```
# đọc file
filename='tedtalk.txt'
lines=[]
count=0
#Max=-1
Max=1000
with open(filename,'r') as f:
    for s in f:
        count+=1
        if count>Max and Max!=-1:
            break
        lines.append(s.strip())
print(len(lines))
print(lines[:5])
```

```
 1000
 ["Here are two reasons companies fail: they only do more of the same, or they only do what's new"
```

```
# tokenize sentences
import nltk
sentences=[]
all_tokens_count=0
for line in lines:
    tokens = nltk.word_tokenize(line.lower())
    all_tokens_count+=len(tokens)
    #sentences.append(tokens)
    sentences.append(['<s>']+tokens+['</s>'])
print('all_tokens_count=',all_tokens_count)
print(len(sentences))
print(sentences[:5])
```

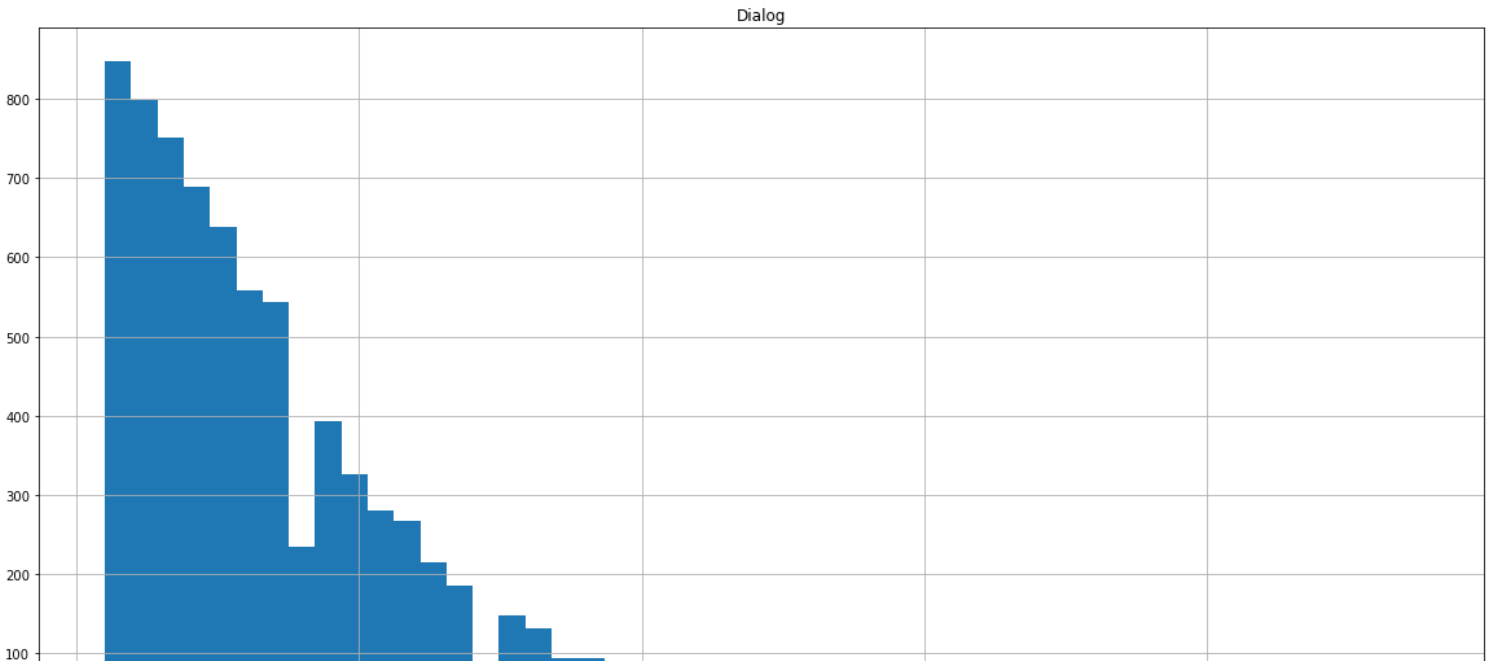
```
 all_tokens_count= 17965
 1000
 [['<s>', 'here', 'are', 'two', 'reasons', 'companies', 'fail', ':', 'they', 'only', 'do', 'more',
```

```
# tạo dữ liệu training w(i-k)...w(i-1) dự đoán w(i)
import random
def get_instances(sent):
    # tạo random t instances từ câu này
    n=len(sent)
    index=set()
    samples=[]
    for i in range(n//2):
        index.add(random.randint(1,n-1))
    for k in index:
        instance=sent[:k+1]
        samples.append(instance)
    return samples

data=[]
for sent in sentences:
```

```
    data.extend(get_instances(sent))
print(len(data))

# test thử hàm get_instances
'''
samples = get_instances(sentences[0])
for instance in samples:
    print(instance)
'''
```

```
7737
'\nsamples = get_instances(sentences[0])\nfor instance in samples:\n    print(instance)\n'
```

```
# bản đồ histogram của data
import pandas as pd
import matplotlib.pyplot as plt
alllength = [len(x) for x in data]
plt.rcParams['figure.figsize'] = (20, 10)
length_df = pd.DataFrame({'Dialog':alllength})
length_df.hist(bins=50)
plt.show()
```




```
import numpy as np
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow import keras
from sklearn.metrics import confusion_matrix,f1_score,classification_report


# lấy max của độ dài để dự đoán từ tiếp theo
max_sequence_len=20

data1 = [' '.join(x) for x in data]

tokenizer = Tokenizer(filters='')
tokenizer.fit_on_texts(data1)
#print(tokenizer.word_index)
```

```
vocab_size = len(tokenizer.word_index) + 1
print('vocab size:',vocab_size)
```

```
vocab size: 2999
```

```
# chuẩn hoá input
data_sequence = tokenizer.texts_to_sequences(data1)
y_data=np.array([x[-1:] for x in data_sequence])
y_data=y_data.ravel()

X_data = pad_sequences(data_sequence,  maxlen=max_sequence_len, padding='pre')
print(X_data[:3])
```

```
[[   0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    1  103   22]
 [   0    0    0    0    0    0    0    0    0    0    0    0    1  103
    22  124  718  173 1183   64]
 [   0    0    0    0    0    0    0    0    0    1  103   22  124  718
   173 1183   64   30   68   37]]
```

```
print(X_data[:3])
print(y_data[:3])
```

```
[[   0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    1  103   22]
 [   0    0    0    0    0    0    0    0    0    0    0    0    1  103
    22  124  718  173 1183   64]
 [   0    0    0    0    0    0    0    0    0    1  103   22  124  718
   173 1183   64   30   68   37]]
[22 64 37]
```

```
#y_data = keras.utils.to_categorical(y_data,num_classes=vocab_size)
print(y_data.shape)
print(y_data[0])
```

```
(7737,)
22
```

```
def get_params():
    return {'max_seq_len': max_sequence_len, 'vocab_size':vocab_size }
params = get_params()

inputs = keras.layers.Input(shape=(params['max_seq_len'],))
embedding = keras.layers.Embedding(params['vocab_size'], 200, input_length=params
print(inputs.shape)
print(embedding.shape)
```

```
(None, 20)
(None, 20, 200)
```

```
# building CNN model
conv1 = keras.layers.Conv1D(filters=32, kernel_size=2, activation='relu')(embeddi
conv1 = keras.layers.Flatten()(conv1)
dense = keras.layers.Dense(128, activation='relu')(conv1)
```

```
dense = keras.layers.Dense(vocab_size, activation='softmax')(dense)
model = keras.Model(inputs=inputs, outputs=dense)


model.compile(loss='sparse_categorical_crossentropy', optimizer='adam',metrics=['
model.summary()
```

```
Model: "model_6"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_7 (InputLayer)         [(None, 20)]              0
_________________________________________________________________
embedding_6 (Embedding)      (None, 20, 200)           599800
_________________________________________________________________
conv1d_1 (Conv1D)            (None, 19, 32)            12832
_________________________________________________________________
flatten_1 (Flatten)          (None, 608)               0
_________________________________________________________________
dense_12 (Dense)             (None, 128)               77952
_________________________________________________________________
dense_13 (Dense)             (None, 2999)              386871
=================================================================
Total params: 1,077,455
Trainable params: 1,077,455
Non-trainable params: 0
_________________________________________________________________
```

```
type(y_data),len(y_data)
```

```
(numpy.ndarray, 7737)
```

```
print(X_data.shape)
model.fit(X_data,y_data,batch_size=100, epochs=3)
```

```
(7737, 20)
Epoch 1/3
78/78 [==============================] - 4s 54ms/step - loss: 6.6931 - accuracy: 0.0685
Epoch 2/3
78/78 [==============================] - 4s 53ms/step - loss: 5.3050 - accuracy: 0.1059
Epoch 3/3
78/78 [==============================] - 4s 54ms/step - loss: 4.6469 - accuracy: 0.2025
<tensorflow.python.keras.callbacks.History at 0x7f5791291970>
```

```
inputs2 = keras.layers.Input(shape=(params['max_seq_len'],))
embedding2 = keras.layers.Embedding(params['vocab_size'], 200, input_length=param
# building CNN model
lstm = keras.layers.LSTM(100)(embedding2)
dense2 = keras.layers.Dense(128, activation='relu')(lstm)
dense2 = keras.layers.Dense(vocab_size, activation='softmax')(dense2)
model2 = keras.Model(inputs=inputs2, outputs=dense2)
model2.compile(loss='sparse_categorical_crossentropy', optimizer='adam',metrics=[
model2.summary()
```

```
Model: "model_7"
_________________________________________________________________
```

```
Layer (type)                 Output Shape              Param #
=================================================================
input_8 (InputLayer)         [(None, 20)]              0
_________________________________________________________________
embedding_7 (Embedding)      (None, 20, 200)           599800
_________________________________________________________________
lstm_5 (LSTM)                (None, 100)               120400
_________________________________________________________________
dense_14 (Dense)             (None, 128)               12928
_________________________________________________________________
dense_15 (Dense)             (None, 2999)              386871
=================================================================
Total params: 1,119,999
Trainable params: 1,119,999
Non-trainable params: 0
_________________________________________________________________
```

```
model2.fit(X_data,y_data,batch_size=100, epochs=3)
```

```
Epoch 1/3
78/78 [==============================] - 7s 87ms/step - loss: 6.7176 - accuracy: 0.0592
Epoch 2/3
78/78 [==============================] - 6s 75ms/step - loss: 5.5215 - accuracy: 0.0719
Epoch 3/3
78/78 [==============================] - 6s 74ms/step - loss: 5.1566 - accuracy: 0.0888
<tensorflow.python.keras.callbacks.History at 0x7f578e6749d0>
```