Interpretation of documents

RC11
Student Number: 23086801

Path:
Input text - processed by Text Model - get associated words or similar tweets
Input text - processed by TheaterLine Model - get similar Theater Line
(Find the theatre scene corresponding to Theater)
Input image - processed by Theater Model - get similar Stage Scenes

These three searchers can become inputs and outputs for each other, either entering text to get an image or entering an image to get text

Summary Process

1 Input text - get associated words or similar tweets

Tweets Model

```
from text_model_class import TextModel
```

```
tweets_model = TextModel(r"C:\Users\SherryAi\MAGA.pkl", min_df=3)
```

```
tweets_model.get_key_words(tweets_model.vectorize('justice'), n=10)
```

```
['court',
 'justice',
 'law',
 'supreme',
 'judge',
 'stand',
 'two',
 'bcm',
 'mueller',
 'voting']
```

```
tweets_model.find_similar_items('women,femenism,equality,equity,rights,advocacy,fairness,activism,inclusion', top_n=9)
```

```
['BLACK LIVES MATTER Leader Arrested On Charges Of CHILD TRAFFICKING http://po.st/BUnu8M #TrumpsterWarriors #MAGA',
 "In honor of Black History month Brandon is giving out crack pipes in the name of racial equity. That's what he thinks is hel
ping black folk? It's what he thinks of black folks. Have you had enough yet? #AmericaFirst",
 'Illegal immigrants are getting better treatment than our veterans. People who were willing to put their life on the line vs
illegals who break the law We need to put America 1st & her citizens #ILLEGALimmigrants #illegals #BorderCrisis #maga #ULTRAMA
GA #patriot #AmericaFirst',
 'Life is so much brighter when we focus on what truly matters-Like put #AmericaFirst so @POTUS & We #MakeAmericaGreatAgain h/
t @CaliGirl11111',
 '#MAGA folks be like:',
 '#Trump not only sucked money from his #MAGA supporters, #Trump sucked the life out of the rest of us! #IndictArrestAndConvic
tTrump',
 '#MAGA Flowers loves #FREEDOM and @KWHKirkland stands for Freedom in Washington State! Come down to the Freedom Protest in #B
ellevue today to see the diverse crowd here to Protest for Freedom!',
 'I support Trump. #MAGA',
 "If these Black Lives Matter Protesters had Jobs in a good economy they wouldn't be available to George Soros $ #Trump #MAG
A"]
```

vectorization method _LSA (Latent Semantic Analysis) :

Dimensionality Reduction: LSA effectively reduces the number of features in a dataset, minimizing noise and capturing the most significant patterns, which facilitates subsequent machine learning processes.
Semantic Discovery: It reveals latent semantic structures within documents by capturing relationships among words, even if they do not co-occur in the same documents.
Improved Information Retrieval: By emphasizing semantic similarity over mere word matching, LSA enhances the quality of information retrieval, making it useful for applications like search engines and document clustering.

2 Input text - get similar Theater Line

Theater Line Model

```
from theaterline_model_class import SubtitleProcessor
```

```
merged_subtitles_path = r"D:\ntlive\merged_subtitles.pkl"
subtitle_processor = SubtitleProcessor(merged_subtitles_path)
query = '''One party believes women have equal rights. #MAGA doesn' t. Vote accordingly. ...
    @Joy_Villa is a #1 Billboard Artist, actress, human rights activist, a strong conservative and a proud member of the ...
    @NRA . You may have seen her at the Grammy Awards rocking a #MAGA dress. ...
    Joy understands that gun rights are women's rights. #CPAC2019 @CPAC . ...
    @HouseDemocrats are focused on delivering results #ForThePeople. It' s time @SenateMajLdr McConnell holds votes on our bills ...
    on healthcare, Rx prices, gun violence, equal rights for ALL Americans, ...
    #DrainTheSwamp & much more. No federal funds for schools that do not employ an equal amount ...
    of conservative instructors and administrators. #MAGA @realDonaldTrump Gun rights are civil rights. With the ...
    @guncollective 's Jon Patton on #AMERICAFirst @SalemMediaGrp'''

similar_results = subtitle_processor.find_similar(query)

for idx, (index, subtitle, score) in enumerate(similar_results):
    print(f"Rank {idx + 1}, Similarity Score: {score}")
    print(f"Index: {index}, Text: {' '.join(subtitle['text'])}")
    print(f"Start Time: {subtitle['start_time']}, End Time: {subtitle['end_time']}, file_name: {subtitle['file_name']}\n")
```

```
Rank 1, Similarity Score: 0.2013974979337379
Index: 27146, Text: I wan buy gun. Gun. You go give me? - Are you stupid? - Dey kill my husband. Mutumina.
Start Time: 01:09:17,800, End Time: 01:09:27,040, file_name: Three Sisters

Rank 2, Similarity Score: 0.18698038648020998
Index: 27308, Text: Madam, just one gun. Things will be better in the morning. I promise.
Start Time: 01:42:45,200, End Time: 01:42:56,320, file_name: Three Sisters

Rank 3, Similarity Score: 0.1838359792227731
Index: 18639, Text: She is no equal for his birth you may do the part of an honest man in it. How know you he loves her? I heard him swear
his affection.
Start Time: 00:29:15,670, End Time: 00:29:21,600, file_name: Much.Ado.About.Nothing.2011.01

Rank 4, Similarity Score: 0.17655006307929613
Index: 17423, Text: uh it was very exciting and and uh I think they they're so focused and they're working so hard on delivering the play
delivering the text delivering
Start Time: 00:02:33,080, End Time: 00:02:43,840, file_name: MacbethIntro

Rank 5, Similarity Score: 0.17609651237667978
```

```
similar_results
```

```
[(27146,
  {'index': 1353,
   'text': ['I wan buy gun.',
    'Gun. You go give me?',
    '- Are you stupid?',
    '- Dey kill my husband.',
    'Mutumina.'],
   'start_time': '01:09:17,800',
   'file_name': 'Three Sisters',
   'end_time': '01:09:27,040'},
  0.2013974979337379),
 (27308,
  {'index': 2001,
   'text': ['Madam,',
    'just one gun.',
    'Things will be better in the morning.',
    'I promise.']
```

vectorization method _TFIDF :

Relevance Highlighting: It emphasizes words that are frequent in a particular document but not common across all documents, helping to identify key terms that make documents unique.
Scalability: TF-IDF can be efficiently applied to large datasets of text, such as collections of subtitles.
Compatibility with Machine Learning: The numeric representation of text as TF-IDF vectors is ideal for feeding into various machine learning algorithms, particularly for tasks like similarity detection and classification.

3 (Find the theatre scene corresponding to Theater)

Theatre lines to corresponding theatre scenes

```
import os

folder_path = r"D:\ntlive\1"

if similar_results:

    matched_drama = similar_results[0][1]['file_name']
    print(matched_drama)
    mp4_name = f"{matched_drama}.mp4"

    mp4_path = os.path.join(folder_path, mp4_name)
    if os.path.exists(mp4_path):
        print(f"Found: {mp4_path}")
    else:
        print(f"No matching file found for: {mp4_name}")
```

```
Three Sisters
Found: D:\ntlive\1\Three Sisters.mp4
```

```
from moviepy.video.io.ffmpeg_tools import ffmpeg_extract_subclip
from moviepy.editor import VideoFileClip
from IPython.display import display, Video
import imageio

mp4_path = os.path.join(folder_path, mp4_name)

start_time_parts = similar_results[0][1]['start_time'].split(',')
start_seconds = int(start_time_parts[0].split(':')[-1]) + 60 * int(start_time_parts[0].split(':')[1]) + 3600 * int(start_time_parts[0].split('

end_time_parts = similar_results[0][1]['end_time'].split(',')
end_seconds = int(end_time_parts[0].split(':')[-1]) + 60 * int(end_time_parts[0].split(':')[1]) + 3600 * int(end_time_parts[0].split(':')[0])

print(start_seconds)
print(end_seconds)
```

Relational interpretation:

Stage scene design is often closely related to the lines, consistent with the mood of the characters, the plot, the atmosphere, so there is a correlation between the stage scene and the lines.

4 Input image - get similar Stage Scenes

Theater Model

```
import tensorflow as tf
import pickle
from theater_model_class import FilmProcessor

model = tf.keras.applications.MobileNet(
    input_shape=(224, 224, 3),
    include_top=False,
    pooling='avg'
)


processor = FilmProcessor(r"D:\sp\spanish stage play", model)
processor.test('q')
```

```
WARNING:tensorflow:From C:\Users\SherryAi\AppData\Roaming\Python\Python311\site-packages\keras\src\losses.py:2976: The name tf.losses.sparse
_softmax_cross_entropy is deprecated. Please use tf.compat.v1.losses.sparse_softmax_cross_entropy instead.

WARNING:tensorflow:From C:\Users\SherryAi\AppData\Roaming\Python\Python311\site-packages\keras\src\backend.py:1398: The name tf.executing_ea
gerly_outside_functions is deprecated. Please use tf.compat.v1.executing_eagerly_outside_functions instead.

WARNING:tensorflow:From C:\Users\SherryAi\AppData\Roaming\Python\Python311\site-packages\keras\src\layers\normalization\batch_normalization.
py:979: The name tf.nn.fused_batch_norm is deprecated. Please use tf.compat.v1.nn.fused_batch_norm instead.
```

```
'q'
```

```
queryFeatures = processor.processImage(r"C:\Users\SherryAi\Desktop\term3\1\222.png")
queryFeatures
```

```
1/1 [==============================] - 0s 353ms/step
```

```
array([0.09042002, 1.2170194 , 0.        , ..., 0.5831064 , 0.33640075,
```

```
matches = processor.findTopFramesByFeatures(queryFeatures, top_n=400)
```

```
from PIL import Image
for i, match in enumerate(matches):
    film_path, second, frame_path = match
    print(f"Top Match {i + 1} - Film: {film_path}, Second: {second}, Frame Image Path: {frame_path}")

    save_path = r'D:\sp\matched frames\{}'.format(os.path.basename(frame_path))
    Image.open(frame_path).save(save_path)
    print(f"Saved Matched Frame Image to: {save_path}")
```

```
Top Match 1 - Film: E:\sp\spanish stage play\TEATRO 'Llueve en Barcelona' (2009).mp4, Second: 840, Frame Image Path: Frames\TEATRO_Llueve_
en_Barcelona_2009.mp4_14.jpg
Saved Matched Frame Image to: D:\sp\matched frames\TEATRO_Llueve_en_Barcelona_2009.mp4_14.jpg
Top Match 2 - Film: E:\sp\spanish stage play\Los dramáticos orígenes de las galaxias espirales (CDN, 2016).mp4, Second: 660, Frame Image P
ath: Frames\Los_dramáticos_orígenes_de_las_galaxias_espirales_CDN_2016.mp4_11.jpg
Saved Matched Frame Image to: D:\sp\matched frames\Los_dramáticos_orígenes_de_las_galaxias_espirales_CDN_2016.mp4_11.jpg
Top Match 3 - Film: E:\sp\spanish stage play\d.juan@simetrico.es La burladora de Sevilla y el Tenorio del siglo XXI · Jesús Campos Teatro
A Teatro.mp4, Second: 3120, Frame Image Path: Frames\d.juansimetrico.es_La_burladora_de_Sevilla_y_el_Tenorio_del_siglo_XXI_Jesús_Campos_Te
atro_A_Teatro.mp4_52.jpg
Saved Matched Frame Image to: D:\sp\matched frames\d.juansimetrico.es_La_burladora_de_Sevilla_y_el_Tenorio_del_siglo_XXI_Jesús_Campos_Teat
ro_A_Teatro.mp4_52.jpg
Top Match 4 - Film: E:\sp\spanish stage play\Los dramáticos orígenes de las galaxias espirales (CDN, 2016).mp4, Second: 3540, Frame Image
Path: Frames\Los_dramáticos_orígenes_de_las_galaxias_espirales_CDN_2016.mp4_59.jpg
Saved Matched Frame Image to: D:\sp\matched frames\Los_dramáticos_orígenes_de_las_galaxias_espirales_CDN_2016.mp4_59.jpg
Top Match 5 - Film: E:\sp\spanish stage play\Las troyanas de Euripides Ernesto Alteiro 2017.mp4, Second: 1860, Frame Image Path: Frames\La
s_troyanas_de_Euripides_Ernesto_Alteiro_2017.mp4_31.jpg
Saved Matched Frame Image to: D:\sp\matched frames\Las_troyanas_de_Euripides_Ernesto_Alteiro_2017.mp4_31.jpg
Top Match 6 - Film: E:\sp\spanish stage play\TEATRO 'Llueve en Barcelona' (2009).mp4, Second: 2820, Frame Image Path: Frames\TEATRO_Llueve
_en_Barcelona_2009.mp4_47.jpg
```

vectorization method _MobileNet :

High-Level Feature Extraction: Leveraging a pre-trained deep learning model allows for the extraction of complex, high-level features from images, which are more effective for tasks such as classification, recognition, and similarity assessment compared to basic pixel data.
Efficiency in Processing: The use of TensorFlow and MobileNet provides an optimized, efficient way to process and analyze large volumes of visual data, making it scalable for extensive video datasets.
Accuracy in Similarity and Retrieval: The vectorized features facilitate accurate similarity measurements and retrieval tasks through methods like Euclidean distance, enabling precise matching and ranking of video frames based on visual content.

vectorization method comparing :

TF-IDF (Term Frequency-Inverse Document Frequency):
TF-IDF emphasizes words that are important to a specific document but not common across all documents. This method is straightforward and scales well to large datasets, making it useful for tasks where identifying unique or significant words is crucial. However, it doesn't account for the context or order of words.

Word2Vec:
This approach learns to represent words in a continuous vector space based on their context within sentences, capturing both the semantic and syntactic essence of words. Word2Vec requires a large amount of text to perform well and focuses on word-level patterns, making it less effective for getting direct insights at the document level.

Detailed process (including SOM)

1 Input text - get associated words or similar tweets

1.1 Functions

Load functions

```
def preprocess(paragraph):
    stop_words = set(stopwords.words("english"))
    lemmatizer = WordNetLemmatizer()

    words = gensim.utils.simple_preprocess(paragraph, min_len=3, deacc=True)
    processed_words = [lemmatizer.lemmatize(word) for word in words if word not in stop_words]
    processed_doc = " ".join(processed_words)
    return processed_doc
def load_list_from_file(file_name):
    try:
        with open(file_name, 'rb') as file:
            lst = pickle.load(file)
        return lst
    except Exception as e:
        print(f"An error occurred: {e}")
        return None
```

The Load function is responsible for loading the required files;

SOM functions

```
def calculateQE(SOM,data):
    sumSqDist = 0
    for d in data:
        g,h = find_BMU(SOM, d)
        v1 = SOM[g,h]
        v2 = d
        sumSqDist += scipy.spatial.distance.cdist([v1],[v2],'cosine')[0][0]
    QE = sumSqDist/len(data)
    return QE

def find_BMU_2(SOM,x):
    somShape = SOM.shape
    simSOM = SOM.reshape((-1, len(x)))
    cos_sims = cosine_similarity([x], simSOM).reshape(somShape[:2])
    return np.unravel_index(np.argpartition(cos_sims, -2, axis=None)[-2], cos_sims.shape)
```

The SOM function is responsible for the architecture of the subsequent SOM;

Visualization functions

```
def visualize_epochs_som(som_versions, epochs, file_name_base):
    """ Visualizes SOMs at each epoch directly in Jupyter Notebook without saving to files. """
    for epoch, som in enumerate(som_versions):
        plt.figure(figsize=(8, 8))
        u_matrix_values = u_matrix(som)  # Ensure that 'u_matrix' is called with the correct SOM state
        plt.imshow(u_matrix_values, cmap=custom_cmap, aspect='equal')
        plt.title(f'U-Matrix at Epoch {epoch + 1}')
        plt.colorbar(label='Distance')
        plt.xticks([])
        plt.yticks([])

        # Show the plot in Jupyter Notebook
        plt.show()

def visualize_som_with_words(SOM, pca_u_matrix, words_list, u_matrix_values, file_name, dpi=300):
    neuron_word_mapping = {}
    for i in range(SOM.shape[0]):
        for j in range(SOM.shape[1]):
            neuron_word_mapping[(i, j)] = []
```

The visualisation function is responsible for the subsequent visualisation of the SOM;

Search functions

```
def preprocess_text(text):
    """ Tokenizes, removes stopwords, and lemmatizes the input text. """
    tokens = word_tokenize(text.lower())
    tokens = [WordNetLemmatizer().lemmatize(token) for token in tokens if token.isalpha() and token not in stopwords.words('e
    return " ".join(tokens)

def vectorize_and_transform_text(input_text, word2vec_model, pca):
    # Tokenize and vectorize the input text using Word2Vec
    words = input_text.split()  # Simple split; consider more complex preprocessing
    word_vectors = [word2vec_model[word] for word in words if word in word2vec_model]

    # If no words found in the model, handle this case
    if not word_vectors:
        return np.zeros(pca.n_components_)

    # Average the vectors (or another method, like sum)
    average_vector = np.mean(word_vectors, axis=0).reshape(1, -1)

    # Apply PCA transformation
    transformed_vector = pca.transform(average_vector)

    return transformed_vector

def activate_and_visualize_SOM(SOM, input_vector, textual_content_mapping, title="SOM Activation"):
    # Ensure the input vector is properly shaped and normalized
    norm_input_vector = input_vector / np.linalg.norm(input_vector) if np.linalg.norm(input_vector) > 0 else input_vector

    # Reshape and normalize the SOM for cosine similarity calculation
```

The search function is responsible for entering words to find BMUs and texts near BMUs from the SOM;

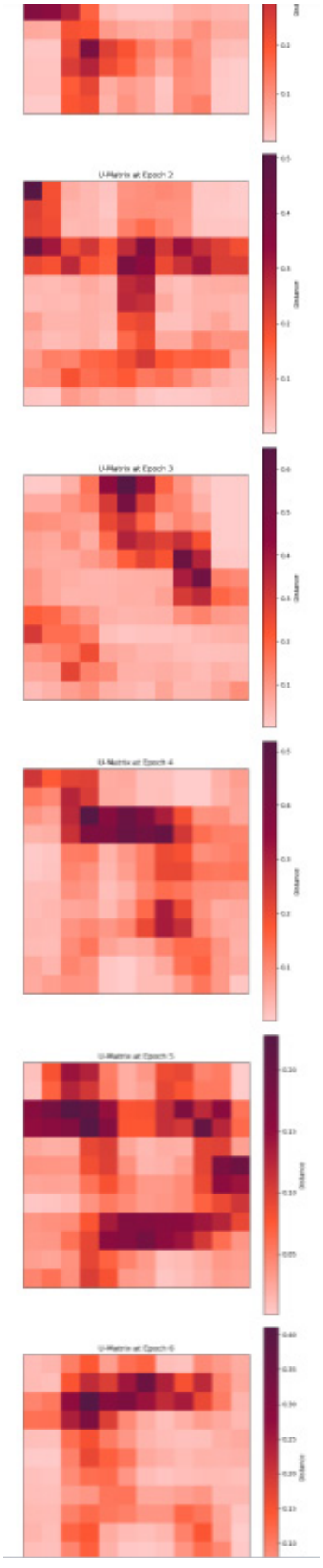
1.2 Load

Load and prepare

```
# Load data
MAGA = load_list_from_file(r"C:\Users\SherryAi\MAGA.pkl")
texts = [preprocess(text) for text in MAGA]
joined_text = " ".join(texts)

# Load Word2Vec model
model_path = r"E:\sp\GoogleNews-vectors-negative300.bin.gz"
model = KeyedVectors.load_word2vec_format(model_path, binary=True)

# Create TF-IDF model
vectorizer = TfidfVectorizer()
tfidf_matrix = vectorizer.fit_transform(texts)
feature_names = vectorizer.get_feature_names_out()

# Identify the top 1000 high-frequency words
high_frequency_1000_index = tfidf_matrix.toarray()[0].argsort()[-1000:]
words_list = [feature_names[i] for i in high_frequency_1000_index]

# Generate word vectors for these high-frequency words
word_vectors = [model[word] for word in words_list if word in model]
word_vectors_matrix = np.array(word_vectors)

# PCA to reduce dimensionality
pca = PCA(n_components=4)
pca_u_matrix = pca.fit_transform(word_vectors_matrix)
pca_u_matrix = normalize(pca_u_matrix, norm='l2')
```

This step loads the file and vectorises it.

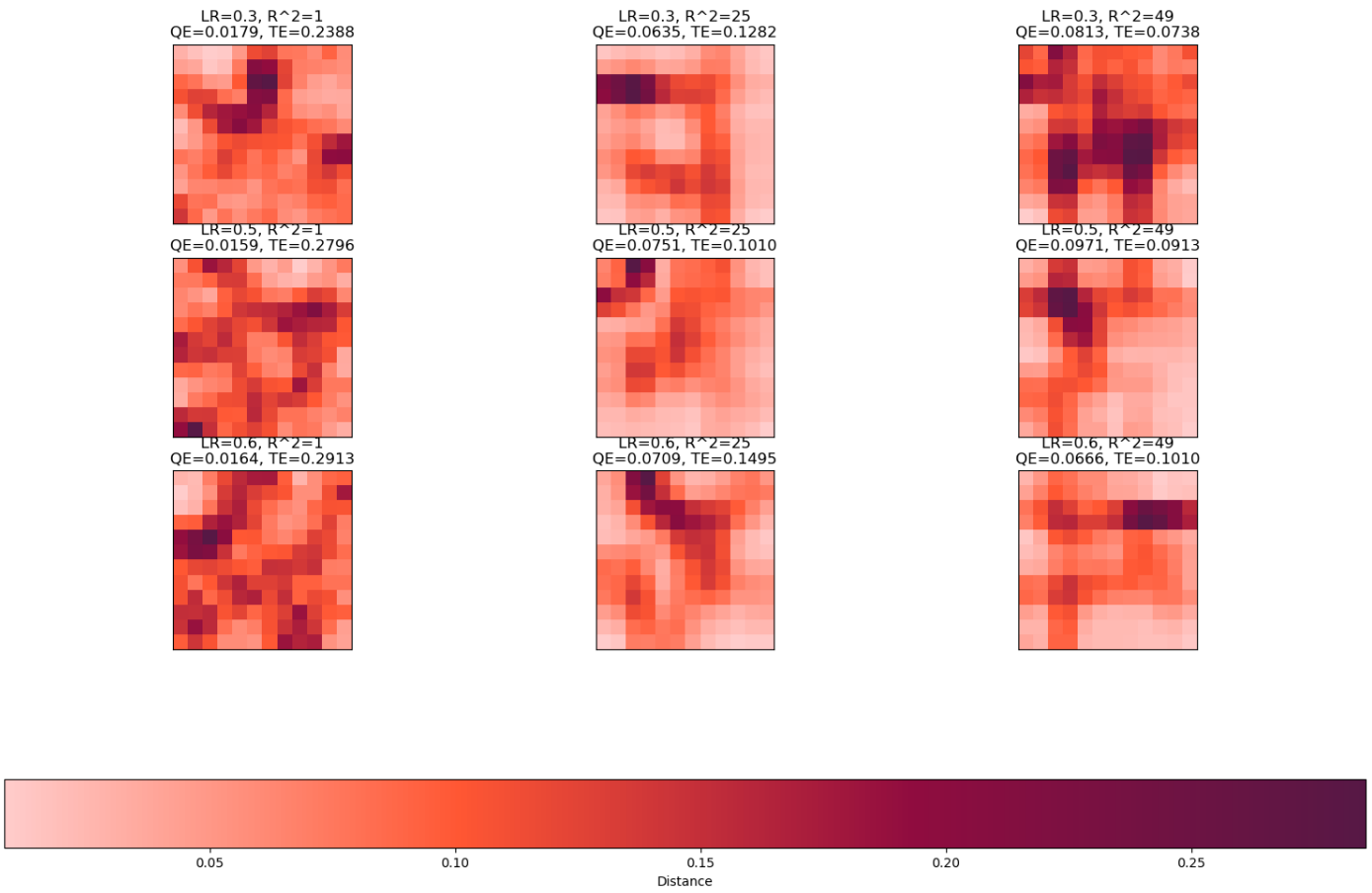
```
learn_rate = 0.5
radius = 5
epochs = 10
SOM = np.random.rand(12, 12, 4)
train_data = pca_u_matrix

best_som, best_errors, som_versions, neuron_texts = train_SOM(SOM, train_data, learn_rate, radius**2, 0.1, 0.1, epochs, texts)

visualize_epochs_som(som_versions, epochs, 'SOM_Visualization')
```



1.3 Visualize for different epochs

This step shows the SOM corresponding to different epochs

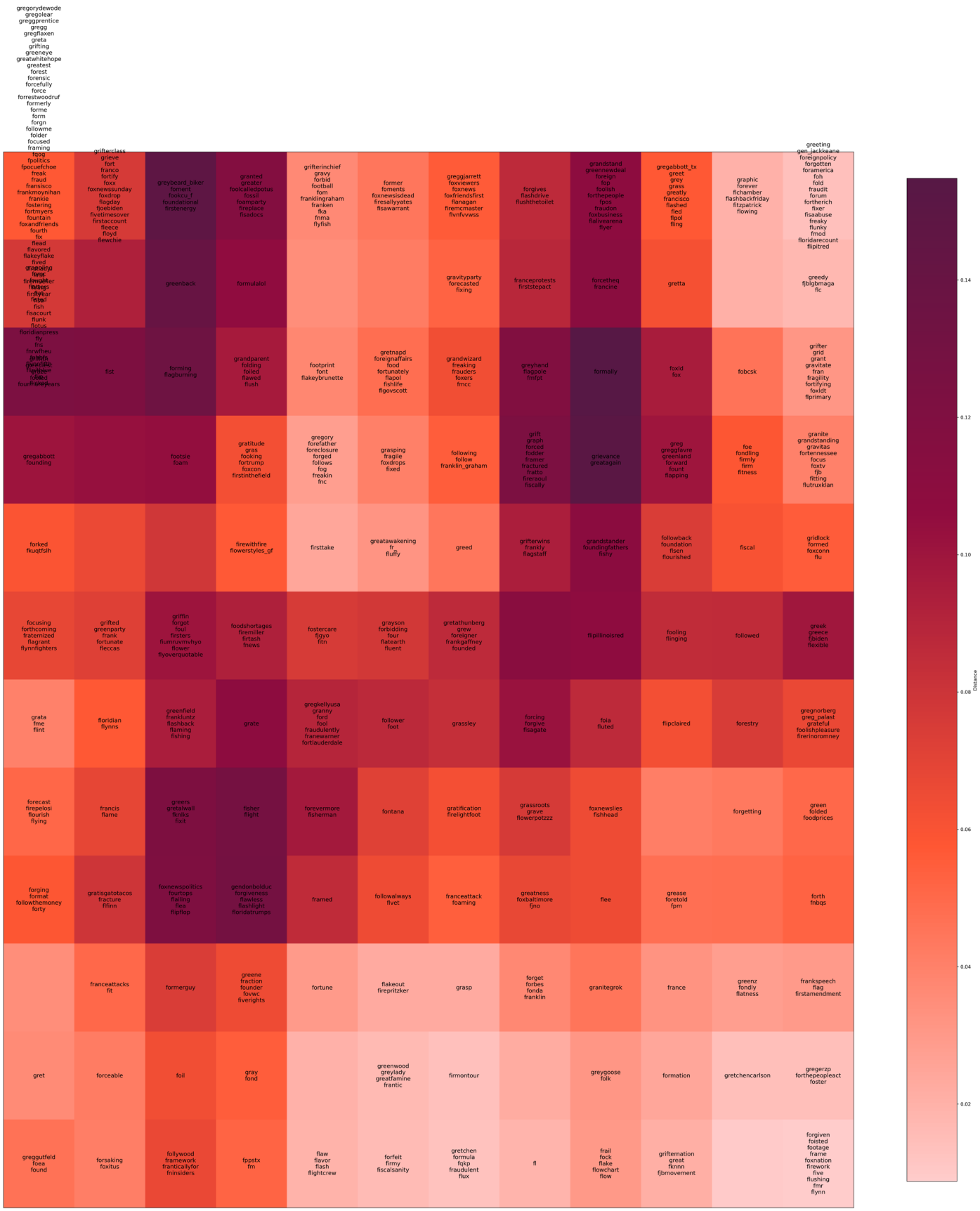
```
best_overall_som, best_overall_error = run_and_visualize_som_training(pca_u_matrix, [0.3, 0.5, 0.6], [1, 5, 7], 12, 12, 4, 20, 1
```



This step shows the SOMs corresponding to different learning rates and radii, since the learning rate and radius need to be smaller at the same time, I set the SOM with the smallest sum of learning rate and radius to be best_overall_SOM.

1.3 Visualize for different lr and r2

```
u_matrix_values = u_matrix(best_overall_som)
visualize_som_with_words(best_overall_som, pca_u_matrix, words_list, u_matrix_values, 'SOM_Visualization.png')
```

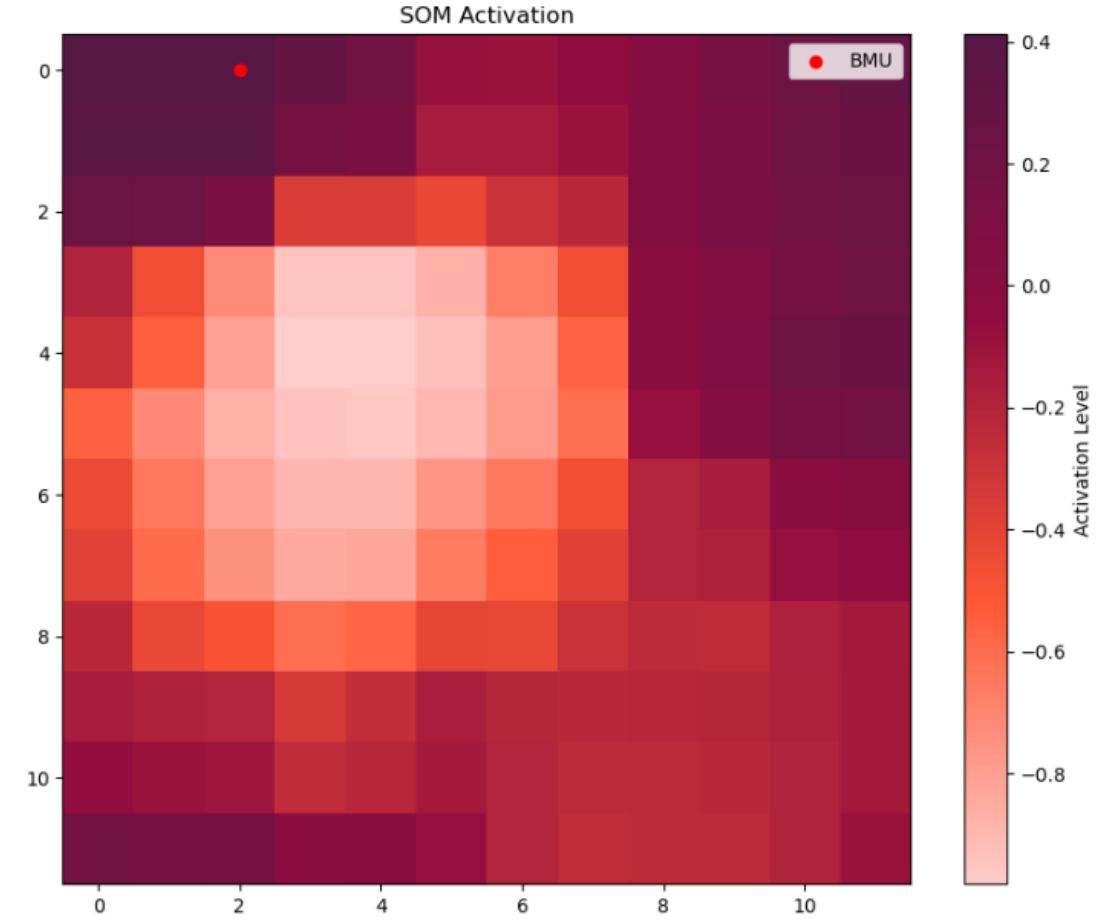


This step shows the text attached to the SOM when it is visualised.

1.3 Visualize with words

Search

```
# Train your SOM and receive the mapping of texts to each neuron
best_som, best_errors, som_versions, neuron_texts = train_SOM(SOM, train_data, learn_rate, radius**2, 0.1, 0.1, epochs, texts)

# Create textual content mapping from neuron_texts
textual_content_mapping = {}
for (i, j), texts in neuron_texts.items():
    textual_content_mapping[(i, j)] = ", ".join(texts) if texts else "No associated text"

# Prepare input text and transform it using previously defined vectorization and PCA transformation
input_text = "BLACK LIVES MATTER Leader Arrested On Charges Of CHILD TRAFFICKING http://po.st/BUnu8M #TrumpsterWarriors #MAGA"
input_vector = vectorize_and_transform_text(input_text, model, pca)

# Use the trained SOM and the input vector to find BMU and display associated text
BMU_index, BMU_content = activate_and_visualize_SOM(best_som, input_vector, textual_content_mapping)
print("BMU Index:", BMU_index)
print("BMU Content:", BMU_content)
```

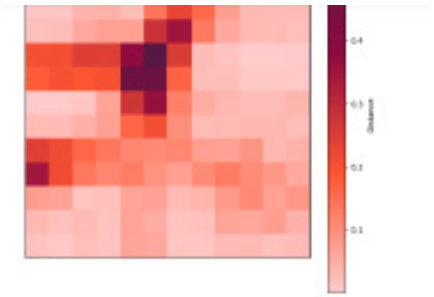


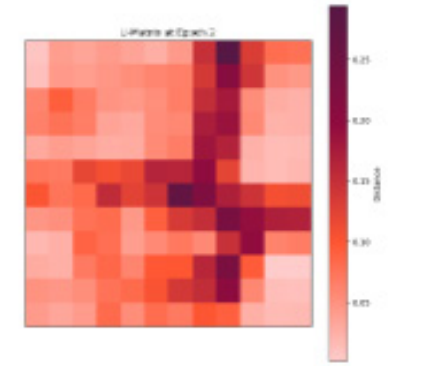

```
BMU Index: (0, 2)
BMU Content: skinny dip sea night belonging get robbed see advice maga guide http magalufevents com magaluf guide, absentee vo
ting tomorrow virginia also phone bank home tweet work teamtrump trumptrain maga, magaluf hottest new event label naughtythera
py kmedication magaluf maga maga, carly endorsing cruz newsworthy fat kid endorsing cake maga, anita conant speak heard stand
anita vote trump trump maga http bit snsgup, maga donald trump fought include jew black palm beach golf course http bit mrqa,
lovely picture pastor god bless hope wonderful easter family maga, trump burlington vermont teamtrump teamdonald trumpwinning
maga woman trump trumpvoter woman trump, ted going get schlonged maga trump http twitter com joshhollingsw status, want speak
front thousand trump supporter someday speak front thousand leatherwood supporter maga, conservative donald trump http youtube
com watch trumpac amykremer imwithtrump americafirst, wow look trump take iowa rest country trump force reckoned maga, donald
trump debate still getting medium coverage raising million war vet fundraiser maga, apply spent summer working abroad http wor
kmagaluf com start making thing happen maga ibiza, wish inform trump sen sasse msnbc defining conservatism http youtu cmi mlus
tcot teaparty maga sovereignty, celebrate tonight trumpfans cause back work tomorrow maga trump, nvcaucus trump win among ever
y demographic including latino stopped let make america great trump maga, leaving bcm maga, cruzcrew end united state nau love
usa cruz norubio alwaystrump votetrump maga http twitter com carolhaydu status, trump lot momentum feel rally today fantastic
know know world know maga, view illegals fed debt afford take world poor need help poor vet americafirst tcot, memory maga, ma
galuf hottest new event label naughtytherapy kmedication magaluf maga maga, watch wish america close choose another cartel mem
ber greed trump maga one last chance, sick elitsts moron protest vote oxymoron frankluntz foxnews tool landmanmarius trump mag
a, really dizappointe diminishes confidence people attack kind compassionate man maga, orlando rally asking start trending vet
trump maga alwaystrump realdonaldtrump, waiting june maga, yard sign game maga, either country realdonaldtrump america america
n great americafirst, let get video retweets trump trump trumptrain realdonaldtrump maga http twitter com dad_trump status, ne
w poll insideradv show trump double digit lead post debate yuge votetrump gapol maga, trump made right enemy vote trump votetr
umpfl votetrumpoh maga http bit rsbj, per math trump track win every state super tuesday beyond trump votetrump maga, thank tr
avis stand strong stopthesteal, legislature back session seems like good time assembly reform caucus meet since never drainthe
swamp, tag mate wont shut maga, gave money tedcruz realdonaldtrump jumped race choice clear america need run like business mag
a, audacity community organizer attack business man sketchy resume realdonaldtrump abc maga http com former chicago community
organizer attack multi billionaire businessman sketchy resume president, realdonaldtrump right candidate right time usa never
crisis vote trump maga http twitter com noopdoggy status, yes sen cruz month experience announced despised senate record worse
elite snob harvard maga trumptrain, sink every ship board time return israel americafirst http twitter com marylovefreedom sta
tus, absentee voting tomorrow virginia also phone bank home tweet work teamtrump trumptrain maga, american long stood silent a
merica keep spiraling downward speak mind time tcot maga, cruz rightwing rigid trump get latino dems rudy fully endorses teamt
rump trumptrain maga, love low key town hall end love love trumptrain maga http twitter com trump_world status, swedish bradst
er mrfumezz tbt maga sweden, great day maga isourturntomaga realdonaldtrump mitchellvii, tedcruz get financial backing egop es
tablishment bush http wapo maga alwaystrump, wdfx cruz dominionism explained rafael cruz preaches mountain dominionism creepyc
ruz maga trump, boom trump supporter maga, report mallorca gonna hottest summer record magaluf bring suncream maga magaluf, ex
cited huge victory realdonaldtrump huge turnout let trumptrain ladiesfortrump maga, ibiza weekender making buzzing maga shame
day away ibizaweekender maga, day maga, time get currency back gold standard shut fed endglobalism endthefed americafirst http
twitter com kennycollins status, america screwed long enough trump maga, well said scottbaio welcome aboard trump maga, trump
built one else stage come close politician gopdebate trump maga, see month maga intirga fergusresort, roll getting holiday boo
ked september maga tenerife majorca decision xxxemalouisexxx, mar tedcruz world would better place newyorkvalues realdonaldtru
mp maga trump http huffingtonpost com entry shirt back stranger subway nyc_, please listen america trillion debt trump bust st
eph barbmuenchen woman trump carl_c_icahn carminezozzora maga, please get vote trump maga trump trumptrain votetrumpnh, hard w
orking tax paying american need change washington career politician votetrump maga potus, yes senronjohnson wisconsin biz man
got politics reason realdonaldtrump maga wiprimary trump http twitter com glennfranco status, end nato build wall take oil tar
iff china realdonaldtrump americafirst, lyingted cruz campaign fundraiser recorded telling donor violate finance law http com
sketchy ted cruz campaign fundraiser recorded telling donor violate finance law realdonaldtrump maga, maga maga maga party htt
p twitter com patriotbygod status, watch trumpga opening video volunteer http youtube com watch kagvtnbv gagop gapol trumptrai
n trump votetrump maga trump, okay trump convinced votetrump maga, nothing medium say trump trustworthy use voice american vot
```

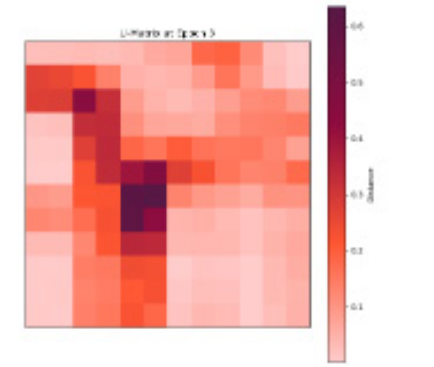
This step demonstrates a search function where text is entered and the most relevant text is obtained from BMU.

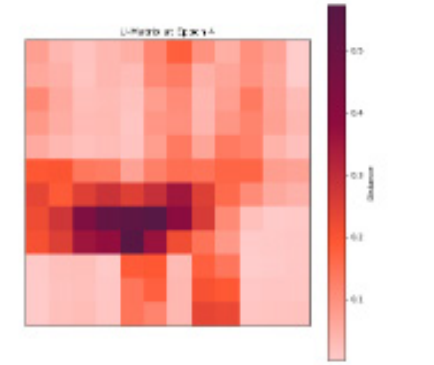
1.4 Search function

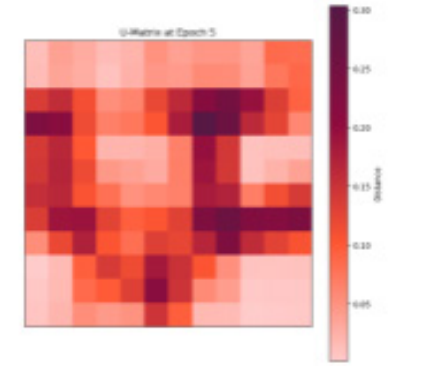
2 Input text - get similar Theater Line





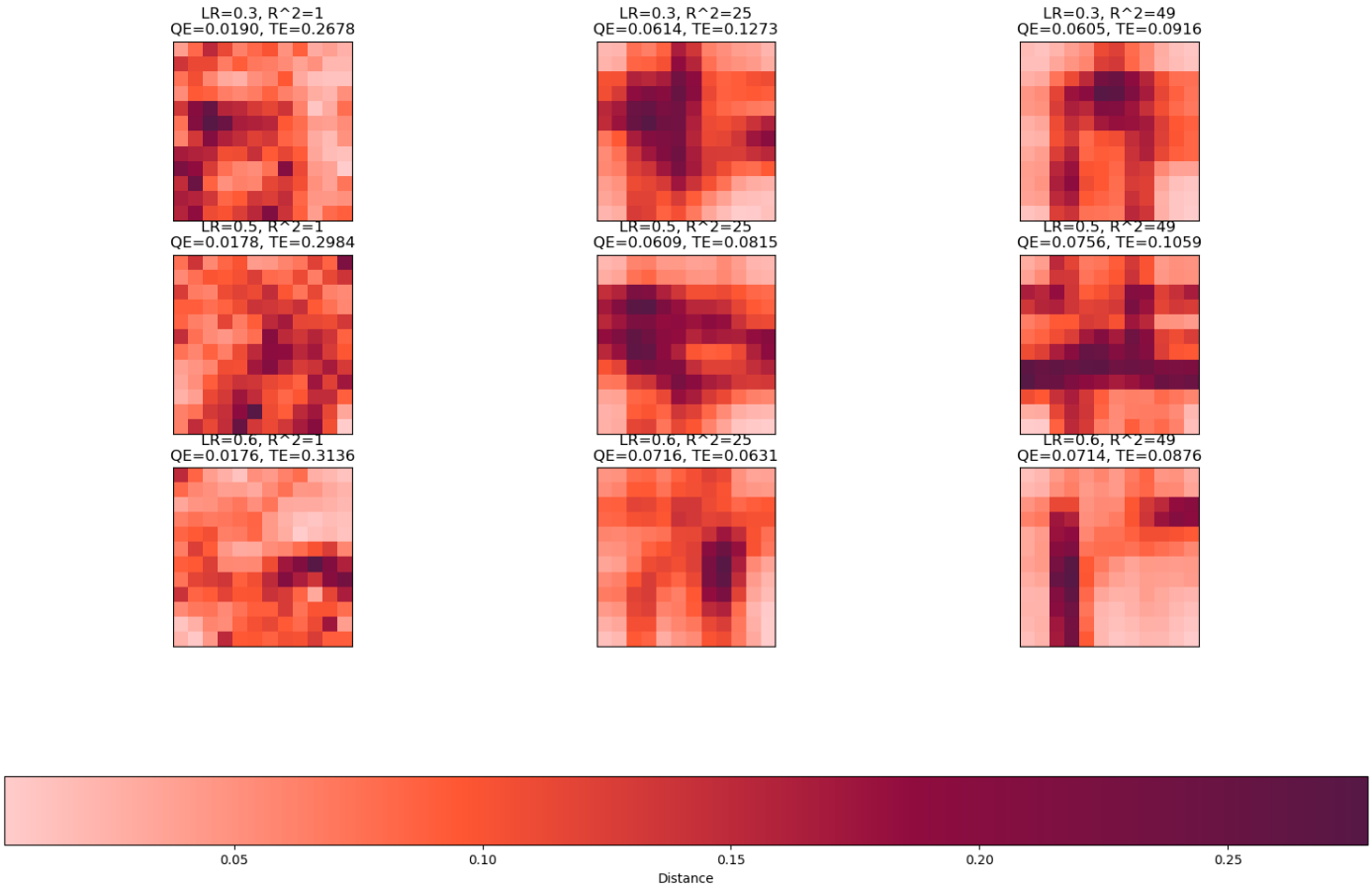




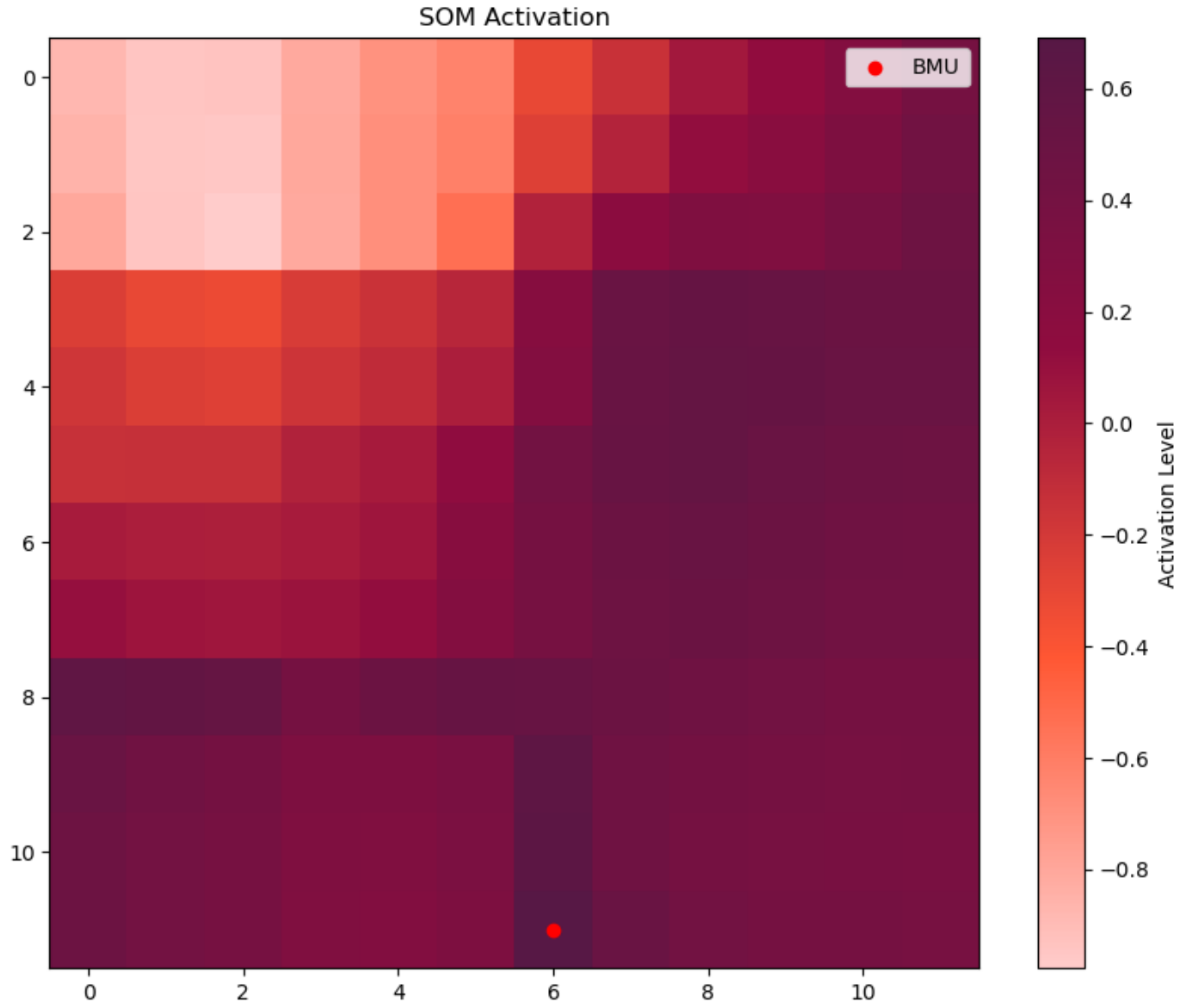


The process is exactly the same as on the previous two pages, so I won't repeat it again, but only show the results.

2.1 Visualize for different epochs

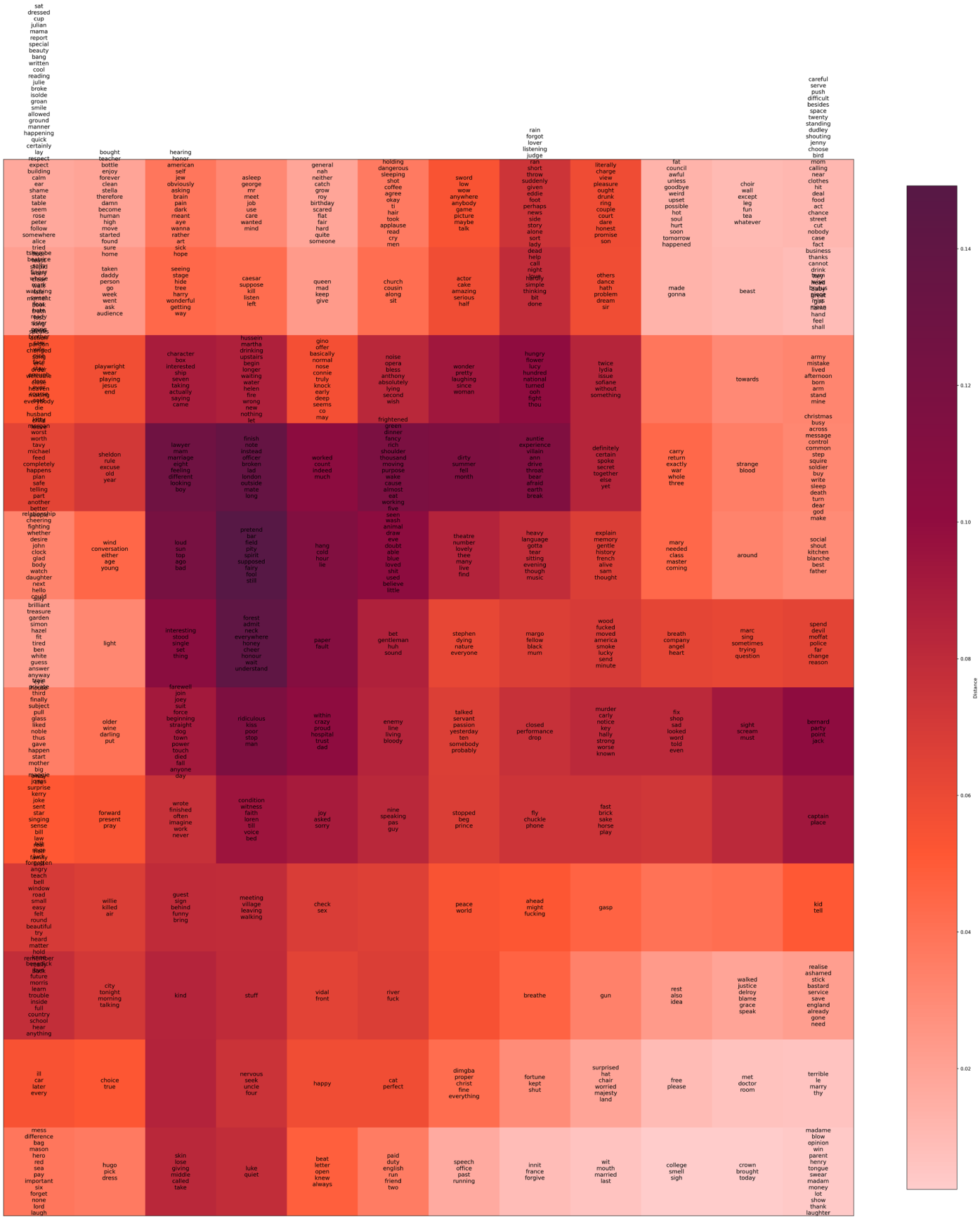


2.2 Visualize for different lr and r2



BMU Index: (11, 6)
BMU Content: eddie suppose somebody asks living mean ask, invitation auditorium imagine nobody ever done show, lot young people also older people lot people like art, horse town skinnier goat, yeah mean always making like remark like know, think theatre entertainment, man ride great machine man responsible man exists, think little bit eddie please crazy start work

2.3 Search function



2.4 Visualize with words

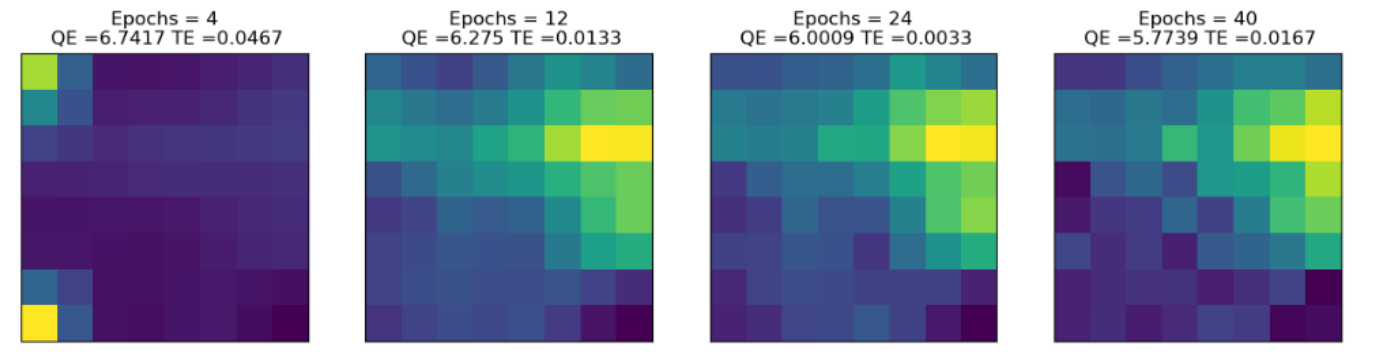
3 (Find the theatre scene corresponding to Theater)

No SOM was done for this step.

4 Input image - get similar Stage Scenes

4.1 Visualize for different epochs

```
fig, ax = plt.subplots(
    nrows=1, ncols=4, figsize=(15, 3.5),
    subplot_kw=dict(xticks=[], yticks=[]))
total_epochs = 0
SOMS = []
for epochs, i in zip([4, 8, 12, 16], range(0,4)):
    total_epochs += epochs
    SOM = train_SOM(SOM, n_train_data, learn_rate = .1, radius_sq = 2, epochs = epochs)
    SOMS.append(SOM)
    QE = round(calculateQE(SOM, n_train_data), 4)
    TE = round(calculateTE(SOM, n_train_data), 4)

    ax[i].imshow(u_matrix(SOM))
    ax[i].title.set_text('Epochs = ' + str(total_epochs) + '\n QE =' + str(QE) + ' TE =' + str(TE))
```



4.2 Visualize with pictures

```
# 读取并预处理图像
selected_image_filename = r"D:\sp\matched frames\TLA_-_Opera__Marina__Obra_Completa.mp4_38.jpg"
selected_image = Image.open(selected_image_filename)

# 将图像的特征提取出来
selected_image_features = processImage(selected_image_filename, model)

# 使用 activate 函数计算激活区域
activated_SOM = activate(selected_image_features, SOM, normalise(train_data, selected_image_features))

# 显示原始图像
plt.subplot(1, 3, 1)
plt.imshow(selected_image)
plt.title('Original Image')
plt.axis('off')

# 显示激活区域
plt.subplot(1, 3, 3)
plt.imshow(np.expand_dims(activated_SOM, axis=2), cmap='viridis')
plt.title('Activation Map')
plt.axis('off')

plt.show()
```

```
1/1 [==============================] - 0s 27ms/step
```

4.3 Search function

Search and Save function

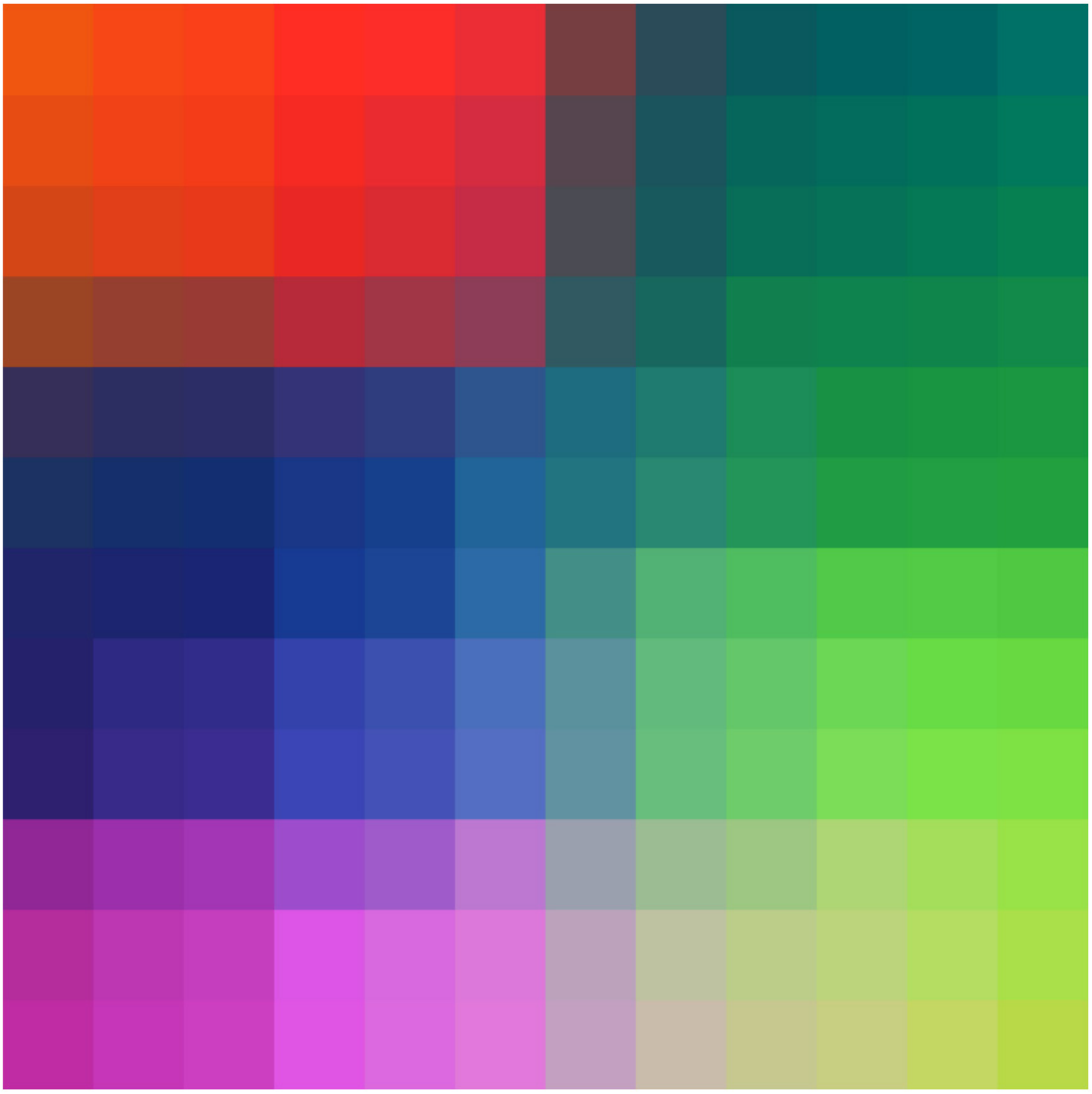
```
output_directory = r"D:\sp\11"

normalized_selected_features = normalise(train_data, selected_image_features)
bmu_index = find_BMU(SOM, normalized_selected_features)
neighbors = get_neighbors(bmu_index, som_height, som_width, radius=1)  # You can adjust radius as needed

neighbor_images = []
for (i, j) in neighbors:
    if SOMimages[i][j]:
        neighbor_images.extend(SOMimages[i][j])

fig, axes = plt.subplots(1, len(neighbor_images) + 1, figsize=(5 * (len(neighbor_images) + 1), 10))  # Adjusted for larger display
axes[0].imshow(selected_image)
axes[0].set_title('Original Image')
axes[0].axis('off')

original_image_save_path = os.path.join(output_directory, 'Original_Image.jpg')
selected_image.save(original_image_save_path)

for idx, fi in enumerate(neighbor_images):
    img_path = os.path.join(r"D:\sp\matched frames", fi['image'])
    img = Image.open(img_path)
    axes[idx + 1].imshow(img)
    axes[idx + 1].set_title(f"Neighbor {idx + 1}")
    axes[idx + 1].axis('off')

    # Save each neighbor image
    neighbor_image_save_path = os.path.join(output_directory, f"Neighbor_{idx + 1}.jpg")
    img.save(neighbor_image_save_path)

plt.show()
```

Enter an image, search for its corresponding BMU, and return twenty images on the BMU.

4.3 Search function

PCA

1 PCA on all cells



Fit PCA on All Cells of the SOM: In this method, you apply PCA directly to the feature vectors of all cells in the SOM grid. This approach treats each cell as an individual data point, calculating the principal components based on the variation observed directly in the SOM's representation of the data.
Fit PCA on the Entire Training Set, Then Apply to SOM Cells: Here, PCA is first fit on the entire dataset used to train the SOM. This captures the principal components that reflect the broad patterns and variance in the overall training data. The transformation defined by this PCA is then applied to each SOM cell to obtain their RGB values.

1 PCA on entire training