Cuisine Clustering and Map Construction
(Analysis of Yelp dataset)

1. Data Preparation

The dataset contains 42,153 businesses, of which 14,303 are restaurants (identified by presence of 'Restaurants' category). After removing all businesses without 'Restaurants' category, the dataset has 1,125,458 reviews in total with 706,646 reviews for the restaurants.

Initial review of the categories for restaurants showed 240 categories with some of these categories assigned very few businesses, most with irrelevant labels like "Dry Cleaning & Laundry" and "Sporting Goods". For similarity analysis I took top 50 categories (based on number of restaurants per category). From this point I will refer to these categories as cuisines. This left 13,647 restaurants with 681,590 reviews. Note that since some restaurants list more than one cuisine, when projecting reviews onto cuisines, reviews for those restaurants end up in more than one cuisine, which brings the grand total number of reviews across all cuisines to 1,332,178. It's important to note that while in previous work (Week 1) we needed to avoid these duplicates to build a topic model across all reviews, here we are computing a similarity model between cuisines. Thus, same review present for more than one cuisine actually helps us identify similar cuisines, so this is a benefit, not a problem.

On import I converted each review to lower case, removed punctuation and stopwords, and applied stemming (using Porter algorithm).

2. Visualization of the Cuisine Map (Task 2.1)

I first concatenated all reviews for each category, ending up with 50 very large documents. Then I used a TF-IDF vectorizer to project these documents onto vector space (using up to 10000 features, and ignoring words that appear in more than 50% of the documents or less than 2 times). Then I used LDA with 100 topics, running 10 passes with 1,000 iterations, and computed similarity matrix using Euclidean distance, converted to similarity as 1/(1+distance) to give measures in [0,1] range, on the results (Fig. 1). Since the matrix is somewhat hard to read, I also produced a bubble chart, showing cuisine pairs with similarity above 0.8 (Fig. 2)

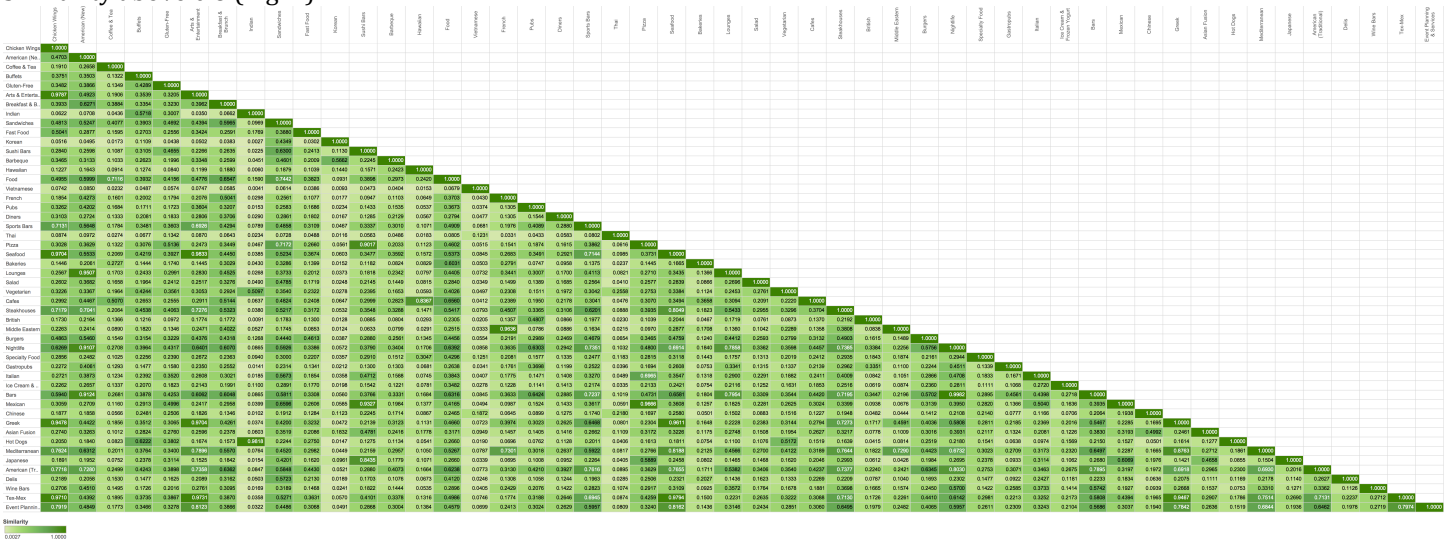

Fig. 1. Similarity Matrix based on 1-grams and Euclidean similarity

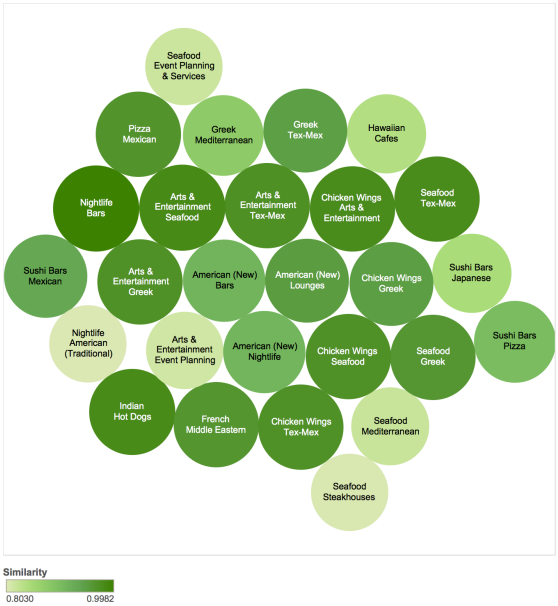

Fig. 2. Bubble chart of most similar cuisine pairs (1-grams and Euclidean similarity)

3. Improving the Cuisine Map (Task 2.2)

I looked at 3 possible improvements:

1. Using 1-, 2- and 3-grams for tokenization, when projecting documents onto vector space (varying text representation)
2. Using cosine similarity (varying similarity function)
3. Instead of 50 large documents, keeping each review as individual document, computing similarities between individual reviews, then aggregating from review-review similarities to cuisine-cuisine similarities (varying both text representation and similarity calculation)

First, I re-calculated the similarity matrix, using cosine similarity (Fig. 3). Next, I decided to vary the text representation, by using 1-, 2-, and 3-grams (same parameters for TF-IDF vectorizer, just adding n-grams), and also calculated similarity matrices, using first Euclidean distance (Fig. 4) and cosine similarity (Fig. 5). Cosine similarity produced better results, as expected: Euclidean measure is not very good for text-based vectors, because Euclidean measure focuses on magnitude of the vectors, which in case of text comparison is less important than the direction of the vectors, which we capture with cosine similarity. While using Euclidean similarity,

going to n-grams gave us worse results than 1-grams, producing less pronounced groupings, going to n-grams with cosine similarity gave the best result of the four.

For example, let's consider Indian cuisine (which rarely gets grouped with anything else). In 1-gram Euclidean, it got very high similarity with Hot Dogs (very strange), and somewhat smaller with Buffets and Vegetarian (probably the only 2 categories that make sense, since many Indian restaurants in US are buffet-style and there's a large selection of vegetarian dishes in Indian cuisine). In n-gram Euclidean, it's marked as similar to almost every category. In 1-gram cosine, it's not much different from 1-gram Euclidean, but in n-gram cosine Hot Dog similarity is close to 0 and Vegetarian is the top similarity, with Buffet as 2nd, which looks very good.

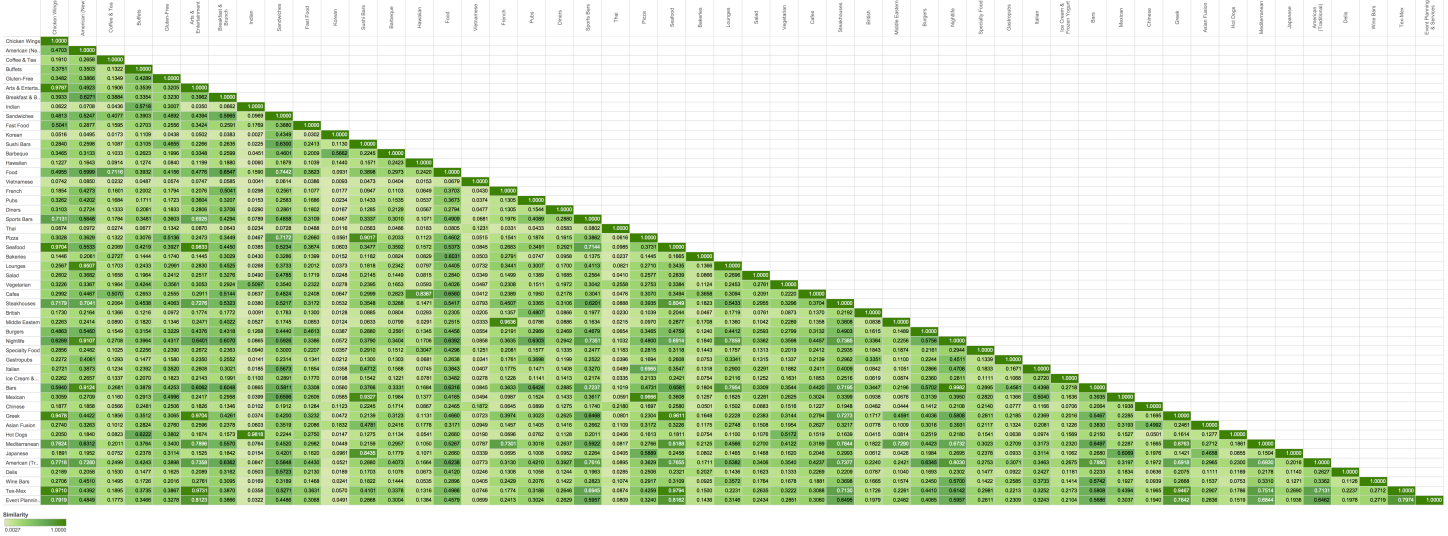

Fig 3. Similarity Matrix based on 1-grams and cosine similarity

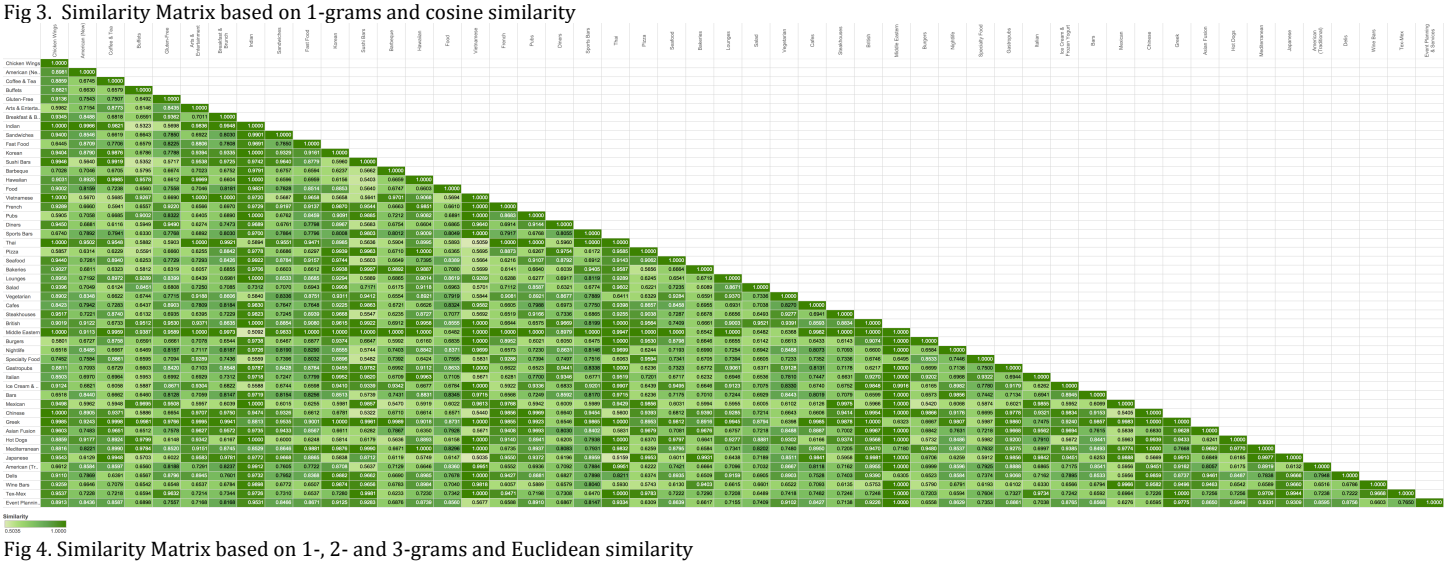

Fig 4. Similarity Matrix based on 1-, 2- and 3-grams and Euclidean similarity

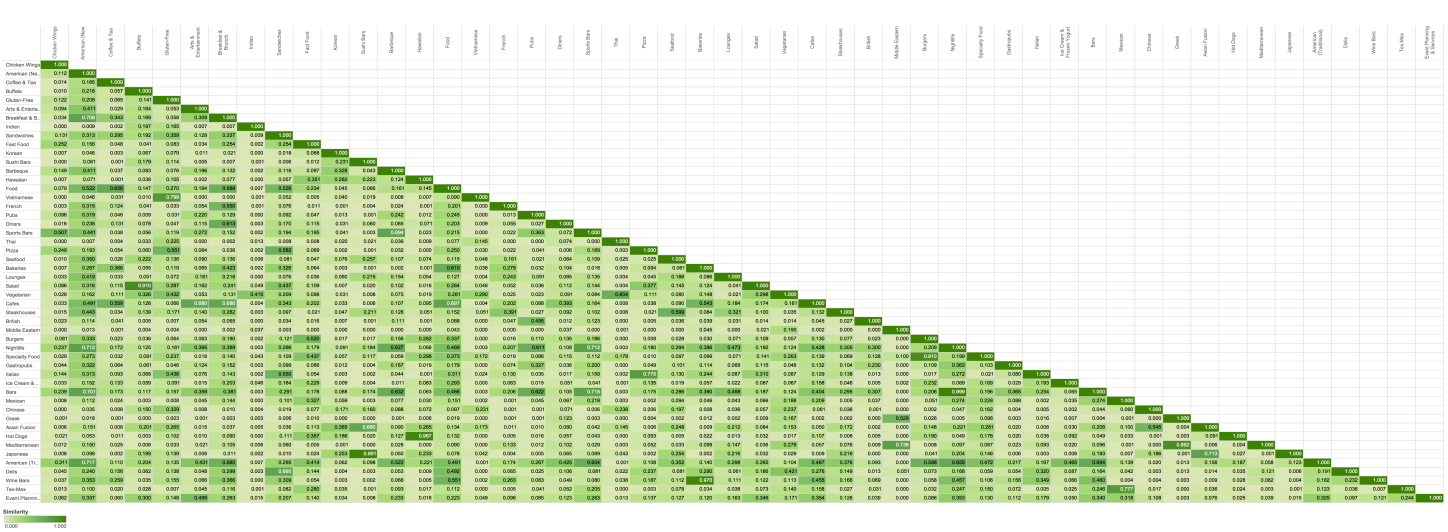

Fig. 5. Similarity Matrix based on 1-, 2- and 3-grams and cosine similarity

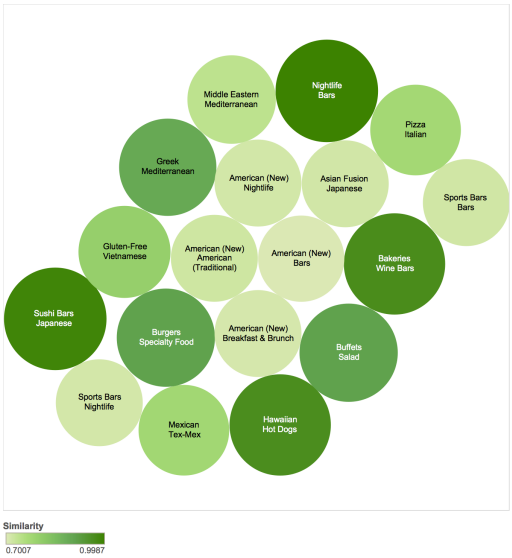

Fig. 6. Bubble chart of most similar cuisine pairs (n-grams and cosine similarity)

I also produced the bubble chart showing most similar cuisine pairs (Fig. 6). I also tried to do similarity matrix between individual reviews, to aggregate similarities to category level from individual review pairs, however that has proven computationally prohibitive (1,332,178 reviews would mean n*(n-1)/2 = 887,348,445,753 similarities to compute. I ran it for 20 hours on my laptop before having to kill it. This is a good calculation to run on a distributed cluster, since it's easy to parallelize, but I don't have one at hand).

4. Incorporating Clustering in Cuisine Map (Task 2.3)

Since similarity matrix built based on n-grams and cosine similarity looked most promising, that's the one I proceed with. Having a pre-computed similarity matrix is a good input for clustering. First I used Affinity Propagation algorithm, which automatically determines the number of clusters by iteratively examining each pair of cuisines to determine if one is a good representation of the other, until convergence. The algorithm resulted in 12 clusters, as follows (Fig. 7). It's not a bad split, but I also wanted to try forcing the number of clusters. So, I used KMeans (by way of passing my pre-computed similarity matrix to Spectral Clustering algorithm), with 5, 6, 7 and 10 clusters (Fig. 8).

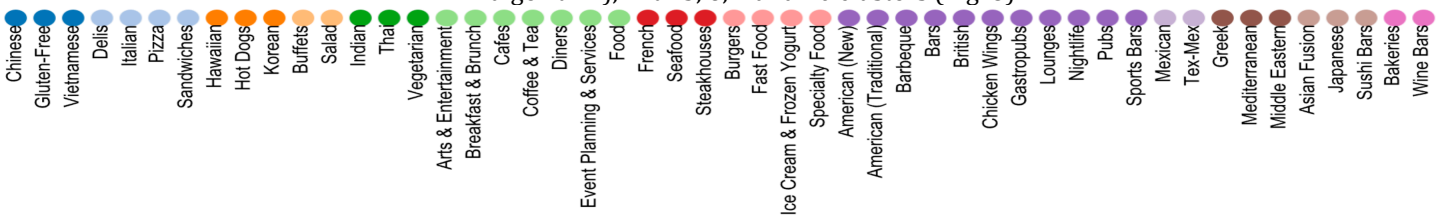

Fig. 7. 12-cluster grouping with Affinity Propagation

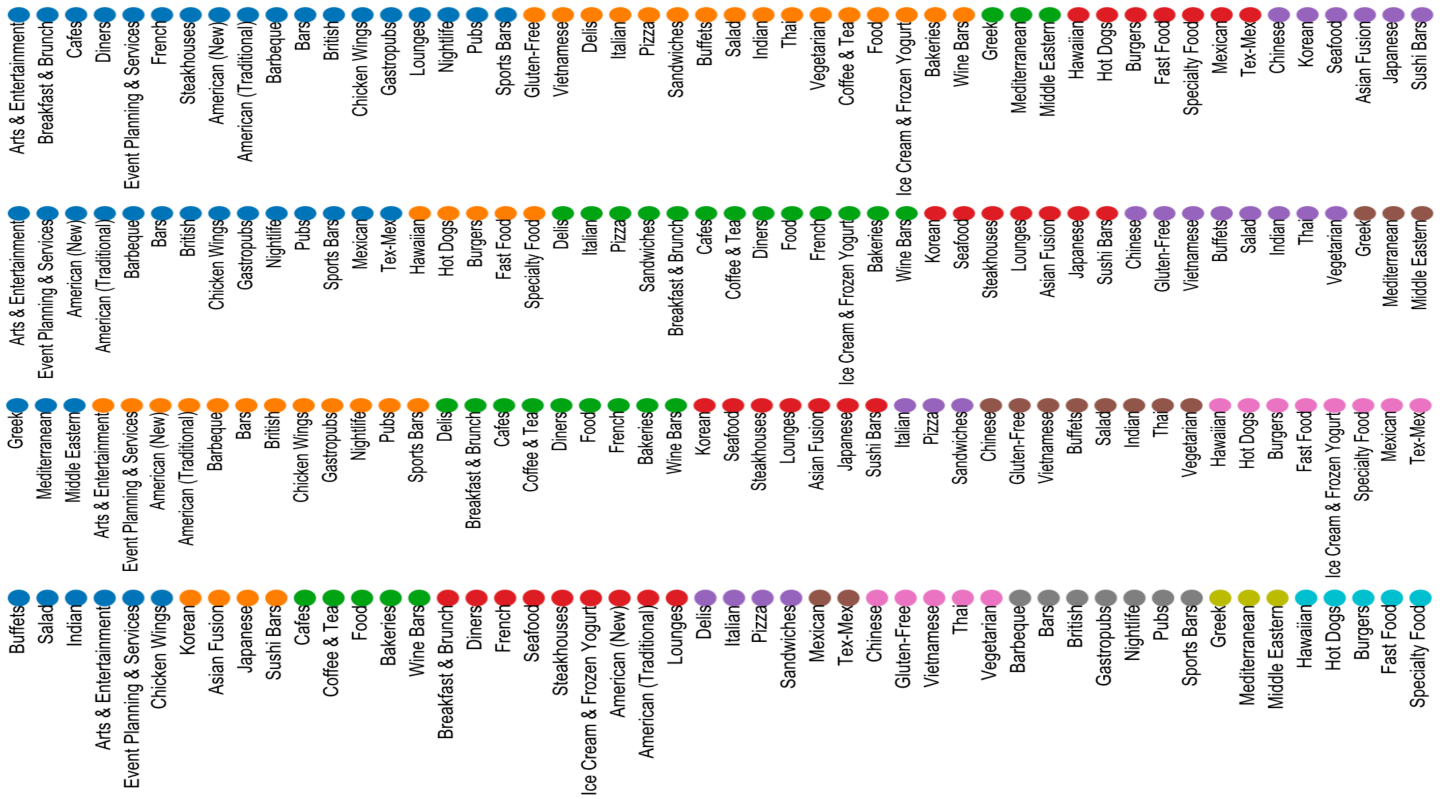

Fig. 8. 5, 6, 7 and 10 cluster grouping with KMeans

The most interesting observation here is that Greek/Mediterranean/Middle-Eastern is the most stable cluster that gets identified early on and stays a separate cluster. And both algorithms identify it. Comparing the clusters, the 12-cluster split made by Affinity Propagation made the most sense to me. You can see very clearly identified groups (from right to left on Fig. 7): Asian, Lunch/Pizza, Lunch, Indian, Generic Service Categories, Dinner, Lunch/Snack, Bars, Mexican, Mediterranean, Japanese, Other.

Applying cluster colors to similarity matrix does not make a useful visual (Fig. 9), so in the interest of space I did not include matrices with KMeans cluster colors. Instead, see Fig. 10-13 for bubble charts showing cuisines with cluster colors; bubble size is proportional to number of restaurants providing that cuisine.

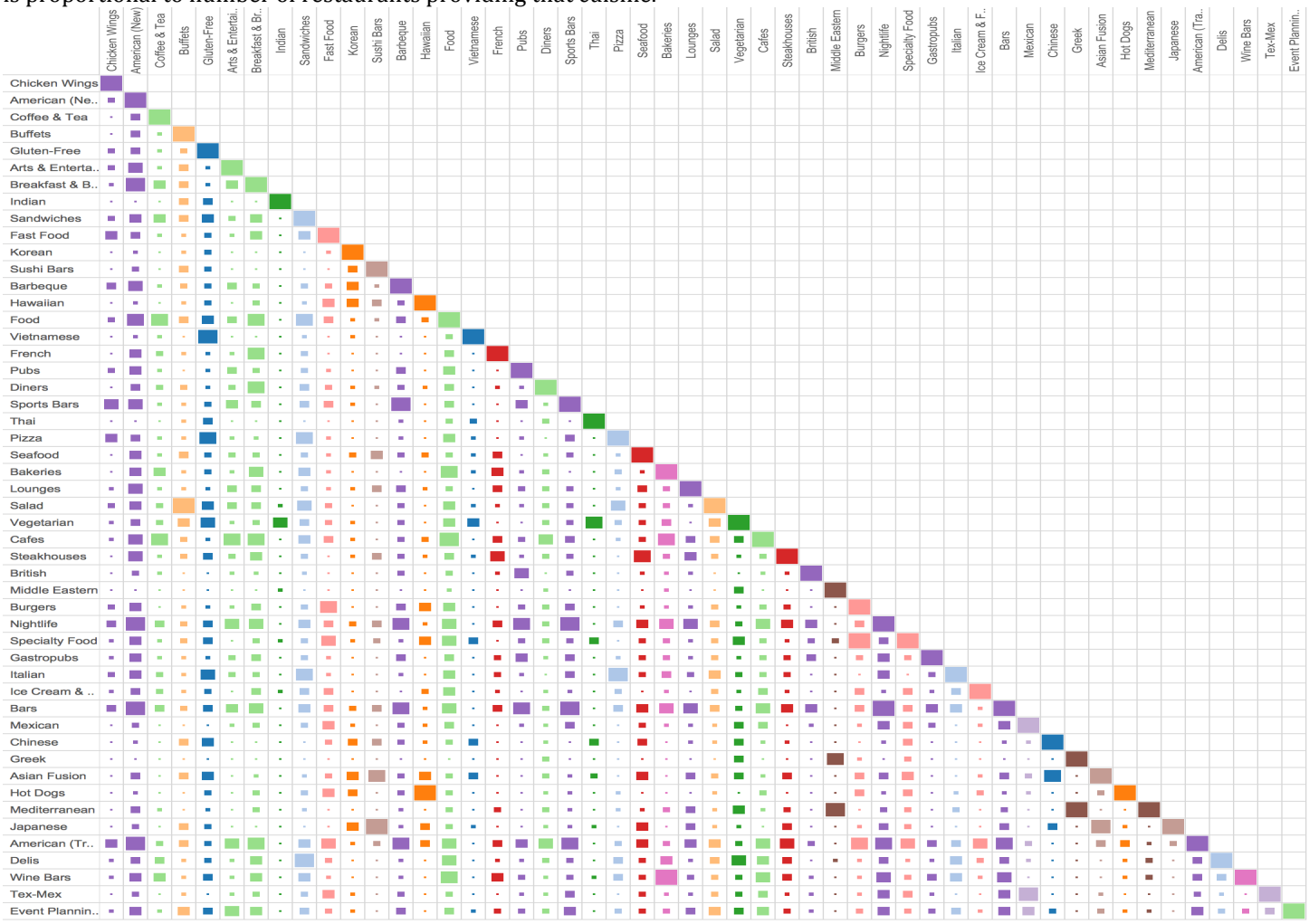

Fig 9. Similarity Matrix with 12 clusters (size of the box corresponds to similarity measure)

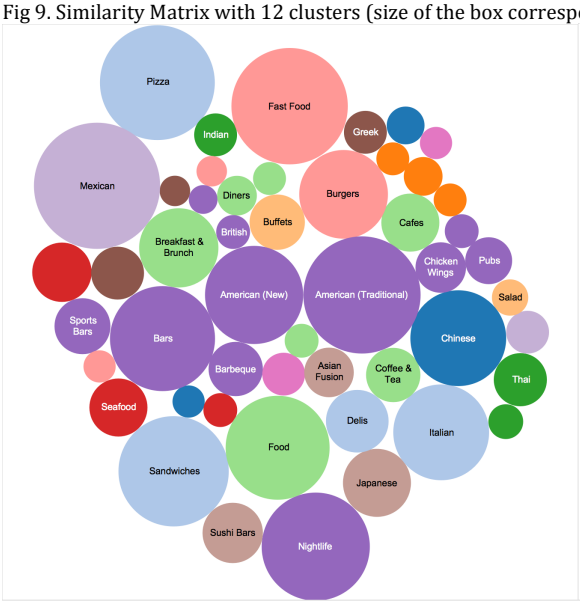

Fig. 10. Affinity Propagation clustering

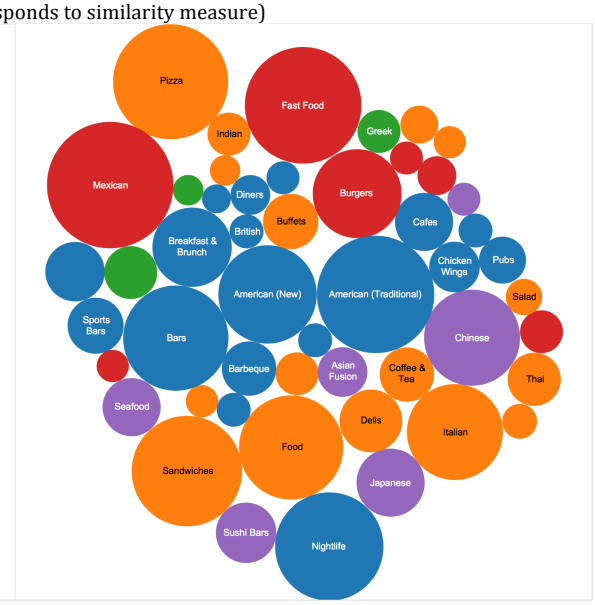

Fig. 11. KMeans clustering (5 clusters)

Fig. 12. KMeans clustering (7 clusters)

Fig. 13. KMeans clustering (10 clusters)

How is this useful? Well, we could use it for improving a search system, for example. When someone searches for a restaurant for specific cuisine, and there's not enough restaurants meeting the search criteria (not enough close by, or not enough high-rated restaurants), we could suggest restaurants with cuisines from the same cluster that meet other search criteria. And looking at bubble charts (Fig. 10), helps us see that someone looking for a Pub is much more likely to find a place they like if we also show Bars, or that someone looking for Middle-Eastern cuisine is much more likely to find a place if we also show Mediterranean (follow the link to the interactive version of these graphs at the end of the document to see this one).

5. Tools Used

Data processing: Python with nltk, gensim and sklearn libraries
Visualization: Tableau, with publishing to TableauPublic
Processing results were exported to CSV files from Python for visualization in Tableau.
Interactive version of the charts is available at:

- Fig. 1: https://public.tableau.com/views/Task2_6/SimilarityMatrix1-gramEucledian?:embed=y&:display_count=yes&:showTabs=y&:toolbar=no
- Fig. 2: https://public.tableau.com/views/Task2_6/MostSimilarPairs1-gramEuclidean?:embed=y&:display_count=yes&:showTabs=y&:toolbar=no
- Fig. 3: https://public.tableau.com/views/Task2_6/SimilarityMatrix1-gramsCosine?:embed=y&:display_count=yes&:showTabs=y&:toolbar=no
- Fig. 4: https://public.tableau.com/views/Task2_6/SimilarityMatrix3-gramEuclidean?:embed=y&:display_count=yes&:showTabs=y&:toolbar=no
- Fig. 5: https://public.tableau.com/views/Task2_6/SimilarityMatrix3-gramcosine?:embed=y&:display_count=yes&:showTabs=y&:toolbar=no
- Fig. 6: https://public.tableau.com/views/Task2_6/MostSimilarPairs3-gramcosine?:embed=y&:display_count=yes&:showTabs=y&:toolbar=no
- Fig. 9: https://public.tableau.com/views/Task2_6/SimilarityMatrixwithClusters?:embed=y&:display_count=yes&:showTabs=y&:toolbar=no
- Fig. 10: https://public.tableau.com/views/Task2_6/AffinityPropagation?:embed=y&:display_count=yes&:showTabs=y&:toolbar=no
- Fig. 11: https://public.tableau.com/views/Task2_6/KMeans5?:embed=y&:display_count=yes&:showTabs=y&:toolbar=no
- Fig. 12: https://public.tableau.com/views/Task2_6/Kmeans7?:embed=y&:display_count=yes&:showTabs=y&:toolbar=no
- Fig. 13: https://public.tableau.com/views/Task2_6/KMeans10?:embed=y&:display_count=yes&:showTabs=y&:toolbar=no