In [5]:
```python
import io

import random

import string # to process standard python strings

import warnings

warnings.filterwarnings('ignore')

import numpy as np

from sklearn.feature_extraction.text import TfidfVectorizer # convert a collection

from sklearn.metrics.pairwise import cosine_similarity

from nltk.stem import WordNetLemmatizer
```

In [6]:
```python
#!pip install nltk

import nltk
```

In [9]:
```python
#installing NLTK packages
nltk.download('popular', quiet=True) # for downloading packages

nltk.download('punkt') # first-time use only

nltk.download('wordnet') # first-time use only
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\Vaish\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\Vaish\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
```

Out[9]:
```
True
```

In [23]:
```python
#reading the corpus
f = open('input.txt','r',errors = 'ignore')
raw = f.read()
raw = raw.lower()#converts to lowercase
```

In [25]:
```python
#tokenization
import nltk

sent_tokens = nltk.sent_tokenize(raw)# converts to list of sentences

word_tokens = nltk.word_tokenize(raw)# converts to list of words
```

In [27]:
```python
lemmer = nltk.stem.WordNetLemmatizer()

#WordNet is a semantically-oriented dictionary of English included in NLTK.
```

```
def LemTokens(tokens):

    return [lemmer.lemmatize(token) for token in tokens]



remove_punct_dict = dict((ord(punct), None) for punct in string.punctuation)



def LemNormalize(text):

    return LemTokens(nltk.word_tokenize(text.lower().translate(remove_punct_dict)))
```

In [29]:
```
GREETING_INPUTS = ("hello", "hi", "greetings", "what's up","hey",\

                   "how are you?")

GREETING_RESPONSES = ["hi", "hey", "hi there", "hello", \

                      "I am glad! You are talking to me", \

                      "I am fine! How about you?"]

def greeting(sentence):

    for word in sentence.split():

        if word.lower() in GREETING_INPUTS:

            return random.choice(GREETING_RESPONSES)
```

In [31]:
```
def response(user_response):

    robo_response=''

    sent_tokens.append(user_response)

    TfidfVec = TfidfVectorizer(tokenizer=LemNormalize, stop_words='english')

    tfidf = TfidfVec.fit_transform(sent_tokens)

    vals = cosine_similarity(tfidf[-1], tfidf)

    idx=vals.argsort()[0][-2]

    flat = vals.flatten()

    flat.sort()

    req_tfidf = flat[-2]

    if(req_tfidf==0):

        robo_response=robo_response+"I am sorry! I don't understand you."
        return robo_response

    else:

        robo_response = robo_response+sent_tokens[idx]

        return robo_response
```

In [37]:
```
flag=True

print("SABot: My name is SABot. How can I assist you?. If you want to exit, type By

while(flag==True):

    user_response = input()

    user_response=user_response.lower()

    if(user_response!='bye'):

        if(user_response=='thanks' or user_response=='thank you' ):

            flag=False

            print("SABot: You are welcome...")

        else:

            if(greeting(user_response)!=None):

                print("SABot: "+greeting(user_response))

            else:

                print("SABot: ",end=" ")

                print(response(user_response))

                sent_tokens.remove(user_response)

    else:

        flag=False

        print("SABot: Bye! take care...")
```

```
SABot: My name is SABot. How can I assist you?. If you want to exit, type Bye!
SABot: hi
SABot: hello
SABot:  I am sorry! I don't understand you.
SABot:  I am sorry! I don't understand you.
SABot: I am fine! How about you?
SABot:  I am sorry! I don't understand you.
SABot: Bye! take care...
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```