```
data=['Artifical intelligence is the ability of a machine to think and learn','There are three types of Artifical intelligen
```

```
from sklearn.feature_extraction.text import TfidfVectorizer,CountVectorizer
```

```
cv=CountVectorizer()
result=cv.fit_transform(data)
print(result)
```

```
  (0, 3)        1
  (0, 4)        1
  (0, 5)        1
  (0, 9)        1
  (0, 0)        1
  (0, 8)        1
  (0, 7)        1
  (0, 13)       1
  (0, 11)       1
  (0, 1)        1
  (0, 6)        1
  (1, 3)        1
  (1, 4)        1
  (1, 8)        1
  (1, 10)       1
  (1, 2)        1
  (1, 12)       1
  (1, 14)       1
```

```
import pandas as pd
names=cv.get_feature_names()
print(names)
pd.DataFrame(result.toarray(),columns=names)
```

```
['ability', 'and', 'are', 'artifical', 'intelligence', 'is', 'learn', 'machine', 'of', 'the', 'there', 'think', 'three'
```

```
cv1=TfidfVectorizer()
result1=cv1.fit_transform(data)
print(result1)
```

```
  (0, 6)        0.32412344955584815
  (0, 1)        0.32412344955584815
  (0, 11)       0.32412344955584815
  (0, 13)       0.32412344955584815
  (0, 7)        0.32412344955584815
  (0, 8)        0.23061650387901603
  (0, 0)        0.32412344955584815
  (0, 9)        0.32412344955584815
  (0, 5)        0.32412344955584815
  (0, 4)        0.23061650387901603
  (0, 3)        0.23061650387901603
  (1, 14)       0.42567716283146345
  (1, 12)       0.42567716283146345
  (1, 2)        0.42567716283146345
  (1, 10)       0.42567716283146345
  (1, 8)        0.3028728072833121
  (1, 4)        0.3028728072833121
  (1, 3)        0.3028728072833121
```

```
import pandas as pd
names=cv1.get_feature_names()
print(names)
pd.DataFrame(result1.toarray(),columns=names)
```

```
['ability', 'and', 'are', 'artifical', 'intelligence', 'is', 'learn', 'machine', 'of', 'the', 'there', 'think', 'three'
```

	ability	and	are	artifical	intelligence	is	learn	machine	of	the	there	th
0	0.324123	0.324123	0.000000	0.230617	0.230617	0.324123	0.324123	0.324123	0.230617	0.324123	0.000000	0.324
1	0.000000	0.000000	0.425677	0.302873	0.302873	0.000000	0.000000	0.000000	0.302873	0.000000	0.425677	0.000

```
from gensim.models import Word2Vec
```

```
from gensim.utils import simple_preprocess
```

```
from gensim.utils import simple_preprocess
```

```
store=[simple_preprocess(i) for i in data]
store
```

```
[['artifical',
  'intelligence',
  'is',
  'the',
  'ability',
  'of',
  'machine',
  'to',
  'think',
  'and',
  'learn'],
 ['there', 'are', 'three', 'types', 'of', 'artifical', 'intelligence']]
```

```
wordvec=Word2Vec(store,min_count=1,size=30)
```

```
wordvec.most_similar('think')
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:1: DeprecationWarning: Call to deprecated `most_similar` (
  """Entry point for launching an IPython kernel.
[('intelligence', 0.2951553165912628),
 ('there', 0.20442068576812744),
 ('types', 0.1837206482887268),
 ('and', 0.1438174545764923),
 ('the', 0.1329154074192047),
 ('is', 0.07777770608663559),
 ('are', 0.06542398780584335),
 ('learn', 0.006135251373052597),
 ('of', -0.04308314248919487),
 ('to', -0.07751388847827911)]
```

```
input=[10,20,30,40,50,60,70,80,90]
```

```
x=[]
y=[]

def split_data(input):
  for i in range(len(input)):
    last=i+2
    if(last>len(input)-1):
      break
    else:
      a=input[i:last]
      b=input[last]
      x.append(a)
      y.append(b)
  return x,y
```

```
X,Y=split_data(input)
```

```
X
```

```
[[10, 20], [20, 30], [30, 40], [40, 50], [50, 60], [60, 70], [70, 80]]
```

```
Y
```

```
[30, 40, 50, 60, 70, 80, 90]
```

```
import numpy as np
X=np.array(X)
X
```

```
array([[10, 20],
       [20, 30],
       [30, 40],
       [40, 50],
       [50, 60],
```

```
            [60, 70],
            [70, 80]])
```

```
import numpy as np

Y=np.array(Y)
Y
```

```
     array([30, 40, 50, 60, 70, 80, 90])
```

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(X,Y,test_size=0.2,random_state=10)
```

```
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
     (5, 2)
     (2, 2)
     (5,)
     (2,)
```

```
x_train=x_train.reshape(x_train.shape[0],x_train.shape[1],1)
x_test=x_test.reshape(x_test.shape[0],x_test.shape[1],1)
```

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Flatten,Dropout,Dense,BatchNormalization,LSTM
```

```
model=Sequential()
model.add(LSTM(100,activation='relu',input_shape=(2,1)))
model.add(Dense(1))
```

```
WARNING:tensorflow:Layer lstm_1 will not use cuDNN kernel since it doesn't meet the cuDNN kernel criteria. It will use
```

```
model.compile(optimizer='adam',loss='mae',metrics=['mse','mae'])
```

```
model.fit(x_train,y_train,validation_data=(x_test,y_test),epochs=100)
```

```
Epoch 26/100
1/1 [==============================] - 0s 31ms/step - loss: 2.1304 - mse: 11.5183 - mae: 2.1304 - val_loss: 2.1955 -
Epoch 27/100
1/1 [==============================] - 0s 31ms/step - loss: 2.0843 - mse: 11.2082 - mae: 2.0843 - val_loss: 2.2458 -
Epoch 28/100
1/1 [==============================] - 0s 32ms/step - loss: 2.0375 - mse: 10.9078 - mae: 2.0375 - val_loss: 2.2944 -
Epoch 29/100
1/1 [==============================] - 0s 32ms/step - loss: 1.9902 - mse: 10.6155 - mae: 1.9902 - val_loss: 2.3413 -
Epoch 30/100
1/1 [==============================] - 0s 34ms/step - loss: 1.9627 - mse: 10.3303 - mae: 1.9627 - val_loss: 2.3920 -
Epoch 31/100
1/1 [==============================] - 0s 32ms/step - loss: 1.9902 - mse: 10.2404 - mae: 1.9902 - val_loss: 2.3776 -
Epoch 32/100
1/1 [==============================] - 0s 31ms/step - loss: 1.9878 - mse: 10.0477 - mae: 1.9878 - val_loss: 2.3051 -
Epoch 33/100
1/1 [==============================] - 0s 37ms/step - loss: 1.9502 - mse: 9.7360 - mae: 1.9502 - val_loss: 2.1804 -
Epoch 34/100
1/1 [==============================] - 0s 34ms/step - loss: 1.8929 - mse: 9.3664 - mae: 1.8929 - val_loss: 2.0793 -
Epoch 35/100
1/1 [==============================] - 0s 33ms/step - loss: 1.8712 - mse: 9.1814 - mae: 1.8712 - val_loss: 2.0009 -
Epoch 36/100
1/1 [==============================] - 0s 31ms/step - loss: 1.8200 - mse: 9.1266 - mae: 1.8200 - val_loss: 1.8717 -
Epoch 37/100
1/1 [==============================] - 0s 31ms/step - loss: 1.7897 - mse: 8.9955 - mae: 1.7897 - val_loss: 1.7701 -
Epoch 38/100
1/1 [==============================] - 0s 33ms/step - loss: 1.7416 - mse: 8.9957 - mae: 1.7416 - val_loss: 1.6208 -
Epoch 39/100
1/1 [==============================] - 0s 32ms/step - loss: 1.7194 - mse: 8.9144 - mae: 1.7194 - val_loss: 1.4919 -
Epoch 40/100
1/1 [==============================] - 0s 31ms/step - loss: 1.7064 - mse: 8.8131 - mae: 1.7064 - val_loss: 1.3808 -
Epoch 41/100
1/1 [==============================] - 0s 31ms/step - loss: 1.6897 - mse: 8.6922 - mae: 1.6897 - val_loss: 1.2857 -
Epoch 42/100
1/1 [==============================] - 0s 33ms/step - loss: 1.6697 - mse: 8.5529 - mae: 1.6697 - val loss: 1.2047 -
```

```
Epoch 43/100
1/1 [==============================] - 0s 31ms/step - loss: 1.6466 - mse: 8.3967 - mae: 1.6466 - val_loss: 1.1364 -
Epoch 44/100
1/1 [==============================] - 0s 36ms/step - loss: 1.6227 - mse: 8.2252 - mae: 1.6227 - val_loss: 1.0919 -
Epoch 45/100
1/1 [==============================] - 0s 35ms/step - loss: 1.5988 - mse: 8.1255 - mae: 1.5988 - val_loss: 1.0567 -
Epoch 46/100
1/1 [==============================] - 0s 35ms/step - loss: 1.5741 - mse: 8.0027 - mae: 1.5741 - val_loss: 1.0298 -
Epoch 47/100
1/1 [==============================] - 0s 36ms/step - loss: 1.5463 - mse: 7.8559 - mae: 1.5463 - val_loss: 1.0104 -
Epoch 48/100
1/1 [==============================] - 0s 34ms/step - loss: 1.5157 - mse: 7.6860 - mae: 1.5157 - val_loss: 0.9981 -
Epoch 49/100
1/1 [==============================] - 0s 39ms/step - loss: 1.4822 - mse: 7.4941 - mae: 1.4822 - val_loss: 0.9923 -
Epoch 50/100
1/1 [==============================] - 0s 31ms/step - loss: 1.4643 - mse: 7.2824 - mae: 1.4643 - val_loss: 1.0043 -
Epoch 51/100
1/1 [==============================] - 0s 34ms/step - loss: 1.4459 - mse: 7.1890 - mae: 1.4459 - val_loss: 0.9536 -
Epoch 52/100
1/1 [==============================] - 0s 33ms/step - loss: 1.4244 - mse: 7.0368 - mae: 1.4244 - val_loss: 0.8466 -
Epoch 53/100
1/1 [==============================] - 0s 33ms/step - loss: 1.4044 - mse: 6.8402 - mae: 1.4044 - val_loss: 0.7676 -
Epoch 54/100
1/1 [==============================] - 0s 34ms/step - loss: 1.3771 - mse: 6.7560 - mae: 1.3771 - val_loss: 0.7144 -
```

```
a=[10,20]
a=np.array(a)
a.shape
```

```
(2,)
```

```
a=a.reshape(1,2,1)
```

```
model.predict(a)
```

```
array([[26.15179]], dtype=float32)
```