Steps followed based on process yelp restaurant

GitHub (for reference on implementation): https://github.com/atif-github-venture/data-mining-capstone (**Note**: The data and large files are excluded, they can be generated by execution)
Make sure to add 'data' and 'output' directory when cloning. Also add the unzipped data provided as is in the data folder as inputs.

File: **Task2/solution.py**

Executing using '--cuisine' argument

- Load business json line by line
    - If restaurant in the category and size of categories more than 1
    - Add that business id as restaurant id and add the star rating of that business 'rest2rate' business id
    - If category is other than 'restaurant', continue appending the business id to 'cat2rid' (category to restaurant id)
- Save the restaurant id with rating map in a file 'restaurantIds2ratings.txt' and also save cat2rid in a pickle file format.
- If business id is in restaurant ids save above, build restaurant to review id 'rest2revID' and dump in a pickle file format. This is a map of business ids to all the reviews mentioned in the review corpus.
- Now (from category to restaurant id) take all number of reviews (based on reviews id) for a category and compute the total reviews of category if the restaurant id is in restaurant to review id.
    - If the categories total number of reviews are above 30, add it to 'valid_cats' dictionary. (approx. 183)
- Now we save the total number of cuisines (in this case 20) based on random samples.
    - For each 'category' from cat2rid, if the corresponding restaurant id is in rest2revID, save sample of restaurant id to category 'sample_rid2cat' (basically a dictionary of restaurant id with its list of categories.
- Now to save all the reviews for a category
    - Read each review from 'review corpus'
    - If restaurant id is in sample derived above, build category's review list and star list and save them all under 'categories' directory

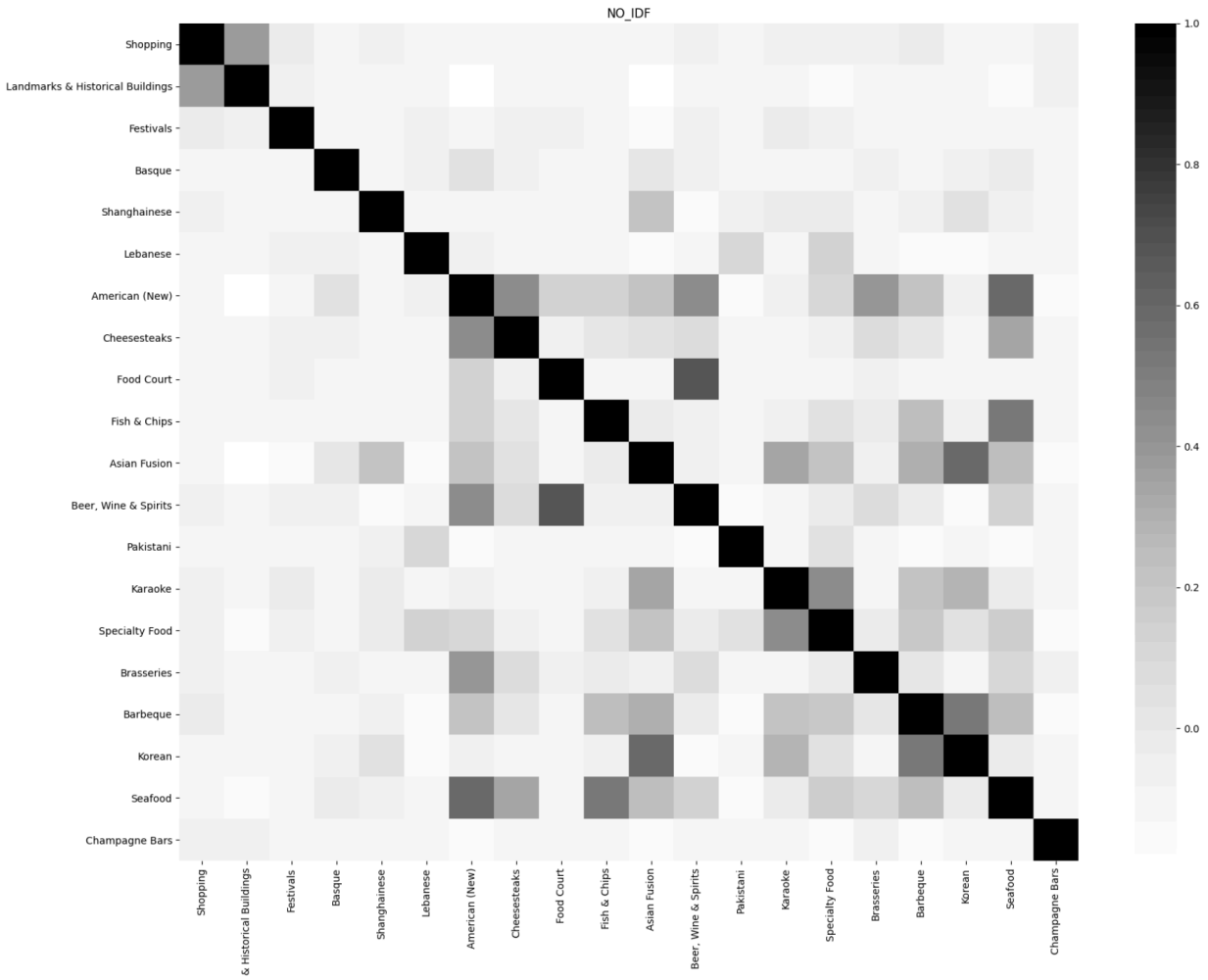
Executing using '--matrix' argument

- Create a list of text from cuisines (in this case 20 sample)
- Based on vectorizer, extract features
    - Max features = 10,000
    - Stop words: English

    - Use idf = true
    - Max document frequency = 0.5 and min df as 2
- Using LDA model (from genism) and topic=50
    - Compute cosine similarity and save the matrix in a file (for visualization)
    - And also save cuisine indices based on how the categories directory were read.
- Generating graph using seaborn and matplotlib libraries.
- The opacity of each cell is the similarity - with a higher opacity for a higher similarity.
    - Refer bar on right for similarity strength
- Deliberately kept the same color theme, to see visual differences.

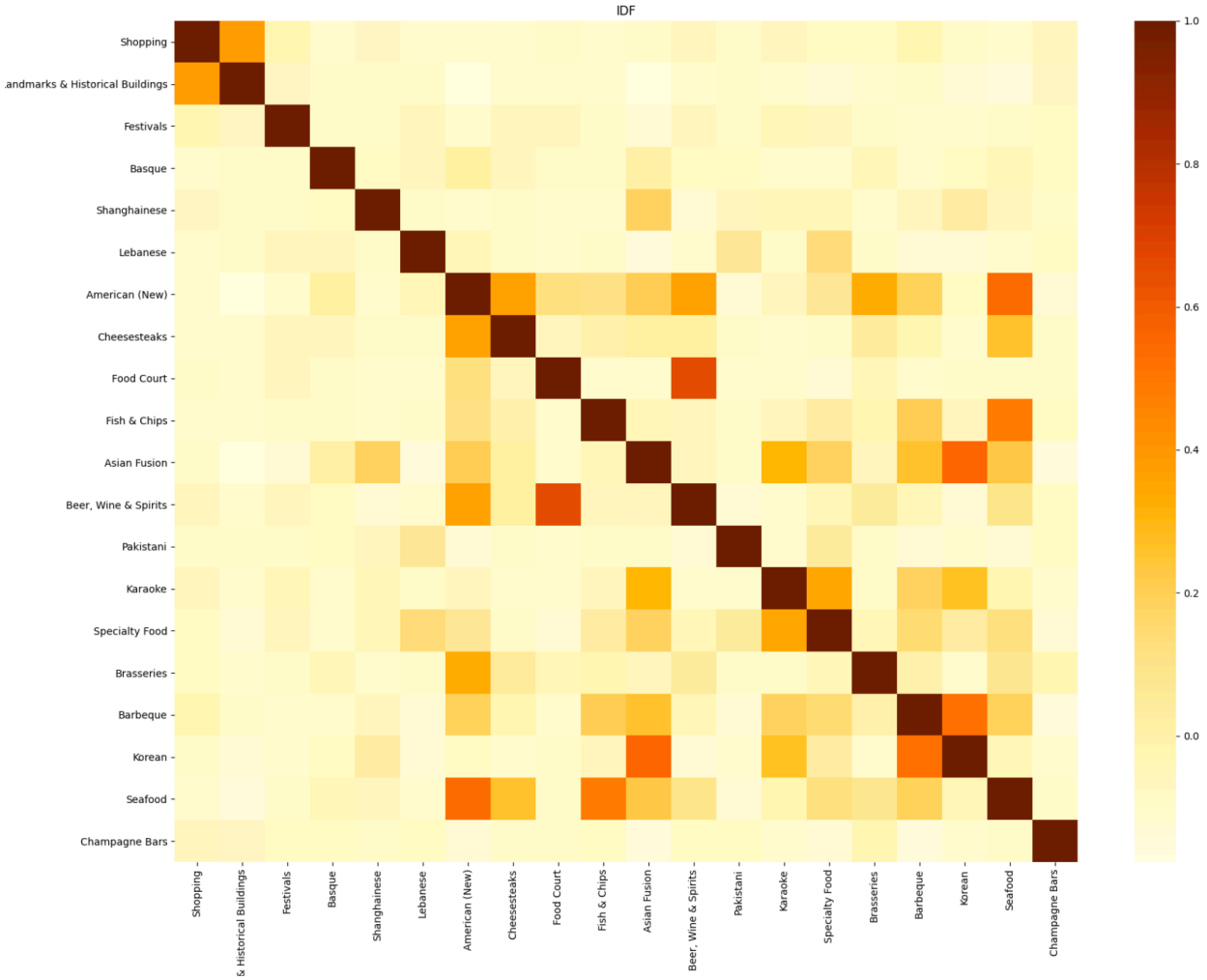
Task 2.1 Visualization of the Cuisine Map

- **No IDF** - Per the program passing variables (cosine similarity)
    - Idf = False
    - cmap = 'Greys'



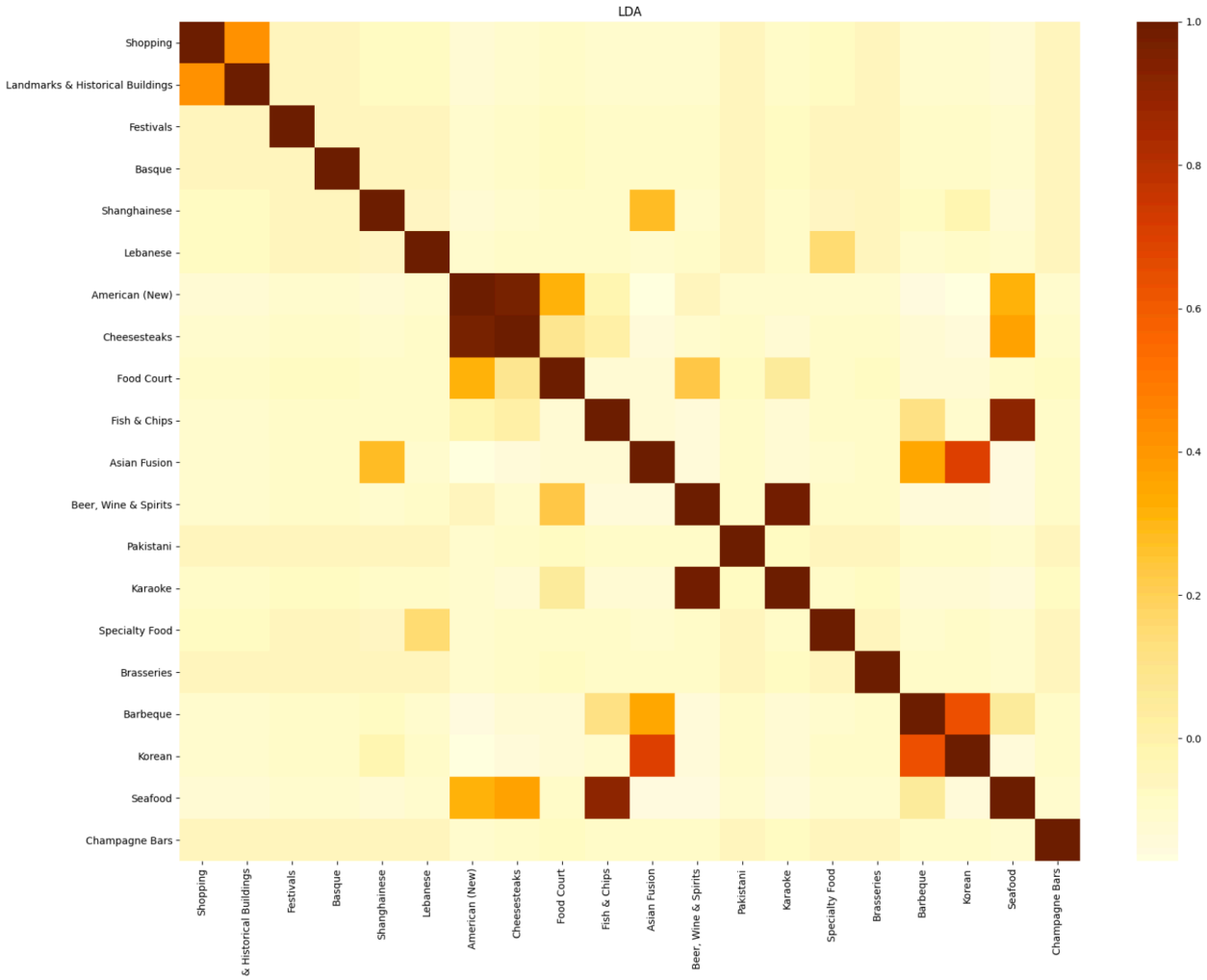
sim_matrix(idf=False, sublinear_tf=False, cmap='Greys', max_df=0.5, min_df=2, method='NO_IDF', similarity='cosine')

- **IDF** - Per the program passing variables (cosine similarity)
  - Idf = True
  - cmap = 'YlOrBr'
  - colors used only for better distinction



sim_matrix(idf=True, sublinear_tf=False, cmap='YlOrBr', max_df=0.5, min_df=2, method='IDF', similarity='cosine')

- **LDA** - Per the program passing variables (cosine similarity)
  - Idf = True
  - cmap = 'YlOrBr'
  - colors used only for better distinction




```
sim_matrix(idf=True, sublinear_tf=False, cmap='YlOrBr', max_df=0.5, min_df=2, method='LDA', similarity='cosine')
```

Task 2.1 – Observation & Description

Based on the program description above the computation of similarity is on all the aggregated reviews text per category of cuisine. And the similarity between 2 different cuisines are computed and visualized.

Image 1 – is on gray scale without the IDF usage in similarity computation. No modeling has been used. Directly the cosine similarity has been applied to generate the view.

Image 2 – Similar to the approach of Image 1, with difference of IDF being used here. Visually there is no obvious change with the used parameters and settings between the IDF and no IDF. However, there are more probability (high score) in image 2 which makes the cell opaquer.
This could be obvious changing other parameters, but we can address those later on.

Image 3 – is on using the IDF and LDA model. All using same dataset of cuisine generated from the coding explained at the beginning. Observing the image difference with that of image 2, very few pairs have got very high similarity, but reduced the similarity between other pairs from image 2 in terms of pair strength.

Note:

- 'max_df' - When building the vocabulary ignore terms that have a document frequency strictly higher than the given threshold (corpus-specific stop words)
- 'min_df' - When building the vocabulary ignore terms that have a document frequency strictly lower than the given threshold. This value is also called cut-off in the literature.

As mentioned above we will have to try different tuners such as max_df, min_df or suing the stop words or lemmatization/stemming differently.

Task 2.2: Improving Clustering Results

Varying the text representation

Let's try to tweak other parameters specially in terms text representation. Other parameters such as max_df, min_df, smooth_idf and sublinear_tf.

smooth_idf (default=True) - Smooth idf weights by adding one to document frequencies, as if an extra document was seen containing every term in the collection exactly once. Prevents zero divisions.

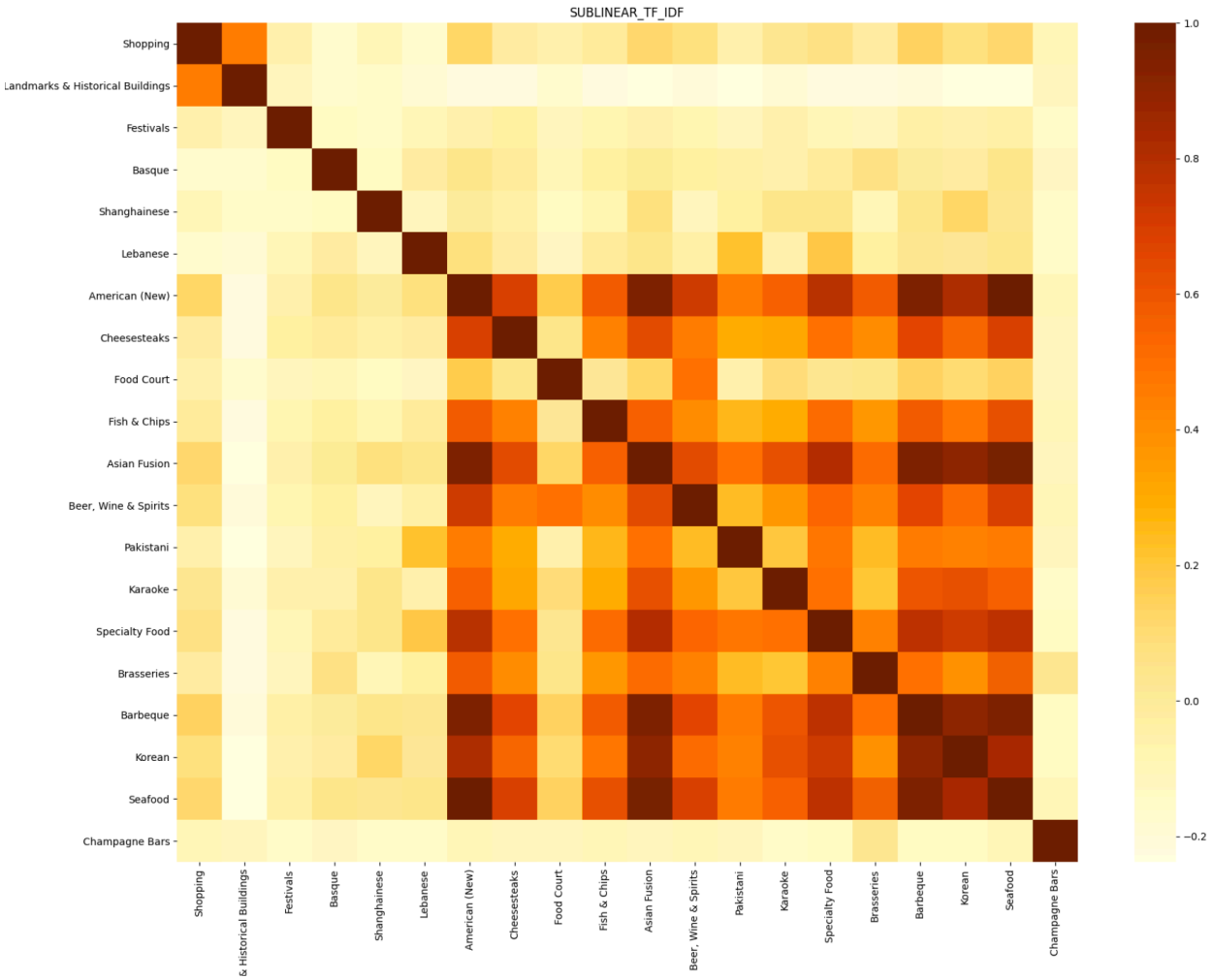
sublinear_tf (default=False) - Apply sublinear tf scaling, i.e. replace tf with 1 + log(tf).

Setting below parameter (cosine similarity):

- smooth_idf = True
- sublinear_tf = True
- max_df = 0.5
- min_df = 2

Sublinear tf with IDF - Comparing the result, we obtained from 'TFIDF' and this, below we can evidently see how the similarity has been improvised by adjusting the representation of text. Lower indices of cuisines have more similarity than upper indices. It seems unlikely that 20 occurrences of a term in a document truly carry twenty times the significance of a single occurrence A common modification is to use instead the logarithm of the term frequency, which assigns a weight given by:

$$\mathrm{wf}_{t,d} = \begin{cases} 1 + \log \mathrm{tf}_{t,d} & \text{if } \mathrm{tf}_{t,d} > 0 \\ 0 & \text{otherwise} \end{cases}.$$



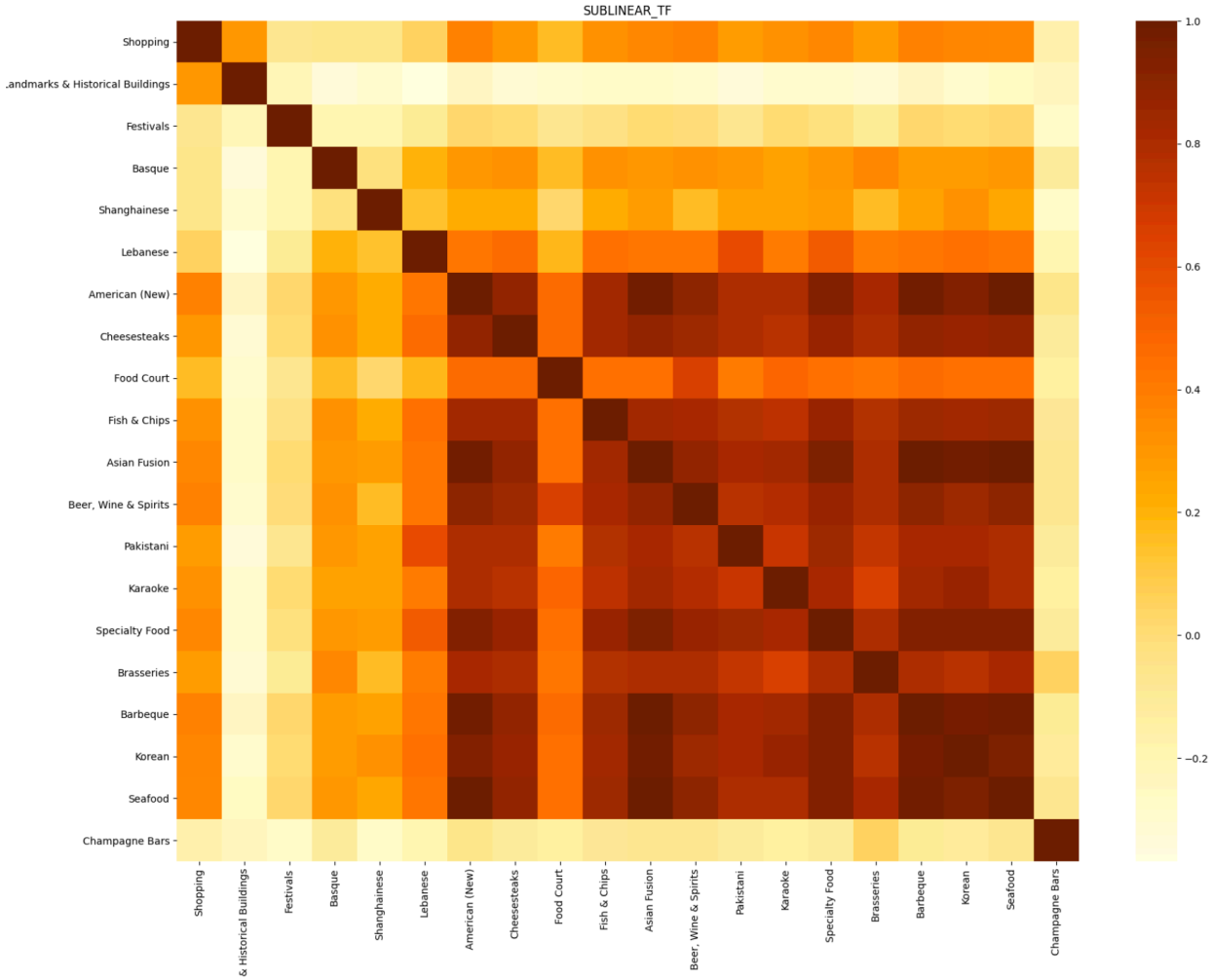
sim_matrix(idf=True, sublinear_tf=True, cmap='YlOrBr', max_df=0.5, min_df=2, method='SUBLINEAR_TF_IDF', similarity='cosine')

Setting below parameter (cosine similarity):

- smooth_idf = True
- sublinear_tf = True
- max_df = 0.8
- min_df = 4

Having the sublinear transformation set to true, tweaking max and min document frequency parameters results increasing similarity compared to output with max_df=0.5 and min_df=2.

One key thing to note is increasing intensity of opaqueness of cells in this output. The more obvious cuisines like seafood, barbeque, Pakistani, have more similarity showing below.

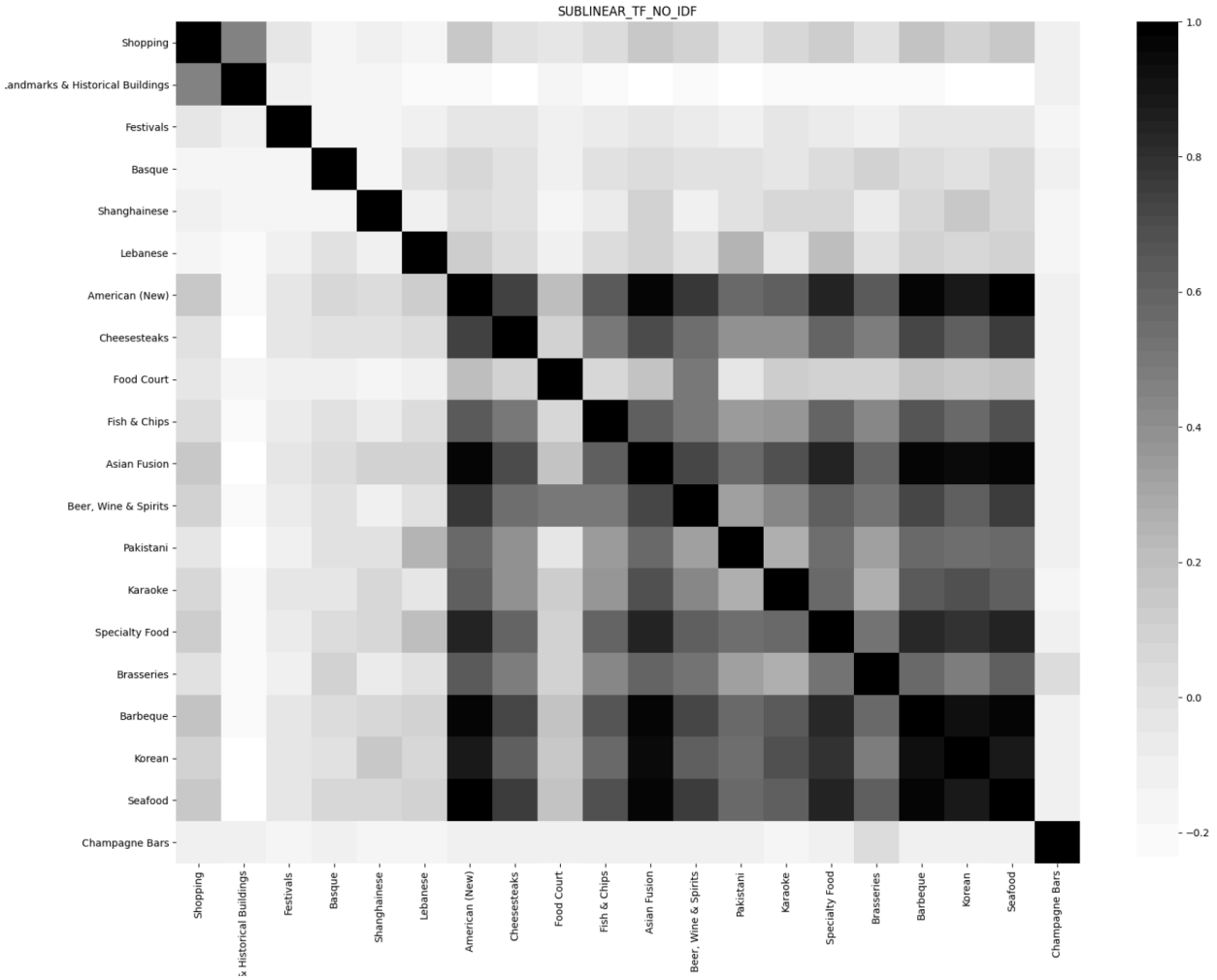



```
sim_matrix(idf=True, sublinear_tf=True, cmap='YlOrBr', max_df=0.8, min_df=4, method='SUBLINEAR_TF', similarity='cosine')
```

Setting below parameter (cosine similarity):

- smooth_idf = True
- sublinear_tf = True
- max_df = 0.5
- min_df = 2
- idf=False

Sublinear tf with noIDF - Below comparing result by applying sublinear transformation on noIDF, with that of noIDF, we observe the lower indices have got more denser similarity result, but in overall other cells in general have also got increased similarity.

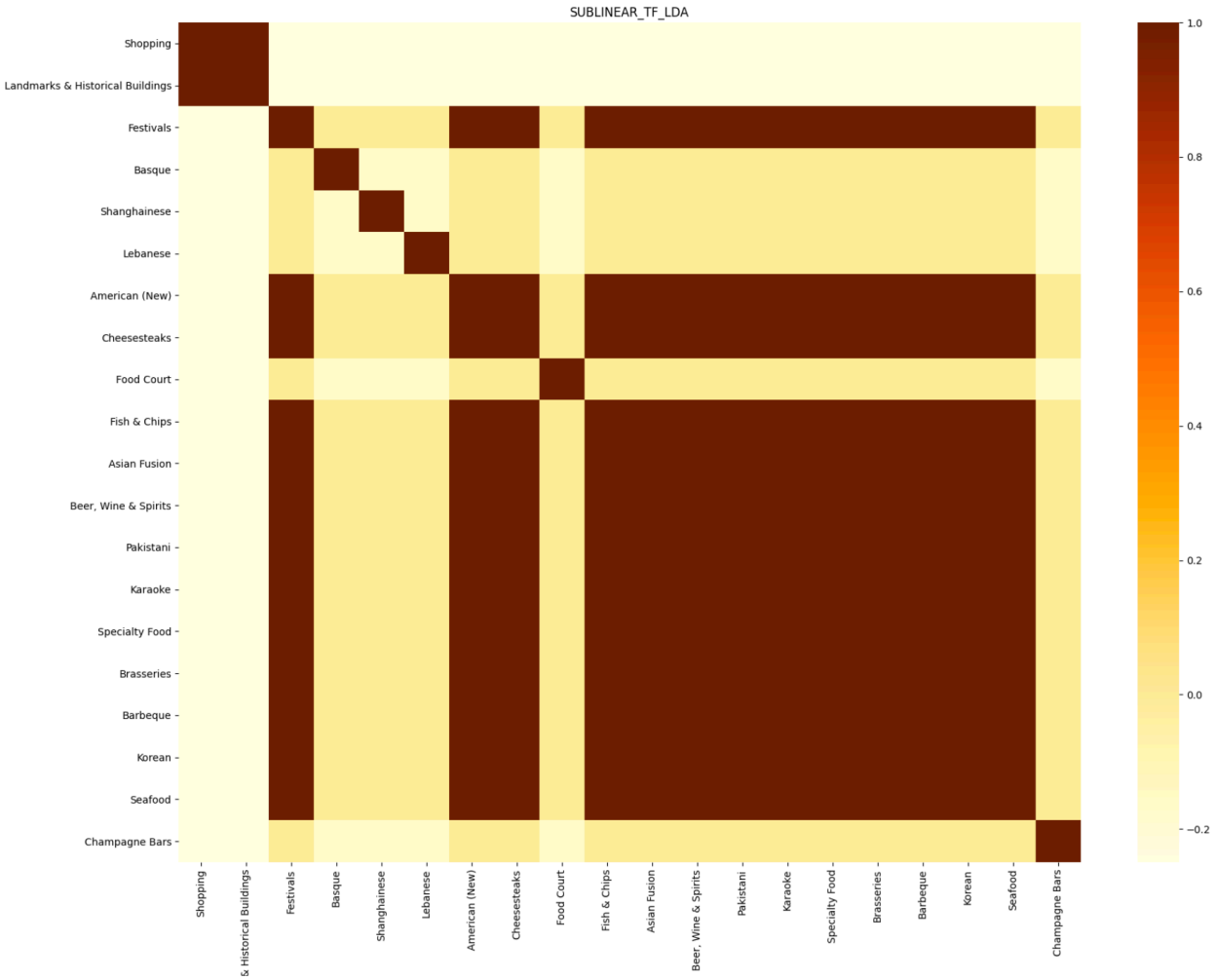



```
sim_matrix(idf=False, sublinear_tf=True,  cmap='Greys',   max_df=0.5, min_df=2,
method='SUBLINEAR_TF_NO_IDF', similarity='cosine')
```

Setting below parameter (cosine similarity):

- smooth_idf = True
- sublinear_tf = True
- max_df = 0.5
- min_df = 2

Sublinear tf with LDA – Here the results comparing to LDA does not seem very convincing as the lower indices or any significantly matching cells have got complete similarity and other minimum similarity cells also have got boost. The results seems to be overfitting.



sim_matrix(idf=True, sublinear_tf=True, cmap='YlOrBr', max_df=0.5, min_df=2, method='SUBLINEAR_TF_LDA', similarity='cosine')

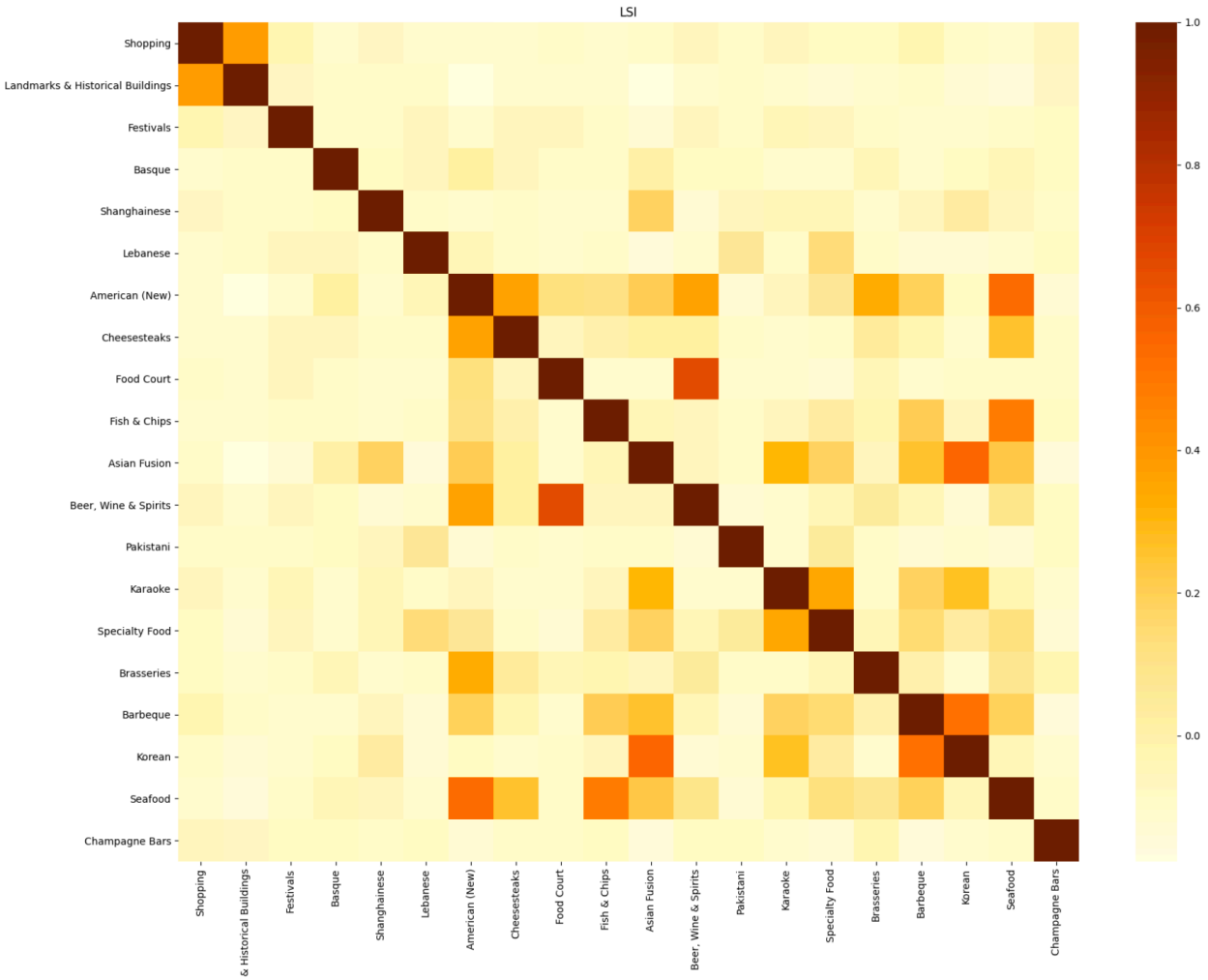
Varying the similarity function

Usually tf-idf/bag-of-words matrices contain a lot of noise. Applying LSI model can help with this problem, so we can achieve better quality similarities. But in this case, we try with TFIDF and LSI and see what results we get and we shall compare to that of TFIDF or LDA.

Let's model with LSI and other parameters as default (cosine similarity):

- max_df=0.5
- min_df=2
- smooth_idf=True
- sublinear_tf=False
- number of topics = 50

Here when just comparing with TFIDF, there is visually no difference that's evident. But compared to LDA this method due to obvious differences in computation on latent semantic analysis grounds, shows the output puts the weightage differently in few numbers of the cells decreasing the similarity strength.



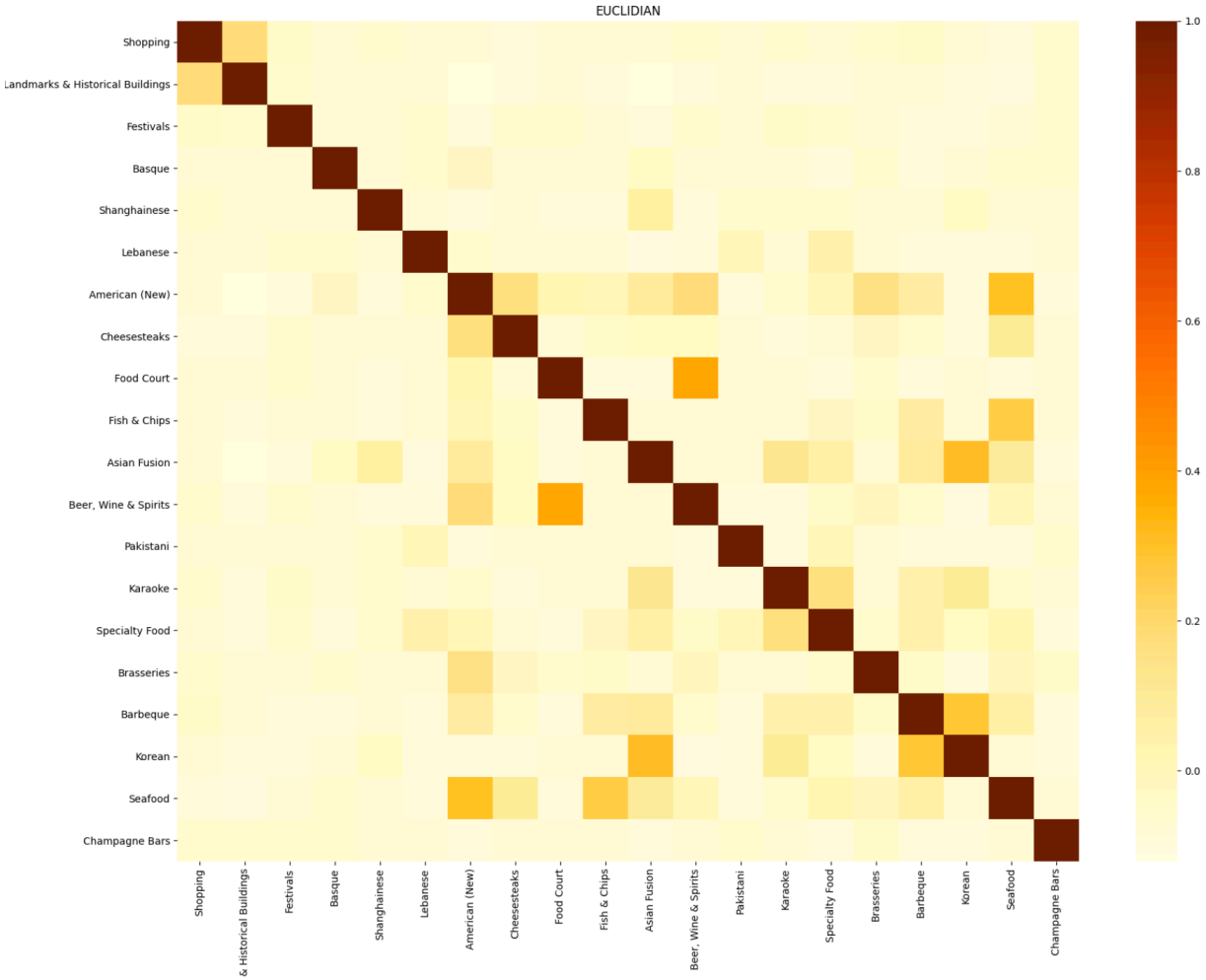

```
sim_matrix(idf=True, sublinear_tf=False, cmap='YlOrBr', max_df=0.5, min_df=2, method='LSI', similarity='cosine')
```

Let's model with Euclidian distance (TFIDF)

- max_df=0.5
- min_df=2
- smooth_idf=True
- sublinear_tf=False
- number of topics = 50

What we observe is there is not much change comparing to 'TFIDF' with cosine distance in how this changes the similarity between the cuisines, rather it increases the probability

(opaqueness) and makes it denser. But overall similarity remains pretty much same. As we know, due to normalization, all the TF-IDF vectors point into the same segment (positive values) and have similar weights, so this explains the behavior.



sim_matrix(idf=True, sublinear_tf=False, cmap='YlOrBr', max_df=0.5, min_df=2, method='EUCLIDIAN', similarity='euclidian')

Task 2.3: Incorporating Clustering in Cuisine Map

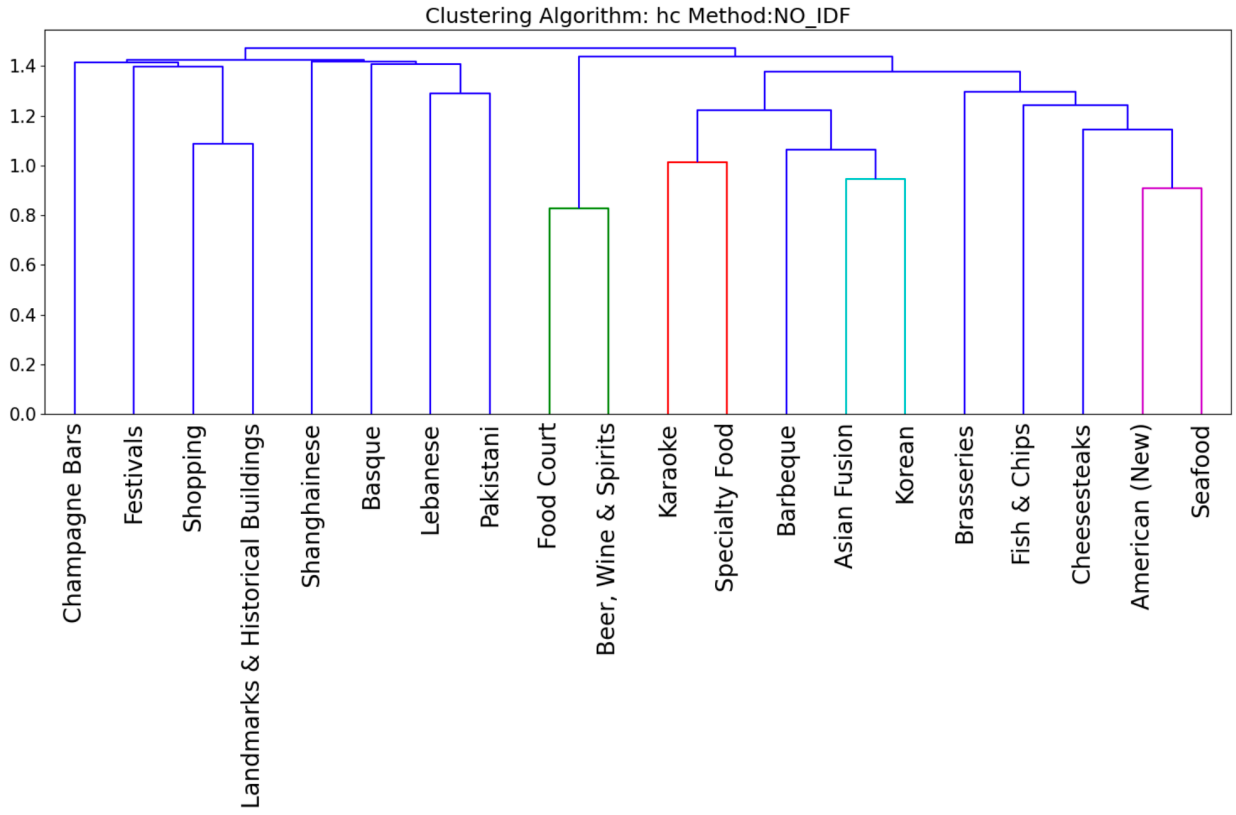
Trying the hierarchy clustering and visualization with dendrogram – "average" link

I prefer the ease in readability and closely relate what cuisines are similar and how the overall clusters are formed threshold after threshold. The results below are very clear.

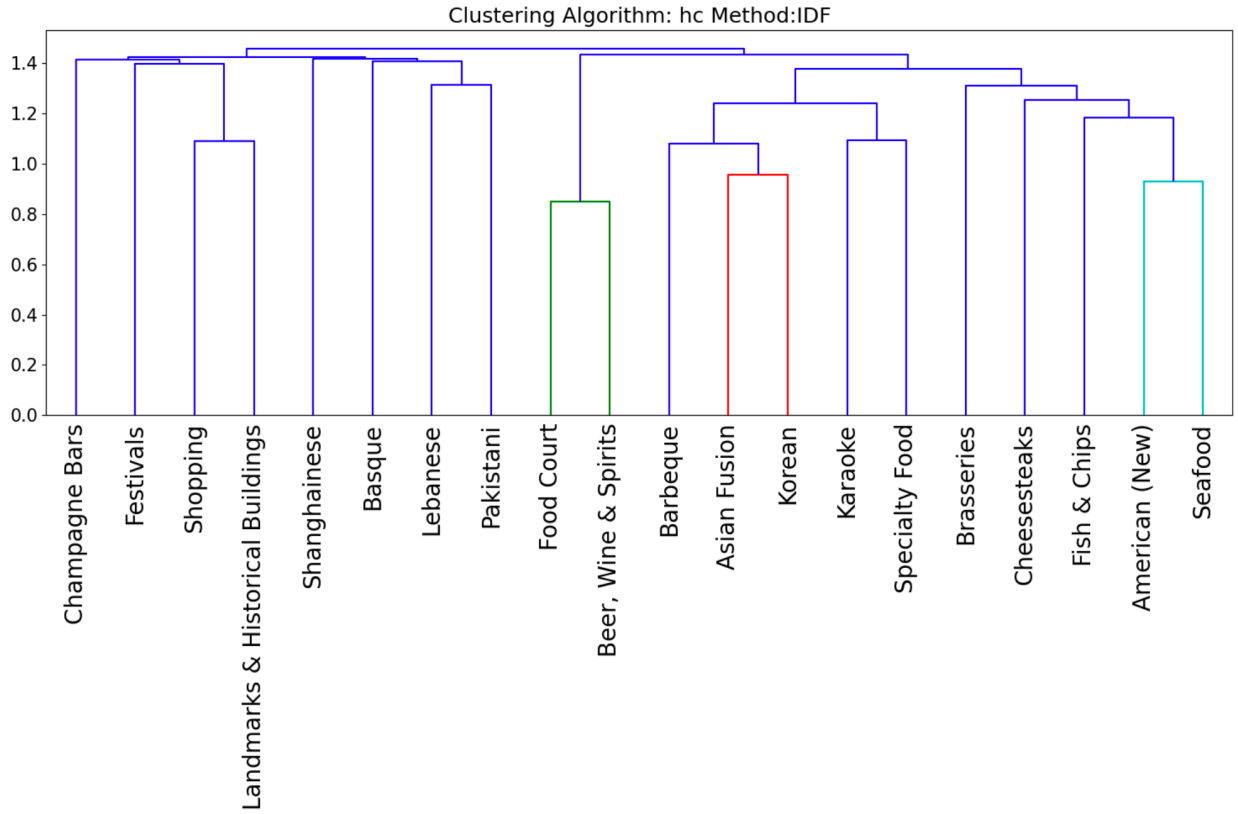
From Image 1 & 2: As we can see first clustering is Food court and beer, and in nature of the cuisines, they seem to be pretty close. Also, Asian fusion and Korean, makes sense as they would have similar food offerings. Fish chips, seafood, American as we go up have been

clustered together. There are some similarities between Lebanese and Pakistani, but I would not bet over 30%. Clustering of shopping and land marks is also convincing.
There is not any difference in clustering between IDF and no IDF.

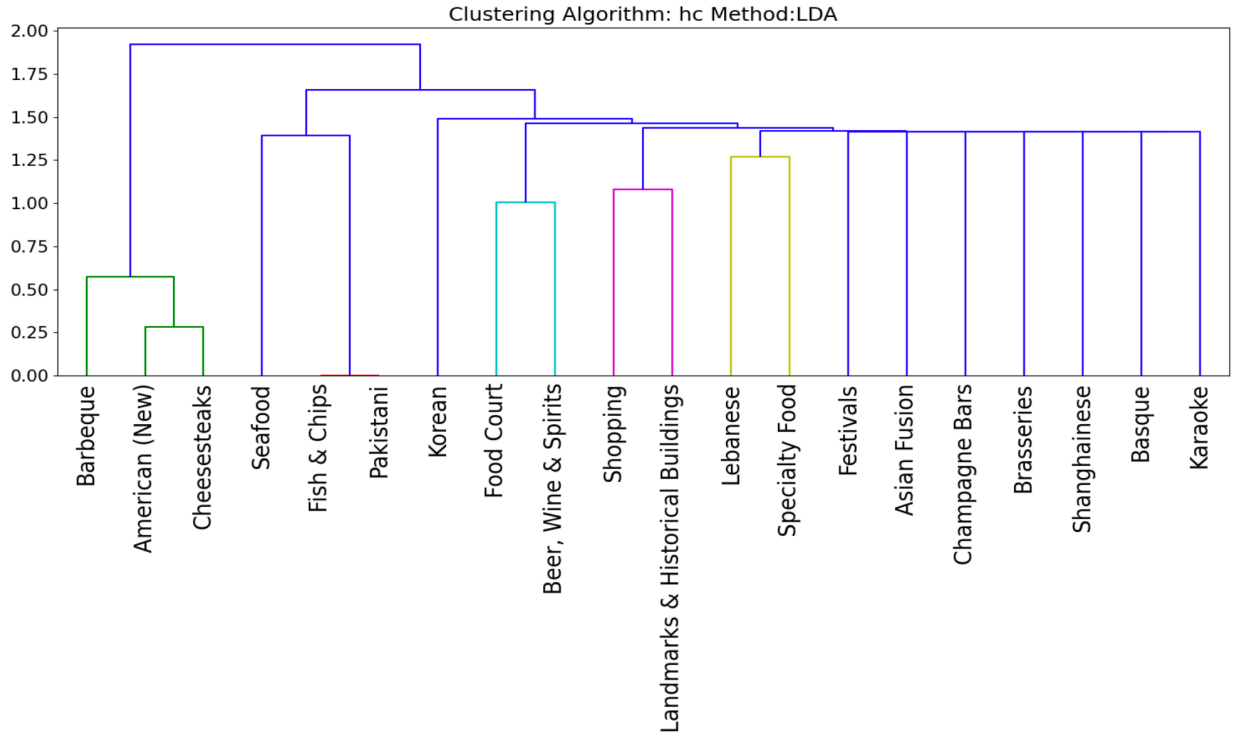
I like how we have got the clustering applied with LDA. Barbeque, American and cheesesteaks fall in same cluster, food court and beer in same, sensible combinations based on similarity.



sim_matrix(idf=False, sublinear_tf=False, cmap='Greys', max_df=0.5, min_df=2, method='NO_IDF', c_algo='hc')



sim_matrix(idf=True, sublinear_tf=False, cmap='YlOrBr', max_df=0.5, min_df=2, method='IDF', c_algo='hc')

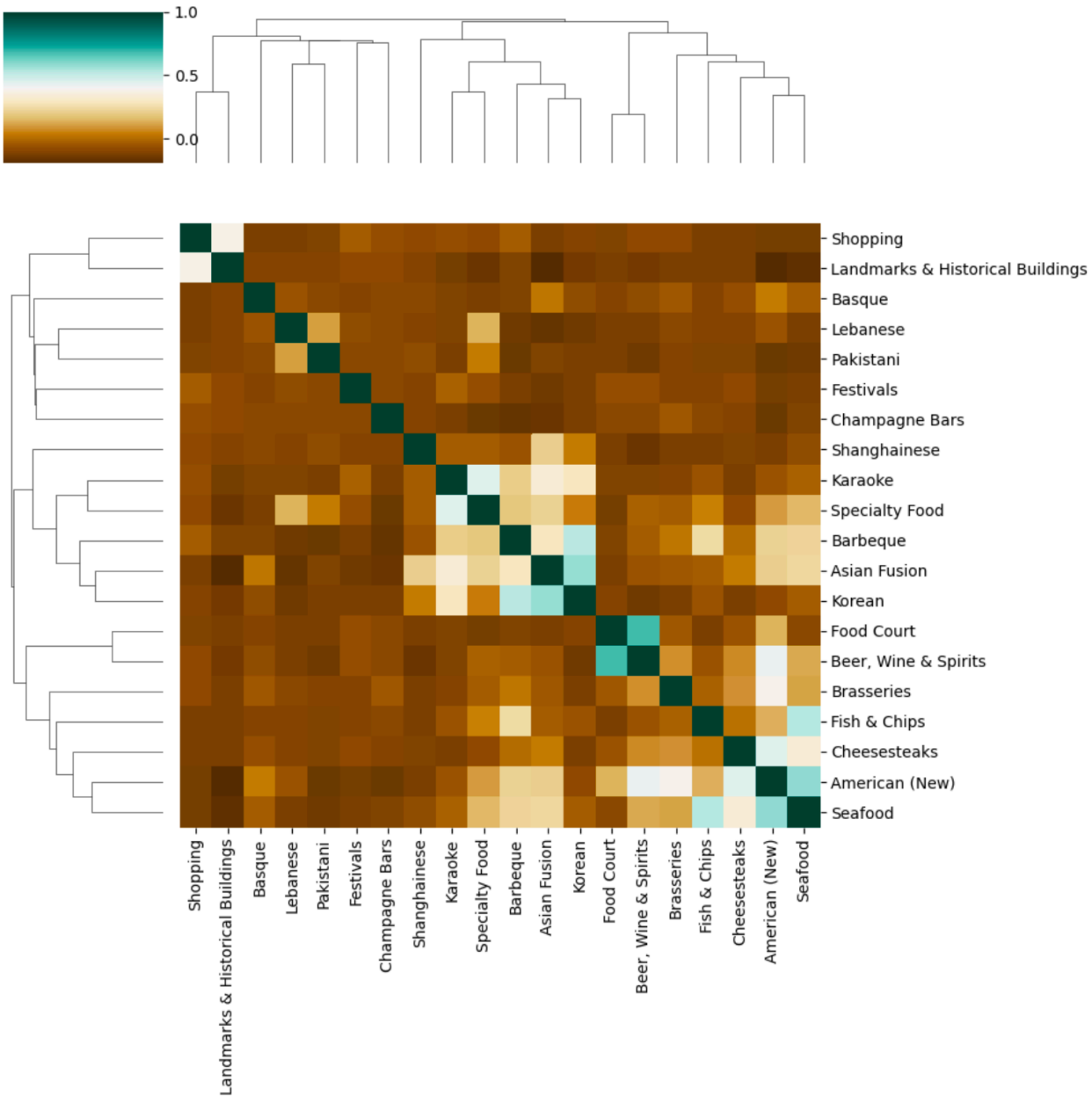


sim_matrix(idf=True, sublinear_tf=False, cmap='YlOrBr', max_df=0.5, min_df=2, method='LDA', c_algo='hc')

Spectral clustering with 5 clusters

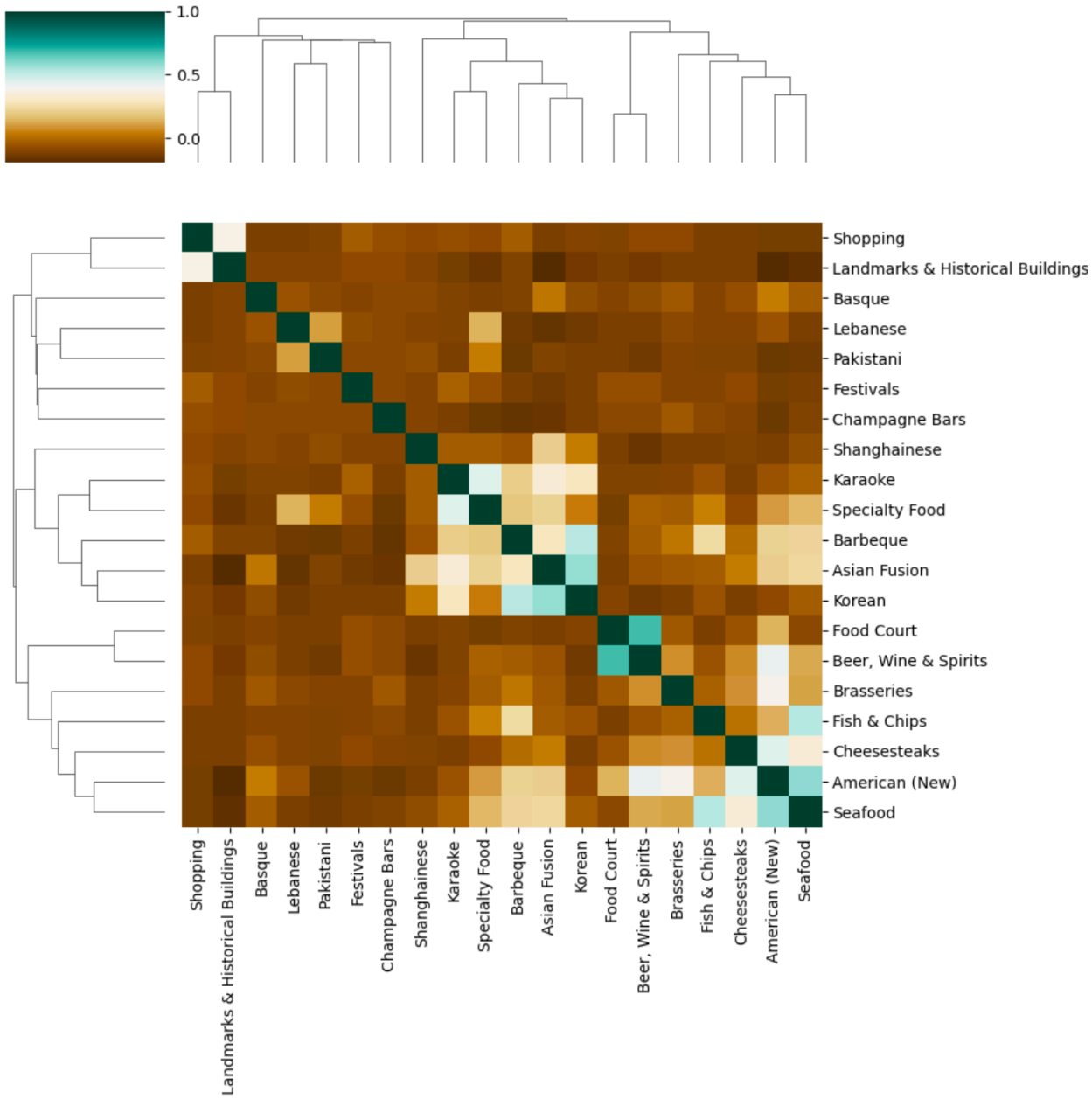
Applying classical clustering techniques to data might not work very well. Having too many dimensions, the number of words in lexicon, to reasonably do any clustering of the documents. Hence ot make it work well, directly with a matrix of similarity scores; it is called spectral clustering.

The spectral clustering splits our documents into number of clusters groups and returns the id of each group.



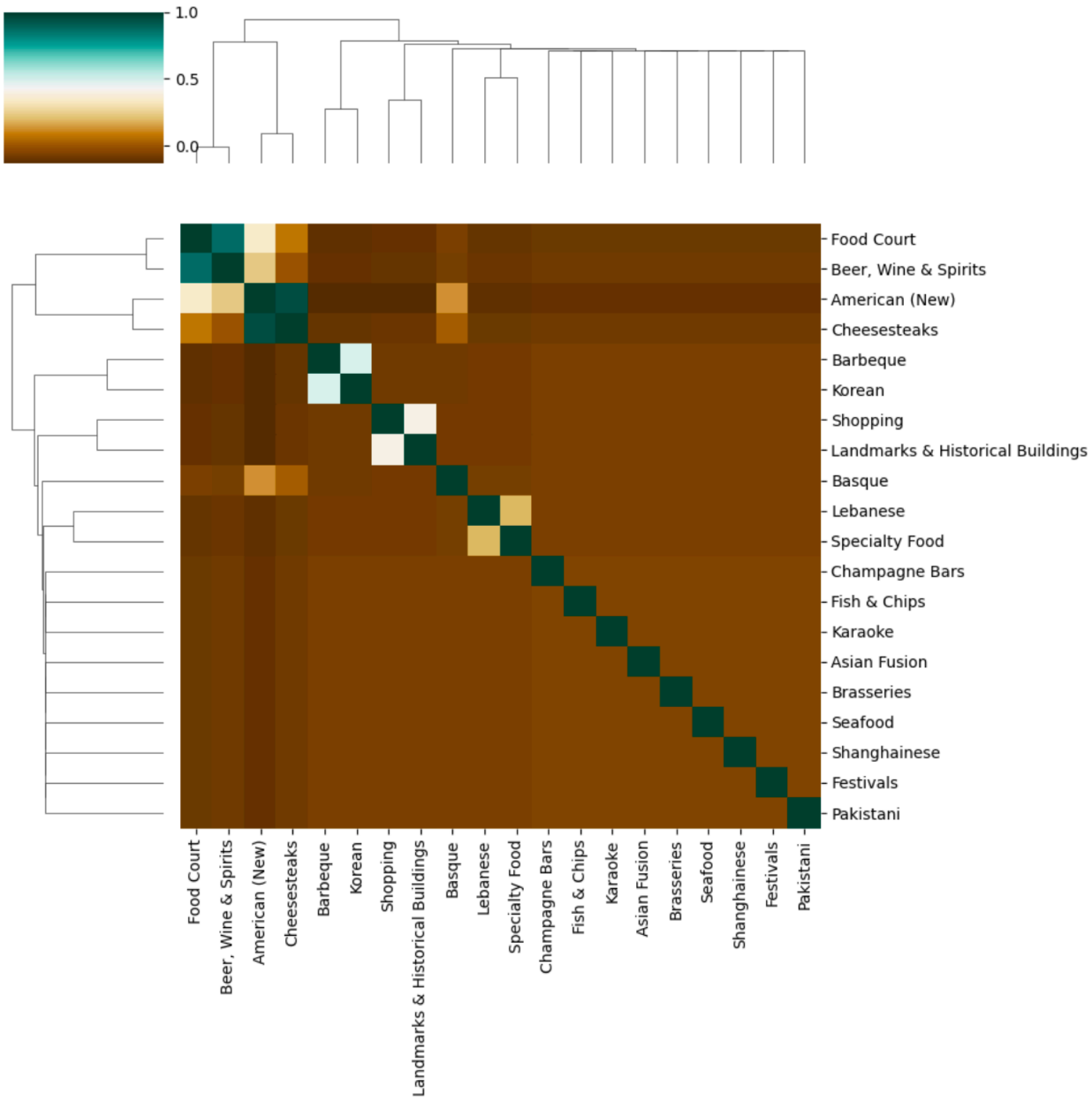
Clusters labels - [3 3 2 0 2 1 0 0 4 0 2 4 1 2 2 0 2 2 0 0] – NO IDF

sim_matrix(idf=False, sublinear_tf=False, cmap='Greys', max_df=0.5, min_df=2, method='NO_IDF', n_clusters=5, c_algo='spectral')



Cluster labels - [3 3 4 0 4 2 0 0 1 0 4 1 2 4 4 0 4 4 0 0] – IDF

sim_matrix(idf=False, sublinear_tf=False, cmap='Greys', max_df=0.5, min_df=2, method='IDF', n_clusters=5, c_algo='spectral')



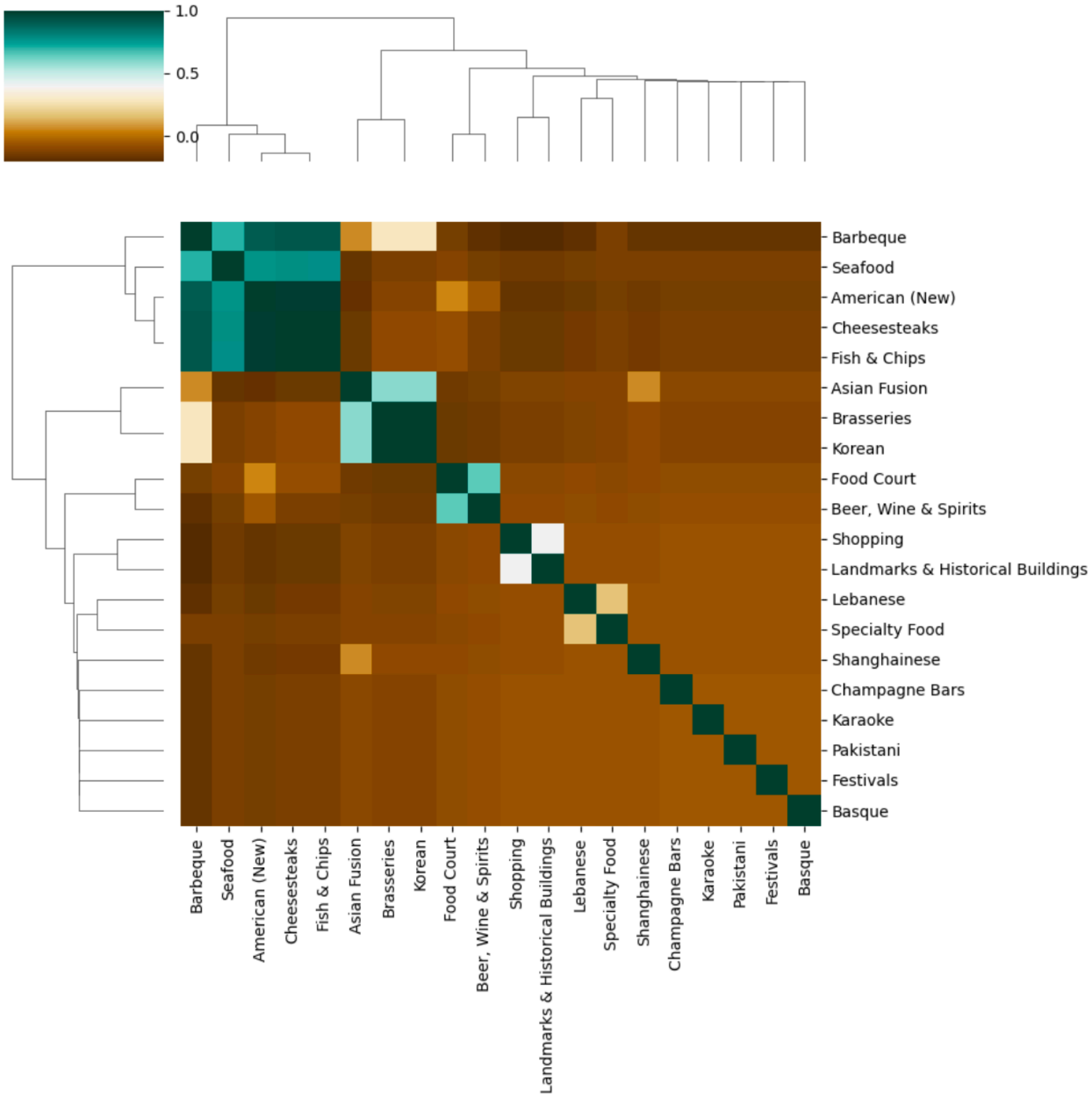
Cluster labels - [0 0 3 3 3 2 3 3 1 3 3 1 3 3 2 3 4 4 3 3] – LDA

```
sim_matrix(idf=False, sublinear_tf=False,  cmap='Greys',  max_df=0.5, min_df=2, method='LDA', n_clusters=5,
c_algo='spectral')
```

Spectral clustering with 3 clusters

I am surprised that changing the cluster to 3, there is no obvious difference in the visualization for IDF/noIDF with that compared to 5 clusters. However even the results in LDA has just lowered most of the lower indicies similarity and for sure its not wise to say cluster 3 is an optimal value

here. Since it clusters all the lower indices together. Its interesting to compare this to heirarchical clustering.



Cluster labels - [2 2 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0] – LDA

Inspirations

https://matplotlib.org/3.2.1/tutorials/colors/colormaps.html
https://www.absentdata.com/python-graphs/create-a-heat-map-with-seaborn/
https://seaborn.pydata.org/generated/seaborn.heatmap.html

https://github.com/igor-sokolov/dataminingcapstone/blob/master/Capstone%20project%202.ipynb
https://scipython.com/book/chapter-6-numpy/examples/vstack-and-hstack/
http://text2vec.org/similarity.html#cosine_similarity_with_lsa