Song of the Day!

Natural language Processing Project

จัดทำโดย

64050285 เอกภาพ สุขเกษม

64050543 ปริยากร ประมูลศิลป์

64050624 ยศพล ดิษฐปาน

เสนอ

อาจารย์ จักรพันธ์ เตไชยา

รายงานฉบับนี้เป็นส่วนหนึ่งของวิชา

SPECIAL TOPIC IN COMPUTER SCIENCE 1 รหัสวิชา 05506049

คณะวิทยาศาสตร์ สาขาวิทยาการคอมพิวเตอร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

Abstract

Introduction/motivation

ในวันที่เราเจอเรื่องดีๆจนอยากจะเล่าให้ใครสักคนฟัง ในวันที่เราเจอปัญหามากมายจนอยากระบายให้ใครสักคนปลอบใจ หรืออยากได้เพลงสักเพลงที่เข้ากับความรู้สึกเรา แต่ปัญหาคือเราจะระบายทุกอย่างให้ใครฟัง แล้วใครจะรับฟังเรา แล้วเราจะค้นหาเพลงที่เข้ากับความรู้สึกเราได้อย่างไร

เราจึงได้เลือกทำ โปรเจค Songs of the day! ขึ้นมาโดย Songs of the day! เป็น web application ที่มี input box ที่จะถามสั้น ๆ ว่า how was your day? ที่แปลว่าวันนี้เป็นอย่างไรบ้าง

ซึ่งสามารถพิมพ์บอกเล่าเรื่องราวสิ่งที่เราพบเจอมาและความรู้สึกในแต่ละวันเหมือนเล่าให้เพื่อนสักคนฟัง และ Songs of the day! จะเป็นเพื่อนที่จะคอยปลอบโยนเราด้วยเสียงเพลงโดยการแนะนำเพลงที่เหมาะสมหรือตรงกับอารมณ์และความรู้สึกให้เรา

Problem statement

ดนตรีมีบทบาทสำคัญและมีอิทธิพลต่ออารมณ์และอารมณ์ ในบางครั้งเราไม่สามารถค้นหาเพลงที่เข้ากับความรู้สึกของเราในเวลานั้นได้ จึงอาจจะต้องมีต้องมีตัวช่วยในการคัดเลือกเพลงที่เข้ากับความรู้สึกของเราในขณะนั้น จึงได้มีการสร้างโปรเจค Songs of the day! ขึ้น เพื่อแนะนำเพลงให้อย่างเหมาะสมและตรงกับความรู้สึก โดยใช้ Natural Language Processing ในการใช้แก้ปัญหาที่กล่าวไปข้างต้น

Summarized result of the project

web application ที่รับ input จาก user และแนะนำเพลงที่เข้ากับอารมณ์และความรู้สึกของ user

Contribution

จุดประสงค์หลักของ Project นี้คือการแนะนำเพลงที่เหมาะสมกับอารมณ์และความรู้สึกของ user โดยมีวิธีการดังนี้

1. Train Doc2Vec Model
2. ทำ vector representation ของเพลงและ input ของ user
3. หา Cosine similarity ที่มีมุมระหว่าง เพลงและ input ของ user น้อยที่สุด
4. แนะนำเพลงให้กับ User

Data section

ใช้ Taylor Swift Spotify Data โดยเลือกใช้เพียง เนื้อเพลงในการ Train Doc2Vec Model เท่านั้น และทำ vector representation ของเพลง เพื่อทำ Cosine similarity เทียบกับ vector representation ของ user มี data ทั้งหมดจำนวน 167 เพลง

Methodology

Overall Idea : สร้างโมเดลที่เกิดจากการเรียนรู้จากเนื้อเพลง เพื่อให้โมเดลคุ้นเคยกับเนื้อเพลงมากที่สุด แล้วใช้โมเดลนั้นเพื่อสร้าง vector representation ทั้งตัวเนื้อเพลงที่เอามาใช้เทรนเอง และ user input จากนั้นใช้ cosine similarity เพื่อหา vector ของเพลง และ user input ที่ใกล้เคียงกันมากที่สุดคือเพลงที่เราจะแนะนำ

How does the method works

1. Preprocess ทำการเตรียมเนื้อเพลง โดยหาเนื้อเพลงจาก google แล้วมานำมาใส่ใน csvจากนั้น เราจะนำเนื้อเพลงมาใส่ลงใน colab โดยจะต้องเอาตัวขึ้นบรรทัดใหม่ออก “/n” แล้วจากนั้นก็สร้าง dataframe มาเพื่อใช้สำหรับเก็บชื่อเพลง และเนื้อเพลงแล้วนำไปเทรนต่อไป

```
import pandas as pd
data = pd.read_csv('/content/drive/MyDrive/NLP_Project/spotify_dataset/spotify_taylorswift.csv',
                   usecols = ['lyrics', 'name'])
data['lyrics'] = data['lyrics'].astype(str)
remove_newline = lambda x: x.replace('\n', '')
data['lyrics'] = data['lyrics'].apply(remove_newline)
print(data)
```

จากนั้นก็เข้าสู่การ preprocess เริ่มจากการ tokenize โดยใช้ pretrain model จาก bert เพื่อตัดเนื้อเพลงออกมาเป็นคำ ๆ เมื่อตัดเสร็จจะทำการ lemmatize คำเหล่านั้น เพื่อเปลี่ยนให้เป็นรูป infinitive (verb 0) เช่น saw เป็น see เป็นต้น เมื่อได้คำในรูป infinitive แล้ว ก็จะเพิ่ม tag ให้แต่ละคำ เพื่อนำไปใช้ในการเทรนด้วย (ในตอนแรกที่เริ่มทำ กลุ่มของเราได้เลือกใช้ tag เฉพาะ noun, verb และ adjective แต่ด้วยข้อมูลที่นำมาเทรนยังไม่ได้เยอะมาก ผลลัพธ์เลยยังไม่ถึงจุดที่น่าพอใจ เลยตัดสินใจที่จะใช้ทุก tag ไปเลย) และทำการเอา stop word ต่าง ๆ ออก 2 ครั้ง โดยใช้จาก library gensim และ nltk เสร็จแล้วเราก็จะได้ข้อมูลที่พร้อมสำหรับสร้างโมเดล

```
tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')
data['tokenized'] = data.lyrics.apply(lambda x: tokenizer.tokenize(x))

lemmatizer = WordNetLemmatizer()
data['lemmatize'] = data['tokenized'].apply(lambda x: [lemmatizer.lemmatize(lyric) for lyric in x])

tagged_tokens=[]
for token in data['lemmatize'].to_list():
    tagged_tokens.append(nltk.pos_tag(token))

tag_list = []
for song in tagged_tokens:
    tag_list.append([(word,tag) for word, tag in song])
    #if tag.startswith('NN') or tag.startswith('VB') or tag.startswith('JJ')])
```

```
word_list = []
for song in tag_list:
    word_list.append([word for word, tag in song])

data['processed'] = [' '.join(song) for song in word_list]

#1st remove stopwords
data['processed'] = data['processed'].apply(remove_stopwords)

#2nd remove stopwords
stop = set(stopwords.words('english'))
def process(text):
    text = text.lower()
    # only preserve words that have three or more characters
    text_filt = re.findall(r'\b[a-zA-Z]{3,}\b', text)
    words_filt = [w for w in text_filt if w not in stop]
    return words_filt
data['dictionary'] = data['processed'].apply(process)

# Preprocess your data
processed_data = []
for i, song in enumerate(data["dictionary"]):
  processed_data.append(TaggedDocument(song, [i]))
```

2. Doc2vec หลังจาก preprocess ข้อมูลเรียบร้อยแล้ว เราจะนำข้อมูลที่เป็นเนื้อเพลงเหล่านั้นมาเทรนโมเดลโดยใช้โมเดล Doc2vec ขนาด 100 dimensions โดยเทรนทั้งหมด 20 epochs เสร็จแล้วเราจะนำโมเดลนั้นมาสร้าง vector representation ของเพลงทั้งหมดด้วย

```
# Initialize the Doc2Vec model
model = Doc2Vec(vector_size=100, window=5, min_count=1, workers=4, epochs=20)

# Build the vocabulary
model.build_vocab(processed_data)

# Train the model
model.train(processed_data, total_examples=model.corpus_count, epochs=model.epochs)

# Generate a vector representation for each song
song_vectors = [model.dv[i] for i in range(len(data))]
```

3. Cosine similarity จากนั้นเราจะรับ input และทำการ clean input เพื่อที่จะได้นำโมเดล มาสร้าง vector representation ได้ (input ที่จะใช้ต้องมีรูปแบบเหมือนกับที่นำเข้าไปเทรนในโมเดล) เมื่อ clean input เรียบร้อยแล้ว เราก็จะนำ vector representation ของทุกเพลง มาเทียบ (cosine similarity) กับ vector representation ของ user’s input และหา 5 อันดับที่ vector มีความใกล้เคียงกันมากที่สุด

```python
def input_def(input_text):

  # Preprocess the text and generate a vector representation
  input_text_words = tokenizer.tokenize(input_text.lower())

  stop_words = set(stopwords.words('english'))
  input_text_words = [word for word in input_text_words if not word in stop_words]

  input_text_vector = model.infer_vector(input_text_words)
  return input_text_words, input_text_vector

user_input = "Today I met my ex-boyfriend but he was standing with another girl, I
input_text_words, input_text_vector = input_def(user_input)
print(input_text_vector)
```

```python
from sklearn.metrics.pairwise import cosine_similarity

# Compute the cosine similarity between the new text and each song vector
similarities = cosine_similarity([input_text_vector], song_vectors)[0]

#Find the index of the most similar song
most_similar_index = similarities.argmax()
# print(most_similar_index)
# print(data.loc[most_similar_index])

#top 5
top_indices = similarities.argsort()[::-1][:5]
for index in top_indices:
    print(data.loc[index]['name'])
```

https://colab.research.google.com/drive/1P1aUtppdM2FJKOlWhqIu7K5O3gPSYBVL?usp=sharing (Colab)

Evaluation

1. **USE** เราได้ใช้ pretrain model เพื่อสร้าง vector representation ของทุกเพลง และ user's input เหมือนกัน เพื่อเปรียบเทียบระหว่าง pretrain model และ model ที่เราเทรนขึ้นมาเอง https://colab.research.google.com/drive/1lXbcM_sxVwEy37yB6SYRQhBCPmP1ALI-?usp=sharing (Colab)

```python
import numpy as np
# Load the Taylor Swift songs dataset
data = pd.read_csv('/content/drive/MyDrive/NLP_Project/spotify_dataset/spotify_taylorswift.csv',
                   usecols=['lyrics', 'name'])
data['lyrics'] = data['lyrics'].astype(str)
data['lyrics'] = data.lyrics.apply(lambda x: re.sub(r'\W+', ' ', x))

def input_def(text):
  # Preprocess the text and generate an embedding using the USE model
  input_embedding = model([text])[0]
  return input_embedding

user_input = "Today I met my ex-boyfriend but he was standing with another girl, I can't get over
input_embedding = input_def(user_input)

# Compute the cosine similarity between the new text embedding and each song lyrics embedding
song_lyrics_embeddings = model(data['lyrics'].tolist())
similarities = pd.Series(cosine_similarity([input_embedding], song_lyrics_embeddings)[0])

# Find the index of the most similar song
most_similar_index = similarities.idxmax()
top_indices = np.argsort(similarities)[::-1][:10]

# Print the name and lyrics of the most similar song
#print(data.loc[most_similar_index, 'name'])

# Print the top 5 most similar songs
print("Top 5 most similar songs:")
for i in top_indices:
    print(data.loc[i]['name'])
```

```python
import pandas as pd
import tensorflow_hub as hub
import re
from sklearn.metrics.pairwise import cosine_similarity

# Load the USE model
module_url = "https://tfhub.dev/google/universal-sentence-encoder/4"
model = hub.load(module_url)
```

2. **ROUGE Score** ใช้สำหรับนับจำนวนคำจาก user's input ที่พบในเนื้อเพลง โดยนับเป็น n-grams (ในที่นี้ใช้ 1-gram) กล่าวคือ จำนวนคำที่ match / จำนวนคำในเนื้อเพลงทั้งหมด

```
# Compute the word overlap score between the new text and the most similar song
song_words = set(data.loc[most_similar_index]["tokenized"])

# Define the stop words
#stop_words = set(stopwords.words('english'))

# Remove the stop words from the list
input_words = tokenizer.tokenize(user_input.lower())
input_words_set = set(input_words)

print(len(song_words.intersection(input_words_set)))
print(len(input_words_set))
word_overlap_score = len(song_words.intersection(input_words_set)) / len(input_words_set)
print("Word overlap score: ", word_overlap_score)
```

3. **Overlap Score** ใช้สำหรับหาจำนวนคำจาก user's input ว่ามีจำนวนกี่คำที่ไปพบในเนื้อเพลง กล่าวคือ จำนวนคำที่ match / จำนวนคำทั้งหมดใน user's input

```
# Compute the ROUGE score between the new text and the most similar song
rouge = Rouge()
song_text = ' '.join(data.loc[most_similar_index]["tokenized"])
rouge_score = rouge.get_scores(song_text, user_input)[0]['rouge-1']['f']
print("ROUGE score: ", rouge_score)
```

จากนั้นเราจะใช้ ROUGE และ Overlap Score จากทั้งโมเดลของเรา และ USE มาเปรียบเทียบกัน

Conclusion

Overall contribution : สำหรับ project ที่ทำในขณะนี้ยังไม่สามารถรองรับภาษาไทยได้ และยังไม่ได้มีความแม่นยำมากตามที่คาดหวัง(overlap ~32% from 500 sampling) แต่สามารถแนะนำเพลงได้ตามจุดประสงค์ และเมื่อนำไปเปรียบเทียบกับ pretrain model (USE) แล้ว ผลลัพธ์ที่ได้ไม่ได้ห่างกันมาก (± 10%) แม้โมเดลของเราไม่ได้เทรนด้วยข้อมูลที่มาก

490	Today was a pro	my tears ricoche	[12, 58]	0.2068965517	0.09944751056	Today Was A Fa	[15, 58]	0.2586206897	0.1137440729	0
491	Today was a cha	A Place in this W	[8, 53]	0.1509433962	0.067039103	Gorgeous	[13, 53]	0.2452830189	0.1052631551	0
492	Today was a lazy	False God	[10, 48]	0.2083333333	0.07954545157	Today Was A Fa	[14, 48]	0.2916666667	0.1067961139	0
493	Today was a soc	Dress	[14, 55]	0.2545454545	0.08938547171	It's Nice To Have	[18, 55]	0.3272727273	0.1244019111	0
494	Today was a lea	Welcome To Nev	[15, 48]	0.3125	0.1016949122	this is me trying	[16, 48]	0.3333333333	0.1062801906	0
495	Today was a gre	gold rush	[15, 75]	0.2	0.1276595709	It's Nice To Have	[20, 75]	0.2666666667	0.1467889876	0
496	Today was a bus	Out Of The Woo	[9, 57]	0.1578947368	0.07821228736	The Moment I Kı	[13, 57]	0.2280701754	0.1052631551	0
497	One of the most	Begin Again	[17, 85]	0.2	0.1280788136	Sad Beautiful Tri	[20, 85]	0.2352941176	0.154506434	0
498	I read a book ab	the last great am	[15, 70]	0.2142857143	0.1347150221	Sad Beautiful Tri	[21, 70]	0.3	0.1614349741	0
499	I heard a story a	Our Song	[16, 82]	0.1951219512	0.1407035136	Lover	[23, 82]	0.2804878049	0.1834061099	0
500	A love story that	Mine - POP Mix	[17, 91]	0.1868131868	0.1531100436	Sad Beautiful Tri	[23, 91]	0.2527472527	0.1841004145	0
501	I watched a mov	Death By A Thou	[18, 84]	0.2142857143	0.1319796915	Sad Beautiful Tri	[19, 84]	0.2261904762	0.123348014	0
502			Doc2vec	0.3213174365	0.08519234355		USE	0.3555695961	0.09152288073	12
503				Overlap	rouge score			Overlap	rouge score	Common song

https://docs.google.com/spreadsheets/d/13FdWEvA2WI28W03OWzIj8iLLSzY9VGEejLnTGvYCxpA/edit?usp=sharing (CSV)

Future work :

- ในอนาคตเราอยากจะทำให้สามารถแนะนำเพลงจำนวนได้มากขึ้น และหลากหลายแนว และศิลปินมากขึ้น
- พัฒนา Overlap score และ Performance ให้ดีมากขึ้น
- และเนื่องจากการใช้ Doc2vec ซึ่งเป็น unsupervised learning ทำให้ไม่ได้มีความแม่นยำมากนัก หากมีโอกาสได้พัฒนาต่อไปจึงอยากจะลองหาวิธีทำ supervised learning เพื่อให้แม่นยำมากขึ้น และ
- ทำ ภาษาไทย เพื่อให้คนไทยใช้
- ในเว็บไซต์เราได้ทำฟอร์มเพื่อให้ user กรอก feedback และคาดหวังว่าจะนำ feedback ในส่วนนี้ไปพัฒนาต่อไป

Website : https://macro-theater-387412.as.r.appspot.com/

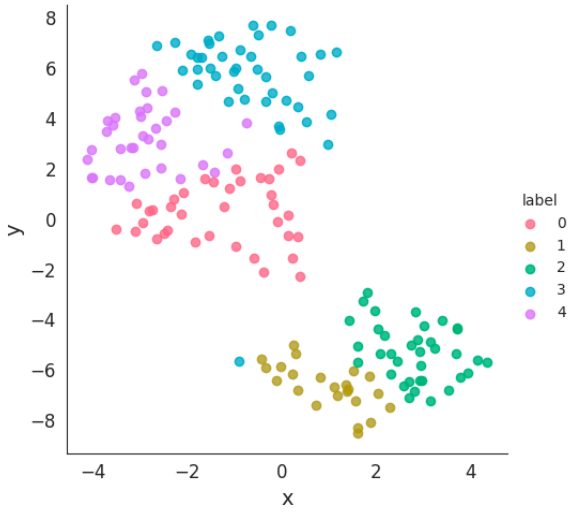
Our Journey

ในตอนแรกเราอยากจะทำ sentiment analysis เพื่อวิเคราะห์อารมณ์ของผู้ใช้เป็นคะแนนตั้งแต่ 0 - 1 และแนะนำเพลงให้เหมาะสมจากการเทียบกับ valance score ของ spotify track dataset ซึ่งมีค่า 0 - 1 ไล่จากความรู้สึกไม่ดีไปยังความรู้สึกดี แต่หลังจาก present งานไปในรอบแรกเกิดปัญหาว่าวิธีการทำมันค่อนข้างดิบไป และตัว spotify valance score ที่มีอยู่นั้นเราไม่ทราบ feature ว่าถูกแยกเป็น score จากอะไร และไม่ทราบว่าแต่ละ range ของ score มีความหมายว่าอย่างไร

เราจึงได้หาวิธีใหม่ โดยการนำ spotify track dataset มาทำ k-means clustering โดยใช้ feature danceability, acousticness, energy, tempo ในการแบ่ง cluster แล้วนำไปเทียบกับการทำ sentiment analysis ในฝั่ง user จากการทำ Logistic Regression model

เกิดปัญหาในฝั่งเพลง พบว่าเกิดปัญหาจากการที่ ใน spotify track dataset ซึ่งเป็น dataset ที่ใหญ่มากประกอบไปด้วยเพลง และไม่ใช่เพลง ไม่ว่าจะเป็น การบรรยายที่มีแค่เสียงพูด หรือเสียงลม เสียงฝน เพลงที่มีแค่เสียงดนตรี ซึ่งทำให้เมื่อแบ่ง cluster ออกมาแล้วไม่เป็นกลุ่มเท่าที่ควร

จึงได้แก้ปัญหาโดยการเปลี่ยน dataset โดยการนำเพียงเพลงของ Taylor Swift มาใช้ในการทำ cluster ซึ่งปัญหาที่พบก็คือไม่สามารถแบ่งเพลงตามความหมายได้ แบ่งได้แค่จังหวะ และแนวเพลงเท่านั้น ทำให้ไม่สามารถสื่อถึงอารมณ์ของเพลงตามเนื้อเพลงได้ เพลงที่ไม่ควรจะอยู่ด้วยกันก็อยู่ด้วยกัน เช่นเพลง I Knew You Were Trouble ที่มีเนื้อหาในแง่ลบในเชิงโกรธแค้น และเพลง Back to December ที่มีเนื้อหาเศร้าและคิดถึงแฟนเก่า กลับอยู่ใน cluster เดียวกัน



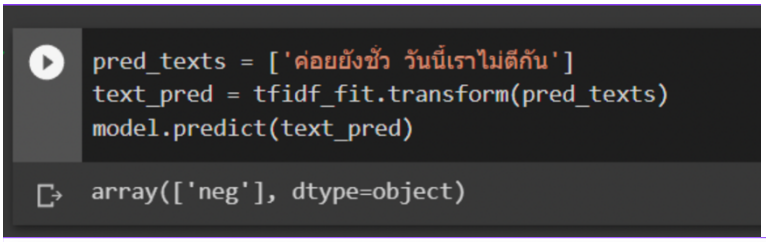

```
songs['label'] = y_kmeans

# shuffle dataset

songs = songs.sample(frac=1)
songs['label'].value_counts()
```

```
3    41
0    37
2    36
4    34
1    23
Name: label, dtype: int64
```

```
songs[songs['label'] == 3]
```

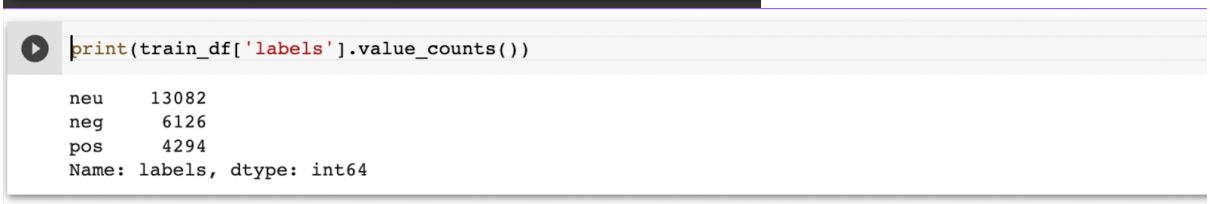
2	Teardrops On My Guitar - Radio Single Remix	0.621	0.28800	0.417	0.0231	3
84	So It Goes...	0.574	0.12200	0.610	0.0732	3
43	I Almost Do	0.567	0.01730	0.481	0.0270	3
40	I Knew You Were Trouble.	0.622	0.00454	0.469	0.0363	3
83	Look What You Made Me Do	0.766	0.20400	0.709	0.1230	3
107	Afterglow	0.756	0.13000	0.449	0.0344	3
102	Death By A Thousand Cuts	0.712	0.45400	0.732	0.0629	3
96	The Man	0.777	0.07670	0.658	0.0540	3
80	I Did Something Bad	0.696	0.06790	0.602	0.1590	3
89	Dress	0.719	0.03290	0.469	0.0533	3
54	Come Back...Be Here	0.483	0.00471	0.548	0.0254	3
59	Welcome To New York	0.789	0.03480	0.634	0.0323	3
18	Speak Now	0.708	0.10100	0.601	0.0306	3
52	Begin Again	0.530	0.19900	0.526	0.0263	3
81	Don't Blame Me	0.615	0.10600	0.534	0.0386	3
56	Treacherous - Original Demo Recording	0.828	0.17500	0.640	0.0355	3
45	Stay Stay Stay	0.729	0.30700	0.748	0.0245	3
17	Back To December	0.517	0.02020	0.606	0.0289	3

และในส่วนของ user's input sentiment analysis ก็เกิดปัญหาที่ dataset (PyThaiNLP Wisesight Sentiment) มีความเอนเอียงของ dataset มีค่า negative กับ neutral มากทำให้ทำ positive ออกมาได้อย่างไม่ดีนัก



```
pred_texts = ['ค่อยยังชั่ว วันนี้เราไม่ดีกัน']
text_pred = tfidf_fit.transform(pred_texts)
model.predict(text_pred)
```

```
array(['neg'], dtype=object)
```



```
print(train_df['labels'].value_counts())
```

```
neu    13082
neg     6126
pos     4294
Name: labels, dtype: int64
```

จึงได้แก้ปัญหาต่อโดยการเตรียม dataset เนื้อเพลงของ Taylor Swift แล้วนำมาทำ Doc2vec และนำ user's input ทำ doc2vec เช่นกัน หลังจากนั้นหา cosine similarity แล้วให้แนะนำเพลงที่มีความคล้ายคลึงที่สุดออกมา พบว่าสามารถแนะนำเพลงออกมาได้ค่อนข้างน่าพอใจในระดับหนึ่ง สามารถแนะนำเพลงได้ตามจุดประสงค์ที่ตั้งไว้ แต่ยังคงไม่แม่นยำอย่างที่ตั้งใจเอาไว้ แต่ทั้งนี้ทั้งนั้น ความแม่นยำขึ้นอยู่กับ input ว่ามีความ relate กับเนื้อเพลงใน dataset เพียงใด หาพูดถึงเรื่องอื่นเช่นเรื่องงาน เรื่องปัญหาสังคม ก็ไม่ตรง หากเป็นปัญหาเรื่องความรักหรือชีวิตประจำวันค่อนข้างตรง และขึ้นอยู่กับ input ของ user หาก input ของ user มีความยาวก็จะทำให้มีความแม่นยำมากขึ้น

ปัญหาต่อไปคือการทำเว็บไซต์ เริ่มต้นใช้ nodejs เพราะเนื่องจาก deploy ง่ายและมีประสบการณ์การใช้งาน หลังจากสร้างโปรแกรมไปจนถึงช่วงนำตัวโมเดลเข้ามาใช้ในโปรแกรม เกิดปัญหาไม่สามารถอ่านค่าของโมเดลได้ ตอนแรก donwload ออกมาเป็น model.h5 เพราะคิดว่า tensorflow จะรองรับ แต่ปรากฏว่าไม่ tensorflow รองรับเฉพาะ xxx.json file จึงได้มีการพยายามลอง donwload ออกมาเป็น model.json แทนแต่ก็ไม่สามารถใช้งานได้เนื่องจากไฟล์ถูก save ออกมาผิดรูปแบบ จึงได้มีการเปลี่ยนแปลงมาใช้ python ซึ่งมี pickle ที่สามารถนำ variable จากโปรแกรมอื่น(ซึ่งทางเราได้ดึง word2vec มาใช้)มาใช้ได้และเลือกใช้ flask framework เพราะเป็น framework ที่นิยมและมีตัวอย่างมากมายหลังจากที่กำหนดค่าต่าง ๆ แล้วลองรันเป็น local host ไม่มีปัญหาจึงทดลอง deploy โดยเราได้ deploy ใน google cloud โดยใช้ virtual machine ของ google ในการ deploy และสามารถ deploy ได้ปกติไม่มีปัญหาใด ๆ

Website : https://macro-theater-387412.as.r.appspot.com/