Nama : Virgie Yunita Salsabil

NIM : 21110022

Kelas : S1-SD02A

Import Libraries

```
# Library untuk operasi input/output
import io
# Library untuk menghasilkan bilangan acak atau pemilihan elemen acak
import random
# Library untuk manipulasi string
import string
# Library untuk memberikan peringatan
import warnings
# Library untuk operasi array numerik dan manipulasi data
import numpy as np
# Library untuk pemrosesan teks menggunakan TF-IDF
from sklearn.feature_extraction.text import TfidfVectorizer
# Library untuk mengukur kemiripan kosinus antara dua set dokumen
from sklearn.metrics.pairwise import cosine_similarity
# Menyaring peringatan agar tidak ditampilkan
warnings.filterwarnings('ignore')
```

```
# Instalasi library Natural Language Toolkit (nltk) menggunakan pip
!pip install nltk
```

```
Requirement already satisfied: nltk in /usr/local/lib/python3.10/dist-packages (3.8.1)
Requirement already satisfied: click in /usr/local/lib/python3.10/dist-packages (from nltk) (8.1.7)
Requirement already satisfied: joblib in /usr/local/lib/python3.10/dist-packages (from nltk) (1.3.2)
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.10/dist-packages (from nltk) (2023.6.3)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from nltk) (4.66.1)
```

```
# Impor library NLTK
import nltk
# Import library untuk lemmatization
from nltk.stem import WordNetLemmatizer
# Pengunduhan data popular dari NLTK
nltk.download('popular', quiet=True)
#nltk.download('punkt') # first-time use only
#nltk.download('wordnet') # first-time use only
```

```
True
```

Reading in the corpus

Program ini menggunakan halaman Wikipedia sebagai corpus dari chatpotdengan cara meng-Copy konten halaman dan menyimpannya dalam file ‘chatbot.txt’

```
# Membuka file 'chatbot.txt' untuk dibaca ('r')
# Parameter 'errors='ignore'' digunakan untuk mengabaikan karakter yang tidak dapat di-decode
f = open('chatbot.txt', 'r', errors='ignore')

# Membaca seluruh isi file dan menyimpannya dalam variabel 'raw'
# raw kini berisi semua data dari corpus per baris (raw)
raw = f.read()

# Mengonversi semua teks dalam 'raw' menjadi huruf kecil (lowercase)
raw = raw.lower()
```

Tokenisasi

```
# Tokenisasi adalah memilah-milah dokumen ke kalimat-kalimat,
# kemudian memilah setiap kalimat menjadi sekumpulan kata kata
sent_tokens = nltk.sent_tokenize(raw) # converts dokumen corpus ke kalimat-kalimat
word_tokens = nltk.word_tokenize(raw) # converts dokumen corpus ke kata-kata
```

Preprocessing

```
# Membuat objek lemmatizer menggunakan WordNetLemmatizer dari NLTK
lemmer = nltk.stem.WordNetLemmatizer()

# WordNet adalah kamus berorientasi semantik Bahasa Inggris yang disertakan dalam NLTK.
# Fungsi LemTokens(tokens) untuk melakukan lemmatization pada setiap token dalam list 'tokens'
def LemTokens(tokens):
    return [lemmer.lemmatize(token) for token in tokens]

# Membuat kamus yang akan digunakan untuk menghapus tanda baca dari teks
remove_punct_dict = dict((ord(punct), None) for punct in string.punctuation)

# Fungsi LemNormalize(text) untuk melakukan normalisasi teks dengan tokenisasi, lemmatization, dan konversi huruf kecil
def LemNormalize(text):
    return LemTokens(nltk.word_tokenize(text.lower().translate(remove_punct_dict)))
```

Keyword matching

```
# Daftar kata-kata pembuka yang mungkin diterima sebagai salam
GREETING_INPUTS = ("hello", "hi", "greetings", "sup", "what's up", "hey", "hai")

# Respon-respon yang mungkin diberikan sebagai tanggapan terhadap salam
GREETING_RESPONSES = ["hi", "hey", "*nods*", "hi there", "hello", "I am glad! You are talking to me"]

# Fungsi greeting(sentence) untuk menentukan respon terhadap kalimat yang mengandung kata-kata pembuka
def greeting(sentence):
    # Melakukan iterasi pada setiap kata dalam kalimat
    for word in sentence.split():
        # Memeriksa apakah kata dalam bentuk huruf kecil terdapat dalam daftar kata-kata pembuka
        if word.lower() in GREETING_INPUTS:
            # Jika ditemukan kata pembuka, maka pilih secara acak salah satu respon dari daftar respon
            return random.choice(GREETING_RESPONSES)
```

Generating Response

After the initial preprocessing phase, we need to transform text into a meaningful vector (or array) of numbers. The bag-of-words is a representation of text that describes the occurrence of words within a document. It involves two things:

- A vocabulary of known words.
- A measure of the presence of known words.

Why is it is called a “bag” of words? That is because any information about the order or structure of words in the document is discarded and the model is only concerned with whether the known words occur in the document, not where they occur in the document. The intuition behind the Bag of Words is that documents are similar if they have similar content. Also, we can learn something about the meaning of the document from its content alone. For example, if our dictionary contains the words {Learning, is, the, not, great}, and we want to vectorize the text “Learning is great”, we would have the following vector: (1, 1, 0, 0, 1).

TF-IDF Approach

A problem with the Bag of Words approach is that highly frequent words start to dominate in the document (e.g. larger score), but may not contain as much “informational content”. Also, it will give more weight to longer documents than shorter documents. One approach is to rescale the frequency of words by how often they appear in all documents so that the scores for frequent words like “the” that are also frequent across all documents are penalized. This approach to scoring is called Term Frequency-Inverse Document Frequency, or TF-IDF for short, where:

Term Frequency: is a scoring of the frequency of the word in the current document.

- TF = (Number of times term t appears in a document)/(Number of terms in the document)

Inverse Document Frequency: is a scoring of how rare the word is across documents.

- IDF = 1+log(N/n), where, N is the number of documents and n is the number of documents a term t has appeared in.

Cosine Similarity

Tf-idf weight is a weight often used in information retrieval and text mining. This weight is a statistical measure used to evaluate how important a word is to a document in a collection or corpus

- Cosine Similarity (d1, d2) = Dot product(d1, d2) / ||d1|| * ||d2||

where d1,d2 are two non zero vectors. To generate a response from our bot for input questions, the concept of document similarity will be used. We define a function response which searches the user's utterance for one or more known keywords and returns one of several possible responses. If it doesn't find the input matching any of the keywords, it returns a response:" I am sorry! I don't understand you"

```
# Fungsi untuk menetapkan respon jawaban
def response(user_response):
    robo_response = ''  # Pada tahap awal, respon mesin diisi karakter kosong
    sent_tokens.append(user_response)  # Pertanyaan user ditokenisasi dan ditambahkan ke corpus

    # Posisi paling bawah (yaitu posisi -1)
    TfidfVec = TfidfVectorizer(tokenizer=LemNormalize, stop_words='english')
    tfidf = TfidfVec.fit_transform(sent_tokens)  # Token dari pertanyaan user di-vektorisasi
    vals = cosine_similarity(tfidf[-1], tfidf)  # Hitung similarity setiap token corpus dengan token pertanyaan

    # Sortir jarak similarity setiap token corpus dengan token pertanyaan
    # Mengambil indeks token yang memiliki similarity tertinggi (indeks paling rendah) dengan pertanyaan pengguna
    idx = vals.argsort()[0][-2]

    # Meratakan dan mengurutkan nilai similarity untuk mendapatkan similarity tertinggi kedua
    flat = vals.flatten()
    flat.sort()

    # Menyimpan nilai similarity tertinggi kedua
    req_tfidf = flat[-2]

    # Jika pertanyaan dan semua token corpus jaraknya tinggi, maka pertanyaan tidak memiliki jawaban
    if req_tfidf == 0:
        robo_response = robo_response + "Please reply, I don't understand your questions"
        return robo_response
    else:
        # Jika jaraknya terendah, maka token tersebut dipakai sebagai jawaban
        robo_response = robo_response + sent_tokens[idx]
        return robo_response
```

Finally, we will feed the lines that we want our bot to say while starting and ending a conversation depending upon user's input.

```
# Inisialisasi variabel 'flag' untuk menjalankan loop
flag=True
print("Mesin: My name is Robo. I will answer your queries about Chatbots. If you want to exit, type Bye!")

# Memulai loop utama, akan terus berjalan selama 'flag' bernilai True
while(flag==True):
    # Meminta user untuk memasukkan pertanyaan
    user_response = input("Masukkan pertanyaan :")
    # Mengubah input pengguna menjadi huruf kecil untuk konsistensi
    user_response=user_response.lower()

    # Memeriksa jika user tidak keluar
    if(user_response!='bye'):
        # Jika pengguna mengucapkan thanks/thankyou, menghentikan loop & memberikan balasan
        if(user_response=='thanks' or user_response=='thank you' ):
            flag=False #tandai proses berhenti
            print("Mesin: You are welcome..") #balasan thank you
        else:
            if(greeting(user_response)!=None): #jika response adalah kalimat greeting
                print("Mesin: "+greeting(user_response)) #tampilkan kalimat greeting
            else: #jika bukan kalimat greeting
                print("Mesin: ",end="")
                print(response(user_response)) #memproses user answer disini
                sent_tokens.remove(user_response) #user answer dihapus setelah dimunculkan
                print("----------------------------------------")

    # Jika pengguna ingin keluar, mengubah nilai flag menjadi False untuk menghentikan loop
    else:
        flag=False
        # Menampilkan pesan perpisahan dan pemisah untuk setiap iterasi
        print("Mesin: Bye! take care..")
        print("========================")
```

```
Mesin: My name is Robo. I will answer your queries about Chatbots. If you want to exit, type Bye!
Masukkan pertanyaan :hi
Mesin: hi there
Masukkan pertanyaan :What are the categories of chatbot usage?
Mesin: analytics
```

```
the usage of the chatbot can be monitored in order to spot potential flaws or problems.
----------------------------------------
Masukkan pertanyaan :Who coined the term "ChatterBot" and when?
Mesin: the term "chatterbot" was originally coined by michael mauldin (creator of the first verbot, julia) in 1994 to describe these
----------------------------------------
Masukkan pertanyaan :What is the difference between older chatbots like ELIZA and PARRY compared to newer generations?
Mesin: while eliza and parry were used exclusively to simulate typed conversation, many chatbots now include functional features suc
----------------------------------------
Masukkan pertanyaan :How is the chatbot development process structured, and what are its key phases?
Mesin: chatbot creation
the process of creating a chatbot follows a pattern similar to the development of a web page or a mobile app.
----------------------------------------
Masukkan pertanyaan :What is the impact of artificial intelligence (AI) usage on jobs, according to Forrester?
Mesin: according to forrester (2015), ai will replace 16 percent of american jobs by the end of the decade.chatbots have been used i
----------------------------------------
Masukkan pertanyaan :thank you
Mesin: You are welcome..
```

KESIMPULAN

- Berdasarkan output di atas, fungsi ***cosine similarity*** dalam program berjalan dengan baik. Ketika pengguna memasukkan pertanyaan, program menghitung kemiripan cosinus antara vektor TF-IDF dari pertanyaan tersebut dengan vektor TF-IDF dari seluruh korpus (pertanyaan sebelumnya). Hasil similarity digunakan untuk menentukan jawaban terbaik dengan memilih pertanyaan sebelumnya yang paling mirip.
- Contohnya, ketika pengguna memasukkan pertanyaan "What are the categories of chatbot usage?" program memberikan jawaban yang relevan dengan mencocokkan kata kunci dan struktur pertanyaan dengan pertanyaan yang ada dalam korpus. Ini menunjukkan bahwa cosine similarity berfungsi dengan baik dalam memahami konteks pertanyaan dan memberikan respons yang sesuai.