CSE4077- Recommender Systems

J Component – Project Report

VACATION RECOMMENDER BASED ON USER PRIORITIES

By

19MIA1077 Gaurav Trivedi

Integrated M. Tech CSE with Specialization Business Analytics

Submitted to

Dr.A.Bhuvaneswari,
Assistant Professor Senior,
SCOPE, VIT, Chennai

School of Computer Science and Engineering

Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

November 2022

BONAFIDE CERTIFICATE

Certified that this project report entitled "Vacation Recommender based on user inputs" is a bonafide work of Gaurav Trivedi, 19MIA1077

who carried out the J-component under my supervision and guidance. The contents of this Project work, in full or in parts, have neither been taken from any other source nor have been submitted to any other Institute or University for award of any degree or diploma and the same is certified

Dr.A.Bhuvaneswari,

Assistant Professor Senior,

SCOPE, VIT, Chennai

ABSTRACT

A good relaxing vacation is required for every individual. But when it comes to vacation planning, we usually spend maximum amount of time finalizing the location. In the process of finalizing the location, we follow a recommender model. We basically start with shortlisting features of each location like food, nightlife, Environment, Art & culture, history, etc. and then we prioritize this fields and find suitable location for vacation. This is nothing but a type of recommendation system. In this project, with help of techniques like web scrapping, natural language processing, Mongo DB and recommender system. I will create a full stack application which will take inputs or priorities of user and recommend a list of suitable destination for vacation.

ACKNOWLEDGEMENT

We wish to express our sincere thanks and deep sense of gratitude to our project guide, **Dr. A. Bhuvaneswari Assistant** Professor, School of Computer Science Engineering, for his consistent encouragement and valuable guidance offered to us in a pleasant manner throughout the course of the project work.

We express our thanks to our HOD Dr. Sivabalakrishanan for his support throughout the course of this project.

We also take this opportunity to thank all the faculty of the School for their support and their wisdom imparted to us throughout the course.

We thank our parents, family, and friends for bearing with us throughout the course of our project and for the opportunity they provided us in undergoing this course in such a prestigious institution.

Gaurav Trivedi

19MIA1077

Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

School of Computing Science and Engineering

VIT Chennai

Vandalur - Kelambakkam Road, Chennai - 600 127

FALL SEM 22-23

Worklet details

Programme	Integrated M. Tech CSE, Specialization BA	
Course Name / Code	CSE4077 Recommender Systems	
Slot	E1 + TE1	
Faculty Name	Dr. A. Bhuvaneswari	
Component	J – Component	
J Component Title	Vacation Recommender based on user priorities	
Team Members Name | Reg. No	Gaurav Trivedi	19MIA1077

Team Members(s) Contributions – Tentatively planned for implementation:

Worklet Tasks	*Contributor's Names*
Database connection and integration using Pymongo	Gaurav Trivedi
Preprocessing	
Model building	
Visualization	
Technical Report writing	
Presentation preparation	

TABLE OF CONTENTS

INTRODUCTION

In this project, we will be web scrapping data from famous tour guide column of New York Times called “36 hours in….”. In this column the writer spends 36 hours in a particular place and writes an article consisting of every detail about the tour. There are more than 180 different articles implies it has details about 180 different places. I will be using selenium and beautiful soup to extract the articles and store it in Mongo DB with using Pymongo. Once the data is stored, we will apply NLP techniques like TF-IDF (Term Frequency- Inverse document frequency) and NMF (Non-negative matrix factorization). We are using this technique as TF-IDF weights each word depending on its importance across a document and NMF clusters the words into a predefined number of topics based on these weights. But as usual we need a lot of NLP preprocessing which will detailed in upcoming sections. After the NLP state we have topics consisting of some words which represents a place most accurately. From this, I could then rank the prevalence of each topic across each of the articles. NMF provides a “score” for each topic among each article. Although the score itself is arbitrary, the higher the number more prevalent the topic. By normalizing these scores across each of the articles I was able to obtain a ranking of topics for each city!

By ranking each topic among the places, I will be able to match destinations with the personal rankings of what an individual prefers when looking for a vacation! This will be achieved by building a recommender which uses cosine similarity to match user with destination and return the recommendations.

LITERATURE SURVEY

Sl no	Title	Author / Journal name / Year	Technique	Result
1	A Multi-Level Tourism Destination Recommender System	Hend Alrasheed_, Arwa Alzeer, Arwa Alhowimel, Nora shameri, Aisha Althyabi Elsevier 2020	Multi-level recommender using web scrapping and user-user,user-destination recommendation	
2	A novel tourism recommender system in the context of social commerce	*Leila Esmaeili , Shahla Mardani , Seyyed Alireza Hashemi Golpayegani , Zeinab Zanganeh Madar* Expert Systems with Applications 2020	proposed a social-hybrid recommender system based on trust, reputation, similarity, and social communities. The proposed system contains five main components that can be tailored to the type of application, and personalize based on the needs.	Precision – 91.64% Recall – 69.35% F measure 78.95
3	A recommender system for tourism industry using cluster ensemble and prediction machine learning techniques	Mehrbakhsh Nilashi, Karamollah Bagherifard , Mohsen Rahmani, Vahid Rafe. Computers & Industrial Engineering - 2017	ANFIS and SVR as prediction techniques. Principal Component Analysis (PCA) as a dimensionality reduction technique. Self-Organizing Map (SOM) and Expectation Maximization (EM) as two well-known clustering techniques. The main work was the use of clustering ensemble method in the recommendation model.	Precision – 94.2% F1 – 0.932 MAE – 0.761

4	Detection of tourists attraction points using Instagram profile	Ksenia D. Mukhina, Stepan V. Rakitin, Alexander A. Visheratin International Conference on Computational Science, ICCS 2017, 12-14 June 2017, Zurich, Switzerland	presented an extension for tourist and locals' detection from social media data by using time frames. This approach allows to identify the person without complete information and full profile. Saint Petersburg was used as an example: top places for visitors and residents were compared and the results showed that tourists and locals have different activity trends during the year.	Graphs were plotted to show the trend of customers or tourists in the location based on predictions.
5	Intelligent tourism recommender systems: A survey	Joan Borrαs, Antonio Moreno, Aida Valls Expert Systems with Applications 2014	Content – Based recommender system which uses data like location, social media posts, tweets, etc. to identify key tags for a particular geo-spatial location and match it with user requirements to recommend palces	An accurate application which can show nearby famous locations to the user based on his/her location.
6	On the design of individual and group recommender systems for tourism	Inma Garcia, Laura Sebastia, Eva Onaindia	Grouping the users based on interests, passing their demographic, content-based preferences and general likes to item selector which will produce demographic recommendation, content-based recommendation and general liking-based recommendations, which will be passed to hybrid technique model and final recommendations will be generated.	Precision – 90.0% Recall – 0.30 F value – 0.59

PROBLEM STATEMENT

Most of the vacation recommender systems works on tourists' comments and ratings, but these models lack the essence of destination. Most of the travelers visits a destination based on user recommendations without analyzing the features of the destination. This sometimes results in great experience but sometimes the tourists' expectations are not meet.

In order to avoid this a recommender model is needed which can justify the user requirements, the mood of tourists, in order to recommend destinations which can provide a comforting and relaxing vacation.

OBJECTIVES

In this project, I am aiming at developing a recommender model which can take user moods as input and map it to destinations which are similar or can provide a whole some experience.

In order to develop the model, we will be using LF with TFIDF, LMF with TFIDF and NMF with count Vectorizer and find the best algorithm which can describe the destinations. Later, user inputs will be compared with feature vectors of destination and similar locations using cosine similarity will be given as output.

TOOLS UTILIZED

1. Python
   Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built-in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

2. Mongo DB
   MongoDB stores data in flexible, JSON-like documents, meaning fields can vary from document to document and data structure can be changed over time. The document model maps to the objects in your application code, making data easy to work with Ad hoc queries, indexing, and real time aggregation provide powerful ways to access and analyse your data. MongoDB is a distributed database at its core, so high availability, horizontal scaling, and geographic distribution are built in and easy to use. MongoDB is free to use.

DATABASE

The dataset will be web scrapped from "36 hours in…" column from New York Times using Selenium and beautiful soup.

ALGORITHMS AND TECHNIQUES USED

1. Web Scrapping
   In order to retrieve the articles, we will write a python script which will retrieve the articles and store in a vector as [destination name, country name and articles.

2. MongoDB client
   In order to store the articles in Mongo DB, we will use python library, Pymongo which will help us to create a mongo DB client and store our articles with unique id, destination name, country name and article. Later we will retrieve these articles for our data cleaning and model building.

3. Natural Language Processing
   In order to clean the data and to generate the key features from the article we will use NLP techniques like stemming and lemmatization. In order to use NLP, we will use python library NLTK.

4. Count Vectorizer
   Machines cannot understand characters and words. So, when dealing with text data we need to represent it in numbers to be understood by the machine. Count vectorizer is a method to convert text to numerical data.

5. TF-IDF
   TF-IDF stands for “Term Frequency — Inverse Document Frequency”. This is a technique to quantify words in a set of documents. We generally compute a score for each word to signify its importance in the document and corpus.

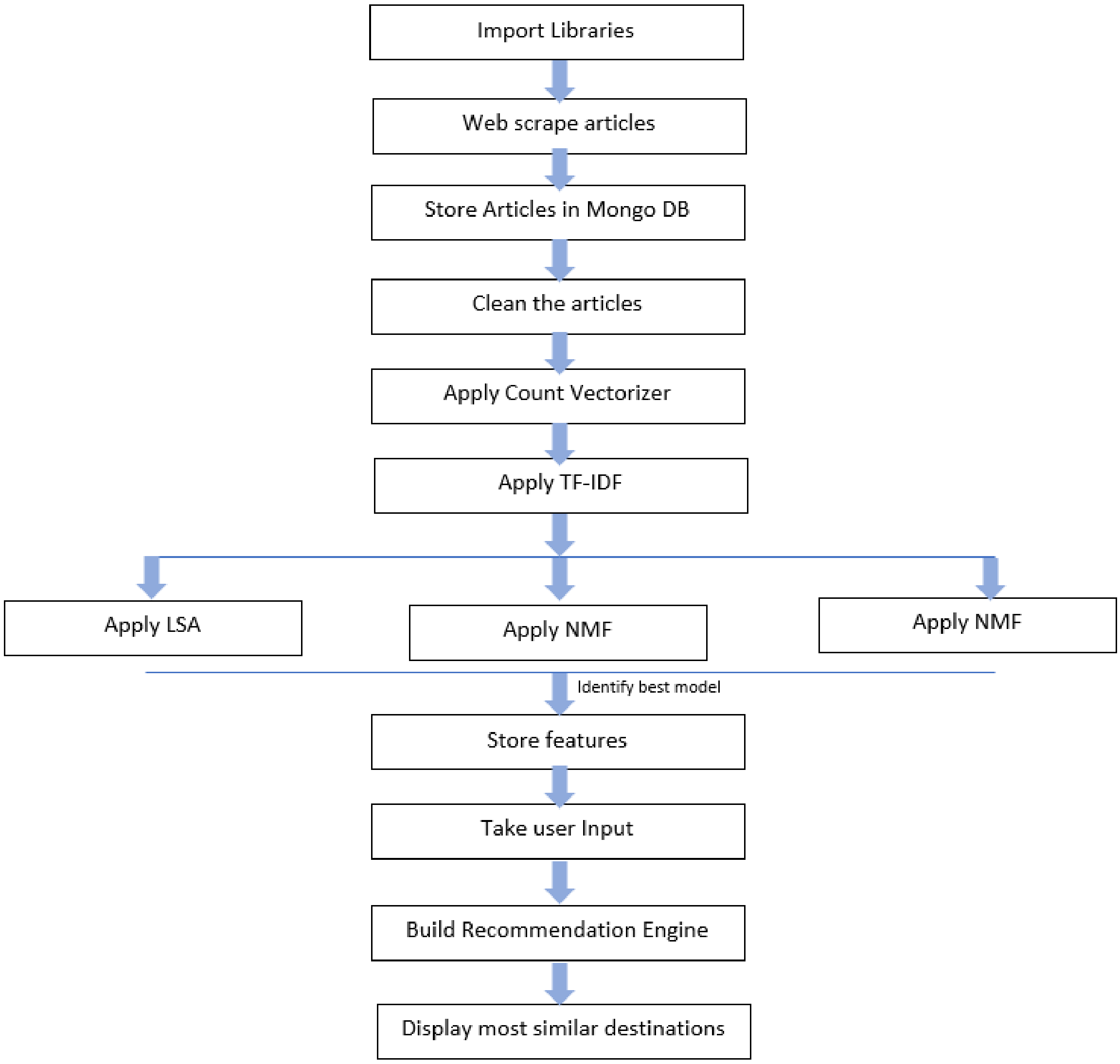
6. Latent Dirichlet Allocation (LDA)
   Latent Dirichlet Allocation (LDA) is a popular topic modelling technique to extract topics from a given corpus. The term latent conveys something that exists but is not yet developed. In other words, latent means hidden or concealed.

7. Non-Negative Matrix Factorization (NMF)
   Non-Negative Matrix Factorization (NMF) decomposes (or factorizes) high-dimensional vectors into a lower-dimensional representation. These lower-dimensional vectors are non-negative which also means their coefficients are non-negative. Using the original matrix (A), NMF will give you two matrices (W and H). W is the topics it found and H is the coefficients (weights) for those topics. In other words, A is articles by words (original), H is articles by topics and W is topics by words.

8. Principal Component Analysis (PCA)
   Principal component analysis, or PCA, is a dimensionality-reduction method that is often used to reduce the dimensionality of large data sets, by transforming a large set of variables into a smaller one that still contains most of the information in the large set.

9. Cosine Similarity
   Cosine similarity measures the similarity between two vectors of an inner product space. It is measured by the cosine of the angle between two vectors and determines whether two vectors are pointing in roughly the same direction. It is often used to measure document similarity in text analysis.

METHODOLOGY



"I need a vacation, but where should I go?"

Import Libraries

In [2]:

```
# General libraries
import numpy as np
import pandas as pd
import seaborn as sns
import random

# Data lists for scraping
from project4data import cities, countries
from webscrape import country

# Scraping
import os
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
import time
import requests
from bs4 import BeautifulSoup

# MongoDB
from pymongo import MongoClient

# NLP cleaning
#from cleaning import clean_article

# Vectorising
import copy
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer

# Topic Modelling & Dimensionality Reduction
from sklearn.decomposition import TruncatedSVD
from sklearn.decomposition import NMF
from sklearn.metrics.pairwise import cosine_similarity

# PCA
from sklearn.decomposition import PCA

# Recommender Engine
from recommender import recommendation,return_countries
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
e:\19mia1077\year 4\sem 7\Recomender\Project\where-to-vacation-nlp-project-master\nlp_vac
ation_fullcode.ipynb Cell 3 in <cell line: 38>()
     <a href='vscode-notebook-cell:/e%3A/19mia1077/year%204/sem%207/Recomender/Project/wh
ere-to-vacation-nlp-project-master/nlp_vacation_fullcode.ipynb#W2sZmlsZQ%3D%3D?line=34'>3
5</a> from sklearn.decomposition import PCA
     <a href='vscode-notebook-cell:/e%3A/19mia1077/year%204/sem%207/Recomender/Project/wh
ere-to-vacation-nlp-project-master/nlp_vacation_fullcode.ipynb#W2sZmlsZQ%3D%3D?line=36'>3
7</a> # Recommender Engine
---> <a href='vscode-notebook-cell:/e%3A/19mia1077/year%204/sem%207/Recomender/Project/wh
ere-to-vacation-nlp-project-master/nlp_vacation_fullcode.ipynb#W2sZmlsZQ%3D%3D?line=37'>3
8</a> from recommender import recommendation,return_countries

File e:\19mia1077\year 4\sem 7\Recomender\Project\where-to-vacation-nlp-project-master\re
commender.py:5, in <module>
      2 import random
      3 from project4data import cities, countries
----> 5 with open('../ranked_df.pickle','rb') as read_file:
      6     ranked_df = pickle.load(read_file)
      9 def recommendation(inputs, original_df=ranked_df, iteration=1):
```

```
FileNotFoundError: [Errno 2] No such file or directory: '../ranked_df.pickle'
```

Put data into MongoDB

In [11]:

```
# Import the required libraries
import os

# Define the location of the directory
path =r"E:\19mia1077\year 4\sem 7\Recomender\Project\where-to-vacation-nlp-project-master
\Articles"

articles = []
# Change the directory
os.chdir(path)

def read_files(file_path):
   with open(file_path, 'r') as file:
      articles.append(file.read())

# Iterate over all the files in the directory
for file in os.listdir():
   if file.endswith('.txt'):
      # Create the filepath of particular file
      file_path =f"{path}/{file}"
      read_files(file_path)
```

In [13]:

```
# Create dictionary for each: _id, City, Country, Article
mongo_list = []
for i in range(len(articles)):
    mongo_list.append({'_id': i+1, 'City':cities[i], 'Country': countries[i], 'Article':
articles[i]})
```

In [14]:

```
# Create and add full article data into MongoDB
client = MongoClient()
db = client['project4_fletcher']
collection = db['travel_articles']
#collection.insert_many(mongo_list)
```

In [15]:

```
# Test
cursor = list(db.travel_articles.find({'_id': 1}, {'_id':1, 'City':1, 'Article':1}))
cursor
```

Out[15]:

```
[{'_id': 1,
  'City': 'Amsterdam',
  'Article': "BELIEVE it or not, there are far more intoxicating reasons to visit Amsterd
am these days than its infamous coffee shops or its red-light district. Not since the Dut
ch Golden Age has Amsterdam seen such a creative boom. All along the harbor and in the ci
ty's South Axis area, futuristic buildings designed by architects like Renzo Piano and Ra
fael Viñoly have been going up — a modernist foil to the city's venerable canal houses. T
he country known for Rembrandt and Franz Hals also has modern day counterparts in Amsterd
am design stars like Marcel Wanders and Tord Boontje. And the restaurant scene is finally
catching up with the rest of Europe. Amsterdam is angling to become Europe's creative cap
ital, and it's doing so without even inhaling.\n\nFriday\n\n4:30 p.m.\n1) GET SOME WHEELS
\n\nFirst things first. Renting a bike is key in Amsterdam; you can avoid expensive taxi
rides and feel like a local from the start. Don't be nervous. Two-wheels rule the roads,
```

it's difficult to get lost and there are bike paths everywhere. A central spot to find a
safe and sturdy bicycle is Orangebike (Singel 233; 31-20-528-9990; www.orangebike.nl) nea
r Dam Square. Rentals are 9.50 euros a day, or about $13.20 at $1.39 to the euro. They al
so offer amusing excursions, including bike tours of cafes, beaches and gay landmarks.\n\
n5 p.m.\n2) GET YOUR BEARINGS\n\nThe tower at Westerkerk (Prinsengracht 281; www.westerke
rk.nl) recently reopened after a major restoration. At more than 279 feet tall, it's the
highest church tower in Amsterdam and, on clear day, affords stunning views of the entire
city, including the glittering modern towers of the South Axis neighborhood. It's also wh
ere Rembrandt was buried and where Queen Beatrix and Prince Claus were married.\n\n6 p.m.
\n3) UNWIND AT VYNE\n\nA few blocks away on the Prinsengracht, you'll find Vyne (Prinseng
racht 411; 31-20-344-6408; www.vyne.nl), a trendy new wine bar from the owners of Envy, t
he hotspot restaurant next door. Although the space was designed by the ultramodern local
architectural firm Concrete, the bar is cozy and sensual with a wall of backlit wine case
s and sexy suede banquettes.\n\n7:30 p.m.\n4) DINE WITH STRANGERS\n\nThe next best thing
to having cool, connected friends in town is hiring a few. The quirky company Like-a-Loca
l (31-20-530-1460; www.like-a-local.com) connects tourists with friendly locals who will
invite you into their homes and cook dinner for you. Choose the 1930s-style apartment of
a graphic designer couple, dinner on a house boat or a traditional three-course Dutch mea
l in a classic canal house. These dinners are not only fun and informative, but they're a
great deal as well, with prices ranging from 23 to 35 euros, including a lot of wine.\n\n
10 p.m.\n5) DANCE WITH GROWN-UPS\n\nThe popular nightclub Jimmy Woo's may be good times i
f you're 22, but anyone older might feel like a chaperone. Go instead to the Mansion (Hob
bemastraat 2; 31-20-616-6664; www.the-mansion.nl), an upscale saloon that feels like a me
mbers-only club for stylish 30-somethings. Fortunately, you don't have to be a member or
know one to be admitted to this three-story town house. It has an over-the-top Asian rest
aurant with ostrich wallpaper and mirrored tables, three cocktail bars with pretty lighti
ng and hotshot bartenders, and a cool dance floor that vibrates with some of Europe's up-
and-coming D.J.'s.\n\nSaturday\n\n10 a.m.\n6) FLAT FOOD\n\nThe Dutch love their pancakes.
Pancakes with cheese, pancakes with butter and even pancake sushi. You name it, stick it
in a pancake and they'll eat it. You'll find all these options and much more at the newly
opened and often packed Pancakes! restaurant (Berenstraat 38; 31-20-528-9797; www.pancake
samsterdam.com). A pancake with endive, ham, Camembert cheese and raspberry sauce is 9.80
euros.\n\n11:30 a.m.\n7) DRESSED TO THE NINES\n\nYou're already in the renowned shopping
zone known as the Nine Streets (www.theninestreets.com), loaded with funky independent st
ores, so you might as well do some consuming. On the same street as Pancakes! is Fashion
Flairs (31-20-620-2104) a new women's shop that sells flirty and glamorous dresses. In th
e opposite direction on Wolvenstraat is Spoiled (Wolvenstraat 19; 31-20-626-3818; www.spo
iled.nl), a fashion emporium with a denim bar and a hair salon. For avant-garde clothing,
there's Van Ravenstein (Keizersgracht 359; 31-20-639-0067; www.van-ravenstein.nl) which c
arries Dries Van Noten, Viktor & Rolf and Balenciaga. And for home accessories with a sen
se of humor, drop into DR Wonen (Hartenstraat 27, 31-20-489-2808).\n\n1:30 p.m.\n8) LUNCH
, ART, DESIGN\n\nThe menu is simple (pastas, sandwiches and salads) but the view is subli
me from the trendy restaurant 11 (Oosterdokskade 3-5; 31-20-625-5999; www.ilove11.nl) hid
den at the top of the Post Building on the harbor. While you're there, stop off at the se
cond and third floors to check out the contemporary art at the temporary location of the
Stedelijk Museum (31-20-573-2911; www.stedelijk.nl). Through Sept. 16, the museum is show
casing 30 artists from the Netherlands. Afterward, hop on your bike toward the city's Flo
wer Market. You're heading to Droog Design (Staalstraat 7; 31-20-523-5050; www.droogdesig
n.nl), the country's iconic design collective and gallery, where you'll get a peek at fut
ure designers. A show this summer features student work from the esteemed Design Academy
Eindhoven.\n\n7:30 p.m.\n9) CONCEPTUAL DINING\n\nThey may be out of the way, but the city
's most buzzed-about restaurants are worth the trip. Across the harbor from Central Stati
on is the Hotel de Goudfazant (Aambeeldstraat 10 H; 31-20-636-5170; www.hoteldegoudfazant
.nl), a sweeping industrial space with an enormous modern chandelier and raw concrete sur
faces. Stylish locals order unpretentious but tasty fare like roasted chicken and oyster
ceviche. (A three-course meal is 28.50 euros). The conceptual Restaurant As (Prinses Iren
estraat 19, 31-20-644-0100; www.platform21.com) is in the South Axis area. It's part of P
latform 21, an experimental space for design and fashion. Diners sit at long communal tab
les while meals are cooked in a Tuscan oven. The three- to five-course menu changes daily
and recently included asparagus with pecorino, wild sea bass with fennel stalks, and bari
goule of artichokes and cipollini onions. A three-course dinner runs 36 euros.\n\n10 p.m.
\n10) COCKTAILS AT THE FACTORY\n\nThe Westergasfabriek — a former gasworks that now house
s a complex of galleries, cafes and performance spaces — is still a local secret. It come
s to life at night when crowds of young adults in T-shirts and jeans meet for drinks at t
he laid back Pacific Parc restaurant and bar (Polonceaukade 23; 31-20-488-7778; www.pacif
icparc.nl). The cavernous space is a mishmash of recycled tables and chairs, a massive ch
andelier made from found objects, and a D.J. booth that spins different types of music ev
ery night. Like the rest of Amsterdam, there are no pretensions or velvet ropes.\n\nMidni
ght\n11) THE PARTY DOESN'T END\n\nOn the other end of the Westergasfabriek is Flex (Pazza
nistraat 1; 31-20-486-2123; www.flexbar.nl), an intimate new two-room insider spot that f
eatures a mix of local and international D.J.'s. It attracts a crowd similar to Pacific P
arc, but these are the people who want to keep the party going after 2 a.m.\n\nSunday\n\n

```
Noon\n12) HIT THE BEACH\n\nIt's only logical that water-lined Amsterdam was among the fir
st European cities to try the urban beach concept. One of the most stylish is on the slop
ed roof of the Renzo Piano-designed NEMO, Amsterdam's Science and Technology Center (Oost
erdok 2, 31-20-531-3233; www.e-nemo.nl). You'll find more than sand and stunning harbor v
iews; there are also D.J.'s, big cozy beanbags and tapas-style snacks.\n\nThe Basics\n\nK
LM (www.klm.com) and other major carriers fly from Kennedy Airport in New York to Amsterd
am. A recent Web search showed round-trip fares starting at around $1,044. From Amsterdam
's Schiphol Airport, a taxi into town costs about 40 euros, or about $56 at $1.39 to the
euro. Once in the city, the best way to get around is by bike or on foot.\n\nFashion-cons
cious travelers stay at the Dylan (Keizersgracht 384; 31-20-530 2010; www.dylanamsterdam.
com), a modern boutique hotel with a chic lounge and 41 rooms starting at 435 euros.\n\nC
elebrities tend to choose the Intercontineal Amstel (Professor Tulpplein 1; 31-20-622-60
60; www.amsterdam.intercontinental.com). Doubles currently start at about 475 euros a nig
ht.\n\nA new favorite among stylish urbanites is the College Hotel near the Van Gogh Muse
um (Roelof Hartstraat 1, 31-20- 571-1511; www.collegehotelamsterdam.com). Doubles start a
t about 250 euros.\n\nThere is also Miauw Suites Amsterdam (Hartenstraat 36; 31-20-717-34
29; www.miauw.com), scheduled to open last Friday. Rooms at the hotel, located in the Nin
e Streets area, start at 145 euros.\n\nFor a bed-and-breakfast, try Steel (Staalstraat 32
; www.staywithsteel.com ). Prices start at 150 euros a night, with a minimum stay of two
nights.\n"}]
```


Clean text data

In [16]:

```python
import re
import string
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from unicodedata import normalize, category
from nltk.tag import pos_tag
from nltk.stem import WordNetLemmatizer
from nltk.stem.lancaster import LancasterStemmer
```

In [17]:

```python
def clean_article(article):

    # Find all proper nouns and names in article - keep to one side
    nouns_list = []
    for tup in pos_tag(word_tokenize(article)):
        if 'NNP' in tup:
            nouns_list.append(tup[0].lower())

    # Remove accents (keep as just letters)
    article = ''.join([c for c in normalize('NFD',article.lower()) if category(c) != 'Mn
'])
    article = re.sub(r'[-|—|’|“|”|£]', ' ', article)
    article = re.sub('\(.*?\)', '', article)

    # Tokenize article in words
    tokens = nltk.word_tokenize(article)

    # Remove all punctuation & numbers
    for idx, word in enumerate(tokens):
        tokens[idx] = "".join(l for l in word if l not in string.punctuation)

    for idx, word in enumerate(tokens):
        if re.compile('\w*\d\w*').match(word):
            tokens[idx] = ''

    # Remove stopwords
    stops = list(stopwords.words('english'))

    for idx, word in enumerate(tokens):
```

```
        if word in stops:
            tokens[idx] = ''

    # Remove 'credit' from words
    for idx, word in enumerate(tokens):
        if "credit" in word:
            tokens[idx] = ''
        if "euro" in word:
            tokens[idx] = ''
        if "city" in word:
            tokens[idx] = ''

    # Remove spaces and 2 letter words and words over 15 letters
    final_list = []
    for word in tokens:
        if 2<len(word)<15:
            final_list.append(word)

    # Remove nouns and names
    total_list = []
    for word in final_list:
        if word not in nouns_list:
            total_list.append(word)

    # Lemmatize words & stem
    for idx, word in enumerate(total_list):
        new_word = WordNetLemmatizer().lemmatize(word, pos='v')
        new_word = LancasterStemmer().stem(new_word)
        total_list[idx] = new_word


    return ' '.join(total_list)
```

In [18]:

```
# Clean all articles from scraped data
clean_articles = []

for city_id in range(1,14):
    art = list(db.travel_articles.find({'_id': city_id}, {'_id':0, 'Article':1}))
    for obj in art:
        clean_articles.append(clean_article(str(obj['Article'])))
```

Count Vectorize

In [19]:

```
# Make a copy of the articles for count vectorising & tfidf
cv_articles = copy.deepcopy(clean_articles)
tf_articles = copy.deepcopy(clean_articles)
```

In [20]:

```
cv = CountVectorizer(min_df=0.15, max_df=0.9)
cv_X = cv.fit_transform(cv_articles)
```

In [21]:

```
cv_array = pd.DataFrame(cv_X.toarray(), index=cities, columns=cv.get_feature_names())
```

```
c:\Python310\lib\site-packages\sklearn\utils\deprecation.py:87: FutureWarning: Function g
et_feature_names is deprecated; get_feature_names is deprecated in 1.0 and will be remove
d in 1.2. Please use get_feature_names_out instead.
  warnings.warn(msg, category=FutureWarning)
```

In [22]:

```
cv_array.head()
```

Out[22]:

	abund	acc	access	accommod	accompany	account	acr	across	act	ad	...	workshop	world	worthy	woul
Amsterdam	0	0	1	0	0	0	0	1	0	0	...	0	0	0	(
Bali	1	0	1	1	0	0	0	0	1	0	...	1	0	0	:
Bangalore	0	0	0	0	0	1	1	2	1	0	...	0	0	0	(
Bangkok	0	0	0	0	0	0	1	3	0	0	...	0	0	0	(
Barcelona	0	0	0	0	2	0	0	1	0	0	...	0	1	0	(

5 rows × 1259 columns

TF-IDF

In [23]:

```
tfidf = TfidfVectorizer(min_df=0.1, max_df=0.9, ngram_range=(1,3))
tf_X = tfidf.fit_transform(tf_articles)
```

In [24]:

```
tf_array = pd.DataFrame(tf_X.toarray(), index=cities, columns=tfidf.get_feature_names())
```

```
c:\Python310\lib\site-packages\sklearn\utils\deprecation.py:87: FutureWarning: Function g
et_feature_names is deprecated; get_feature_names is deprecated in 1.0 and will be remove
d in 1.2. Please use get_feature_names_out instead.
  warnings.warn(msg, category=FutureWarning)
```

In [25]:

```
tf_array.head()
```

Out[25]:

	abund	acc	access	accommod	accompany	account	account sav	account sav map	acr	across	...	worthy	
Amsterdam	0.00000	0.0	0.031353	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.026699	...	0.0	0.
Bali	0.04087	0.0	0.027239	0.036242	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.0	0.
Bangalore	0.00000	0.0	0.000000	0.000000	0.000000	0.038536	0.043457	0.043457	0.034719	0.049329	...	0.0	0.
Bangkok	0.00000	0.0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.032644	0.069570	...	0.0	0.
Barcelona	0.00000	0.0	0.000000	0.000000	0.091812	0.000000	0.000000	0.000000	0.000000	0.026054	...	0.0	0.

5 rows × 1452 columns

Topic Modelling

LSA and TFIDF

In [50]:

```
tf_array.shape
```

Out[50]:

```
(13, 1452)
```

In [26]:

```
lsa = TruncatedSVD(5)
doc_topic = lsa.fit_transform(tf_array)
lsa.explained_variance_ratio_.round(3)
```

Out[26]:

```
array([0.   , 0.107, 0.101, 0.093, 0.089])
```

In [27]:

```
# Create dataframe of each word and weighting of word per article
topic_word = pd.DataFrame(lsa.components_.round(3),
             columns = tfidf.get_feature_names())
topic_word
```

```
c:\Python310\lib\site-packages\sklearn\utils\deprecation.py:87: FutureWarning: Function g
et_feature_names is deprecated; get_feature_names is deprecated in 1.0 and will be remove
d in 1.2. Please use get_feature_names_out instead.
  warnings.warn(msg, category=FutureWarning)
```

Out[27]:

	abund	acc	access	accommod	accompany	account	account sav	account sav map	acr	across	...	worthy	would	writ	year
0	0.011	0.012	0.032	0.016	0.018	0.016	0.012	0.012	0.019	0.044	...	0.021	0.022	0.018	0.067
1	-0.002	-0.004	-0.008	0.011	-0.021	0.005	0.023	0.023	-0.002	-0.027	...	-0.022	0.040	-0.021	0.041
2	0.022	-0.020	0.002	-0.015	-0.036	0.006	-0.009	-0.009	0.022	0.022	...	-0.012	-0.007	-0.036	0.021
3	-0.001	0.010	-0.019	0.003	0.010	-0.035	-0.029	-0.029	-0.010	-0.059	...	-0.054	0.027	0.010	-0.039
4	0.019	0.015	0.017	0.001	-0.033	-0.018	-0.024	-0.024	-0.029	-0.014	...	0.017	0.080	-0.033	0.032

5 rows × 1452 columns

In [28]:

```
# Function to show most important words in different topics
def display_topics(model, feature_names, no_top_words, topic_names=None):

    '''Displays topics and word in order of importance for each topic
    Inputs: Model being used for analysis, topic names, number of words to
    present for each category'''

    for idx, topic in enumerate(model.components_):
        if not topic_names or not topic_names[idx]:
            print("\nTopic ", idx)
        else:
            print("\nTopic: '",topic_names[idx],"'")
        print(", ".join([feature_names[i] for i in topic.argsort()[:-no_top_words - 1:-1
]]))
```

In [29]:

```
display_topics(lsa, tfidf.get_feature_names(),15)
```

Topic 0

```
Topic  0
bar, loc, two, tour, win, night, start, spac, serv, off, includ, get, around, plac, rup

Topic  1
rup, im, col, per, plac, vil, germ, decad, around rup, brunch, serv, beach, import, cocon
ut, vib

Topic  2
glow, whit, window, serv, soar, templ, gold, smal, wal, hotel, food, amid, green, soft, b
el

Topic  3
tour, rat, lin, stil, sint, view, vil, crowd, rol, rid, start, district, get, are, spring

Topic  4
germ, er, form, per night, stil, night, preserv, apart, coff, would, cloth, sens, swa, li
st, smal
```

```
c:\Python310\lib\site-packages\sklearn\utils\deprecation.py:87: FutureWarning: Function g
et_feature_names is deprecated; get_feature_names is deprecated in 1.0 and will be remove
d in 1.2. Please use get_feature_names_out instead.
  warnings.warn(msg, category=FutureWarning)
```

In [30]:

```
# Create dataframe of topic array (Vt in equation)
Vt = pd.DataFrame(doc_topic.round(5),
            index = cities)
Vt.head()
```

Out[30]:

	0	1	2	3	4
Amsterdam	0.58047	-0.27604	-0.34965	0.11991	0.02722
Bali	0.51132	0.33861	0.13021	0.18364	0.29464
Bangalore	0.53958	0.32685	0.05675	-0.35526	-0.39994
Bangkok	0.53015	-0.34342	0.42785	-0.16519	0.10012
Barcelona	0.53848	-0.14419	-0.27519	0.28201	-0.37409

In [31]:

```
# Check similarity between articles
cos_sim = pd.DataFrame(cosine_similarity(Vt).round(), index=cities, columns=cities)
cos_sim.head()
```

Out[31]:

	Amsterdam	Bali	Bangalore	Bangkok	Barcelona	Berlin	Boston	Budapest	Cape Town	Dubai	Goa	Hong Kong	Jaipur
Amsterdam	1.0	0.0	0.0	0.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	1.0	0.0
Bali	0.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	1.0	1.0	1.0
Bangalore	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	1.0
Bangkok	0.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	1.0	1.0
Barcelona	1.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	1.0	0.0	0.0	1.0	0.0

Observations:

- **A lot of similarity between a lot of cities - possibly due to the way each article is structured?**

NMF and TFIDF - [Best]

In [32]:

```
nmf_model = NMF(6, random_state=27) # Try dfferent topic numbers
doc_topic_nmf_tfidf = nmf_model.fit_transform(tf_array)
```

In [33]:

```
topic_word = pd.DataFrame(nmf_model.components_.round(3), #.components_ gets it out of NM
F
             columns =tfidf.get_feature_names())
topic_word
```

```
c:\Python310\lib\site-packages\sklearn\utils\deprecation.py:87: FutureWarning: Function g
et_feature_names is deprecated; get_feature_names is deprecated in 1.0 and will be remove
d in 1.2. Please use get_feature_names_out instead.
  warnings.warn(msg, category=FutureWarning)
```

Out[33]:

	abund	acc	access	accommod	accompany	account	account sav	account sav map	acr	across	...	worthy	would	writ	year
0	0.000	0.033	0.043	0.038	0.078	0.003	0.005	0.005	0.024	0.033	...	0.003	0.000	0.078	0.017
1	0.000	0.000	0.034	0.016	0.000	0.046	0.054	0.054	0.037	0.056	...	0.016	0.000	0.000	0.126
2	0.029	0.000	0.035	0.000	0.000	0.023	0.000	0.000	0.028	0.067	...	0.025	0.000	0.000	0.062
3	0.000	0.000	0.031	0.000	0.044	0.001	0.000	0.000	0.000	0.077	...	0.081	0.000	0.044	0.062
4	0.000	0.038	0.000	0.001	0.000	0.000	0.001	0.001	0.000	0.000	...	0.002	0.078	0.000	0.055
5	0.043	0.000	0.029	0.038	0.000	0.000	0.000	0.000	0.000	0.000	...	0.000	0.088	0.000	0.022

6 rows × 1452 columns

In [34]:

```
display_topics(nmf_model, tfidf.get_feature_names(), 15)
```

```
Topic  0
loc, spac, bar, cours, feat, start, two, bik, eleg, three, win, tour, crowd, town, design

Topic  1
rup, might, col, im, serv, around, ev, century, weekend, dom, beach, year, around rup, da
y, origin

Topic  2
glow, whit, soar, bar, gold, gallery, bel, window, templ, brit, cool, high, vint, smal, f
ood

Topic  3
list, therm, many, atmosph, origin, pork, jew, pub, whiskey, sev, cuisin, brand, neighb,
star, modern

Topic  4
germ, form, er, sens, preserv, cloth, two, apart, stil, swa, flo, night, coff, three, tou
r

Topic  5
plac, ric, vil, coconut, smal, mass, stil, bargain, swim, found, carv, milk, less, los, p
ric
```

```
c:\Python310\lib\site-packages\sklearn\utils\deprecation.py:87: FutureWarning: Function g
et_feature_names is deprecated; get_feature_names is deprecated in 1.0 and will be remove
d in 1.2. Please use get_feature_names_out instead.
  warnings.warn(msg, category=FutureWarning)
```

In [35]:

```
# Create dictionary of City and most prominent topic
city_topic_nmf_tfidf = dict(zip(cities, np.argmax(doc_topic_nmf_tfidf, axis=1)))
```

In [36]:

```
# Check each topic to observe cities
for number in range(0,7):
    for k,v in city_topic_nmf_tfidf.items():
        if v == number:
            print(f"Topic: {number}, {k}")
```

```
Topic: 0, Amsterdam
Topic: 0, Barcelona
Topic: 0, Boston
Topic: 1, Bangalore
Topic: 1, Cape Town
Topic: 1, Goa
Topic: 1, Jaipur
Topic: 2, Bangkok
Topic: 2, Dubai
Topic: 2, Hong Kong
Topic: 3, Budapest
Topic: 4, Berlin
Topic: 5, Bali
```

In [37]:

```
# Check prevelance of topic for each city
H = pd.DataFrame(doc_topic_nmf_tfidf.round(5),
             index = cities)
H.head(5)
```

Out[37]:

	0	1	2	3	4	5
Amsterdam	0.36155	0.00000	0.00000	0.02354	0.06639	0.0000
Bali	0.00000	0.00000	0.00000	0.00000	0.00000	0.9149
Bangalore	0.00000	0.48915	0.00000	0.00000	0.00000	0.0000
Bangkok	0.00000	0.00000	0.61695	0.00000	0.00000	0.0000
Barcelona	0.37963	0.00000	0.00000	0.00000	0.00000	0.0018

In [38]:

```
cos_sim_nmf = pd.DataFrame(cosine_similarity(H).round(), index=cities, columns=cities)
cos_sim_nmf.head()
```

Out[38]:

	Amsterdam	Bali	Bangalore	Bangkok	Barcelona	Berlin	Boston	Budapest	Cape Town	Dubai	Goa	Hong Kong	Jaipur
Amsterdam	1.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0
Bali	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Bangalore	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	1.0
Bangkok	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	1.0	1.0
Barcelona	1.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0

- **NMF gives better categories and splits the cities into better topics than LSA**

NMF and CountVectoriser

In [39]:

```
nmf_model = NMF(6, random_state=27)
doc_topic = nmf_model.fit_transform(cv_array)
```

```
c:\Python310\lib\site-packages\sklearn\decomposition\_nmf.py:1692: ConvergenceWarning: Ma
ximum number of iterations 200 reached. Increase it to improve convergence.
  warnings.warn(
```

In [40]:

```
topic_word = pd.DataFrame(nmf_model.components_.round(3),
             columns =cv.get_feature_names())
topic_word
```

```
c:\Python310\lib\site-packages\sklearn\utils\deprecation.py:87: FutureWarning: Function g
et_feature_names is deprecated; get_feature_names is deprecated in 1.0 and will be remove
d in 1.2. Please use get_feature_names_out instead.
  warnings.warn(msg, category=FutureWarning)
```

Out[40]:

	abund	acc	access	accommod	accompany	account	acr	across	act	ad	...	workshop	world	worthy	would
0	0.000	0.789	1.351	1.102	1.205	0.337	0.881	0.998	0.185	0.000	...	0.000	0.705	0.141	0.000
1	0.698	0.000	1.316	0.000	0.000	0.678	0.826	3.177	1.353	0.000	...	0.000	0.021	0.629	0.000
2	0.000	0.000	3.172	0.000	3.094	0.826	0.239	10.707	4.009	3.214	...	0.000	6.433	6.990	0.000
3	0.665	0.000	1.610	0.781	0.000	0.418	0.109	0.691	1.005	0.395	...	0.505	0.000	0.000	1.269
4	0.000	0.000	0.000	0.000	0.000	0.044	0.591	0.621	0.000	0.534	...	1.053	0.064	0.000	0.000
5	0.000	0.266	0.000	0.012	0.000	0.031	0.000	0.000	0.024	0.000	...	0.000	0.000	0.029	0.551

6 rows × 1259 columns

In [41]:

```
display_topics(nmf_model, cv.get_feature_names(), 15)
```

```
Topic  0
loc, bar, spac, two, feat, includ, start, win, design, tour, also, cours, three, din, sto
p

Topic  1
bar, whit, gallery, tradit, food, glow, wood, soar, light, spac, two, class, bel, vint, a
ir

Topic  2
many, list, loc, origin, lat, off, neighb, star, bout, win, modern, glass, atmosph, sev,
cuisin

Topic  3
plac, century, beach, ev, lov, coconut, two, day, im, lif, vil, art, ric, mass, year

Topic  4
tour, serv, wal, around, back, courtyard, gard, near, ceil, hotel, window, get, walk, jew
elry, flo

Topic  5
form, germ, two, flo, er, loc, night, apart, three, stil, tour, win, cloth, caf, build
```

```
c:\Python310\lib\site-packages\sklearn\utils\deprecation.py:87: FutureWarning: Function g
et_feature_names is deprecated; get_feature_names is deprecated in 1.0 and will be remove
d in 1.2. Please use get_feature_names_out instead.
  warnings.warn(msg, category=FutureWarning)
```

In [43]:

```
city_topic = dict(zip(cities, np.argmax(doc_topic, axis=1)))
```

In [44]:

```
# Check each topic to observe cities
for number in range(0,7):
```

```
    for k,v in city_topic.items():
        if v == number:
            print(f"Topic: {number}, {k}")
```

```
Topic: 0, Barcelona
Topic: 0, Boston
Topic: 1, Bangkok
Topic: 1, Dubai
Topic: 2, Budapest
Topic: 3, Bali
Topic: 3, Bangalore
Topic: 3, Goa
Topic: 4, Hong Kong
Topic: 4, Jaipur
Topic: 5, Amsterdam
Topic: 5, Berlin
Topic: 5, Cape Town
```

In [45]:

```
H2 = pd.DataFrame(doc_topic.round(5),
             index = cities)
H2.head()
```

Out[45]:

	0	1	2	3	4	5
Amsterdam	0.37112	0.00000	0.01354	0.00000	0.00000	0.38603
Bali	0.00000	0.00000	0.00000	0.73965	0.00000	0.00000
Bangalore	0.05632	0.00062	0.05241	0.31095	0.27229	0.00000
Bangkok	0.00000	0.68641	0.00000	0.00000	0.00000	0.00000
Barcelona	0.32316	0.00000	0.00005	0.07965	0.06349	0.00000

In [46]:

```
# Cosine similarity between all cities
cos_sim_nmf = pd.DataFrame(cosine_similarity(H).round(), columns=cities, index=cities)
cos_sim_nmf.head(10)
```

Out[46]:

	Amsterdam	Bali	Bangalore	Bangkok	Barcelona	Berlin	Boston	Budapest	Cape Town	Dubai	Goa	Hong Kong	Jaipur
Amsterdam	1.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0
Bali	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Bangalore	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	1.0
Bangkok	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	1.0	1.0
Barcelona	1.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
Berlin	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Boston	1.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
Budapest	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0
Cape Town	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	1.0
Dubai	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	1.0	1.0

Observations:

- **More change in cosine similarity - all seem to be either 0 or 1.**

PCA - Reduce 6 topics to 2 for demonstration

- **Best topics came from using TFIDF and NMF - use for demonstration and recommender engine**

In [47]:

```
# normalise numbers across df
norm_df = H.div(H.sum(axis=1), axis=0)
```

In [48]:

```
# Run PCA to reduce to 2 components
pca = PCA(n_components=2)
pca.fit(norm_df)
pcafeatures = pca.transform(norm_df)
```

In [49]:

```
# Create df
pca_df = pd.DataFrame(pcafeatures, index=cities)

# Add topic and countries column
pca_df['Topics'] = list(np.argmax(doc_topic_nmf_tfidf, axis=1))
pca_df['Countries'] = countries
pca_df.head()
```

Out[49]:

	0	1	Topics	Countries
Amsterdam	0.629722	-0.102486	0	The Netherlands
Bali	-0.034958	0.063002	5	Indonesia
Bangalore	-0.540057	-0.601952	1	India
Bangkok	-0.288668	0.786218	2	Thailand
Barcelona	0.786744	-0.123356	0	Spain

In [50]:

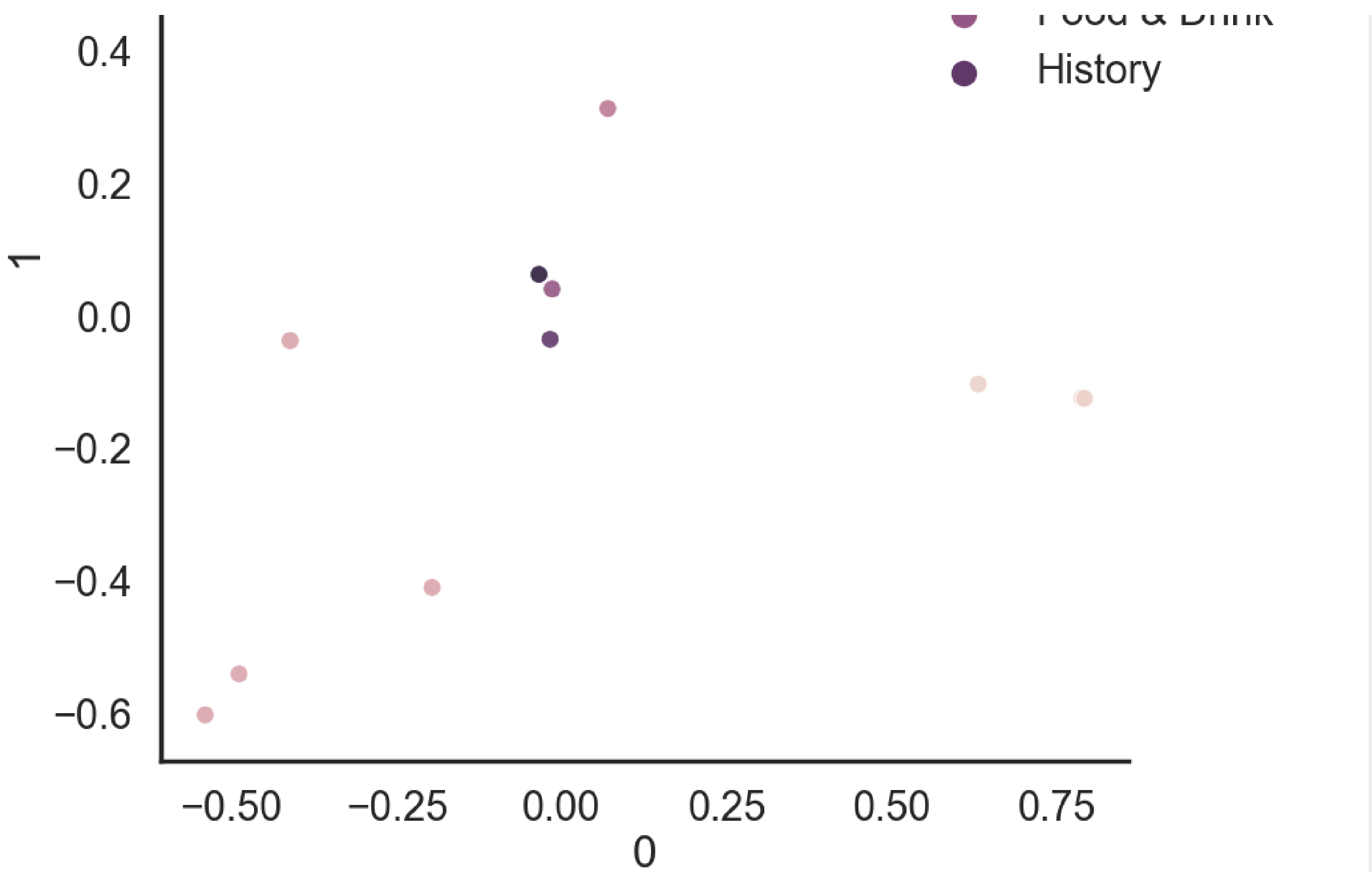
```
# Save for Tableau dashboard
pca_df.to_csv('twodim2.csv', index=True)
```

In [51]:

```
topic_list=['Walking/Tours', 'Nightlife', 'Tropical', 'Art/Culture', 'Food & Drink', 'Hi
story']
```

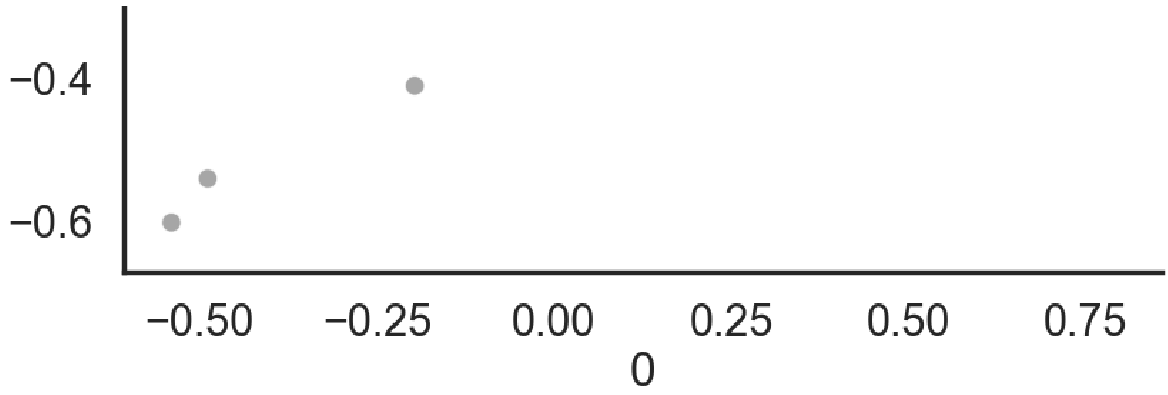
In [52]:

```
# Create scatter plots and highlight a couple of features
sns.set_style("white")
sns.set_context("poster")
six_topic_plot = sns.pairplot(x_vars=[0], y_vars=[1], data=pca_df, hue='Topics', markers
='o', height=9, plot_kws=dict(s=100, alpha=0.9));
six_topic_plot._legend.remove()
six_topic_plot.fig.legend(labels=topic_list, frameon=False);

six_topic_plot.savefig('topic.png')
```



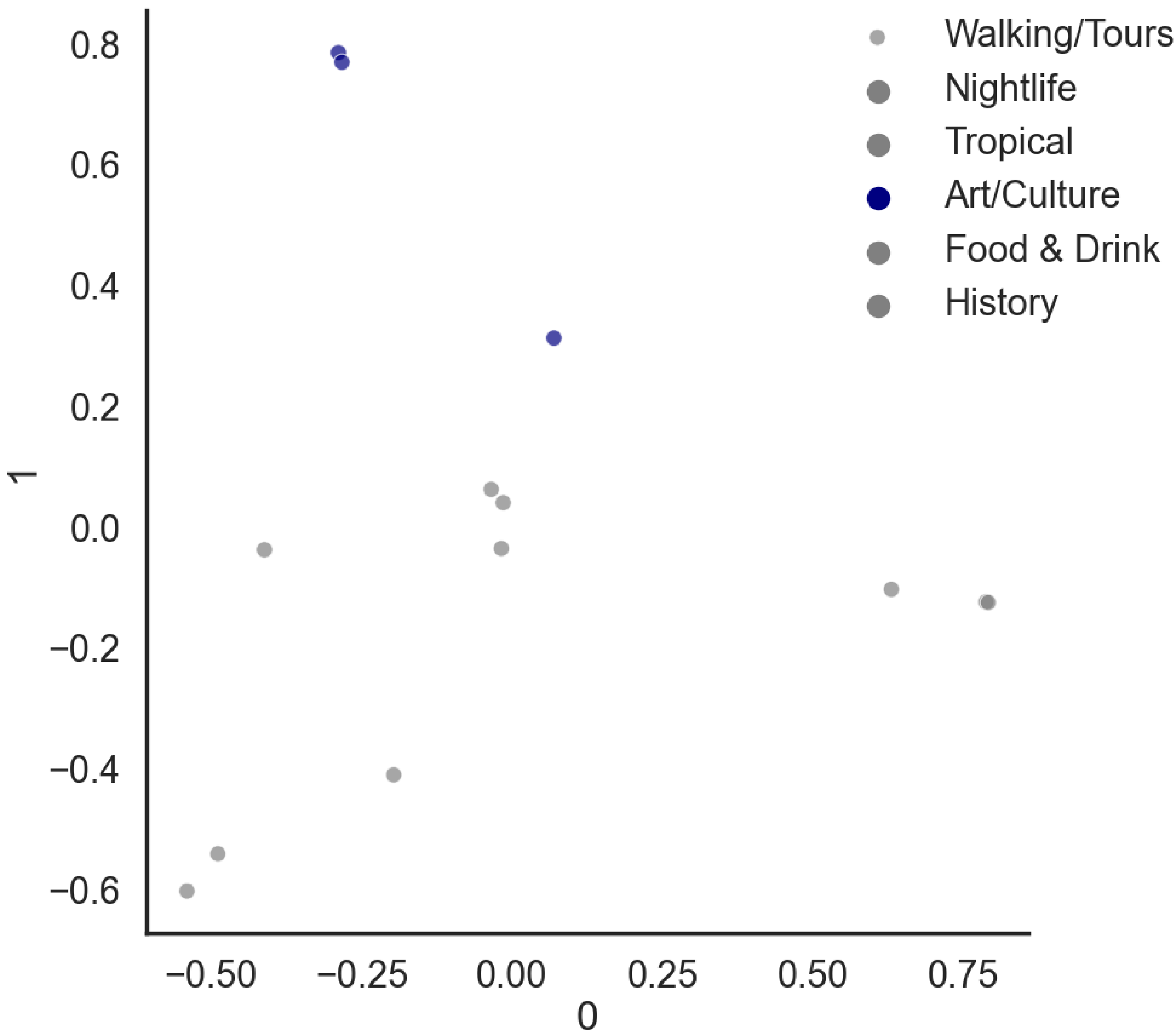
In [53]:

```
# Highlight art
sns.set_style("white")
sns.set_context("poster")
highlight_art = sns.pairplot(x_vars=[0], y_vars=[1], data=pca_df, hue='Topics', markers=
'o', palette=['grey','grey','grey','navy','grey','grey'], height=9, plot_kws=dict(s=95,
alpha=0.7));
highlight_art._legend.remove()
highlight_art.fig.legend(labels=topic_list, frameon=False);

highlight_art.savefig('art.png')
```



In [54]:

```
# Highlight Tropical
sns.set_style("white")
sns.set_context("poster")
highlight_trop = sns.pairplot(x_vars=[0], y_vars=[1], data=pca_df, hue='Topics', markers
='o', palette=['grey','grey','navy','grey','grey','grey'], height=9, plot_kws=dict(s=95,
alpha=0.7));
highlight_trop._legend.remove()
highlight_trop.fig.legend(labels=topic_list, frameon=False)

highlight_trop.savefig('trop.png');
```



Recommender Engine

- **Recommend based on topics - e.g. Rank which is most important & compare to topic distribution for each city**

In [55]:

```
# Function to rank topics
def rank(var_list):
    '''Takes a list of numbers and ranks them based on highest to lowest'''
    sorted_unique = sorted(set(var_list), reverse=True)
    order_dict = {val: i for i, val in enumerate(sorted_unique, 1)}
    return [order_dict[val] for val in var_list]
```

In [56]:

```
# Separate norm_df rows into lists
li = []
for i in range(0,len(norm_df)):
    df_list = list(norm_df.iloc[i])
    li.append(rank(df_list))
```

In [59]:

```
# Create df with rankings of each topic for each country
ranked_df = pd.DataFrame(li, columns=topic_list, index=cities)
ranked_df.head()
```

In [ ]:

```
import pickle
ranked_df.to_pickle('ranked_df.pickle')
```

In [65]:

```
# Test manually for flask app
walking_rank = int(input())
nightlife_rank = int(input())
tropical_rank = int(input())
art_rank = int(input())
fandd_rank = int(input())
history_rank = int(input())
```

In [66]:

```
inputs = {'Walking/Tours' : walking_rank,
          'Nightlife':nightlife_rank,
          'Tropical':tropical_rank,
          'Art/Culture':art_rank,
          'Food & Drink':fandd_rank,
          'History':history_rank}
```

In [68]:

```
city_recommendations = recommendation(inputs)
correct_countries = return_countries(city_recommendations)
dict(zip(city_recommendations, correct_countries))
```

Out[68]:

```
{'Boston': 'USA'}
```

CONCLUSION

Out of listed three techniques, NLP with TF-IDF showed more efficient feature vector distribution. Using this feature vectors and user input, cosine similarity was computed to give recommendations to user.

Moreover, Principal component analysis shows that any place in the database can be described using 6 feature – Walking/tours, Nightlife, Wildlife, Art/culture, Food and Drinks, History.

FUTURE WORKS

A potential next step could be to optimize our model with more articles and expanding the feature vectors. Creating a full stack application for seamless user experience. Moreover, the model can be merged with other recommendation techniques in order to produce hybrid recommendation model which will help in providing more user centric recommendations.

GITHUB

https://github.com/GauravTrivedi1099/CSE4077_Recommender_Systems_J_component

REFERENCES

[1] A Multi-Level Tourism Destination Recommender System - Hend Alrasheed_, Arwa Alzeer, Arwa Alhowimel, Nora shameri, Aisha Althyabi

[2] A novel tourism recommender system in the context of social commerce - Leila Esmaeili , Shahla Mardani , Seyyed Alireza Hashemi Golpayegani , Zeinab Zanganeh Madar.

[3] A recommender system for tourism industry using cluster ensemble and prediction machine learning techniques - Mehrbakhsh Nilashi, Karamollah Bagherifard , Mohsen Rahmani , Vahid Rafe.

[4] Detection of tourists' attraction points using Instagram profiles - Ksenia D. Mukhina, Stepan V. Rakitin, Alexander A. Visheratin

[5] Intelligent tourism recommender systems: A survey - Joan Borras, Antonio Moreno, Aida Valls

[6] On the design of individual and group recommender systems for tourism - Inma Garcia, Laura Sebastia, Eva Onaindia