Step 0: Import Packages

import gensim from gensim.models import Word2Vec

```
#5. Implement the Continuous Bag of Words (CBOW) Model. Stages can be:
a. Data preparation
b. Generate training data
c. Train model
d. Output
```

```
import matplotlib.pyplot as plt
import seaborn as sns
import matplotlib as mpl
import matplotlib.pylab as pylab
import numpy as np
%matplotlib inline
from nltk.tokenize import sent_tokenize, word_tokenize
import warnings
warnings.filterwarnings(action = 'ignore')
import gensim
from gensim.models import Word2Vec
import re
import bs4 as bs
import urllib.request
import nltk
```

```
nltk.download('punkt')
nltk.download('stopwords')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
True
```

Step 1(a) : Download or collect data. or Prepare Corpus

```
 #a. Data preparation
scrapped_data=urllib.request.urlopen("https://en.wikipedia.org/wiki/Machine_learning")
article=scrapped_data.read()
paresed_article=bs.BeautifulSoup(article,'lxml')
paragraphs=paresed_article.find_all('p')
article_text=""
for p in paragraphs:
  article_text+=p.text
sentences=article_text


print(article_text)
```

```
Classification of machine learning models can be validated by accuracy estimation
In addition to overall accuracy, investigators frequently report sensitivity and s
Machine learning poses a host of ethical questions. Systems that are trained on da
While responsible collection of data and documentation of algorithmic rules used b
AI can be well-equipped to make decisions in technical fields, which rely heavily
Other forms of ethical challenges, not related to personal biases, are seen in hea
Since the 2010s, advances in both machine learning algorithms and computer hardwar
A physical neural network or Neuromorphic computer  is a type of artificial neural
Embedded Machine Learning is a sub-field of machine learning, where the machine le
Software suites containing a variety of machine learning algorithms include the fo
```

Instead of lengthy text on wiki, we can try for a small text as well

```
sentences="""Alice 23 opened the door and found that it led into a small 90
passage, not much larger than a rat-hole:  she knelt down and
looked along the passage into the loveliest garden you ever saw.
How she longed to get out of that dark hall, and wander about
among those beds of bright flowers and those cool fountains, but
she could not even get her head through the doorway; `and even if
my head would go through,' (thought) $poor Alice, `it would be of
very little use without my shoulders.  Oh, how I wish
I could shut up like a telescope!  I think I could, if I only
know how to begin.'  For, you see, so many out-of-the-way things
had happened lately, that Alice had begun to think that very few
things indeed were really impossible.


"""
```

```
sentences = re.sub('[^A-Za-z0-9]+', ' ', sentences)
sentences = re.sub(r'(?:^| )\w(?:$| )', ' ', sentences).strip()
print(sentences)
```

```
Alice 23 opened the door and found that it led into small 90 passage not much larger
```

Step 1(b) : Prepare Dataset

1. Remove all special characters and digits
2. Remove all punctuation marks
3. bring all letters to lower case
4. Tokenize the document into sentences and words
5. Remove all stop words.

Step 2: Generate Training Data

```
# remove special characters
sentences = re.sub('[^A-Za-z]+', ' ', sentences)

# remove 1 letter words
sentences = re.sub(r'(?:^| )\w(?:$| )', ' ', sentences).strip()

# lower all characters
sentences = sentences.lower()

all_sent=nltk.sent_tokenize(sentences)
all_words=[nltk.word_tokenize(sent) for sent in all_sent]

from nltk.corpus import stopwords
for i in range(len(all_words)):
  all_words[i]=[w for w in all_words[i] if w not in stopwords.words('english')]
data =all_words
data1=data[0]
```

Step 3 : Train the model

Use Gensim word2vec for training the model.

```
model1 = gensim.models.Word2Vec(data, min_count = 1,vector_size = 52, window = 5)
```

```
vocabulary=list(model1.wv.index_to_key)
print(vocabulary)
```

```
['could', 'alice', 'passage', 'think', 'things', 'even', 'head', 'get', 'would', 'eve
```

```
wrd='door'
#wrd=['subset','machine', 'learning','closely','related']
v1=model1.wv[wrd]
similar_words=model1.wv.most_similar(wrd)
for x in similar_words:
  print(x)
```

```
('beds', 0.36491504311561584)
('much', 0.3305249512195587)
('shut', 0.32979297637939453)
('cool', 0.25908932089805603)
('wish', 0.243193581700325)
('oh', 0.24176433682441711)
('begun', 0.2212926298379898)
('begin', 0.17681987583637238)
('loveliest', 0.14279094338417053)
('things', 0.13509944081306458)
```

Prepare Context_words_list

```
print(data1)
```

```
['alice', 'opened', 'door', 'found', 'led', 'small', 'passage', 'much', 'larger', 'ra
```

```
#print(data)

dat = []

for i in range(0, len(data) ):
    context = [data1[i - 2], data1[i - 1], data1[i+1], data1[i + 2]]
    target = data1[i]
    dat.append((context, target))
print(dat[:5])
#print(dat[1][0])
```

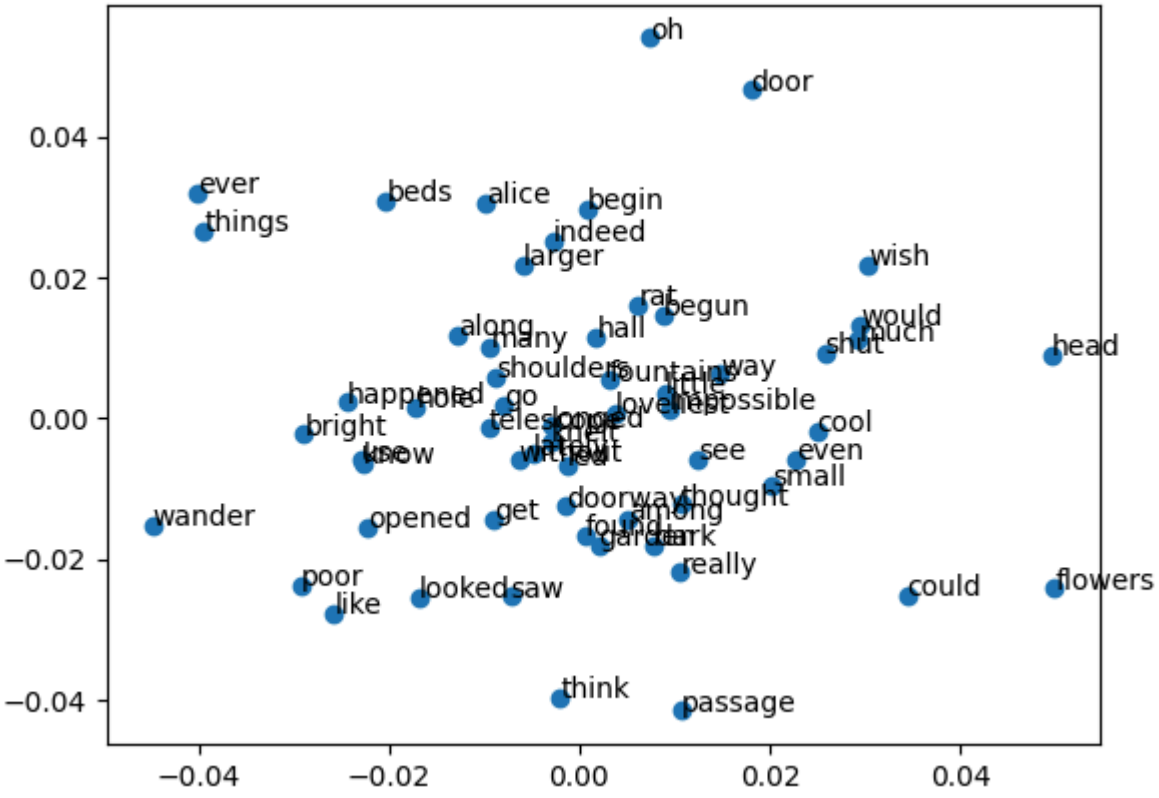
```
[(['really', 'impossible', 'opened', 'door'], 'alice')]
```

```
#for val in dat:
 # print(val[0],val[1])
i=0
print(dat[i][0],dat[i][1])
print(model1.predict_output_word(dat[i][0]))
```

```
['really', 'impossible', 'opened', 'door'] alice
[('even', 0.016949415), ('lately', 0.016949397), ('among', 0.016949339), ('saw', 0.01
```

```
 #d. Output
from sklearn.decomposition import PCA
from matplotlib import pyplot
X = model1.wv.vectors
pca = PCA(n_components=2)
result = pca.fit_transform(X)

pyplot.scatter(result[:, 0], result[:, 1])
word =list(model1.wv.index_to_key)
for i, word in enumerate(word):
   pyplot.annotate(word, xy=(result[i, 0], result[i, 1]))
pyplot.show()
```



Start coding or generate with AI.