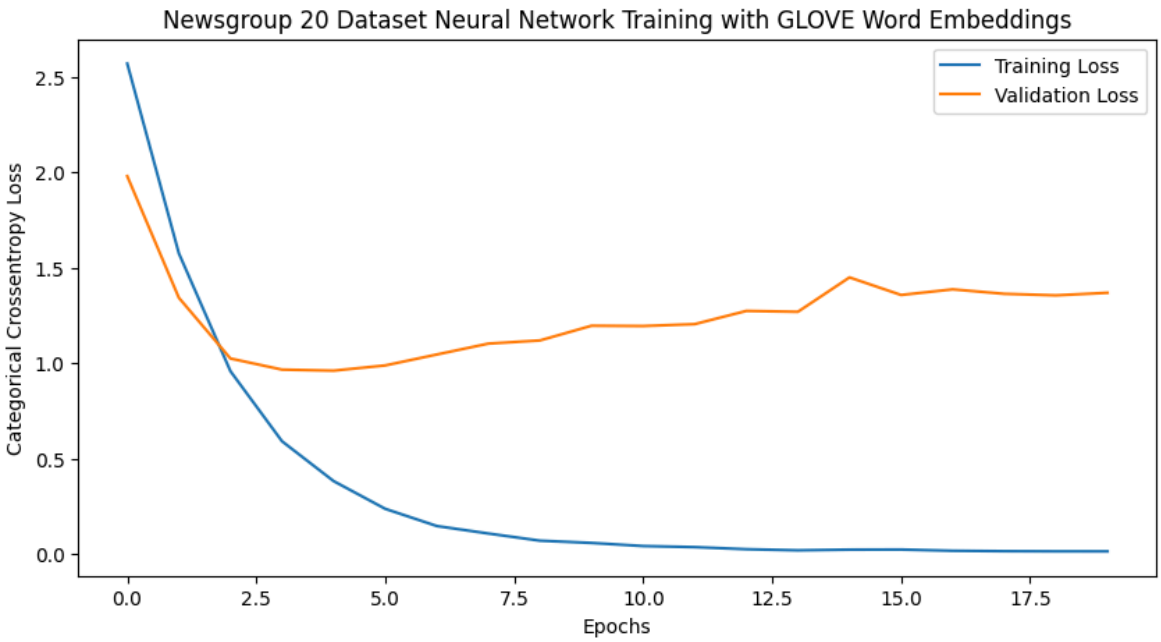
In [ ]:
```
history1 = model.fit(X_train_token, y_train, epochs=20, validation_data=(X_test_
```

```
Epoch 1/20
89/89 ━━━━━━━━━━━━━━━━━━━━ 74s 792ms/step - accuracy: 0.1212 - loss: 2.8294 - val
_accuracy: 0.3225 - val_loss: 1.9792
Epoch 2/20
89/89 ━━━━━━━━━━━━━━━━━━━━ 69s 776ms/step - accuracy: 0.4114 - loss: 1.7482 - val
_accuracy: 0.5466 - val_loss: 1.3427
Epoch 3/20
89/89 ━━━━━━━━━━━━━━━━━━━━ 69s 772ms/step - accuracy: 0.6492 - loss: 1.0311 - val
_accuracy: 0.6633 - val_loss: 1.0245
Epoch 4/20
89/89 ━━━━━━━━━━━━━━━━━━━━ 69s 779ms/step - accuracy: 0.7871 - loss: 0.6327 - val
_accuracy: 0.7083 - val_loss: 0.9663
Epoch 5/20
89/89 ━━━━━━━━━━━━━━━━━━━━ 69s 781ms/step - accuracy: 0.8762 - loss: 0.3876 - val
_accuracy: 0.7282 - val_loss: 0.9608
Epoch 6/20
89/89 ━━━━━━━━━━━━━━━━━━━━ 70s 783ms/step - accuracy: 0.9289 - loss: 0.2390 - val
_accuracy: 0.7443 - val_loss: 0.9882
Epoch 7/20
89/89 ━━━━━━━━━━━━━━━━━━━━ 69s 774ms/step - accuracy: 0.9553 - loss: 0.1548 - val
_accuracy: 0.7484 - val_loss: 1.0454
Epoch 8/20
89/89 ━━━━━━━━━━━━━━━━━━━━ 83s 786ms/step - accuracy: 0.9681 - loss: 0.1089 - val
_accuracy: 0.7528 - val_loss: 1.1025
Epoch 9/20
89/89 ━━━━━━━━━━━━━━━━━━━━ 69s 775ms/step - accuracy: 0.9793 - loss: 0.0685 - val
_accuracy: 0.7600 - val_loss: 1.1190
Epoch 10/20
89/89 ━━━━━━━━━━━━━━━━━━━━ 69s 774ms/step - accuracy: 0.9791 - loss: 0.0641 - val
_accuracy: 0.7564 - val_loss: 1.1960
Epoch 11/20
89/89 ━━━━━━━━━━━━━━━━━━━━ 83s 783ms/step - accuracy: 0.9875 - loss: 0.0371 - val
_accuracy: 0.7592 - val_loss: 1.1945
Epoch 12/20
89/89 ━━━━━━━━━━━━━━━━━━━━ 82s 780ms/step - accuracy: 0.9880 - loss: 0.0369 - val
_accuracy: 0.7646 - val_loss: 1.2045
Epoch 13/20
89/89 ━━━━━━━━━━━━━━━━━━━━ 69s 777ms/step - accuracy: 0.9925 - loss: 0.0258 - val
_accuracy: 0.7689 - val_loss: 1.2736
Epoch 14/20
89/89 ━━━━━━━━━━━━━━━━━━━━ 69s 781ms/step - accuracy: 0.9949 - loss: 0.0183 - val
_accuracy: 0.7704 - val_loss: 1.2690
Epoch 15/20
89/89 ━━━━━━━━━━━━━━━━━━━━ 70s 783ms/step - accuracy: 0.9940 - loss: 0.0223 - val
_accuracy: 0.7525 - val_loss: 1.4491
Epoch 16/20
89/89 ━━━━━━━━━━━━━━━━━━━━ 82s 786ms/step - accuracy: 0.9898 - loss: 0.0304 - val
_accuracy: 0.7634 - val_loss: 1.3571
Epoch 17/20
89/89 ━━━━━━━━━━━━━━━━━━━━ 70s 786ms/step - accuracy: 0.9961 - loss: 0.0142 - val
_accuracy: 0.7679 - val_loss: 1.3865
Epoch 18/20
89/89 ━━━━━━━━━━━━━━━━━━━━ 70s 784ms/step - accuracy: 0.9953 - loss: 0.0181 - val
_accuracy: 0.7683 - val_loss: 1.3632
Epoch 19/20
89/89 ━━━━━━━━━━━━━━━━━━━━ 70s 789ms/step - accuracy: 0.9943 - loss: 0.0173 - val
_accuracy: 0.7690 - val_loss: 1.3551
Epoch 20/20
89/89 ━━━━━━━━━━━━━━━━━━━━ 81s 784ms/step - accuracy: 0.9951 - loss: 0.0176 - val
_accuracy: 0.7723 - val_loss: 1.3685
```

In [ ]:
```
metrics_df = pd.DataFrame(history1.history)
print(metrics_df)
```

```
    accuracy      loss  val_accuracy  val_loss
0   0.185699  2.568511      0.322491  1.979157
1   0.467739  1.575433      0.546601  1.342673
2   0.673237  0.958261      0.663303  1.024523
3   0.804755  0.591166      0.708311  0.966294
4   0.877055  0.382652      0.728226  0.960839
5   0.926905  0.237579      0.744291  0.988202
6   0.956868  0.147444      0.748407  1.045448
7   0.968181  0.108241      0.752788  1.102472
8   0.979052  0.070739      0.759957  1.119043
9   0.981439  0.058928      0.756373  1.196021
10  0.985770  0.042629      0.759161  1.194531
11  0.988687  0.036988      0.764604  1.204473
12  0.992399  0.026013      0.768853  1.273582
13  0.994697  0.020222      0.770446  1.269041
14  0.992929  0.023840      0.752523  1.449136
15  0.992399  0.024322      0.763409  1.357114
16  0.994785  0.017865      0.767924  1.386522
17  0.995846  0.015691      0.768322  1.363185
18  0.995316  0.014883      0.768986  1.355098
19  0.995934  0.014781      0.772305  1.368527
```

In [ ]:
```
plt.figure(figsize=(10,5))
plt.plot(metrics_df.index, metrics_df.loss)
plt.plot(metrics_df.index, metrics_df.val_loss)
plt.title('Newsgroup 20 Dataset Neural Network Training with GLOVE Word Embeddin
plt.xlabel('Epochs')
plt.ylabel('Categorical Crossentropy Loss')
plt.legend(['Training Loss', 'Validation Loss'])
plt.show()
```



In [ ]:
```
plt.figure(figsize=(10,5))
plt.plot(metrics_df.index, metrics_df.accuracy)
plt.plot(metrics_df.index, metrics_df.val_accuracy)
plt.title('Newsgroup 20 Dataset Neural Network Training with GLOVE Word Embeddin
plt.xlabel('Epochs')
plt.ylabel('Acuracy')
```

```
plt.legend(['Training Accuracy', 'Validation Accuracy'])
plt.show()
```