ch5-IntroductionToSentimentAnalysis

หน้า 1: บทนำเกี่ยวกับการวิเคราะห์ความรู้สึก

- **การวิเคราะห์ความรู้สึก (Sentiment Analysis)** คือการขุดค้นความคิดเห็น (Opinion Mining) หรือการพิจารณาความคิดเห็นจากข้อความของผู้เขียน โดยเน้นที่การเข้าใจอารมณ์ ความรู้สึก และความคิดเห็นของผู้เขียนที่มีต่อเรื่องใดเรื่องหนึ่ง เช่น ความคิดเห็นเชิงบวกหรือเชิงลบเกี่ยวกับภาพยนตร์ ผลิตภัณฑ์ หรือบริการ

หน้า 2: องค์ประกอบในระบบการวิเคราะห์ความรู้สึก

- ระบบการวิเคราะห์ความรู้สึกประกอบด้วย:
    1. **ความคิดเห็นหรืออารมณ์ (Opinion/Emotion):** แสดงออกมาเป็นความรู้สึกเชิงบวก (positive), เชิงกลาง (neutral), หรือเชิงลบ (negative) และสามารถเป็นการแสดงอารมณ์เชิงคุณภาพ (joy, anger) หรือเชิงปริมาณ (การให้คะแนนจาก 1 ถึง 10)
    2. **หัวเรื่อง (Subject):** สิ่งที่ผู้เขียนกำลังพูดถึง เช่น ผลิตภัณฑ์ หนังสือ หรือภาพยนตร์
    3. **ผู้ที่แสดงความคิดเห็น (Opinion holder):** บุคคลหรือองค์กรที่ให้ความคิดเห็นนั้น

หน้า 3: ทำไมต้องวิเคราะห์ความรู้สึก

- การวิเคราะห์ความรู้สึกมีประโยชน์หลายด้าน เช่น:
    - **การติดตามบนโซเชียลมีเดีย (Social Media Monitoring):** ช่วยให้ทราบว่าผู้คนพูดถึงแบรนด์อย่างไร ไม่ใช่แค่จำนวนแต่รวมถึงลักษณะการพูดถึงด้วย
    - **การติดตามแบรนด์ (Brand Monitoring):** เพื่อเข้าใจปฏิสัมพันธ์ของลูกค้ากับแบรนด์ สิ่งที่พวกเขาพึงพอใจหรือไม่พึงพอใจ

หน้า 4: การวิเคราะห์รีวิวภาพยนตร์

- การใช้ชุดข้อมูลตัวอย่างจากเว็บไซต์ IMDB โดยแบ่งข้อมูลเป็นสองส่วนคือ **ข้อความรีวิว (text)** และ **ฉลาก (label)** ที่บ่งบอกความรู้สึกของรีวิวว่าเป็นบวก (1) หรือ ลบ (0)

หน้า 5: จำนวนรีวิวที่เป็นบวกและลบ

- สามารถใช้คำสั่ง .value_counts() บนคอลัมน์ label เพื่อตรวจสอบจำนวนรีวิวที่เป็นบวกและลบ เช่น:
    - จำนวนรีวิวที่เป็นบวก (1)
    - จำนวนรีวิวที่เป็นลบ (0)

หน้า 6: เปอร์เซ็นต์ของรีวิวที่เป็นบวกและลบ

- การหาสัดส่วนของรีวิวที่เป็นบวกและลบสามารถทำได้โดยการใช้สูตร:

python

Copy code

data.label.value.count() / len(data)

หน้า 7: ความยาวของรีวิวที่ยาวที่สุด

- ใช้ฟังก์ชัน .str.len() เพื่อหาความยาวของรีวิวที่ยาวที่สุด และใช้ฟังก์ชัน max() เพื่อค้นหาความยาวที่มากที่สุด

หน้า 8: การปฏิบัติการ

- ในหน้านี้จะเริ่มฝึกปฏิบัติในการสำรวจข้อมูลเบื้องต้น โดยการโหลดชุดข้อมูลรีวิวภาพยนตร์

หน้า 9: สำรวจข้อมูลจำนวนรีวิว

- ขั้นตอนในการสำรวจจำนวนรีวิวทั้งบวกและลบ และหาเปอร์เซ็นต์ของแต่ละประเภท

หน้า 10: การหาความยาวของรีวิว

- ใช้คอลัมน์ **text** เพื่อหาความยาวของรีวิวทั้งยาวที่สุดและสั้นที่สุด

หน้า 11: ประเภทของการวิเคราะห์ความรู้สึกและแนวทางการประยุกต์ใช้

- มีการวิเคราะห์ในระดับต่างๆ:
    1. **ระดับเอกสาร** (**Document Level)**: ดูที่ข้อความทั้งหมด
    2. **ระดับประโยค** (**Sentence Level)**: ดูว่าประโยคใดเป็นบวก ลบ หรือกลาง
    3. **ระดับคุณลักษณะ** (**Aspect Level)**: พิจารณาคุณลักษณะต่างๆ ของผลิตภัณฑ์ เช่น การแสดงความคิดเห็นเกี่ยวกับกล้องที่ดี แต่แบตเตอรี่ที่ไม่ดีของโทรศัพท์เครื่องหนึ่ง

หน้า 12: ประเภทของอัลกอริธึมการวิเคราะห์ความรู้สึก

- **อิงตามกฎ** (**Rule-based)**: ใช้พจนานุกรมของคำที่มีคะแนนความรู้สึกเชิงบวกหรือเชิงลบ เช่น "nice" ได้คะแนน +2, "terrible" ได้ -3
- **อัตโนมัติ** (**Automated)**: ใช้การเรียนรู้ของเครื่อง (Machine Learning) โดยใช้ข้อมูลประวัติที่มีการติดฉลากความคิดเห็นเพื่อทำนายความคิดเห็นใหม่

หน้า 13: การคำนวณค่า Valence ของข้อความ

- การใช้ไลบรารี **TextBlob** ใน Python สำหรับการคำนวณคะแนนความคิดเห็น (polarity) ซึ่งวัดตั้งแต่ -1 ถึง 1 โดย -1 หมายถึงลบมาก และ 1 หมายถึงบวกมาก

หน้า 14: การปฏิบัติการ I

- การฝึกฝนการตรวจจับความรู้สึกของข้อความด้วยการใช้ TextBlob ในการสร้างข้อความและวิเคราะห์ความคิดเห็นจากข้อความนั้น

หน้า 15: การตรวจจับความรู้สึกของรีวิวภาพยนตร์

- การนำข้อความรีวิวภาพยนตร์ (ตัวอย่างจาก Titanic) มาใช้และคำนวณความคิดเห็นจากเนื้อหาด้วยการสร้างออบเจ็กต์ TextBlob

หน้า 16: โมดูลที่จำเป็นสำหรับการวิเคราะห์ความรู้สึก

- การใช้โมดูลสำคัญในการวิเคราะห์ข้อความ เช่น:
    - **pandas** สำหรับการจัดการข้อมูล
    - **matplotlib** สำหรับการสร้างกราฟ
    - **nltk** สำหรับการประมวลผลภาษาธรรมชาติ (NLP)

หน้า 17: การนำเข้าข้อมูลและการทำความสะอาดข้อความ

- การโหลดข้อมูลตัวอย่างและการทำความสะอาดข้อความ โดยลบตัวอักษรที่ไม่ใช่ตัวอักษรภาษาอังกฤษออก และแปลงข้อความทั้งหมดเป็นตัวพิมพ์เล็ก

หน้า 18: การสร้างคะแนนความรู้สึก (Sentiment Polarity Scores)

- การสร้างคะแนนความรู้สึกโดยใช้ VADER lexicon ซึ่งเหมาะสมกับการวิเคราะห์ความรู้สึกจากโพสต์บนโซเชียลมีเดีย รีวิวสินค้า และการตอบกลับแบบสำรวจ

หน้า 19: การวนลูปเพื่อคำนวณคะแนนความรู้สึก

- การใช้ลูปใน Python เพื่อวิเคราะห์ความรู้สึกของข้อความแต่ละแถวในข้อมูลตัวอย่าง และจัดเก็บผลลัพธ์ในตาราง

หน้า 20: การรวมข้อมูลผลลัพธ์เข้ากับตารางข้อมูลต้นฉบับ

- การรวมตารางข้อมูลผลลัพธ์จากการวิเคราะห์เข้ากับตารางข้อมูลต้นฉบับเพื่อแสดงผลลัพธ์ขั้นสุดท้าย

sentiment_1

Pandas

- เป็นไลบรารีในภาษา Python ที่ใช้สำหรับการวิเคราะห์และจัดการข้อมูลที่มีโครงสร้าง (structured data) โดยเฉพาะข้อมูลในรูปแบบตาราง (table)
  - เช่น DataFrame ซึ่งสามารถใช้จัดการข้อมูลที่มาจากแหล่งต่าง ๆ เช่น CSV, Excel, SQL Database และอื่น ๆ ได้อย่างมีประสิทธิภาพ
- Pandas ยังมีฟังก์ชันที่ช่วยในการจัดการและทำความสะอาดข้อมูล
  - เช่น การจัดการค่าที่หายไป การจัดการค่าที่ซ้ำกัน
- Pandas สามารถรวมข้อมูลจากหลายแหล่ง และจัดกลุ่มข้อมูลเพื่อการวิเคราะห์ที่ง่ายขึ้น
- นอกจากนี้ Pandas ก็มีฟังก์ชันที่ช่วยในการคำนวณทางคณิตศาสตร์
  - เช่น การคำนวณค่าเฉลี่ย ค่าเบี่ยงเบนมาตรฐาน การหาผลรวม และอื่น ๆ เป็นต้น

[2]
```
# นำเข้าไลบรารี Pandas สำหรับการจัดการข้อมูลแบบตาราง
import pandas as pd
```

[3]
```
import io
from google.colab import files

# อัปโหลดไฟล์จากเครื่องผู้ใช้
uploaded = files.upload()
# อ่านไฟล์ CSV ที่อัปโหลดเข้ามาและแปลงเป็น DataFrame ของ Pandas
movies = pd.read_csv(io.BytesIO(uploaded['Train_small.csv']))
```

```
เลือกไฟล์ Train_small.csv
• Train_small.csv(text/csv) - 1047477 bytes, last modified: 15/10/2567 - 100% done
Saving Train_small.csv to Train_small.csv
```

- 0: the number of negative reviews
- 1: the number of positive reviews

[4]
```
# นับจำนวนรีวิวที่เป็นบวกและลบจากคอลัมน์ 'label'
print('Number of positive and negative reviews: ', movies.label.value_counts())
```

```
Number of positive and negative reviews:  label
0    104
1     97
Name: count, dtype: int64
```

[5]
```
# คำนวณสัดส่วนของรีวิวที่เป็นบวกและลบ โดยหารด้วยจำนวนรีวิวทั้งหมด
print('Proportion of positive and negative reviews: ', movies.label.value_counts() / len(movies))
```

```
Proportion of positive and negative reviews:  label
0    0.517413
1    0.482587
Name: count, dtype: float64
```

[6] #รีวิวที่ยาวที่สุดและสั้นที่สุดมีความยาวเท่าไร

```
# คำนวณความยาวของแต่ละรีวิวจากคอลัมน์ 'text'
length_reviews = movies.text.str.len()
# หาความยาวรีวิวที่ยาวที่สุด
print("the longest revies:", max(length_reviews))
# หาความยาวรีวิวที่สั้นที่สุด
print("the shortest revies:", min(length_reviews))
```

```
the longest revies: 28433.0
the shortest revies: 1.0
```

[7] # แสดงความยาวของรีวิวแต่ละรายการ

```
print(length_reviews)
```

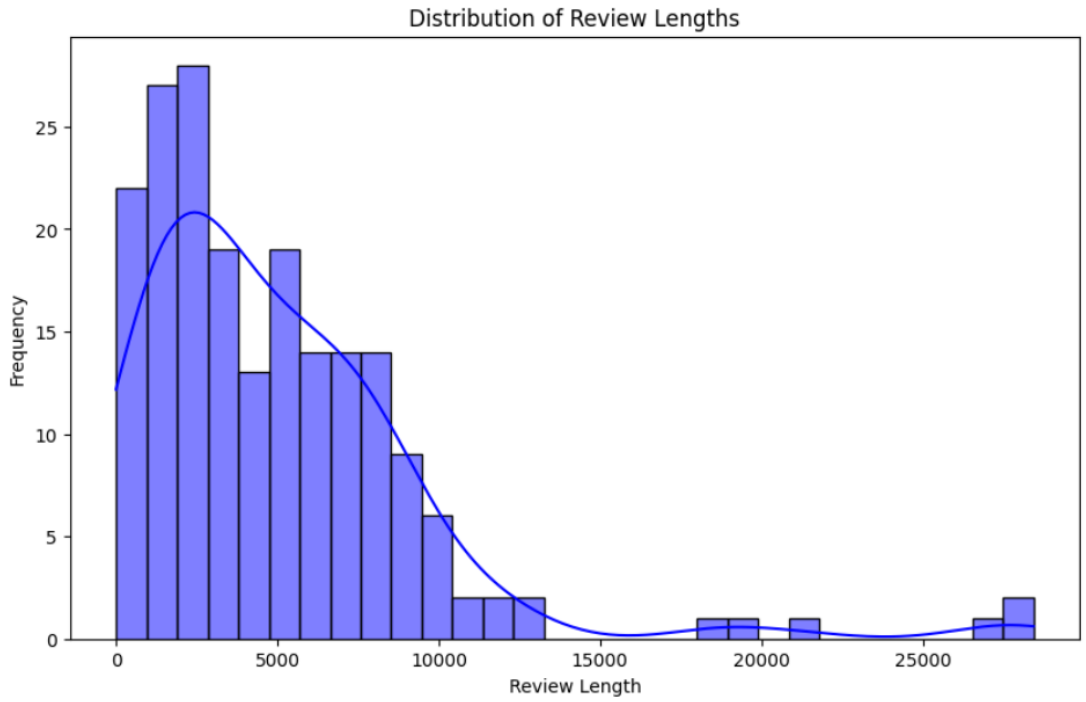
```
0      4996.0
1      4250.0
2      7786.0
3      3261.0
4       958.0
       ...
196      38.0
197   28433.0
198   10198.0
199   11779.0
200    7293.0
Name: text, Length: 201, dtype: float64
```

seaborn is a Python data visualization library built on top of **matplotlib**. It provides a *high-level interface* for creating attractive and informative statistical graphics. **seaborn** คือไลบรารีแสดงภาพข้อมูล Python ที่สร้างขึ้นบน matplotlib โดยไลบรารีนี้ให้อินเทอร์เฟซระดับสูงสำหรับการสร้างกราฟิกสถิติที่น่าสนใจและให้ข้อมูล

histplot

- คือฟังก์ชันในไลบรารี seaborn ที่ใช้ในการสร้าง histogram ซึ่งเป็นกราฟที่แสดงการกระจายตัวของข้อมูล
- ข้อมูลถูกแบ่งออกเป็นช่วง ๆ และแสดงจำนวนข้อมูลที่ตกอยู่ในแต่ละช่วง (frequency)
- การสร้าง histogram ช่วยให้เข้าใจลักษณะการกระจายตัว เช่น
  - ข้อมูลมีการกระจายอย่างสม่ำเสมอหรือไม่
  - มีความโน้มเอียงไปทางใด

```python
# นำเข้าไลบรารี Matplotlib และ Seaborn สำหรับการสร้างกราฟ
import matplotlib.pyplot as plt
import seaborn as sns

# สร้างกราฟ histogram เพื่อแสดงการกระจายตัวของความยาวรีวิว
plt.figure(figsize=(10, 6))  # กำหนดขนาดกราฟ
sns.histplot(length_reviews, bins=30, kde=True, color='blue')  # สร้าง histogram พร้อม KDE
plt.title('Distribution of Review Lengths')  # ตั้งชื่อกราฟ
plt.xlabel('Review Length')  # ตั้งชื่อแกน x
plt.ylabel('Frequency')  # ตั้งชื่อแกน y
plt.show()  # แสดงกราฟ
```



- **โหลดไฟล์ CSV:** ผู้ใช้จะอัปโหลดไฟล์ CSV ที่มีข้อมูลรีวิวเข้ามา จากนั้นไฟล์จะถูกแปลงเป็น DataFrame ด้วย Pandas เพื่อให้สามารถจัดการข้อมูลได้ง่าย
- **นับจำนวนรีวิวบวกและลบ:** โค้ดจะนับจำนวนรีวิวที่เป็นบวก (1) และลบ (0) ในข้อมูล พร้อมแสดงผลจำนวนและสัดส่วนของรีวิวบวกและลบ
- **วิเคราะห์ความยาวรีวิว:** โค้ดคำนวณความยาวของแต่ละรีวิวในหน่วยตัวอักษร และระบุรีวิวที่มีความยาวมากที่สุดและน้อยที่สุด
- **สร้างกราฟ histogram:** โค้ดสร้างกราฟแสดงการกระจายตัวของความยาวรีวิวทั้งหมด โดยใช้ histogram และ Kernel Density Estimate (KDE) เพื่อแสดงรูปแบบการกระจายของข้อมูลรีวิว

โค้ดนี้ช่วยให้สามารถวิเคราะห์ข้อมูลรีวิวในเชิงสถิติ เช่น จำนวนรีวิวบวก/ลบ ความยาวของรีวิว และแสดงข้อมูลในรูปแบบกราฟที่เข้าใจง่าย

sentiment_2

Detecting the sentiment

- Polarity (คะแนนความรู้สึก) วัดในช่วง [-1.0 ถึง 1.0] โดยที่
  - -1.0 หมายถึงความรู้สึกเชิงลบมาก
  - 0 หมายถึงเป็นกลาง, และ
  - +1.0 หมายถึงความรู้สึกเชิงบวกมาก
- Subjectivity (การให้ความเห็นในมุมมองของตัวเองเป็นหลัก) วัดในช่วง [0.0 ถึง 1.0] โดยที่
  - 0.0 หมายถึงมีความเป็นกลางมาก และ
  - 1.0 หมายถึง มีการแสดงความรู้สึกหรือความคิดเห็นส่วนตัว

```
[18]  # นำเข้าไลบรารี TextBlob สำหรับการวิเคราะห์ข้อความ
      from textblob import TextBlob
```

```
[19]  # ข้อความตัวอย่าง
      text = "You are so beautiful"
      # สร้างวัตถุ TextBlob จากข้อความ
      blob_two_cities = TextBlob(text)
```

```
[20]  # แสดงผลความรู้สึก (polarity และ subjectivity)
      print(blob_two_cities.sentiment)
```

```
Sentiment(polarity=0.85, subjectivity=1.0)
```

↑ TextBlob ถูกใช้เพื่อวิเคราะห์ข้อความ text โดยตรวจสอบคะแนน polarity (ความรู้สึก) และ subjectivity (ความคิดเห็นส่วนตัว)

What is the sentiment of a movie review? การวิจารณ์หนังมีอารมณ์อย่างไร?

```
[22]  import io
      from google.colab import files

      # อัปโหลดไฟล์
      uploaded = files.upload()

      # อ่านเนื้อหาของไฟล์ที่อัปโหลด
      with io.StringIO(uploaded['titanic.txt'].decode('utf-8')) as f:
          titanic = f.read()

      # แสดงเนื้อหาของไฟล์
      print(titanic)
```

```
เลือกไฟล์  titanic.txt
• titanic.txt(text/plain) - 10323 bytes, last modified: 15/10/2567 - 100% done
Saving titanic.txt to titanic.txt
Titanic directed by James Cameron presents a fictional love story on the historical setting of the Titanic. The plot is simple, noncomplicated, or not for those who love plots that twist and turn and keep you in suspense. The end o
```

```
[25] # สร้างวัตถุ TextBlob สำหรับไฟล์ Titanic
     blob_titanic = TextBlob(titanic)
```

```
[26] # แสดงผลความรู้สึกของข้อความ
     print(blob_titanic.sentiment)
```

```
Sentiment(polarity=0.2024748060772906, subjectivity=0.4518248900857597)
```

↑ ในส่วนนี้ไฟล์ titanic.txt ถูกอัปโหลดและอ่านข้อมูลด้วย TextBlob เพื่อวิเคราะห์ความรู้สึก

```
[27] # นำเข้าไลบรารี nltk และดาวน์โหลดโมดูลสำหรับแยกประโยค (sentence tokenization)
     import nltk
     nltk.download('punkt')
```

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

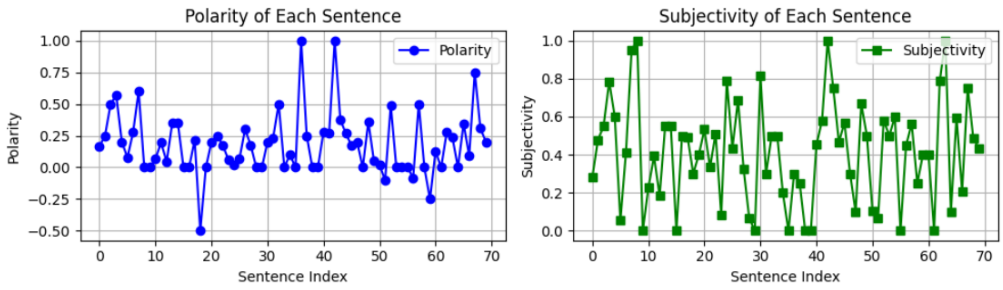
```
# ตัวแปรสำหรับเก็บผลรวมของคะแนน polarity และ subjectivity
polarity_sum = 0
subjectivity_sum = 0
sentence_count = 0  # ตัวนับจำนวนประโยค

# วนลูปประโยคแต่ละประโยคในไฟล์ที่อ่าน
for sentence in blob_titanic.sentences:
    sentiment = sentence.sentiment  # วิเคราะห์ความรู้สึกของแต่ละประโยค
    polarity_sum += sentiment.polarity  # บวกคะแนน polarity ลงในผลรวม
    subjectivity_sum += sentiment.subjectivity  # บวกคะแนน subjectivity ลงในผลรวม
    sentence_count += 1  # เพิ่มจำนวนประโยคที่นับ

# คำนวณค่าเฉลี่ยของ polarity และ subjectivity ถ้ามีประโยคอยู่
average_polarity = polarity_sum / sentence_count if sentence_count > 0 else 0
average_subjectivity = subjectivity_sum / sentence_count if sentence_count > 0 else 0

# แสดงค่าเฉลี่ยของ polarity และ subjectivity
print(f"Average Polarity: {average_polarity}")
print(f"Average Subjectivity: {average_subjectivity}")
```

```
Average Polarity: 0.1825313324241896
Average Subjectivity: 0.4141866724545295
```

- nltk ถูกใช้สำหรับแยกประโยค (sentence tokenization)
- for loop จะวิเคราะห์ความรู้สึกของแต่ละประโยคในไฟล์ และคำนวณค่าเฉลี่ยของ polarity และ subjectivity

```python
# นำเข้าไลบรารี matplotlib สำหรับการสร้างกราฟ
import matplotlib.pyplot as plt

# สร้างลิสต์เพื่อเก็บค่า polarity และ subjectivity ของแต่ละประโยค
polarity_values = []
subjectivity_values = []

# วนลูปวิเคราะห์ความรู้สึกของแต่ละประโยค
for sentence in blob_titanic.sentences:
    sentiment = sentence.sentiment  # วิเคราะห์ความรู้สึกของแต่ละประโยค
    polarity_values.append(sentiment.polarity)  # เก็บค่า polarity
    subjectivity_values.append(sentiment.subjectivity)  # เก็บค่า subjectivity

# เริ่มการสร้างกราฟ
plt.figure(figsize=(10, 3))  # กำหนดขนาดของกราฟ (กว้าง 10 หน่วย สูง 3 หน่วย)

# สร้างกราฟสำหรับค่า polarity
plt.subplot(1, 2, 1)  # แบ่งกราฟเป็น 1 แถว 2 คอลัมน์ และเลือกกราฟแรก
plt.plot(polarity_values, marker='o', color='b', label='Polarity')  # วาดกราฟเส้นสำหรับ polarity
plt.title('Polarity of Each Sentence')  # ตั้งชื่อกราฟ
plt.xlabel('Sentence Index')  # ตั้งชื่อแกน x
plt.ylabel('Polarity')  # ตั้งชื่อแกน y
plt.grid(True)  # แสดงเส้นตารางในกราฟ
plt.legend()  # แสดงคำอธิบายกราฟ
```

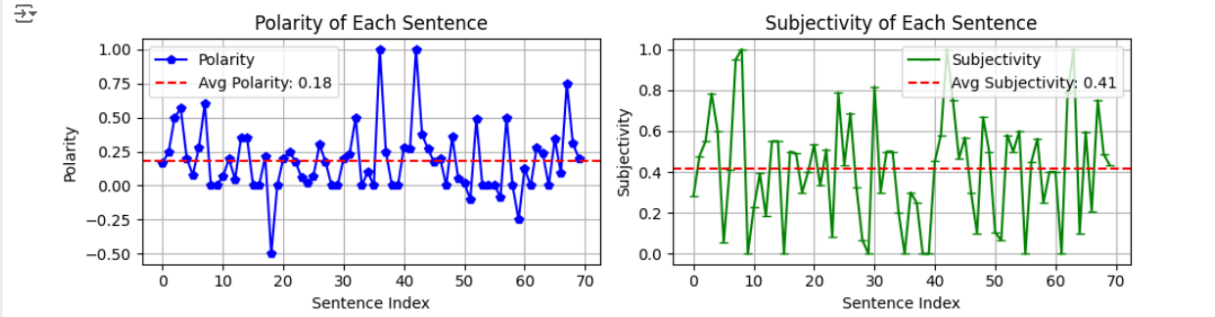
```python
# สร้างกราฟสำหรับค่า subjectivity
plt.subplot(1, 2, 2)  # เลือกกราฟที่ 2
plt.plot(subjectivity_values, marker='s', color='g', label='Subjectivity')  # วาดกราฟเส้นสำหรับ subjectivity
plt.title('Subjectivity of Each Sentence')  # ตั้งชื่อกราฟ
plt.xlabel('Sentence Index')  # ตั้งชื่อแกน x
plt.ylabel('Subjectivity')  # ตั้งชื่อแกน y
plt.grid(True)  # แสดงเส้นตารางในกราฟ
plt.legend()  # แสดงคำอธิบายกราฟ

# แสดงกราฟทั้งหมด
plt.tight_layout()  # จัดวางกราฟให้อยู่ในพื้นที่ที่กำหนด
plt.show()  # แสดงกราฟบนหน้าจอ
```



การสร้างกราฟแสดงผลความรู้สึกแต่ละประโยค แสดงกราฟการกระจายตัวของ polarity และ subjectivity ของแต่ละประโยคในไฟล์ titanic.txt

```python
# นำเข้าไลบรารี matplotlib อีกครั้ง
import matplotlib.pyplot as plt

# คำนวณค่าเฉลี่ยของ polarity และ subjectivity
average_polarity = sum(polarity_values) / len(polarity_values) if polarity_values else 0
average_subjectivity = sum(subjectivity_values) / len(subjectivity_values) if subjectivity_values else 0

# เริ่มการสร้างกราฟใหม่
plt.figure(figsize=(10, 3))  # กำหนดขนาดของกราฟ

# สร้างกราฟสำหรับ polarity พร้อมแสดงเส้นค่าเฉลี่ย
plt.subplot(1, 2, 1)  # แบ่งกราฟเป็น 1 แถว 2 คอลัมน์ และเลือกกราฟแรก
plt.plot(polarity_values, marker='p', color='b', label='Polarity')  # วาดกราฟเส้นสำหรับ polarity
plt.axhline(y=average_polarity, color='r', linestyle='--', label=f'Avg Polarity: {average_polarity:.2f}')  # วาดเส้นค่าเฉลี่ย polarity
plt.title('Polarity of Each Sentence')  # ตั้งชื่อกราฟ
plt.xlabel('Sentence Index')  # ตั้งชื่อแกน x
plt.ylabel('Polarity')  # ตั้งชื่อแกน y
plt.grid(True)  # แสดงเส้นตารางในกราฟ
plt.legend()  # แสดงคำอธิบายกราฟ

# สร้างกราฟสำหรับ subjectivity พร้อมแสดงเส้นค่าเฉลี่ย
plt.subplot(1, 2, 2)  # เลือกกราฟที่สอง
plt.plot(subjectivity_values, marker='_', color='g', label='Subjectivity')  # วาดกราฟเส้นสำหรับ subjectivity
plt.axhline(y=average_subjectivity, color='r', linestyle='--', label=f'Avg Subjectivity: {average_subjectivity:.2f}')  # วาดเส้นค่าเฉลี่ย subjectivity
plt.title('Subjectivity of Each Sentence')  # ตั้งชื่อกราฟ
plt.xlabel('Sentence Index')  # ตั้งชื่อแกน x
plt.ylabel('Subjectivity')  # ตั้งชื่อแกน y
plt.grid(True)  # แสดงเส้นตารางในกราฟ
plt.legend()  # แสดงคำอธิบายกราฟ
```

```python
# แสดงกราฟทั้งหมด
plt.tight_layout()  # จัดวางกราฟให้อยู่ในพื้นที่ที่กำหนด
plt.show()  # แสดงกราฟบนหน้าจอ
```



การคำนวณค่าเฉลี่ยและแสดงผลในกราฟ

- marker: สัญลักษณ์สำหรับจุดในกราฟ เช่น 'o' (วงกลม), 'p' (ห้าเหลี่ยม), 's' (สี่เหลี่ยม)
- linestyle: รูปแบบของเส้นในกราฟ เช่น '-' (เส้นตรง), '--' (เส้นประ)

สรุป: โค้ดนี้เป็นการวิเคราะห์ความรู้สึกของข้อความจากไฟล์ titanic.txt และแสดงผลความรู้สึกในเชิง polarity (ความรู้สึกบวกหรือลบ) และ subjectivity (ความคิดเห็นส่วนตัว) ของแต่ละประโยคในรูปแบบกราฟ

marker

- 'o': วงกลม
- 's': สี่เหลี่ยม
- '^': รูปสามเหลี่ยมชี้ขึ้น
- 'v': รูปสามเหลี่ยมชี้ลง
- '>': รูปสามเหลี่ยมชี้ขวา
- '<': รูปสามเหลี่ยมชี้ซ้าย
- 'D': รูปสี่เหลี่ยมขนมเปียกปูน
- 'p': รูปห้าเหลี่ยม
- 'h': รูปหกเหลี่ยม (หัว)
- 'H': รูปหกเหลี่ยม (ตัวใหญ่)
- '+': เครื่องหมายบวก
- 'x': เครื่องหมายกากบาท
- '|': เครื่องหมายแนวตั้ง
- '_': เครื่องหมายแนวนอน

linestyle

- '-': เส้นตรง (solid line)
- '--': เส้นประ (dashed line)
- '-.': เส้นประ-จุด (dash-dot line)
- ':': เส้นจุด (dotted line)

sentiment_3

[1] 17s

```
# นำเข้าไลบรารีที่จำเป็น
import re
import pandas as pd
import matplotlib.pyplot as plt
import nltk
from nltk.sentiment.vader import SentimentIntensityAnalyzer

# ดาวน์โหลดข้อมูล 'vader_lexicon' สำหรับใช้ในการวิเคราะห์ความรู้สึกด้วย VADER
nltk.download('vader_lexicon')
```

```
[nltk_data] Downloading package vader_lexicon to /root/nltk_data...
True
```

```
[2]  # นำเข้าโมดูลที่ใช้ในการอัปโหลดไฟล์จากเครื่องผู้ใช้ใน Google Colab
     from google.colab import files
     import io  # สำหรับการจัดการ input/output ในรูปแบบของไบนารีสตรีม

     # เปิดหน้าต่างให้อัปโหลดไฟล์จากเครื่องผู้ใช้
     uploaded = files.upload()

     # อ่านเนื้อหาของไฟล์ .xlsx ที่อัปโหลดมา โดยใช้ io.BytesIO เนื่องจากเป็นไฟล์ไบนารี
     with io.BytesIO(uploaded['TeamHealthRawDataForDemo.xlsx']) as f:
         df = pd.read_excel(f, engine='openpyxl')  # อ่านไฟล์ Excel และเก็บข้อมูลใน DataFrame

     # สร้างคอลัมน์ใหม่ชื่อ 'row_id' โดยกำหนดค่าเป็นลำดับของแถว (เริ่มจาก 1)
     df["row_id"] = df.index + 1

     # แสดงข้อมูล 10 แถวแรกของ DataFrame เพื่อดูข้อมูลเบื้องต้น
     print(df.head(10))
```

```
เลือกไฟล์ TeamHealthR...ForDemo.xlsx
• TeamHealthRawDataForDemo.xlsx(application/vnd.openxmlformats-officedocument.spreadsheetml.sheet)
Saving TeamHealthRawDataForDemo.xlsx to TeamHealthRawDataForDemo.xlsx
  Period Manager    Team                                          Response  \
0 2019-Q1   Mgr 1  Team 1  We're a fun team that works well together and ...
1 2019-Q1   Mgr 1  Team 1  we have a sound and collaborative team focused...
2 2019-Q1   Mgr 1  Team 1  we work well as a team, we have fun together, ...
3 2019-Q1   Mgr 1  Team 1  I fell pretty good about the health of our tea...
4 2019-Q1   Mgr 1  Team 1  happy with team's overall health and good dyna...
5 2019-Q1   Mgr 1  Team 1                                            Solid
6 2019-Q1   Mgr 1  Team 1  The Team 2 team is a  collaborative group prod...
7 2019-Q1   Mgr 1  Team 1  We have great teamwork.  We have a lot of fun....
8 2019-Q1   Mgr 1  Team 1  We feel good about our teamwork, process, tech...
9 2019-Q1   Mgr 1  Team 2  A blast!  Always working towards delivering mo...

   row_id
0       1
1       2
2       3
3       4
4       5
5       6
6       7
7       8
8       9
9      10
```

```
[3]  # สร้าง DataFrame ย่อยที่มีเฉพาะคอลัมน์ 'row_id' และ 'Response' สำหรับการวิเคราะห์
     df_subset = df[['row_id', 'Response']].copy()

     # ทำความสะอาดข้อมูลในคอลัมน์ 'Response'
     # ลบอักขระที่ไม่ใช่ตัวอักษร (เช่น ตัวเลข สัญลักษณ์) โดยแทนที่ด้วยช่องว่าง
     df_subset['Response'] = df_subset['Response'].str.replace("[^a-zA-Z#]", " ")

     # แปลงข้อความทั้งหมดเป็นตัวพิมพ์เล็ก เพื่อความสม่ำเสมอในการวิเคราะห์
     df_subset['Response'] = df_subset['Response'].str.casefold()

     # แสดงข้อมูล 10 แถวแรกหลังจากทำความสะอาดข้อมูล
     print(df_subset.head(10))
```

```
   row_id                                          Response
0       1  we're a fun team that works well together and ...
1       2  we have a sound and collaborative team focused...
2       3  we work well as a team, we have fun together, ...
3       4  i fell pretty good about the health of our tea...
4       5  happy with team's overall health and good dyna...
5       6                                             solid
6       7  the team 2 team is a  collaborative group prod...
7       8  we have great teamwork.  we have a lot of fun....
8       9  we feel good about our teamwork, process, tech...
9      10  a blast!  always working towards delivering mo...
```

```python
# สร้าง DataFrame ว่างเพื่อใช้เก็บผลลัพธ์การวิเคราะห์ความรู้สึก
df1 = pd.DataFrame()
df1['row_id'] = ['99999999999']  # กำหนดค่าเริ่มต้นเป็น '99999999999' เพื่อใช้เป็นตัวบ่งชี้ในภายหลัง
df1['sentiment_type'] = 'NA999NA'  # กำหนดค่าเริ่มต้นสำหรับประเภทของความรู้สึก
df1['sentiment_score'] = 0  # กำหนดค่าเริ่มต้นสำหรับคะแนนความรู้สึก
```

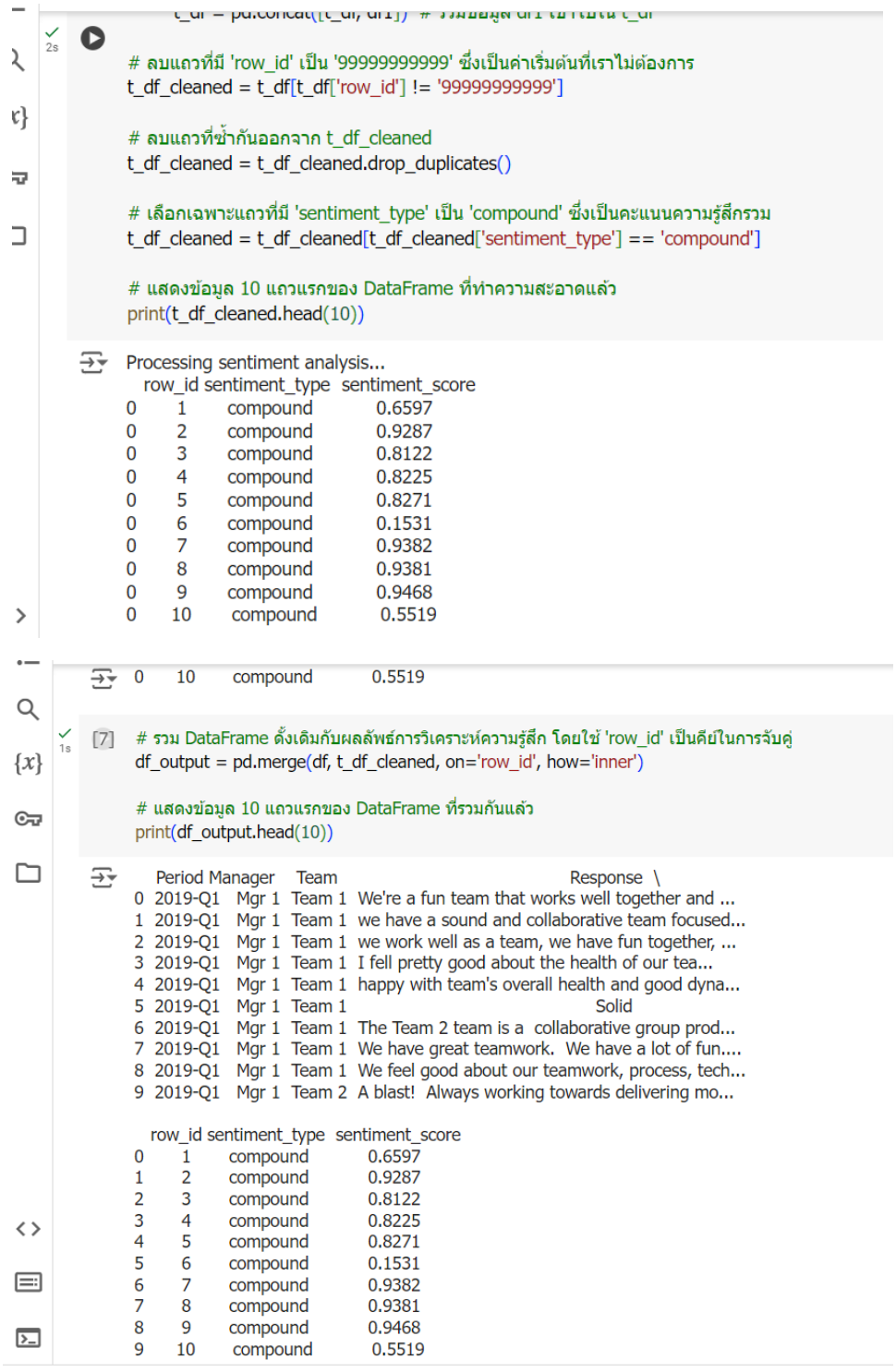
Sentiment Type values are

- **neg** for negative sentiment
- **neu** for neutral sentiment
- **pos** for positive sentiment
- **compound** for an overall score that combines negative, positive, and neutral sentiments into a single score.

ประเภทความรู้สึก ค่าต่างๆ ได้แก่

- **เชิงลบ** สำหรับความรู้สึกเชิงลบ
- **เป็นกลาง** สำหรับความรู้สึกเป็นกลาง
- **เชิงบวก** สำหรับความรู้สึกเชิงบวก
- **รวม** สำหรับคะแนนรวมที่รวมความรู้สึกเชิงลบ เชิงบวก และเป็นกลางเข้าเป็นคะแนนเดียว

```python
# แสดงข้อความเพื่อแจ้งว่ากำลังดำเนินการวิเคราะห์ความรู้สึก
print('Processing sentiment analysis...')

# สร้างวัตถุ SentimentIntensityAnalyzer สำหรับการวิเคราะห์ความรู้สึก
sid = SentimentIntensityAnalyzer()

# กำหนด DataFrame ชั่วคราวเพื่อใช้เก็บผลลัพธ์ระหว่างการวนลูป
t_df = df1

# วนลูปผ่านแต่ละแถวใน df_subset เพื่อวิเคราะห์ความรู้สึก
for index, row in df_subset.iterrows():
    # ดึงข้อความจากคอลัมน์ 'Response' โดยใช้ .iloc[1] (คอลัมน์ที่สอง)
    text = row.iloc[1]

    # ใช้ SentimentIntensityAnalyzer เพื่อคำนวณคะแนนความรู้สึกของข้อความ
    # จะได้ผลลัพธ์เป็นพจนานุกรมที่มีคีย์ 'neg', 'neu', 'pos', และ 'compound'
    scores = sid.polarity_scores(text)

    # วนลูปผ่านแต่ละประเภทของคะแนนความรู้สึกและค่าของมัน
    for key, value in scores.items():
        row_id = row.iloc[0]  # ดึงค่า 'row_id' จากคอลัมน์แรก
        df1['row_id'] = row_id  # กำหนดค่า 'row_id' ใน df1
        df1['sentiment_type'] = key  # กำหนดประเภทของความรู้สึก (เช่น 'neg', 'neu', 'pos', 'compound')
        df1['sentiment_score'] = value  # กำหนดคะแนนของความรู้สึก
        t_df = pd.concat([t_df, df1])  # รวมข้อมูล df1 เข้าไปใน t_df
```



```
t_df = pd.concat([t_df, df1]) # รวมข้อมูล df1 เข้าไปใน t_df

# ลบแถวที่มี 'row_id' เป็น '99999999999' ซึ่งเป็นค่าเริ่มต้นที่เราไม่ต้องการ
t_df_cleaned = t_df[t_df['row_id'] != '99999999999']

# ลบแถวที่ซ้ำกันออกจาก t_df_cleaned
t_df_cleaned = t_df_cleaned.drop_duplicates()

# เลือกเฉพาะแถวที่มี 'sentiment_type' เป็น 'compound' ซึ่งเป็นคะแนนความรู้สึกรวม
t_df_cleaned = t_df_cleaned[t_df_cleaned['sentiment_type'] == 'compound']

# แสดงข้อมูล 10 แถวแรกของ DataFrame ที่ทำความสะอาดแล้ว
print(t_df_cleaned.head(10))
```

```
Processing sentiment analysis...
  row_id sentiment_type  sentiment_score
0      1       compound           0.6597
0      2       compound           0.9287
0      3       compound           0.8122
0      4       compound           0.8225
0      5       compound           0.8271
0      6       compound           0.1531
0      7       compound           0.9382
0      8       compound           0.9381
0      9       compound           0.9468
0     10       compound           0.5519
```

```
0     10       compound           0.5519
```

```
[7] # รวม DataFrame ดั้งเดิมกับผลลัพธ์การวิเคราะห์ความรู้สึก โดยใช้ 'row_id' เป็นคีย์ในการจับคู่
df_output = pd.merge(df, t_df_cleaned, on='row_id', how='inner')

# แสดงข้อมูล 10 แถวแรกของ DataFrame ที่รวมกันแล้ว
print(df_output.head(10))
```

```
   Period Manager    Team                                          Response  \
0  2019-Q1   Mgr 1  Team 1  We're a fun team that works well together and ...
1  2019-Q1   Mgr 1  Team 1  we have a sound and collaborative team focused...
2  2019-Q1   Mgr 1  Team 1  we work well as a team, we have fun together, ...
3  2019-Q1   Mgr 1  Team 1  I fell pretty good about the health of our tea...
4  2019-Q1   Mgr 1  Team 1  happy with team's overall health and good dyna...
5  2019-Q1   Mgr 1  Team 1                                             Solid
6  2019-Q1   Mgr 1  Team 1  The Team 2 team is a  collaborative group prod...
7  2019-Q1   Mgr 1  Team 1  We have great teamwork.  We have a lot of fun....
8  2019-Q1   Mgr 1  Team 1  We feel good about our teamwork, process, tech...
9  2019-Q1   Mgr 1  Team 2  A blast!  Always working towards delivering mo...

   row_id sentiment_type  sentiment_score
0       1       compound           0.6597
1       2       compound           0.9287
2       3       compound           0.8122
3       4       compound           0.8225
4       5       compound           0.8271
5       6       compound           0.1531
6       7       compound           0.9382
7       8       compound           0.9381
8       9       compound           0.9468
9      10       compound           0.5519
```

สรุปการทำงานของโค้ด:

1. **นำเข้าไลบรารีและดาวน์โหลดข้อมูลที่จำเป็น:** นำเข้าไลบรารีที่จำเป็นสำหรับการประมวลผลและวิเคราะห์ข้อมูล รวมถึงดาวน์โหลด vader_lexicon สำหรับการวิเคราะห์ความรู้สึกด้วย VADER

2. **อัปโหลดและอ่านข้อมูลจากไฟล์ Excel**: เปิดหน้าต่างให้อัปโหลดไฟล์ Excel (TeamHealthRawDataForDemo.xlsx) และอ่านข้อมูลเข้าใน DataFrame โดยใช้ Pandas
3. **เตรียมข้อมูล**:
    - สร้างคอลัมน์ row_id เพื่อเก็บหมายเลขแถว
    - สร้าง DataFrame ย่อย df_subset ที่มีเฉพาะคอลัมน์ row_id และ Response
    - ทำความสะอาดข้อความในคอลัมน์ Response โดยลบอักขระที่ไม่ใช่ตัวอักษรและแปลงเป็นตัวพิมพ์เล็ก
4. **เตรียม DataFrame สำหรับเก็บผลลัพธ์**: สร้าง DataFrame ว่าง df1 เพื่อใช้เก็บผลลัพธ์การวิเคราะห์ความรู้สึก โดยกำหนดค่าเริ่มต้นที่ไม่ซ้ำกับข้อมูลจริง
5. **วิเคราะห์ความรู้สึก**:
    - สร้างวัตถุ SentimentIntensityAnalyzer สำหรับการวิเคราะห์ความรู้สึก
    - วนลูปผ่านแต่ละแถวใน df_subset เพื่อวิเคราะห์ความรู้สึกของข้อความในคอลัมน์ Response
    - สำหรับแต่ละข้อความ จะได้รับคะแนนความรู้สึก 4 ประเภท (neg, neu, pos, compound)
    - เก็บผลลัพธ์การวิเคราะห์ลงใน t_df โดยรวมกับข้อมูลเดิมในแต่ละรอบของการวนลูป
6. **ทำความสะอาดผลลัพธ์**:
    - ลบแถวที่มีค่าเริ่มต้นที่เราไม่ต้องการ (เช่น row_id เป็น '99999999999')
    - ลบข้อมูลที่ซ้ำกัน
    - เลือกเฉพาะแถวที่มี sentiment_type เป็น 'compound' ซึ่งเป็นคะแนนความรู้สึกรวมที่มีค่าตั้งแต่ -1 (ลบมาก) ถึง 1 (บวกมาก)
7. **รวมผลลัพธ์กับข้อมูลต้นฉบับ**: ใช้ pd.merge เพื่อรวม DataFrame ดั้งเดิมกับผลลัพธ์การวิเคราะห์ความรู้สึก โดยจับคู่ข้อมูลด้วย row_id
8. **แสดงผลลัพธ์**: แสดงข้อมูล 10 แถวแรกของ DataFrame ที่รวมกันแล้ว เพื่อดูผลลัพธ์ของการวิเคราะห์ความรู้สึก

โค้ดนี้ช่วยให้คุณสามารถวิเคราะห์ความรู้สึกของข้อความในคอลัมน์ Response ของไฟล์ Excel และรวมผลลัพธ์กลับเข้ากับข้อมูลเดิมได้อย่างง่ายดาย

Chapter 6 Building Chatbots I

---

1. การแนะนำแชทบอท (หน้า 2):

- แชทบอทเป็นโปรแกรมซอฟต์แวร์ที่ถูกออกแบบมาเพื่อจำลองการสนทนาของมนุษย์ โดยทำงานผ่านแอปส่งข้อความ
- แนวคิดของซอฟต์แวร์สนทนาไม่ใช่เรื่องใหม่ ย้อนไปถึงยุค 1960 แต่แชทบอทเริ่มมีความนิยมมากขึ้นเรื่อย ๆ ในด้านการตลาด
- เทคโนโลยีของแชทบอทในปัจจุบันใช้ **การประมวลผลภาษาธรรมชาติ (NLP)** และ **ปัญญาประดิษฐ์ (AI)** เพื่อทำความเข้าใจความต้องการของผู้ใช้ และปรับการตอบสนองเพื่อช่วยให้ผู้ใช้บรรลุเป้าหมาย เหมือนผู้ช่วยเสมือนอย่าง Siri หรือ Alexa

---

2. แชทบอททำงานอย่างไร? (หน้า 3):

- แชทบอทสามารถทำงานด้วยคำตอบที่ถูกตั้งโปรแกรมไว้ล่วงหน้า หรือด้วย AI หรือทั้งสองแบบรวมกัน
- แชทบอทจะประมวลผลคำถามของผู้ใช้และให้คำตอบที่ตรงกับคำถามนั้น
- มีแชทบอทหลัก ๆ 2 ประเภท:
    1. **แชทบอทที่ใช้กฎ (Rule-based Chatbots):** ตอบคำถามตามกฎที่ตั้งไว้ล่วงหน้า
    2. **แชทบอทที่ใช้ AI (AI Chatbots):** สามารถเรียนรู้จากการสนทนาในอดีตและตอบโต้ได้อย่างยืดหยุ่น

---

3. แชทบอทที่ใช้กฎ (Rule-based Chatbots) (หน้า 4):

- แชทบอทประเภทนี้ให้คำตอบตามกฎ **if/then** ที่ถูกกำหนดโดยนักออกแบบแชทบอท
- จะตอบเฉพาะเมื่อผู้ใช้ใช้คำสั่งหรือคำที่ถูกโปรแกรมไว้
- ข้อจำกัด: แชทบอทประเภทนี้ไม่สามารถเรียนรู้จากประสบการณ์ที่ผ่านมาได้
- ข้อดี: เป็นแชทบอทที่สร้างได้ง่ายและมีราคาถูกที่สุด
- ตัวอย่างเช่น การรีเซ็ตรหัสผ่าน แชทบอทจะค้นหาคำสำคัญในข้อความ เช่น “reset” และ “password” แล้วจับคู่กับคำตอบที่มีอยู่

---

4. แชทบอทที่ใช้ AI (AI Chatbots) (หน้า 5):

- แชทบอทที่ใช้ AI สามารถสนทนาได้อย่างอิสระกับผู้ใช้
- จำเป็นต้องถูกฝึกฝนด้วยคำตอบที่ถูกกำหนดไว้ในช่วงเริ่มต้น แต่สามารถเรียนรู้จากการสนทนาที่ผ่านมาได้โดยไม่ต้องอัปเดตด้วยมือ
- ใช้เทคโนโลยี **การเรียนรู้ของเครื่อง** (**Machine Learning - ML)** เพื่อวิเคราะห์รูปแบบของข้อความและตัดสินใจตอบกลับได้อย่างมีประสิทธิภาพ
- ใช้ **การประมวลผลภาษาธรรมชาติ** (**NLP)** เพื่อทำความเข้าใจวิธีการสื่อสารของมนุษย์ และเข้าใจบริบทของการสนทนาได้แม้จะมีข้อผิดพลาดในการสะกดคำ

---

5. การใช้งานเนื้อหาและการทำงานของแชทบอท (หน้า 6):

- ในหน้านี้แนะนำวิธีการใช้ **Smalltalk** กับแชทบอทเพื่อให้การสนทนาดูเป็นธรรมชาติ
- คุณจะได้เรียนรู้วิธีใช้ **regular expressions** และ **Machine Learning** เพื่อดึงความหมายจากข้อความที่เป็นอิสระ

---

6. EchoBot I (หน้า 7-9):

- แชทบอทนี้เป็นแชทบอทพื้นฐานที่ใช้ในการตอบกลับข้อความที่ผู้ใช้ส่งมา
- ตัวอย่างการใช้งาน:
    - ผู้ใช้: "Hello!"
    - บอท: "I can hear you, you said: ‘Hello!’"
- มีการสร้างฟังก์ชัน **respond()** เพื่อรับข้อความจากผู้ใช้และตอบกลับข้อความเดิมในรูปแบบที่กำหนด
- ใน EchoBot II (หน้า 8) มีการเพิ่มฟังก์ชัน **send_message()** เพื่อบันทึกข้อความของผู้ใช้และการตอบกลับของบอท
- ใน EchoBot III (หน้า 9) มีการเพิ่มการหน่วงเวลาเล็กน้อยก่อนที่บอทจะตอบกลับเพื่อให้การสนทนารู้สึกเป็นธรรมชาติมากขึ้น โดยใช้โมดูล **time**

---

7. การสร้างบุคลิกภาพให้กับแชทบอท (หน้า 17-21):

- การสร้างบุคลิกภาพให้กับแชทบอททำให้ผู้ใช้งานรู้สึกเหมือนกำลังสนทนากับคนจริง ๆ
- การเพิ่ม **Smalltalk** หรือการสนทนาทั่วไปก่อนที่จะนำเสนอฟังก์ชันหลัก ช่วยให้แชทบอทเข้าถึงได้ง่ายและสนุกสนานมากขึ้น
- ในหน้า 20 ได้มีการนำตัวแปรมาใช้ในคำตอบของแชทบอท เช่น การตอบกลับเกี่ยวกับสภาพอากาศที่เปลี่ยนแปลงตามข้อมูลที่กำหนด

---

8. การเลือกคำตอบและการถามคำถาม (หน้า 22-34):

- แชทบอทที่ดีควรสามารถสร้างความสนใจโดยการถามคำถามเพื่อดึงดูดผู้ใช้งาน
- ในหน้านี้มีการสร้างฟังก์ชัน **respond()** ที่จะเลือกคำตอบแบบสุ่มจากพจนานุกรมที่กำหนดไว้ล่วงหน้า
- นอกจากนี้ยังมีการใช้ **regular expressions** เพื่อจับคู่รูปแบบของข้อความและดึงข้อมูลสำคัญจากคำถามของผู้ใช้

---

9. การประมวลผลข้อความด้วย Regular Expressions (หน้า 35-45):

- Regular expressions (หรือ regex) ใช้สำหรับการจับคู่ข้อความกับรูปแบบที่รู้จัก หรือการดึงข้อความสำคัญ
- ในหน้านี้อธิบายวิธีการใช้ **regex** ในการจับคู่คำถาม เช่น "do you remember" และดึงข้อมูลที่เกี่ยวข้องมาแสดงผล

---

10. การแปลงไวยากรณ์ (หน้า 40):

- หน้านี้แสดงวิธีการใช้ **regex** ในการแปลงคำบุพบท เช่น "I" เป็น "you" และ "my" เป็น "your" เพื่อให้คำตอบของแชทบอทดูเป็นธรรมชาติมากขึ้น

---

11. การรวมทุกอย่างเข้าด้วยกัน (หน้า 49-50):

- ในส่วนสุดท้ายของบทนี้ได้รวมทุกองค์ประกอบที่เราเรียนรู้มาก่อนหน้านี้เข้าด้วยกัน

- เราสร้างฟังก์ชัน **match_rule()** และ **replace_pronouns()** เพื่อให้แชทบอทสามารถจับคู่ข้อความและแปลงคำบุพบทได้อย่างถูกต้อง

bot1.py

```
PS C:\Users\rawip\OneDrive\เดสก์ท็อป\NLP\CH6> pypython .\bot1.py
USER: what's your name?
BOT : I can hear you! you said: what's your name?
USER: bye
BOT: Goodbye!
```

 bot_template และ **user_template**: ใช้สำหรับสร้างข้อความที่จะแสดงผลให้กับผู้ใช้และบอท

 responses: เป็นพจนานุกรมที่เก็บคู่คำถาม-คำตอบที่บอทสามารถตอบได้

 respond(): ฟังก์ชันที่ทำหน้าที่ตรวจสอบข้อความจากผู้ใช้และตอบกลับตามข้อมูลในพจนานุกรม หรือแสดงข้อความที่บอท "ได้ยิน"

 send_message(): ฟังก์ชันนี้จะทำให้บอทสามารถสนทนากับผู้ใช้ โดยรับข้อความจากผู้ใช้ผ่านคอนโซล และตอบกลับโดยใช้ฟังก์ชัน respond

bot2.py

```
PS C:\Users\rawip\OneDrive\เดสก์ท็อป\NLP\CH6> pypython .\bot2.py
USER: what's your name?
BOT : my name is EchoBot
USER: what's the weather today?
BOT : it's sunny!
USER: bye
BOT: Goodbye!
PS C:\Users\rawip\OneDrive\เดสก์ท็อป\NLP\CH6>
```

1. **bot_template** และ **user_template**: ใช้เป็นรูปแบบในการแสดงผลข้อความของบอทและผู้ใช้ โดย {0} เป็นตำแหน่งที่จะแทนที่ด้วยข้อความที่ต้องการแสดง
2. **responses**: เป็นพจนานุกรมที่เก็บคู่คำถามและคำตอบ ซึ่งบอทจะใช้ตอบกลับผู้ใช้หากคำถามตรงกับข้อความที่เก็บไว้ในพจนานุกรมนี้

3. **respond(message):** ฟังก์ชันนี้ตรวจสอบว่าข้อความที่ผู้ใช้ส่งมาตรงกับคำถามในพจนานุกรมหรือไม่ ถ้าตรงจะคืนค่าคำตอบตามที่ระบุไว้ในพจนานุกรม

4. **send_message():** ฟังก์ชันนี้รับข้อความจากผู้ใช้ ถ้าผู้ใช้พิมพ์คำว่า 'bye' บอทจะกล่าวลาและจบการทำงาน ถ้าเป็นข้อความอื่นจะเรียกฟังก์ชัน respond() เพื่อหาคำตอบและแสดงผล

bot3.py

```
PS C:\Users\rawip\OneDrive\เดสก์ท็อป\NLP\CH6> & python .\bot3.py
USER: what's today's weather?
BOT : it's cloudy today
USER: bye
BOT: Goodbye!
PS C:\Users\rawip\OneDrive\เดสก์ท็อป\NLP\CH6>
```

- **bot_template** และ **user_template:** ใช้สำหรับการจัดรูปแบบข้อความที่บอทและผู้ใช้จะพิมพ์ โดย {0} เป็นตำแหน่งที่จะแทนที่ด้วยข้อความที่ต้องการแสดงผล

- **responses:** พจนานุกรมที่เก็บคำถามและคำตอบ โดยคำถามคือ "what's today's weather?" และคำตอบคือ "it's {} today". ในตำแหน่ง {} จะใช้สำหรับใส่ข้อมูลสภาพอากาศที่ได้จากตัวแปร weather_today

- **weather_today:** ตัวแปรที่เก็บข้อมูลสภาพอากาศ ซึ่งสามารถเปลี่ยนแปลงได้ตามความต้องการ

- **respond(message):** ฟังก์ชันที่ตรวจสอบว่าข้อความของผู้ใช้ตรงกับคำถามในพจนานุกรม responses หรือไม่ และถ้าตรง จะใช้ข้อมูลจาก weather_today แทนใน {} เพื่อสร้างข้อความตอบกลับ

- **send_message():** ฟังก์ชันนี้จะรับข้อความจากผู้ใช้ ถ้าผู้ใช้พิมพ์ว่า 'bye' บอทจะกล่าวลาและออกจากการสนทนา ถ้าเป็นข้อความอื่น บอทจะตอบกลับด้วยฟังก์ชัน respond()

bot4.py

```
USER: what's your name?
BOT : my name is EchoBot
USER: what's your name?
BOT : they call me EchoBot
```

1. **bot_template** และ **user_template:** ใช้เป็นรูปแบบในการแสดงข้อความของบอทและผู้ใช้ โดย {0} จะถูกแทนด้วยข้อความที่บอทหรือผู้ใช้พิมพ์

2. **responses**: เก็บรายการคำตอบที่เป็นไปได้สำหรับคำถาม "what's your name?" ซึ่งมีหลายคำตอบและถูกเก็บในรูปแบบของรายการ (list)
3. **random.choice()**: ใช้สำหรับเลือกคำตอบจากรายการคำตอบที่เป็นไปได้ใน responses แบบสุ่ม
4. **respond(message)**: ฟังก์ชันที่ตรวจสอบว่าข้อความที่ผู้ใช้พิมพ์ตรงกับคำถามที่อยู่ใน responses หรือไม่ ถ้าตรง ฟังก์ชันจะเลือกคำตอบแบบสุ่มจาก responses และส่งกลับไปเป็นคำตอบของบอท
5. **send_message()**: ฟังก์ชันนี้จะวนลูปรับข้อความจากผู้ใช้ ถ้าผู้ใช้พิมพ์คำว่า 'bye' บอทจะกล่าวลาและจบการทำงาน แต่ถ้าผู้ใช้พิมพ์คำถามอื่น บอทจะเลือกคำตอบและตอบกลับโดยใช้ฟังก์ชัน respond()

bot5.py

```
PS C:\Users\rawip\OneDrive\เดสก์ท็อป\NLP\CH6> python .\bot5.py
USER: I think you're really great.
BOT : tell me more!
USER: How are you?
BOT : why do you think that?
USER: bye
BOT: Goodbye!
PS C:\Users\rawip\OneDrive\เดสก์ท็อป\NLP\CH6>
```

1. **การกำหนดเทมเพลต**:
   - bot_template = "BOT : {0}": เทมเพลตสำหรับแสดงข้อความที่บอทตอบกลับผู้ใช้
   - user_template = "USER : {0}": เทมเพลตสำหรับแสดงข้อความที่ผู้ใช้พิมพ์ (ในโค้ดนี้ไม่ได้ใช้)
2. **responses**:
   - เป็น **list** ที่เก็บคำตอบที่บอทสามารถตอบกลับได้
   - คำตอบในรายการมี 2 ตัวเลือก:
     - "tell me more!"
     - "why do you think that?"
   - บอทจะสุ่มเลือกคำตอบจากรายการนี้ทุกครั้งที่มีการเรียกใช้งาน
3. **ฟังก์ชัน respond(message)**:
   - ฟังก์ชันนี้ใช้ในการตอบกลับข้อความจากผู้ใช้

- เมื่อผู้ใช้พิมพ์ข้อความเข้ามา ฟังก์ชันจะ **สุ่มเลือกคำตอบ** จากรายการ responses โดยไม่สนใจว่าผู้ใช้พิมพ์อะไร
- ใช้ฟังก์ชัน random.choice(responses) เพื่อสุ่มเลือกคำตอบ

4. **ฟังก์ชัน send_message()**:
   - ฟังก์ชันนี้ใช้เพื่อเริ่มการสนทนาและรับข้อความจากผู้ใช้
   - วนลูปตลอดเวลาจนกว่าผู้ใช้จะพิมพ์คำว่า "bye" เพื่อจบการสนทนา
   - เมื่อได้รับข้อความจากผู้ใช้ จะเรียกฟังก์ชัน respond() เพื่อสุ่มตอบคำตอบจาก responses
   - แสดงผลลัพธ์โดยใช้ bot_template.format(response) ซึ่งจะแสดงคำตอบที่ถูกสุ่มเลือก

ลำดับการทำงาน:

1. ผู้ใช้พิมพ์ข้อความใดๆ
2. ถ้าผู้ใช้พิมพ์คำว่า "bye" บอทจะกล่าวลาและจบการทำงาน
3. ถ้าไม่ใช่ "bye" ฟังก์ชัน respond() จะสุ่มเลือกคำตอบจาก responses แล้วแสดงผลคำตอบนั้น
4. บอทจะวนลูปจนกว่าผู้ใช้จะพิมพ์ "bye"

bot6.py

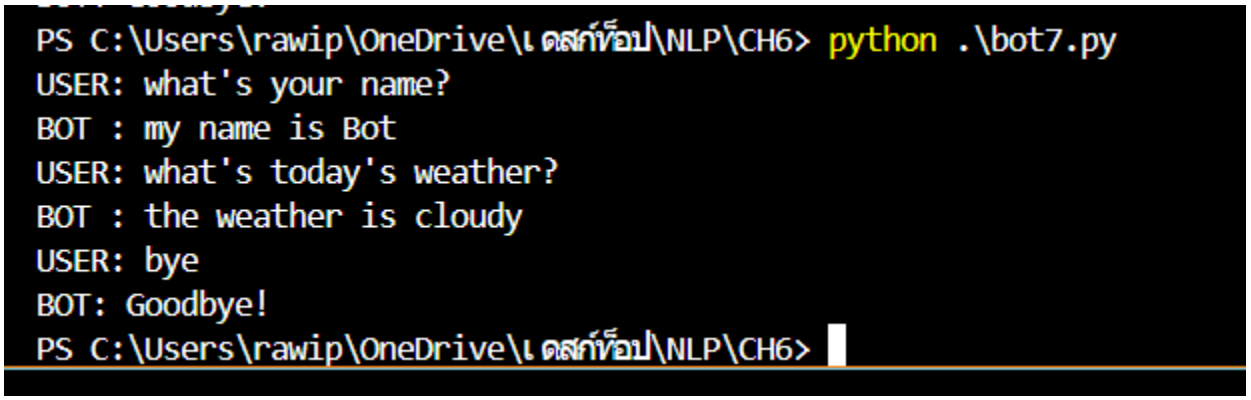
```
PS C:\Users\rawip\OneDrive\เดสก์ท็อป\NLP\CH6> python .\bot6.py
BOT : Hi!
USER : what's today's weather?
BOT : the weather is cloudy
PS C:\Users\rawip\OneDrive\เดสก์ท็อป\NLP\CH6>
```

- **bot_template**: สร้างเทมเพลตสำหรับการแสดงข้อความของบอท
- **name** และ **weather**: กำหนดค่าชื่อของบอทและสภาพอากาศที่บอทจะใช้ตอบกลับ
- **responses**: เป็นพจนานุกรมที่เก็บคำถามและคำตอบที่บอทจะใช้ตอบกลับ เช่น ชื่อบอทและสภาพอากาศ
- **respond(message)**: ฟังก์ชันนี้ตรวจสอบว่าข้อความของผู้ใช้ตรงกับคำถามที่มีใน responses หรือไม่ ถ้าตรงจะคืนค่าคำตอบ ถ้าไม่ตรงจะคืนค่าข้อความเริ่มต้น (default)
- **send_message(message)**: ฟังก์ชันนี้ส่งข้อความของผู้ใช้ไปยังบอทและแสดงผลลัพธ์จากบอท

 input(): รับข้อความจากผู้ใช้ผ่านการพิมพ์

 send_message(value): เรียกใช้ฟังก์ชันเพื่อแสดงผลข้อความตอบกลับจากบอท

bot7.py



 bot_template:

- เป็นเทมเพลตสำหรับการแสดงข้อความของบอท โดย {0} ใช้สำหรับแทนที่ด้วยข้อความที่บอทจะตอบกลับผู้ใช้

 name และ **weather**:

- ตัวแปรที่กำหนดข้อมูล เช่น ชื่อของบอท (name = "Bot") และสภาพอากาศ (weather = "cloudy") ซึ่งจะถูกใช้ในคำตอบที่บอทตอบเมื่อถูกถาม

 responses:

- พจนานุกรมที่เก็บคำถามและคำตอบที่เป็นไปได้สำหรับแต่ละคำถาม โดยใช้การสุ่มเลือกคำตอบจากรายการในแต่ละคำถาม เช่น:
    - ถ้าถูกถามว่า "what's your name?" บอทจะสุ่มตอบจากคำตอบที่เป็นไปได้ 3 แบบ
    - ถ้าถูกถามว่า "what's today's weather?" บอทจะสุ่มตอบจากคำตอบที่เป็นไปได้ 2 แบบ
    - ถ้าเป็นคำถามที่ไม่ตรงกับที่เตรียมไว้ บอทจะตอบข้อความเริ่มต้น "default message"

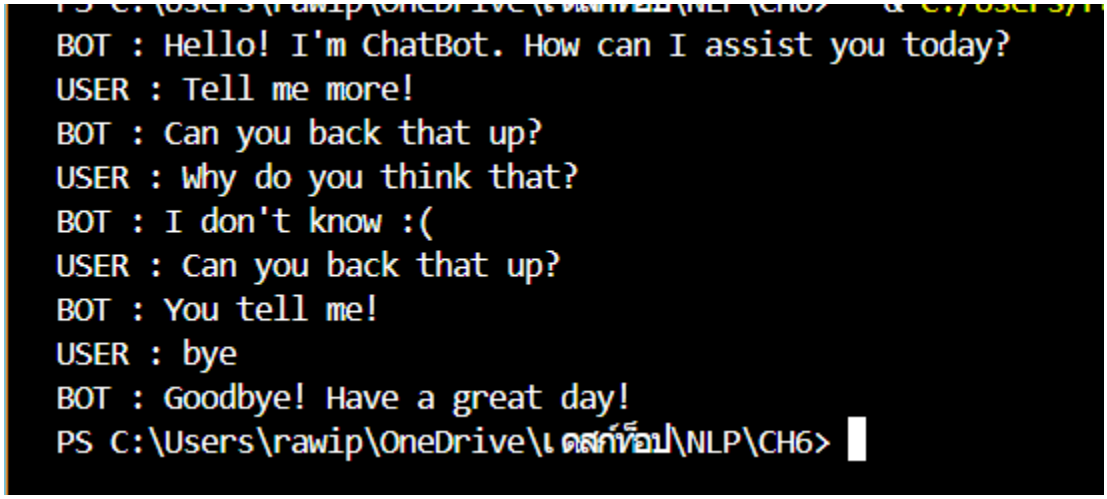
 ฟังก์ชัน respond(message):

- ฟังก์ชันนี้ทำหน้าที่ตรวจสอบว่าข้อความจากผู้ใช้ตรงกับคำถามที่บอทสามารถตอบได้หรือไม่:
    - ถ้าตรง จะสุ่มเลือกคำตอบจากรายการคำตอบที่เตรียมไว้
    - ถ้าไม่ตรง จะสุ่มเลือกคำตอบจากหมวด "default"

 ฟังก์ชัน send_message():

- เป็นฟังก์ชันหลักที่วนลูปเพื่อให้ผู้ใช้สามารถสนทนากับบอทได้
- ถ้าผู้ใช้พิมพ์ "bye" ระบบจะตอบกลับด้วยข้อความลาและจบการสนทนา
- ถ้าผู้ใช้พิมพ์คำถามอื่นๆ ระบบจะเรียกฟังก์ชัน respond() เพื่อตอบกลับ

bot8.py



1. **responses**: พจนานุกรมนี้เก็บคำตอบต่างๆ ที่บอทจะใช้ในการตอบกลับ แบ่งเป็น 3 หมวดหมู่:
    - statement: สำหรับข้อความทั่วไป (ไม่ใช่คำถาม)
    - question: สำหรับข้อความที่เป็นคำถาม (ลงท้ายด้วย ?)
    - default: สำหรับข้อความที่ไม่เข้าใจ
2. **bot_template และ user_template**: เป็นเทมเพลตสำหรับการแสดงข้อความบอทและผู้ใช้ โดย {0} คือข้อความที่จะถูกแทนที่เมื่อแสดงผล
3. **get_response(message)**: ฟังก์ชันนี้ทำหน้าที่แยกแยะว่าข้อความที่ผู้ใช้ส่งเข้ามาเป็นคำถามหรือข้อความทั่วไป จากนั้นสุ่มเลือกคำตอบที่เหมาะสม:
    - ถ้าข้อความลงท้ายด้วย ? ถือว่าเป็นคำถามและเลือกคำตอบจากหมวด question
    - ถ้าเป็นข้อความทั่วไป เลือกคำตอบจากหมวด statement
    - ถ้าไม่มีข้อความหรือข้อความว่างเปล่า บอทจะกระตุ้นให้ผู้ใช้พิมพ์อะไรบางอย่าง

send_message(): ฟังก์ชันนี้จัดการการสนทนาระหว่างผู้ใช้กับบอทในลูปต่อเนื่องจนกว่าผู้ใช้จะพิมพ์ "bye"

- เมื่อได้รับข้อความจากผู้ใช้ บอทจะตรวจสอบข้อความและเลือกคำตอบจากหมวดหมู่ที่เหมาะสม
- เมื่อพิมพ์ "bye" ระบบจะจบการทำงานพร้อมแสดงข้อความอำลา

bot9.py

```
PS C:\Users\rawip\OneDrive\เดสก์ท็อป\NLP\CH6> & python .\bot9.
USER : do you think
BOT : default
USER : do you remember your last birthday
BOT : Why haven't you been able to forget your last birthday
USER : I want a new car
BOT : What's stopping you from getting a new car
USER : if I win the lottery
BOT : What do you think about I win the lottery
USER : bye
BOT : Goodbye!
PS C:\Users\rawip\OneDrive\เดสก์ท็อป\NLP\CH6>
```

 บอทใช้การจับคู่ข้อความแบบ RegEx เพื่อวิเคราะห์ข้อความที่ผู้ใช้พิมพ์

 เมื่อจับคู่ได้ บอทจะสุ่มเลือกคำตอบจากรายการที่เตรียมไว้ใน rules

 บอทจะตอบข้อความที่เกี่ยวข้องกับสิ่งที่ผู้ใช้พิมพ์ เช่น การถามเกี่ยวกับความทรงจำหรือการต้องการบางสิ่ง

 การสนทนาจะดำเนินต่อไปจนกว่าผู้ใช้จะพิมพ์ "bye"

Bot10.py

```
PS C:\Users\rawip\OneDrive\เดสก์ท็อป\NLP\CH6> pypython .\bot10.py
your last birthday
go with you to florida
i had your own castle
PS C:\Users\rawip\OneDrive\เดสก์ท็อป\NLP\CH6>
```

การทำงานของโค้ด:

1. **message.lower()**: ฟังก์ชันนี้จะแปลงข้อความเป็นตัวพิมพ์เล็กทั้งหมดเพื่อให้การเปรียบเทียบกับคำสรรพนามทำได้อย่างถูกต้องและสม่ำเสมอ

2. **การแทนที่สรรพนาม:** โค้ดนี้จะตรวจสอบว่าข้อความมีคำสรรพนามใดอยู่บ้าง และจะแทนที่สรรพนามนั้นด้วยสรรพนามที่เหมาะสม:
    - 'me' ถูกแทนที่ด้วย 'you'
    - 'my' ถูกแทนที่ด้วย 'your'
    - 'your' ถูกแทนที่ด้วย 'my'
    - 'you' ถูกแทนที่ด้วย 'me'
3. **การคืนค่าข้อความ:** ถ้าไม่มีสรรพนามที่ตรงกับเงื่อนไข ฟังก์ชันจะคืนค่าข้อความเดิมกลับมา

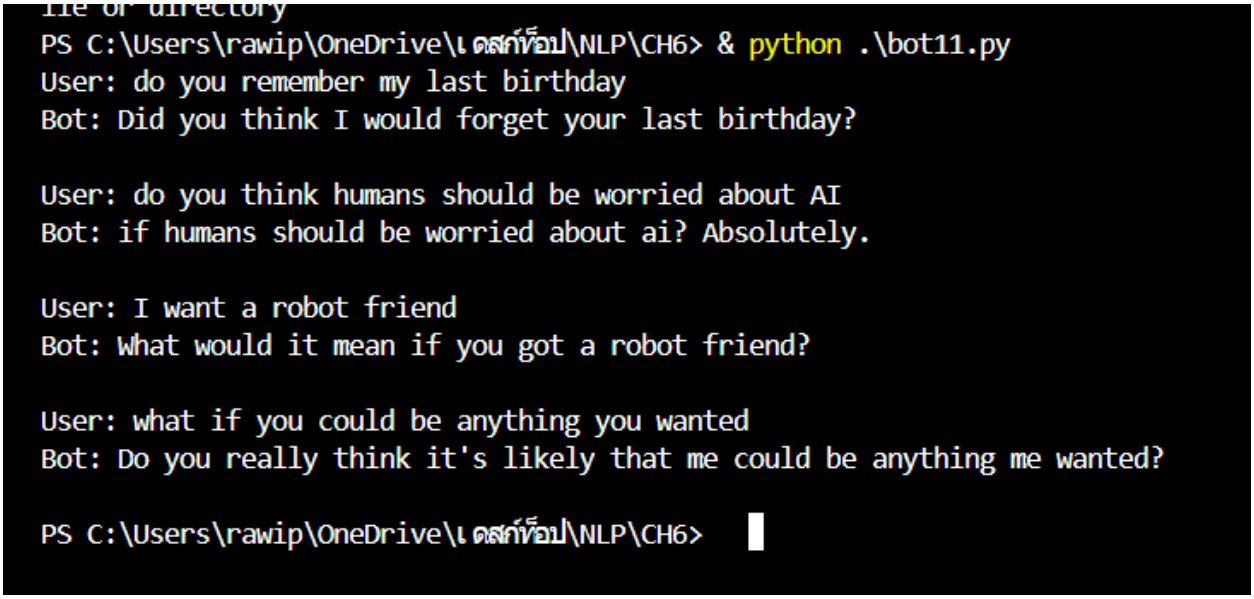
ผลลัพธ์ที่คาดหวัง:

1. "my last birthday" → "your last birthday"
2. "go with me to Florida" → "go with you to Florida"
3. "I had my own castle" → "I had your own castle"

สรุป:

ฟังก์ชันนี้ทำหน้าที่แทนที่สรรพนามในประโยคเพื่อให้ข้อความที่บอทตอบกลับสามารถสะท้อนถึงผู้ใช้ได้อย่างถูกต้อง

Bot11.py



คำอธิบายโดยสรุป:

1. **match_rule(message)**:

- ฟังก์ชันนี้ใช้ Regular Expressions (re.match) เพื่อตรวจสอบว่าข้อความผู้ใช้ตรงกับรูปแบบใดในพจนานุกรม rules และดึงส่วนของข้อความที่ตรงกัน (phrase) เพื่อนำไปใช้ในคำตอบ
- คืนค่าเป็นคู่ของคำตอบและข้อความที่จับคู่ได้ (response และ phrase)

2. **replace_pronouns(phrase):**
   - ฟังก์ชันนี้ทำการแทนที่สรรพนามใน phrase เพื่อให้ข้อความที่ตอบกลับของบอทมีความเหมาะสมกับบริบทของการสนทนา เช่น แทนที่ "my" ด้วย "your"
3. **respond(message):**
   - เรียกใช้ match_rule() เพื่อตรวจสอบว่าข้อความของผู้ใช้ตรงกับกฎใดบ้างใน rules
   - ถ้าพบว่ามีคำตอบที่ตรงกัน จะเรียก replace_pronouns() เพื่อแทนที่สรรพนามในข้อความ และใช้ .format() เพื่อจัดรูปแบบคำตอบให้เหมาะสม
4. **send_message(message):**
   - จำลองการส่งข้อความจากผู้ใช้ไปยังบอท และแสดงผลคำตอบของบอท

การทำงาน:

เมื่อบอทได้รับข้อความจากผู้ใช้ จะทำการจับคู่รูปแบบกับข้อความที่ตรงกับกฎที่ตั้งไว้ หากพบข้อความที่ตรงกัน บอทจะสุ่มตอบคำตอบที่เหมาะสม และทำการแทนที่สรรพนามเพื่อให้การสนทนาดูสมจริง

Bot12.py

```
File or directory
PS C:\Users\rawip\OneDrive\เดสก์ท็อป\NLP\CH6> & python .\bot12.py
USER : hello!
BOT : Hello you! :)
USER : bye byeee
BOT : goodbye for now
USER : thanks very much!
BOT : You are very welcome
PS C:\Users\rawip\OneDrive\เดสก์ท็อป\NLP\CH6>
```

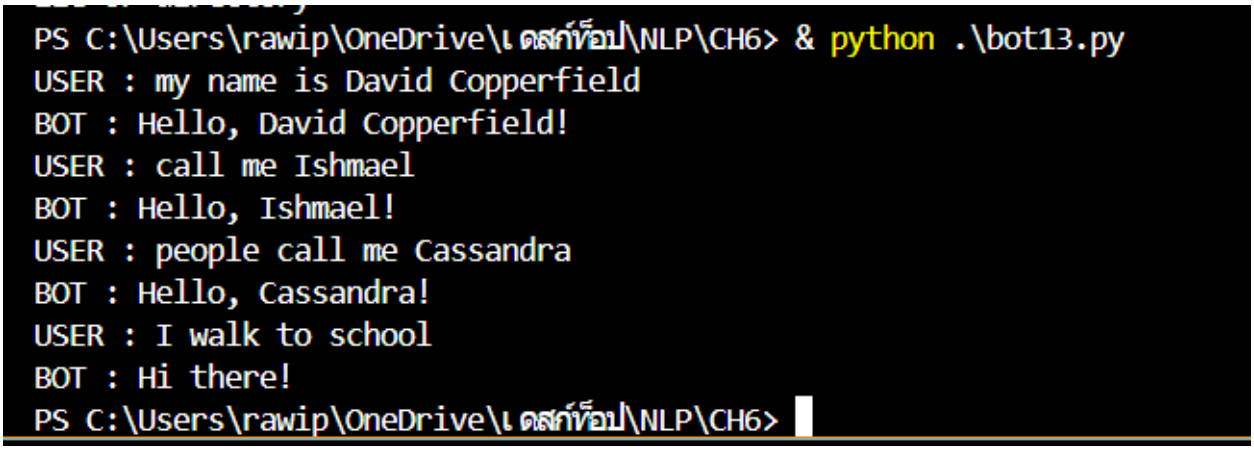
โค้ดนี้เป็นการสร้างแชทบอทที่สามารถตรวจจับความตั้งใจ (intent) ของผู้ใช้จากข้อความที่พิมพ์เข้ามา โดยใช้ Regular Expressions (RegEx) เพื่อตรวจสอบคำสำคัญที่บอทสามารถตอบกลับได้ เช่น คำทักทาย (greet), คำบอกลา (goodbye), และคำขอบคุณ (thankyou)

ส่วนประกอบหลักของโค้ด:

1. **keywords**:
    - เป็นพจนานุกรมที่เก็บคำสำคัญหรือคีย์เวิร์ดสำหรับแต่ละ intent เช่น:
        - คำทักทาย (greet): ประกอบด้วยคำว่า "hello", "hi", และ "hey"
        - คำบอกลา (goodbye): ประกอบด้วยคำว่า "bye", "farewell"
        - คำขอบคุณ (thankyou): ประกอบด้วยคำว่า "thank", "thx"
2. **patterns**:
    - ใช้ Regular Expressions เพื่อสร้างรูปแบบการจับคู่ (patterns) จากคีย์เวิร์ดใน keywords แต่ละ intent จะถูกสร้างด้วยคำสั่ง re.compile() เพื่อให้สามารถตรวจสอบข้อความจากผู้ใช้ได้อย่างง่ายดาย
3. **responses**:
    - พจนานุกรมที่เก็บคำตอบสำหรับแต่ละ intent โดยจะตอบตาม intent ที่ตรวจพบ เช่น:
        - หากตรวจพบคำทักทาย (greet) จะตอบกลับว่า "Hello you! :)"
4. **match_intent(message)**:
    - ฟังก์ชันนี้ทำหน้าที่ตรวจสอบข้อความจากผู้ใช้ว่าตรงกับ intent ใด โดยใช้การตรวจสอบกับ patterns
    - ถ้าพบคำที่ตรงกับ pattern จะคืนค่า intent ที่ตรงกัน
5. **respond(message)**:
    - ฟังก์ชันนี้ใช้ผลลัพธ์จาก match_intent() เพื่อตรวจสอบ intent ที่ตรงกัน และตอบกลับด้วยข้อความที่เก็บไว้ใน responses
    - ถ้าไม่พบ intent ที่ตรงกัน จะใช้คำตอบเริ่มต้น (default)
6. **send_message(message)**:
    - ฟังก์ชันนี้ทำหน้าที่จำลองการสนทนาระหว่างผู้ใช้กับบอท โดยแสดงข้อความผู้ใช้และเรียกใช้ฟังก์ชัน respond() เพื่อตอบกลับจากบอท
7. **สรุป:**

โค้ดนี้ทำให้บอทสามารถตรวจจับและตอบกลับตามประเภทของข้อความ (intent) ที่ผู้ใช้ส่งเข้ามา ไม่ว่าจะเป็นการทักทาย บอกลา หรือขอบคุณ บอทสามารถตรวจสอบข้อความและตอบกลับด้วยข้อความที่เหมาะสมตามรูปแบบที่กำหนด

Bot13.py



โค้ดนี้เป็นการสร้างแชทบอทที่สามารถตรวจจับชื่อของผู้ใช้จากข้อความที่มีคำบอกความเป็นเจ้าของชื่อ เช่น "my name is", "call me", และ "people call me" โดยใช้ Regular Expressions (RegEx) เพื่อตรวจสอบข้อความและดึงชื่อออกมา

ส่วนประกอบหลักของโค้ด:

1. **find_name(message)**:
   - ฟังก์ชันนี้ทำหน้าที่ค้นหาชื่อของผู้ใช้จากข้อความ โดยใช้สองขั้นตอน:
     1. **name_keyword**: ใช้ RegEx เพื่อค้นหาคำที่เป็นคีย์เวิร์ดบ่งบอกว่าผู้ใช้กำลังระบุชื่อ เช่น "name is", "call me", และ "people call me"
     2. **name_pattern**: ใช้ RegEx เพื่อดึงชื่อจากข้อความ โดยตรวจสอบคำที่ขึ้นต้นด้วยอักษรตัวใหญ่ เช่น "David Copperfield"
2. **respond(message)**:
   - ฟังก์ชันนี้จะเรียกใช้ find_name() เพื่อตรวจสอบว่าผู้ใช้ระบุชื่อในข้อความหรือไม่:
     - ถ้าไม่พบชื่อ จะตอบกลับด้วยข้อความ "Hi there!"
     - ถ้าพบชื่อ จะตอบกลับด้วยข้อความ "Hello, {ชื่อ}!"
3. **send_message(message)**:
   - ฟังก์ชันนี้ทำหน้าที่จำลองการสนทนา โดยรับข้อความจากผู้ใช้และแสดงผลการตอบกลับจากบอท

การทำงาน:

เมื่อผู้ใช้ส่งข้อความที่มีคีย์เวิร์ดระบุชื่อ เช่น "my name is", "call me", หรือ "people call me" บอทจะตรวจจับชื่อที่มีคำขึ้นต้นด้วยตัวอักษรใหญ่ และตอบกลับผู้ใช้ด้วยการกล่าวทักทายพร้อมกับชื่อที่ตรวจพบ

สรุป:

- บอทสามารถตรวจจับชื่อของผู้ใช้จากข้อความที่มีคีย์เวิร์ด เช่น "my name is" หรือ "call me"
- ถ้าไม่พบชื่อในข้อความ บอทจะตอบกลับด้วยข้อความเริ่มต้น "Hi there!"
- ถ้าพบชื่อในข้อความ บอทจะตอบกลับด้วยการทักทายชื่อของผู้ใช้

voice_assistant.py

โค้ดนี้สร้างผู้ช่วยเสมือน (Virtual Assistant) ที่สามารถตอบคำถามและคำสั่งต่าง ๆ ของผู้ใช้ เช่น บอกวันและเวลา, เปิด Google, ค้นหาใน YouTube, และตรวจสอบข้อมูลจาก Wikipedia โดยใช้ไลบรารี pyttsx3 สำหรับการสังเคราะห์เสียงพูด, webbrowser สำหรับเปิดเว็บเบราว์เซอร์ และ wikipedia สำหรับการค้นหาข้อมูล

คำอธิบายทีละส่วน:

1. **การนำเข้าไลบรารี**:
   - pyttsx3: ไลบรารีสำหรับการแปลงข้อความเป็นเสียงพูด (text-to-speech)
   - webbrowser: ใช้สำหรับเปิดเว็บเบราว์เซอร์
   - datetime: ใช้สำหรับจัดการวันที่และเวลา
   - wikipedia: ใช้ในการค้นหาข้อมูลจาก Wikipedia
2. **ฟังก์ชัน assistant(audio)**:
   - ทำหน้าที่แปลงข้อความที่ส่งเข้ามาในรูปของตัวแปร audio เป็นเสียงพูด โดยใช้ pyttsx3
   - ใช้เสียงที่มีในระบบ โดยเลือกเสียงที่ 2 (เสียงผู้หญิง)
3. **ฟังก์ชัน greeting()**:
   - ผู้ช่วยทักทายผู้ใช้ด้วยเสียงพูดว่า "Hello, I am your Virtual Assistant. How Can I Help You"
4. **ฟังก์ชัน core_code()**:
   - เรียกใช้ฟังก์ชัน greeting() เมื่อเริ่มต้นโปรแกรม
5. **ฟังก์ชัน theDay()**:
   - คำนวณว่าวันนี้เป็นวันอะไร (เช่น Monday, Tuesday) โดยใช้ไลบรารี datetime
   - แปลงวันเป็นข้อความแล้วส่งไปที่ฟังก์ชัน assistant() เพื่อพูดออกมา
6. **ฟังก์ชัน theTime()**:
   - คำนวณเวลาในขณะนั้น และส่งข้อมูลเป็นข้อความ เช่น "The time right now is 10 hours and 45 minutes" ไปยังฟังก์ชัน assistant() เพื่อพูดออกมา
7. **ฟังก์ชัน send_message()**:
   - รอรับคำสั่งจากผู้ใช้ผ่าน input() และตรวจสอบว่าคำสั่งนั้นตรงกับเงื่อนไขใดบ้าง
   - ตัวอย่างเช่น หากผู้ใช้พิมพ์ "what day is it" โปรแกรมจะเรียกฟังก์ชัน theDay() เพื่อตอบคำถาม

```
You: what day is it
BOT: It's Friday

You: what time is it
BOT: The time right now is 12 hours and 30 minutes

You: open google
BOT: Opening Google
# เปิดเว็บ google.com

You: wiki Python
BOT: Checking Wikipedia
BOT: As per Wikipedia
BOT: Python is a programming language that lets you work quickly and integrate systems mor

You: bye
BOT: Exiting. Have a Good Day
```

คำอธิบายเพิ่มเติม:

- **การค้นหา Wikipedia**: หากผู้ใช้พิมพ์คำสั่งที่ขึ้นต้นด้วย "wiki" เช่น "wiki Python" ระบบจะดึงข้อมูลสรุปจาก Wikipedia จำนวน 4 ประโยค และให้บอทอ่านข้อมูลออกเสียง
- **การค้นหาใน YouTube**: ผู้ใช้สามารถสั่งให้ค้นหาวิดีโอใน YouTube โดยพิมพ์คำสั่งที่ขึ้นต้นด้วย "search youtube for" แล้วระบบจะเปิดเว็บเบราว์เซอร์เพื่อค้นหาวิดีโอนั้นใน YouTube
- **การสิ้นสุดการสนทนา**: หากผู้ใช้พิมพ์ "bye" โปรแกรมจะจบการทำงานและกล่าวอำลา

สรุป:

โค้ดนี้สร้างผู้ช่วยเสมือนที่สามารถตอบคำถามเกี่ยวกับวันและเวลา, เปิด Google, ค้นหาใน YouTube, และตรวจสอบข้อมูลจาก Wikipedia พร้อมทั้งอ่านผลลัพธ์เป็นเสียง

Chapter 6 - Building Chatbots in Python II:

หน้า 1: ทำความเข้าใจกับ Intent และ Entity

- หน้านี้เริ่มด้วยการแนะนำแนวคิดของ **Intent** และ **Entity** ซึ่งเกี่ยวข้องกับ Natural Language Understanding (NLU)
- **Intent** คือสิ่งที่ผู้ใช้ต้องการสื่อ เช่น คำว่า "I'm hungry" อาจหมายถึงการค้นหาร้านอาหาร
- **Entity** คือข้อมูลที่มีความหมายเฉพาะเจาะจง เช่น วันที่, ประเภทอาหาร, หรือสถานที่
    - ตัวอย่าง: ประเภทอาหาร (cuisine), วันที่ (june 10th), สถานที่ (center of town)

หน้า 2-3: Intent

- Intent คือการทำความเข้าใจสิ่งที่ผู้ใช้ต้องการ ตัวอย่างของ Intent เช่น:
    - greet คือการทักทาย เช่น "hello", "hi"
    - restaurant_search คือการค้นหาร้านอาหาร เช่น "I'm looking for sushi"
    - Intent ไม่ได้มีการจำแนกแบบตายตัว ขึ้นอยู่กับลักษณะของบอทที่สร้าง

หน้า 4: Entity

- หน้านี้จะกล่าวถึงการดึงข้อมูล Entity จากข้อความ เช่น:
    - "Book a table for June 10th at a sushi restaurant in New York City"
    - คำว่า June 10th คือตัวแทนของวันที่, sushi คือประเภทอาหาร, New York City คือตัวแทนของสถานที่

หน้า 5-6: การใช้ Regular Expressions (Regex)

- การใช้ regex เพื่อจับคำหลักในข้อความ โดยใช้เครื่องหมาย | และขอบเขตของคำ \b เพื่อจับคำที่ตรงกับตัวอย่าง เช่น "hello", "hi", หรือ "hey"
    - ตัวอย่างเช่น re.search(r"\b(hello|hey|hi)\b", "hey there!") จะตรวจจับคำว่า "hey"

หน้า 7-8: การใช้ Regex ในการค้นหา Entity

- ใช้ regex ในการค้นหา Entity เช่นชื่อบุคคลในข้อความ เช่น "my name is David Copperfield"
    - การสร้างแพทเทิร์นที่ตรงกับคำที่มีตัวอักษรตัวแรกเป็นตัวพิมพ์ใหญ่ เช่น "David Copperfield"

หน้า 9-10: การจำแนก Intent ด้วย Regex

- สร้างพจนานุกรมที่มีคำหลักและ Intent เช่น greet, goodbye, และ thankyou โดยใช้ regex เพื่อจับคู่ข้อความของผู้ใช้กับ Intent ที่เกี่ยวข้อง
    - ใช้ re.compile() สร้างแพทเทิร์นเพื่อจับคู่คำใน Intent
    - ตัวอย่างโค้ด:

```
keywords = {'greet': ['hello', 'hi', 'hey'], 'goodbye': ['bye', 'farewell']} patterns = {intent: re.compile('|'.join(words)) for intent, words in keywords.items()}
```

หน้า 11-13: การสร้างแชทบอทง่าย ๆ

- แสดงวิธีสร้างแชทบอทที่สามารถตอบสนองข้อความจากผู้ใช้ได้ เช่น ทักทาย ลาก่อน หรือแสดงความขอบคุณ โดยใช้พจนานุกรมและฟังก์ชัน regex เพื่อจำแนก Intent และตอบกลับข้อความ

หน้า 14-15: การใช้ Regex เพื่อดึง Entity

- แสดงวิธีการใช้ regex เพื่อดึงชื่อออกจากประโยค เช่น "call me Ishmael"
    - ใช้ regex เพื่อค้นหาคำที่มีตัวพิมพ์ใหญ่และจับคู่กับ Entity เช่นชื่อของบุคคล
    - ตัวอย่างโค้ดการดึงชื่อด้วย regex

```python
name_keyword = re.compile(r'\b(name is|call me|people call me)\b', re.IGNORECASE)
name_pattern = re.compile(r'\b[A-Z][a-zA-Z]*(?:\s[A-Z][a-zA-Z]*)*\b')
```

หน้า 16-18: การสร้างผู้ช่วยเสมือน (Virtual Assistant)

- แนะนำวิธีสร้างผู้ช่วยเสมือนโดยใช้ไลบรารี pyttsx3 สำหรับการแปลงข้อความเป็นเสียงพูด และ webbrowser เพื่อเปิดเว็บไซต์ เช่น Google หรือ Wikipedia
    - ผู้ช่วยสามารถรับคำสั่งเกี่ยวกับเวลา วัน หรือเปิดเว็บต่างๆ ได้

การอธิบายของแต่ละหน้าจะแสดงแนวคิดหลักและการประยุกต์ใช้ regex ในการจำแนก Intent และดึงข้อมูล Entity

Chapter 6 Building Chatbots III

หน้า 1: แนะนำ Rasa

- **Rasa** เป็นเฟรมเวิร์กโอเพ่นซอร์สที่ใช้ในการสร้าง AI สนทนา เช่น แชทบอทและผู้ช่วยเสียง
- มันมีเครื่องมือสำหรับการทำความเข้าใจภาษาธรรมชาติ (NLU), การจัดการการสนทนา และการสร้างบทสนทนาที่ซับซ้อนและมีบริบท
- Rasa มีสองส่วนหลัก:
    - **Rasa NLU**: ทำหน้าที่ในการเข้าใจสิ่งที่ผู้ใช้ต้องการ โดยดึงข้อมูล **intent** (สิ่งที่ผู้ใช้ต้องการ) และ **entity** (ข้อมูลเฉพาะในข้อความ)
    - **Rasa Core**: จัดการการสนทนา โดยตัดสินใจว่าบอทควรทำอะไรหรือพูดอะไรต่อไป โดยใช้โมเดลการเรียนรู้ของเครื่องและกฎเกณฑ์

หน้า 2: ไฟล์หลักที่เกี่ยวข้องกับ Rasa

- **nlu.yml**: ไฟล์นี้ใช้กำหนดข้อมูลการฝึกสอนให้กับ Rasa NLU เพื่อให้บอทเรียนรู้การดึง intent และ entity จากข้อความผู้ใช้
- **rules.yml**: กำหนดกฎเฉพาะสำหรับเส้นทางการสนทนา ซึ่งจะช่วยบอกพฤติกรรมของบอทในบางสถานการณ์
- **stories.yml**: เก็บข้อมูลการฝึกสอนสำหรับ Rasa Core โดยบันทึกเรื่องราวของบทสนทนาต่าง ๆ
- **config.yml**: กำหนดการตั้งค่าสำหรับการทำงานของ Rasa NLU และการจัดการการสนทนาของ Rasa Core

- **credentials.yml**: กำหนดข้อมูลรับรองสำหรับการเชื่อมต่อกับแพลตฟอร์มภายนอก เช่น Slack, Facebook Messenger, Twilio หรือ webhook ที่กำหนดเอง
- **domain.yml**: เป็นไฟล์สำคัญที่กำหนด intent, entity, slot, response และ action ที่บอทสามารถทำได้
- **endpoints.yml**: กำหนดจุดเชื่อมต่อสำหรับบริการภายนอก เช่น Rasa action server หรือแพลตฟอร์มการส่งข้อความ
- **actions.py**: ไฟล์นี้เก็บคำสั่งเฉพาะที่บอทสามารถทำได้ เช่น การเรียกใช้ API หรือคำนวณบางอย่าง

หน้า 3: การตั้งค่าการฝึกสอน

- หน้านี้อธิบายถึงวิธีการกำหนดข้อมูลการฝึกสอน การตั้งกฎเฉพาะสำหรับเส้นทางการสนทนา และวิธีการสร้างเรื่องราวของการสนทนาที่ Rasa Core จะใช้ในการเรียนรู้

หน้า 4: การติดตั้ง Rasa

- คำสั่งการติดตั้ง Rasa: pip install rasa

หน้า 5: การสร้างโปรเจ็กต์ใหม่ใน Rasa

- ขั้นตอนในการสร้างโปรเจ็กต์ใหม่สำหรับ Rasa:
    - สร้างโฟลเดอร์ใหม่
    - ใช้คำสั่ง rasa init เพื่อเริ่มต้นโปรเจ็กต์
    - สามารถโต้ตอบกับบอทผ่าน terminal ด้วยคำสั่ง rasa shell
    - เรียกใช้เซิร์ฟเวอร์ Rasa ด้วยคำสั่ง rasa run

หน้า 6: การทดสอบบอท

- หน้านี้อธิบายถึงการทดสอบบอท โดยแนะนำให้เรียกใช้โปรเจ็กต์ Rasa และทดสอบการโต้ตอบระหว่างบอทและผู้ใช้

หน้า 7: คำถาม

- หน้าสุดท้ายเป็นสไลด์ที่สรุปเนื้อหาและเปิดโอกาสสำหรับคำถาม

บทนี้เน้นไปที่การใช้งาน Rasa สำหรับการสร้างแชทบอท โดยอธิบายไฟล์และการตั้งค่าที่สำคัญสำหรับการพัฒนาแชทบอท