In [ ]:
```
stop_word = ['ho', 'chi', 'minh', 'saigon', 'hanoi', 'hcmc', 'nguyen', 'vietnam', 'hcm', 'vietnames', 'c
```

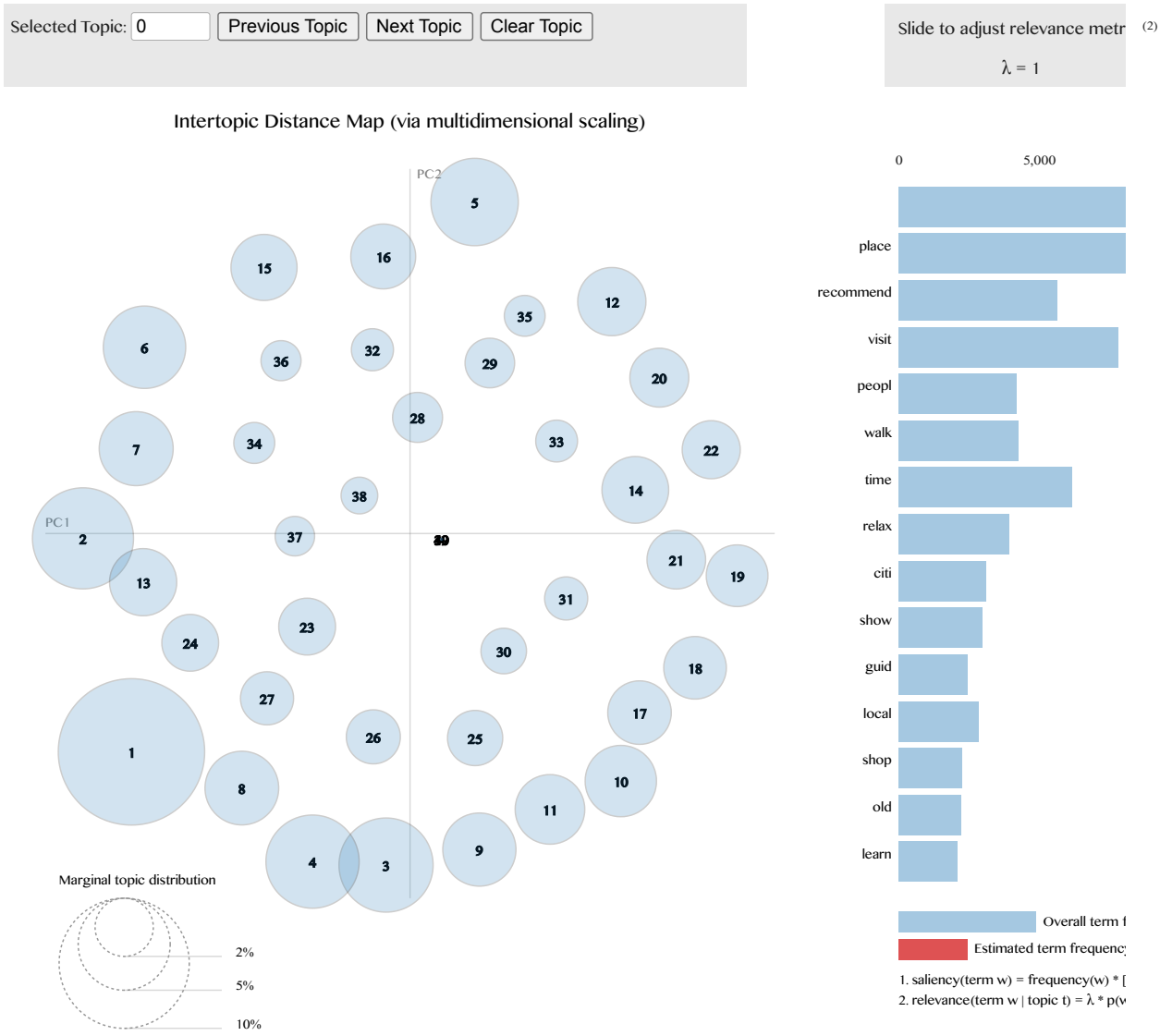
In [ ]:
```
df = df[df['DATE'] >= '2018-01-01']
```

Using gensim

In [ ]:
```
import gensim

preprocessed_texts = df['text_stm']

preprocessed_texts = preprocessed_texts[~preprocessed_texts.isna()]

# Convert string representation of list to actual list of tokens
data_words = [ast.literal_eval(text) for text in preprocessed_texts]

# Step 2: Create a dictionary
id2word = corpora.Dictionary(data_words)

stop_words = stop_word

unwanted_word_ids = [id2word.token2id[word] for word in stop_words if word in id2word.token2id]
id2word.filter_tokens(bad_ids=unwanted_word_ids)

# Step 3: Convert text data to bag-of-words format
corpus = [id2word.doc2bow(text) for text in data_words]

# Step 4: Save corpus and dictionary to disk
corpora.MmCorpus.serialize('corpus.mm', corpus)
id2word.save('dictionary.gensim')

lda_model = gensim.models.ldamodel.LdaModel(corpus=corpus,
                                            id2word=id2word,
                                            num_topics=40,
                                            random_state=100,
                                            update_every=1,
                                            chunksize=100,
                                            passes=10,
                                            alpha='auto')
```

```
huggingface/tokenizers: The current process just got forked, after parallelism has already been used. Dis
abling parallelism to avoid deadlocks...
To disable this warning, you can either:
        - Avoid using `tokenizers` before the fork if possible
        - Explicitly set the environment variable TOKENIZERS_PARALLELISM=(true | false)
huggingface/tokenizers: The current process just got forked, after parallelism has already been used. Dis
abling parallelism to avoid deadlocks...
To disable this warning, you can either:
        - Avoid using `tokenizers` before the fork if possible
        - Explicitly set the environment variable TOKENIZERS_PARALLELISM=(true | false)
huggingface/tokenizers: The current process just got forked, after parallelism has already been used. Dis
abling parallelism to avoid deadlocks...
To disable this warning, you can either:
        - Avoid using `tokenizers` before the fork if possible
        - Explicitly set the environment variable TOKENIZERS_PARALLELISM=(true | false)
huggingface/tokenizers: The current process just got forked, after parallelism has already been used. Dis
abling parallelism to avoid deadlocks...
To disable this warning, you can either:
        - Avoid using `tokenizers` before the fork if possible
        - Explicitly set the environment variable TOKENIZERS_PARALLELISM=(true | false)
```

Out[ ]:



In [ ]:

```
data = df['links_n_review'].tolist()
data_ready = data_words

# Function to extract just the dominant topics from LDA model
def extract_dominant_topics(ldamodel, corpus, texts):
    topic_list = []

    for row_list in ldamodel[corpus]:
        row = row_list[0] if ldamodel.per_word_topics else row_list
        row = sorted(row, key=lambda x: x[1], reverse=True)

        dominant_topic = row[0][0]
        topic_list.append(dominant_topic)

    topics_df = pd.Series(topic_list, name='topic')
    return topics_df

# Assuming lda_model, corpus, and data_ready are already defined
topics_df = extract_dominant_topics(ldamodel=lda_model, corpus=corpus, texts=data_ready)
topics_df = pd.DataFrame(topics_df)
topics_df.head()
```

Out[ ]:

	topic
0	11
1	37
2	27
3	27
4	31

In [ ]:
```
# Get the topics
num_topics = lda_model.num_topics
topics = lda_model.show_topics(num_topics=num_topics, num_words=4, formatted=False)

# Create a list to store each topic's information
topic_data = []

for topic_num, topic_terms in topics:
    words = [word for word, _ in topic_terms]
    topic_data.append({'topic': topic_num, 'Name': words})

# Create a DataFrame
topicn_name = pd.DataFrame(topic_data)

topicn_name
topicn_name.head()
```

Out[ ]:

	topic	Name
0	0	[absolut, last, stun, ca]
1	1	[old, fascin, collect, togeth]
2	2	[guid, say, bring, wall]
3	3	[price, pay, sell, person]
4	4	[attract, name, middl, strong]

Using Bertopic Lda model

In [ ]:
```
# Create a regular expression pattern to match the stopwords
stopwords = r'\b(?:' + '|'.join(stop_word) + r')\b'

# Remove the stopwords from the 'text' column
df['text_stm'] = df['text_stm'].str.replace(stopwords, '', regex=True)

df['text_stm'] = df['text_stm'].str.replace(" '',", '')
```

In [ ]:
```
from bertopic import BERTopic

#the pre-processed text
text = df['text_stm'].tolist()

topic_model = BERTopic(verbose=True)
topics, probs = topic_model.fit_transform(text)
```

```
Batches:   0%|          | 0/605 [00:00<?, ?it/s]
2023-09-04 19:21:04,091 - BERTopic - Transformed documents to Embeddings
2023-09-04 19:21:38,923 - BERTopic - Reduced dimensionality
```

```
huggingface/tokenizers: The current process just got forked, after parallelism has already been used. Dis
abling parallelism to avoid deadlocks...
To disable this warning, you can either:
	- Avoid using `tokenizers` before the fork if possible
	- Explicitly set the environment variable TOKENIZERS_PARALLELISM=(true | false)
huggingface/tokenizers: The current process just got forked, after parallelism has already been used. Dis
abling parallelism to avoid deadlocks...
To disable this warning, you can either:
	- Avoid using `tokenizers` before the fork if possible
	- Explicitly set the environment variable TOKENIZERS_PARALLELISM=(true | false)
huggingface/tokenizers: The current process just got forked, after parallelism has already been used. Dis
abling parallelism to avoid deadlocks...
To disable this warning, you can either:
	- Avoid using `tokenizers` before the fork if possible
	- Explicitly set the environment variable TOKENIZERS_PARALLELISM=(true | false)
huggingface/tokenizers: The current process just got forked, after parallelism has already been used. Dis
abling parallelism to avoid deadlocks...
To disable this warning, you can either:
	- Avoid using `tokenizers` before the fork if possible
	- Explicitly set the environment variable TOKENIZERS_PARALLELISM=(true | false)
2023-09-04 19:21:47,279 - BERTopic - Clustered reduced embeddings
```

In [ ]:

```
#topic_model to dataframe
topics_df = pd.DataFrame(topics, columns=['topic'])
topics_df
```

Out[ ]:

	topic
0	17
1	-1
2	-1
3	0
4	2
...	...
19331	51
19332	-1
19333	23
19334	1
19335	25

19336 rows × 1 columns

In [ ]:

```
#get the name of the topics
topic_info = topic_model.get_topic_info()
topicn_name = topic_info[['Name', 'Topic']]
topicn_name.rename(columns={'Topic': 'topic'}, inplace=True)
topicn_name
```

Out[ ]:

	Name	topic
0	-1_place_walk_visit_street	-1
1	0_museum_war_histori_exhibit	0
2	1_massag_spa_servic_staff	1
3	2_show_puppet_perform_opera	2
4	3_pagoda_buddha_ladi_statu	3
...	...	...
81	80_landmark_architectur_snobbi_clutter	80
82	81_holli_hear_mass_merri	81
83	82_friend_girlfriend_watcher_friendli	82
84	83_mountain_marbl_endem_showlov	83
85	84_delici_food_eat_pedenstrianis	84

86 rows × 2 columns

Dyanmic Topic Model (DTM)

In [ ]:
```
timeslices = df['DATE'].tolist()
```

In [ ]:
```
#merge topic_df with df
df = df.reset_index(drop=True)
topics_df = topics_df.reset_index(drop=True)
new = pd.concat([df, topics_df], axis=1)
new
```

Out[ ]:

	links_destination	links_page	links_n_contributions	links_n_review_title	links_n_review
0	Cai Rang Floating Market	Showing results 201-210 of 1,186	5 contributions	Best trip ever	https://www.tripadvisor.co.uk/ShowUserR
1	Cai Rang Floating Market	Showing results 201-210 of 1,186	9 contributions	Cai Rang floating market	https://www.tripadvisor.co.uk/ShowUserR
2	Cai Rang Floating Market	Showing results 201-210 of 1,186	1 contribution	Wonderful Experience	https://www.tripadvisor.co.uk/ShowUserR
3	Cai Rang Floating Market	Showing results 201-210 of 1,186	1 contribution	Just amazing!!!	https://www.tripadvisor.co.uk/ShowUserR
4	Cai Rang Floating Market	Showing results 201-210 of 1,186	135 contributions	Good market experience	https://www.tripadvisor.co.uk/ShowUserR
...	...	...	...	...	
19331	Ho Chi Minh City's Book Street	Showing results 391-397 of 397	66 contributions	Gorgeous laneway	https://www.tripadvisor.co.uk/ShowUserR
19332	Ho Chi Minh City's Book Street	Showing results 391-397 of 397	251 contributions	A lovely oasis around the corner from Notre Dame	https://www.tripadvisor.co.uk/ShowUserR
19333	Ho Chi Minh City's Book Street	Showing results 391-397 of 397	115 contributions	Worth a trip	https://www.tripadvisor.co.uk/ShowUserR
19334	Ho Chi Minh City's Book Street	Showing results 391-397 of 397	156 contributions	Keep calm & read a book	https://www.tripadvisor.co.uk/ShowUserR
19335	Ho Chi Minh City's Book Street	Showing results 391-397 of 397	24 contributions	Interesting and relaxing place	https://www.tripadvisor.co.uk/ShowUserR

19336 rows × 16 columns

In [ ]:
```
# Create an empty dictionary to store the data slices dataframes
time_topic_slices = {}

# Get unique time and topic values
unique_times = new['DATE'].unique()
unique_topics = new['topic'].unique()

# Split the DataFrame based on the time and topic slices
for t in unique_times:
    time_topic_slices[t] = {}
    df_time_filtered = new[new['DATE'] == t]

    for topic in unique_topics:
        df_time_topic_filtered = df_time_filtered[df_time_filtered['topic'] == topic]
```

```
        if not df_time_topic_filtered.empty:
            time_topic_slices[t][topic] = df_time_topic_filtered
```

In [ ]:
```
# Initialize an empty DataFrame to store the results
dtm = pd.DataFrame()

# Initialize a counter for the 'cluster' column
process_counter = 0

# Loop through the nested dictionary to access each DataFrame
for time_key, topic_dict in time_topic_slices.items():
    for topic_key, df in topic_dict.items():

        # Initialize an empty string to store the concatenated result
        concatenated_string = ''

        # Loop through the DataFrame to concatenate 'text_stm'
        for i in range(len(df)):
            concatenated_string += df.iloc[i]['text_stm'] + ' '

        # Remove the trailing space at the end
        concatenated_string = concatenated_string.rstrip()

        # Create a DataFrame to append
        data_to_append = pd.DataFrame({
            'date': [df['DATE'].iloc[0]],
            'keywords': [df['Keywords'].iloc[0] if 'Keywords' in df.columns else None],
            'string': [concatenated_string],
            'topic': [df['topic'].iloc[0] if 'topic' in df.columns else None],
            'cluster': [process_counter],
            'frequency': len(df)
        })

        # Append data to the empty DataFrame
        dtm = dtm.append(data_to_append, ignore_index=True)

        # Increment the counter
        process_counter += 1
```

In [ ]:
```
dtm
```

Out[ ]:

	date	keywords	string	topic	cluster	frequency
0	2023-06-01	None	['famili', 'book', 'trip', 'market', 'via', 'o...	17	0	4
1	2023-06-01	None	['awesom', 'place', 'spend', 'half', 'day', 's...	-1	1	153
2	2023-06-01	None	['will', 'spend', 'entranc', 'fee', 'plu', 'tr...	0	2	29
3	2023-06-01	None	['long', 'bridg', 'span', 'river', 'da', 'nang...	2	3	19
4	2023-06-01	None	['quit', 'big', 'area', 'mani', 'sceneri', 'ph...	1	4	20
...	...	...	...	...	...	...
2146	2022-02-01	None	['month', 'ago', 'went', 'place', 'nice', 'bea...	69	2146	1
2147	2022-02-01	None	['great', 'street', 'nice', 'stroll', 'see', '...	36	2147	1
2148	2021-05-01	None	['hanoi', 'old', 'quarter', 'good', 'place', '...	-1	2148	3
2149	2021-05-01	None	['good', 'place', 'discov', 'enjoy', 'walk', '...	3	2149	1
2150	2021-09-01	None	['museum', 'post', 'offic', 'great', 'histor',...	0	2150	1

2151 rows × 6 columns

In [ ]:
```
dtm_model = pd.merge(dtm, topicn_name, on='topic', how='left')
dtm_model
```

Out[ ]:

	date	keywords	string	topic	cluster	frequency	Name
0	2023-06-01	None	['famili', 'book', 'trip', 'market', 'via', 'o...	17	0	4	17_book_street_cafe_coffe
1	2023-06-01	None	['awesom', 'place', 'spend', 'half', 'day', 's...	-1	1	153	-1_place_walk_visit_street
2	2023-06-01	None	['will', 'spend', 'entranc', 'fee', 'plu', 'tr...	0	2	29	0_museum_war_histori_exhibit
3	2023-06-01	None	['long', 'bridg', 'span', 'river', 'da', 'nang...	2	3	19	2_show_puppet_perform_opera
4	2023-06-01	None	['quit', 'big', 'area', 'mani', 'sceneri', 'ph...	1	4	20	1_massag_spa_servic_staff
...	...	...	...	...	...	...	...
2146	2022-02-01	None	['month', 'ago', 'went', 'place', 'nice', 'bea...	69	2146	1	69_street_cross_pavement_road
2147	2022-02-01	None	['great', 'street', 'nice', 'stroll', 'see', '...	36	2147	1	36_bar_street_bui_vien
2148	2021-05-01	None	['hanoi', 'old', 'quarter', 'good', 'place', '...	-1	2148	3	-1_place_walk_visit_street
2149	2021-05-01	None	['good', 'place', 'discov', 'enjoy', 'walk', '...	3	2149	1	3_pagoda_buddha_ladi_statu
2150	2021-09-01	None	['museum', 'post', 'offic', 'great', 'histor',...	0	2150	1	0_museum_war_histori_exhibit

2151 rows × 7 columns

In [ ]:
```
from sklearn.feature_extraction.text import CountVectorizer, TfidfTransformer
import numpy as np

# Sample data
documents = dtm_model['string'].tolist()

# Define classes for the documents
clusters = dtm_model['cluster'].tolist()

# Calculate term frequency
vectorizer = CountVectorizer()
X = vectorizer.fit_transform(documents)

# Calculate CTF
cluster_counts = np.bincount(clusters)
ctf = np.zeros_like(X.toarray(), dtype=float)
for i, cluster_id in enumerate(clusters):
    ctf[cluster_id] += X.toarray()[i] / cluster_counts[cluster_id]

# Calculate IDF
transformer = TfidfTransformer(smooth_idf=True, use_idf=True)
transformer.fit(X)
idf = transformer.idf_

# Calculate CTF-IDF
ctf_idf = ctf * idf
```

In [ ]:
```
# Extract feature names (words) from the vectorizer
feature_names = vectorizer.get_feature_names_out()

# Function to get top N words for each class
def get_top_words(matrix, feature_names, top_n=10):
    top_words = {}
    for i, row in enumerate(matrix):
        top_indices = row.argsort()[-top_n:][::-1]
        top_words[i] = [feature_names[idx] for idx in top_indices]
    return top_words

# Get top 10 words for each class based on CTF-IDF values
top_words_per_class = get_top_words(ctf_idf, feature_names, top_n=10)
```

In [ ]:
```
#turn top_words_per_class to a dataframe
top_words_per_class = pd.DataFrame(top_words_per_class)
top_words_per_class = top_words_per_class.T

top_words_per_class['new_words'] = top_words_per_class[0] + ', ' + top_words_per_class[1] + ', ' + top_v
```

```
top_words_per_class = top_words_per_class.drop(columns=[0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
top_words_per_class
```

Out[ ]:

	new_words
0	trip, us, factori, breakfast, love, noodl, aft...
1	massag, recommend, experi, great, siri, cook, ...
2	massag, recommend, amaz, relax, war, excel, we...
3	lee, spa, bridg, also, experi, market, quarter...
4	massag, servic, excel, recommend, good, amaz, ...
...	...
2146	place, month, ago, activ, find, want, went, vi...
2147	great, see, stroll, chang, overal, street, muc...
2148	hanoi, autumn, squid, cake, pho, sad, caramen,...
2149	comeback, enjoy, discov, drive, run, lake, aro...
2150	post, offic, function, postcard, send, great, ...

2151 rows × 1 columns

In [ ]:
```
top_words_per_class['cluster'] = top_words_per_class.index
```

In [ ]:
```
dtm_model = pd.merge(dtm_model, top_words_per_class, on='cluster', how='left')
dtm_model
```

Out[ ]:

	date	keywords	string	topic	cluster	frequency	Name	new_words
0	2023-06-01	None	['famili', 'book', 'trip', 'market', 'via', 'o...	17	0	4	17_book_street_cafe_coffe	trip, us, factori, breakfast, love, noodl, aft...
1	2023-06-01	None	['awesom', 'place', 'spend', 'half', 'day', 's...	-1	1	153	-1_place_walk_visit_street	massag, recommend, experi, great, siri, cook, ...
2	2023-06-01	None	['will', 'spend', 'entranc', 'fee', 'plu', 'tr...	0	2	29	0_museum_war_histori_exhibit	massag, recommend, amaz, relax, war, excel, we...
3	2023-06-01	None	['long', 'bridg', 'span', 'river', 'da', 'nang...	2	3	19	2_show_puppet_perform_opera	lee, spa, bridg, also, experi, market, quarter...
4	2023-06-01	None	['quit', 'big', 'area', 'mani', 'sceneri', 'ph...	1	4	20	1_massag_spa_servic_staff	massag, servic, excel, recommend, good, amaz, ...
...	...	...	...	...	...	...	...	...
2146	2022-02-01	None	['month', 'ago', 'went', 'place', 'nice', 'bea...	69	2146	1	69_street_cross_pavement_road	place, month, ago, activ, find, want, went, vi...
2147	2022-02-01	None	['great', 'street', 'nice', 'stroll', 'see', '...	36	2147	1	36_bar_street_bui_vien	great, see, stroll, chang, overal, street, muc...
2148	2021-05-01	None	['hanoi', 'old', 'quarter', 'good', 'place', '...	-1	2148	3	-1_place_walk_visit_street	hanoi, autumn, squid, cake, pho, sad, caramen,...
2149	2021-05-01	None	['good', 'place', 'discov', 'enjoy', 'walk', '...	3	2149	1	3_pagoda_buddha_ladi_statu	comeback, enjoy, discov, drive, run, lake, aro...
2150	2021-09-01	None	['museum', 'post', 'offic', 'great', 'histor',...	0	2150	1	0_museum_war_histori_exhibit	post, offic, function, postcard, send, great, ...

2151 rows × 8 columns

In [ ]:
```
import plotly.express as px

dtm_model['Name'] = dtm_model['Name'].astype(str)


pio.templates.default = "plotly_white"
fig = px.area(dtm_model, x="date", y="frequency", color="Name", hover_name='new_words', line_group="Name
fig.update_traces(textposition="top right", textfont_color="black")

#whitegrid background

fig.update_layout(showlegend=True)

fig.show()
```

In [ ]:
```
```