Nova Information Management School
BSc in Data Science

Text Mining 2024-2025

Group Project: *"Solving the Hyderabadi Word Soup"*

Project Report

Group 12

João Ferreira, 20221912
Miguel Mendes, 20221904
Rodrigo Maia, 20221934
Maria Mendes 20211624
Ricardo Coelho, 20211633

Table of Contents

1. Introduction

In today's digital world, online free-text reviews have become increasingly crucial for extracting insights into customer satisfaction. This relevance extends to all industries, but it becomes even more critical in restaurants since a numerical evaluation does little to describe the different aspects that characterize the client experience. Text mining can harness this data to uncover relationships and associations that would otherwise go unnoticed.

This project, which meticulously analyzed 10000 Zomato reviews and 105 restaurant profiles from Hyderabad (state of Telangana, India), was designed to address specific information requirements. A restaurant's Zomato score is primarily predicted based on review polarity, and its cuisine type is classified using the reviews' content. Furthermore, we sought to find and explore possible relationships and clusters between dishes mentioned together and understand whether the reviews can be divided into several topics while also trying to comprehend the meaning and importance of the extracted topics.

The knowledge acquired during this semester will be applied throughout this project, from the data understanding phase to the model evaluation. The ultimate goal is to provide the Hyderabad Tourism Board with information to map the distribution of types of cuisine by borough, the relationships between the various types of cuisine and the dishes they comprise, and detect establishments that may require health and safety inspections.

This report describes the methodology and models used, the results obtained, and the insights they offer. We intend to demonstrate the potential of text mining to solve challenges with real-world applications.

2. Literature Review

In this section, we conduct a brief literature review on information extraction techniques to evaluate performance benchmarks across domains, grounding our project in existing knowledge.

We focused on using TfidfVectorizer, and paragraph vector approaches for text mining. Relevant peer-reviewed research exploring these techniques includes Le and Mikolov (2014) and Joachims (2002). These studies highlight that while traditional keyword-based techniques like TfidfVectorizer are valuable for text analysis, the paragraph vector (or doc2vec) performs better at capturing the semantic meaning of longer documents, resulting in more accurate information retrieval and classification.

Zahoor et al. (2020) also comprehensively studied sentiment analysis and classification techniques applied to restaurant reviews. Their research implemented Naïve Bayes, Logistic Regression, Support Vector Machines (SVM), and Random Forest algorithms on restaurant review data, achieving 95% accuracy using Random Forest. This work emphasizes the importance of feature engineering and pre-processing in sentiment analysis tasks. The findings underscore the value of leveraging machine learning models to extract sentiment and classify customer reviews effectively, making it directly relevant to this project's goal of analyzing Zomato reviews.

The studies mentioned above demonstrate that combining keyword-based and semantic-based techniques with robust machine-learning algorithms offers a comprehensive approach to information retrieval and text mining across various fields. Performance benchmarks such as F1-score, recall, and precision remain critical metrics for evaluating model effectiveness.

3. Data Understanding

During the data understanding phase, we focused on the more global aspects of the available data. We gained insights into the columns and addressed a few missing values per the task requirements. To enhance our understanding of the data, we used visualization techniques such as TreeMaps, word clouds, and bar plots with the word frequency.

Finally, we merged our two datasets by adding the cuisine information to the dataset with the reviews.

31. Sentiment Analysis

Concerning the sentiment analysis requirement, we examined the distribution of scores on a scale from 0 to 5, revealing a mean rating of 3.6 and a notable skew towards higher scores, which aligns with our expectations. Notably, there was one review that featured only a “Like” as a rating.

The average length of the reviews was found to be six sentences and 280 characters, which is adequate for conducting sentiment analysis. By employing the bag of words method on the content of each comment, we identified that out of the 20 most frequent words (see Figure 1), only four were not classified as stop words (good, food, place, service) — a finding we anticipated.

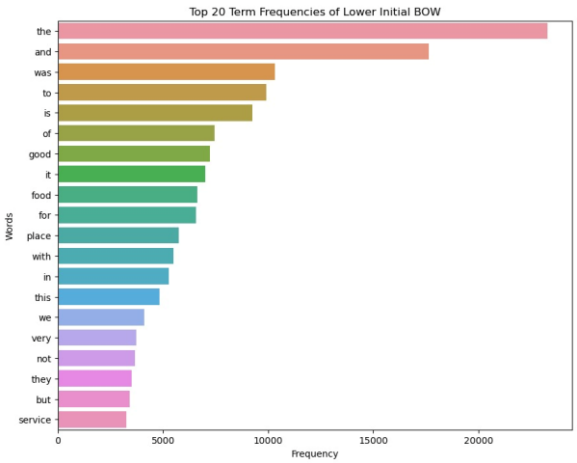

Figure 1: Top 20 Most Frequent Terms in the Bag of Words (BOW) Representation

32. Multilabel Classification

A similar process was undertaken for the second primary requirement: multilabel classification. Initially, it was determined that there were 42 unique types of cuisine present, with Chinese and North Indian foods being the most frequently served (and reviewed) in the analyzed data (see Figure 2). Following this, a co-occurrence analysis was performed to identify which types of cuisine are most commonly served together, revealing that North Indian and Chinese cuisine frequently appear alongside one another. Word clouds were created for each cuisine using Count, Tf-Idf, and Doc2Vec vectorizers, along with TreeMaps for visual analysis.

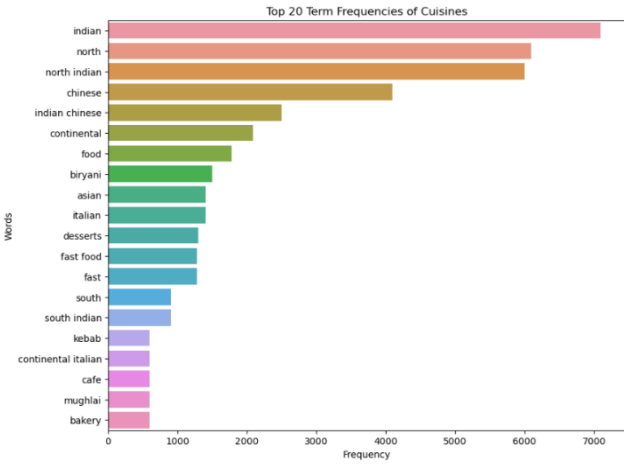

Figure 2: Top 20 Most Frequent Cuisines in the Dataset

The clouds generated with Count and Doc2Vec were notably similar, with only minor differences when compared to those produced with Tf-Idf. However, they exhibited significant alignment with the Chinese, Indian, and Andhra cuisine types. Given this observation, it was anticipated that the model would yield better performance when employing the Tf-Idf vectorizer, as it provided the most diverse output among the three methods evaluated.

33. Clustering

We analyzed the number of reviews that referenced a dish in our dictionary. If a dish was not mentioned, we considered whether it was because the dictionary did not include that dish or because there were no dishes mentioned in the review at all, which was the case with the reviews we examined.

34. Topic Modelling

No specific data understanding was conducted as we only worked with the written reviews that had already been analyzed, both in the general area and other requirements.

4. Data Preparation

41. Sentiment Analysis

To begin the pre-processing phase for sentiment analysis, we removed certain unnecessary features, such as the restaurant name and reviewer profile, and eliminated any missing values from the dataset. Following these steps, we prepared the reviews for the modeling phase through two distinct methods. First, we maintained the original structure of the reviews with emojis encoded. Second, we employed the Punkt Sentence Tokenizer to segment each review into a list of individual phrases. This approach was chosen to compare the effects of analyzing the reviews as a whole against examining them through their separate sentences.

42. Multilabel Classification

To prepare the data for the multi-label classification task, a data frame was created that included restaurant reviews along with a list of associated cuisine types. Following this, a Multi-label Binarizer instance was fitted to the data frame to binarize the various labels found in these lists. Once this process was complete, the data was divided into training and test sets, ensuring it was ready for the models.

43. Co-occurrence Analysis

We began by compiling a list of the most common dishes and ingredients associated with each type of cuisine. Subsequently, we developed a suitable dataset for this task by duplicating the original review dataset and eliminating unnecessary columns. To facilitate the matching process, we created a function called `find_dishes_in_review`. Ultimately, we added a new column to record the dishes mentioned in each review.

Upon reviewing this new column, we discovered that over half of the reviews (5451) referenced at least one dish from our curated list. For the reviews that did not contain any matches, we examined several samples and confirmed that no dishes were mentioned. As a result, we decided not to revise our initial list. Here is one such example:

> *"Great place to chill with friends. Value for money. Love the taste of the food and the service of staff. Especially Praveen was amazing at his job."*

44. Topic Modelling

To meet this requirement, we created a dataset using the same procedures as those used for the clustering task. We then added two columns as a result of applying the preprocessing function from the pipeline. The first column contained the original reviews, converted to lowercase and stripped of stop words, while the second column included a similar version of the reviews but was tokenized.

5. Modelling

51. Sentiment Analysis

This section utilizes two well-known tools: VADER (Valence Aware Dictionary and Sentiment Reasoner) and TextBlob. Both methods are lexicon-based techniques used to assign sentiment polarity to text. Typically, VADER is more effective for informal text, such as social media, while TextBlob offers a broader range of applications. Although VADER was anticipated to yield better results in this specific scenario, both methods were implemented to ensure a comprehensive analysis. Our approach consists of three phases:

511. **Polarity Scoring** – Using both tools, we computed the polarity of the reviews as a whole and separated by sentences (as mentioned previously, we pre-processed the data to allow for this inspection).

512. **Correlation Analysis** – We calculated the correlations between the polarities obtained from the previous step and the ratings associated with the respective reviews using the Pearson coefficient. This analysis was performed independently for both methods.

513. **Model Comparison** – During this phase, we conducted direct comparisons among the four approaches. We determined the RMSE (Root Mean Square Error) and MAPE (Mean Absolute Percentage Error) values for each method. Additionally, we calculated another correlation between the polarities assigned to the reviews using VADER and TextBlob.

Finally, we filtered out some outliers, specifically reviews that had a positive polarity but a poor rating, and vice versa.

52. Multilabel Classification

This stage began with defining a function reused from a class: the Hermetic Classifier. This classifier is designed to streamline text classification tasks by integrating several steps, such as pre-processing, vectorization, training, prediction, and scoring using weighted F1 score. It is a flexible function that adapts to different ways of carrying out the aforementioned steps.

Next, we searched for the best classification models. Given the extensive time it would take to perform a Grid Search — estimated at no less than 15 days — we conducted a total of four Random Searches. These comprised two searches for each of the two adopted classifiers: OneVsRest Classifier and Classifier Chain. We fitted these classifiers using logistic regression and random forest classifier, testing 10 and 5 random combinations of parameters for each, along with five-fold and three-fold cross-validation, respectively.

The vectorization method employed was TF-IDF, as it demonstrated the greatest potential for this task and outperformed other models we tested prior to the Random Searches. All the tested parameters are presented in Tables 1 and 2. Using the best parameters obtained for each classifier, we fitted and evaluated four models on data that had not been seen before.

To complete the modeling stage of this project, a Dummy Classifier was also tested. This was done to provide a clearer interpretation of the results by comparing them with the outcomes from the best classifier.

Table 1 and 2: Optimized Parameters for Multilabel Classification Models

Model Type	Base Estimator	Best Parameters
OneVsRestClassifier	LogisticRegression	`C=37.454`, `penalty='l1'`, `solver='liblinear'`, `class_weight=None`, `d2v_window=10`, `preprocessor__lemmatized=True`, `preprocessor__lowercase=True`, `preprocessor__no_stopwords=True`, `vectorizer=TfidfVectorizer(ngram_range=(1,2), token_pattern='(?u)\b\w+\b')`
OneVsRestClassifier	RandomForestClassifier	`n_estimators=100`, `min_samples_split=2`, `min_samples_leaf=4`, `max_depth=20`, `class_weight='balanced'`, `preprocessor__lemmatized=True`, `preprocessor__lowercase=False`, `preprocessor__no_stopwords=True`, `vectorizer=TfidfVectorizer(ngram_range=(1,2), token_pattern='(?u)\b\w+\b')`

Model Type	Base Estimator	Best Parameters
ClassifierChain	LogisticRegression	`C=43.194`, `penalty='l1'`, `solver='liblinear'`, `class_weight=None`, `d2v_vector_size=200`, `d2v_window=12`, `preprocessor__lemmatized=False`, `preprocessor__lowercase=False`, `preprocessor__no_stopwords=False`, `vectorizer=TfidfVectorizer(token_pattern='(?u)\b\w+\b')`
ClassifierChain	RandomForestClassifier	`n_estimators=200`, `min_samples_split=2`, `min_samples_leaf=4`, `max_depth=20`, `class_weight='balanced'`, `preprocessor__lemmatized=True`, `preprocessor__lowercase=False`, `preprocessor__no_stopwords=True`, `vectorizer=TfidfVectorizer(ngram_range=(1,2), token_pattern='(?u)\b\w+\b')`

53. Co-Occurrence Analysis

The modeling of this task can be explained quite simply. We began by transforming the **None** data points (reviews that did not mention any dishes) into empty lists to facilitate analysis. Next, we identified the maximum number of mentions in a single comment, which was 27, and used this information to create a co-occurrence matrix with a function in our pipeline. It is important to note that before generating this network, we removed some of the more general or common ingredients from the dishes to ensure that our analysis would be more meaningful.

54. Clustering

For the clustering analysis, we started by normalizing our co-occurrence matrix. This step was crucial to ensure that high-frequency items did not overshadow important relationships involving less frequent items. We utilized two specific clustering techniques: OPTICS and HDBSCAN. For each technique, we conducted a grid search to identify optimal hyperparameter values, which were defined in advance. The goal of these grid searches was to achieve the best silhouette score while maximizing the number of clusters. We found that focusing solely on the silhouette score would lead to the creation of only 3 or 4 clusters, which were largely meaningless. Tables 3 and 4 present the values we tested and the best configurations obtained.

Table 3: Hyperparameter Tuning and Results for OPTICS Clustering Algorithm

Hyperparameter	Tested Values	Best Value
`min_samples`	[2, 3, 5, 7, 10]	2
`max_eps`	[0.5, 1, 1.5, 2, 2.5]	0.5
`xi`	[0.001, 0.005, 0.01, 0.05, 0.1]	0.001
`metric`	["euclidean", "cosine", "manhattan"]	cosine
`min_cluster_size`	[2, 5, 10, 20]	2
Number of Clusters	-	22
Silhouette Score	-	-0.0845

Table 4: Hyperparameter Tuning and Results for HDBSCAN Clustering Algorithm

Hyperparameter	Tested Values	Best Value
`min_samples`	[2, 3, 5, 7, 10]	2
`min_cluster_size`	[2, 5, 10, 20]	2
`metric`	["euclidean", "manhattan"]	euclidean
`alpha`	[0.5, 1.0, 1.5]	0.5
`cluster_selection_epsilon`	[0.0, 0.1, 0.2]	0.0
Number of Clusters	-	17
Silhouette Score	-	0.3272

To accomplish this requirement, we used five strategies: Latent Semantic Analysis (LSA) using SKLearn and Gensim (LSI), Latent Dirichlet Analysis (LDA) also using both SKLearn and Gensim, and finally BERTopic.

Starting with SKLearn LSA, we employed TruncatedSVD for dimensionality reduction, which was applied to the BoW (Bag of Words) matrix obtained through CountVectorizer. We tried to optimize the number of components used in this reduction technique by analyzing the variance explained plateau. However, we encountered issues with TruncatedSVD when running it more than two or three times consecutively. As a result, we settled on 100 components, which provided the highest explained variance. Following this, we explored and extracted each topic's composition, determining each word's contribution to specific topics.

We continued with topic modeling by performing LSA using Gensim (LSI). Initially, we created a dictionary that assigned an ID to each token from the tokenized reviews. We then converted each tokenized review into a Bag of Words (BoW) format. To determine the most appropriate number of topics to use, we analyzed the coherence score and concluded that the optimal number would be 50. Finally, we constructed our LSI model, obtaining the composition of each topic and exploring how each document related to the different topics.

The procedure for utilizing LDA in SKLearn was quite straightforward. We conducted a small grid search on an LDA instance to identify the optimal hyperparameter values. Afterward, we fitted the model using the same matrix as in the LSA analysis and conducted a similar exploration. The methodology for gensim mirrored this process, involving a grid search to determine the set of values that produced the best coherence for the final model.

To complete our topic modeling objective, we employed the BERTopic technique. We began by tokenizing each of the preprocessed reviews, creating a new dictionary that assigned an ID to each token, and converting each tokenized review into a Bag of Words (BoW) format, similar to past methods. Following this, we generated a plot demonstrating the variations in coherence scores based on the number of topics explored within a predefined range. From this analysis, we identified the ideal number of topics as 85.

To create the final BERT model, we fitted a BERTopic instance using the preprocessed reviews. To explore the results obtained from this method, we generated various plots that visualized differences between topic groups, the distribution of documents in relation to each topic, possible hierarchies (as certain subjects may encompass others), and finally, the key words that characterize each topic.

6. Evaluation

61. Sentiment Analysis

We built four models: a simple Vader model, a Vader model that splits sentences, a simple TextBlob model, and a TextBlob model that splits sentences. Initially, we compared the models by calculating the correlation between their predictions and the actual ratings. The simple Vader model performed the best, achieving a correlation of 0.7059. Next, we calculated the Root Mean Square Error (RMSE) and Mean Absolute Percentage Error (MAPE) for each model. The model with the lowest RMSE score was the Vader model that splits sentences, which had a score of 0.275. Meanwhile, the simple Vader model had the lowest MAPE score of 0.147. After comparing all the metrics, we concluded that the best model was the simple Vader.

To further validate its reliability, we examined outliers — cases where the model predicted high ratings (rating of 5) but actual ratings were low (rating of 1), and vice versa. The cases where the model predicted good ratings but the actual ratings were poor accounted for 0.90% of the reviews we analyzed. From a sample of five random reviews, three were either confusing for the model due to

mixed or contradictory statements or were presented as bullet points. The remaining two cases were clearer errors made by the model. Conversely, the only instance where the model predicted a bad rating but the actual rating was good represented just 0.01% of the reviews. Upon analyzing this particular review, it became evident that it was not an error of the model; rather, the review was clearly negative. Therefore, it seems the person mistakenly assigned it a good rating.

We also conducted additional analysis of the values predicted by the Vader model:

- We plotted the distribution of the normalized results (see Figure 3).
- We calculated the mean of the normalized ratings, comparing predicted and actual values.
- We created a box plot of the normalized Vader scores against the actual ratings.

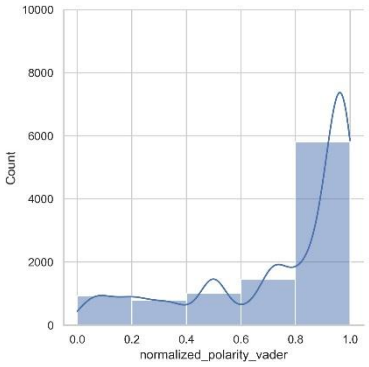

Figure 3: Distribution of Normalized Polarity Scores Predicted by VADER

The discrepancy in normalized scores is about 0.086908, which is equivalent to 0.43454 on a scale of 0 to 5 for the ratings. On average, clients are giving half a star less than what they actually feel according to their reviews. Overall, we were able to accurately predict a restaurant's Zomato score using the polarity of their reviews as input. This suggests that in the future, we can effectively predict expected scores based on new reviews.

62. Multilabel Classification

We decided to build four models, selecting the best from each of the random searches. The results for each model are as follows:

- **OneVsRest Logistic Classifier** - accuracy: 0.202, precision: 0.637, recall: 0.542, f1_weighted: 0.578
- **OneVsRest RandomForest Classifier** - accuracy: 0.055, precision: 0.481, recall: 0.610, f1_weighted: 0.515
- **Logistic ClassifierChain** - accuracy: 0.248, precision: 0.569, recall: 0.517, f1_weighted: 0.535
- **RandomForest ClassifierChain** - accuracy: 0.060, precision: 0.493, recall: 0.620, f1_weighted: 0.522

Based on the weighted F1 score, which is the most important metric given the highly unbalanced data, the OneVsRest Logistic Classifier is the best model. This is further supported by an analysis of the F1 scores per class (table too big to fit in the report) which show that it predicts nearly all classes better than the other models.

To establish a meaningful comparison, we also created a Dummy Classifier as a baseline model. The performance of the baseline model was as follows: accuracy: 0.0301, precision: 0.120, recall: 0.2 and f1_weighted: 0.150.

When focusing on the weighted F1 score, it is evident that our chosen model performs nearly five times better than the baseline (0.578). This means that our model has performed much better than the baseline. However, it is still very far away from being a "good" model. This judgment can be explained by the fact that the reviews are not often very useful for knowing the cuisine type of a restaurant. A big part of them are just comments about their personal experiences and not a description of what they ate.

Looking at the F1 scores per cuisine type, our results appear even more favourable compared to the baseline model, which fails to predict most classes except for North Indian cuisine (with an F1 score of 0.75). Our model, however, can predict multiple classes correctly at least some of the time, which is a

substantial improvement over the baseline. Notably, we are able to predict North Indian cuisine better than the baseline, achieving an F1 score of 0.81.

63. Co-occurrence Analysis and Clustering

In Figure 6, we observe that certain dishes tend to appear together more frequently. These combinations include pizza and pasta, kebab and biryani, biryani and soup, soup and pie, and finally, ice cream and bowl.

When comparing the two models, Optics and HDBSCAN, the latter appears to perform better based on the plots (Figure 4 and 5) and the silhouette score, which is 0.3272 compared to Optics' score of -0.0845. However, after analysing the dishes within each cluster, we found that the clusters produced by Optics yield more meaningful insights. Overall, we conclude that the Optics clusters are preferable, as they are more informative and easier to interpret.

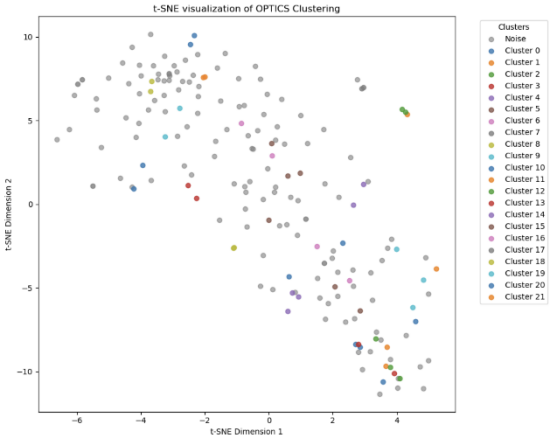

Figure 4: t-SNE Visualization of OPTICS Clustering Results. Silhouette Score for OPTICS: -0.0845

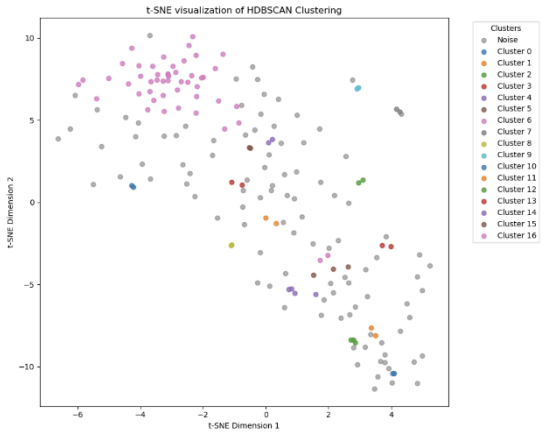

Figure 5: t-SNE Visualization of HDBSCAN Clustering Results. Silhouette Score for HDBSCAN: 0.3272

We successfully formed meaningful clusters using various clustering methods. In both approaches, the clusters created from the co-occurrence matrix were insightful and significant.

We identified several distinct cuisine types, such as Indian, vegetarian, British, and Italian. However, not all clusters clearly represented specific cuisines. Despite this, the number of clusters where a cuisine type was easily recognizable was greater than those that were not.

Table 5: Cluster Topics with Assigned Words and Cuisine Classifications

Topic	Words & Classification	Topic	Words & Classification
0	**Words:** pie, naan, tandoori, pulao **Class:** Indian cuisine with an outlier (pie)	11	**Words:** bake, cheesecake **Class:** Bakery
1	**Words:** curry, grilled **Class:** Indian cuisine	12	**Words:** brownies, cookies **Class:** Bakery
2	**Words:** bowl, crispy, salad **Class:** Vegetarian cuisine	13	**Words:** steak, iced tea **Class:** Steak houses
3	**Words:** soup, gravy **Class:** Multicultural	14	**Words:** chips, fried fish **Class:** British cuisine (fish and chips)
4	**Words:** bbq, shawarma, fried chicken **Class:** KFC-like cuisine	15	**Words:** beef, smoked, shrimp **Class:** Multicultural
5	**Words:** beer, cocktail, pesto **Class:** Bar	16	**Words:** dumplings, loaf **Class:** No common cuisine
6	**Words:** maki, pho **Class:** Asian cuisine	17	**Words:** garlic bread, croissant, bagel, focaccia, brioche **Class:** Café
7	**Words:** appetizer, onion rings **Class:** Entrees	18	**Words:** polenta, pasta salad **Class:** Italian cuisine
8	**Words:** lemonade, barbecue **Class:** American cuisine	19	**Words:** mezze, cheese platter **Class:** European/Mediterranean cuisine
9	**Words:** bread, sandwich, snack **Class:** Café/Snack bar	20	**Words:** shawarma wrap, falafel **Class:** Middle Eastern Street Food
10	**Words:** pasta, crust, spaghetti, risotto **Class:** Italian cuisine	21	**Words:** cucumber salad, couscous **Class:** Vegetarian

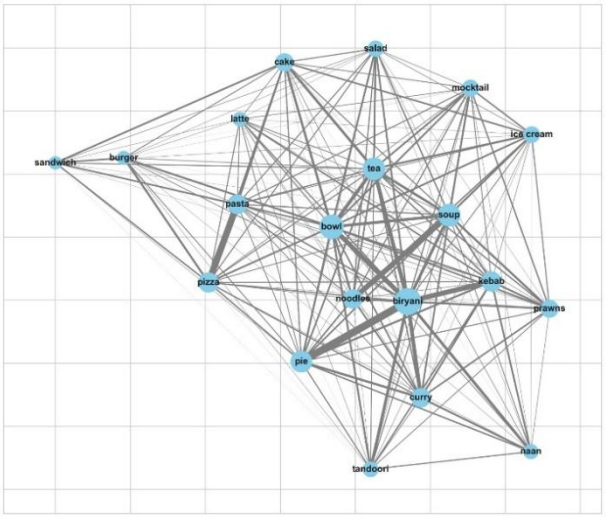

Figure 6: Network Graph of Food Item Co-occurrences

64. Topic Modelling

The models we built were evaluated using different metrics that can not be directly compared:

- LSA with Sklearn – explained variance: 0.457
- LSA with Gensim - coherence score: 0.405
- LDA with Sklearn – perplexity: 2180.84
- LDA with Gensim – log perplexity: -8.54, coherence score: 0.470
- BERTopic - coherence score: 0.438

To further assess the models, we examined how well they classified a random review. The review we analysed (index 2000) was: *"I didn't go and eat at the Dhaba. I had ordered from here. The taste was amazing, and the only issue was the packaging of the order. A must-have is Alu Parantha and lassi."* The model that assigned this review to the most relevant topic was LDA with Gensim, categorizing it with the following keywords and scores: ('place', 0.032), ('food', 0.027), ('good', 0.016), and ('service', 0.0153).

BERTopic is an important model due to its strong results and its ability to generate clearer plots (see Figures 7 and 8), making it easier to interpret the topics compared to the other models:

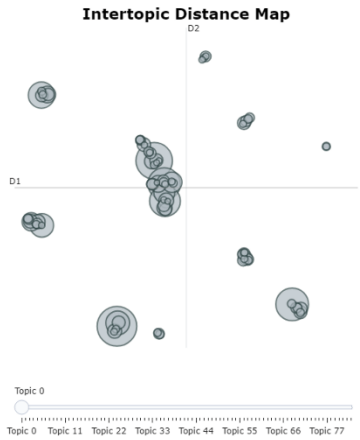

Figure 7: Intertopic Distance Map Generated by BERTopic

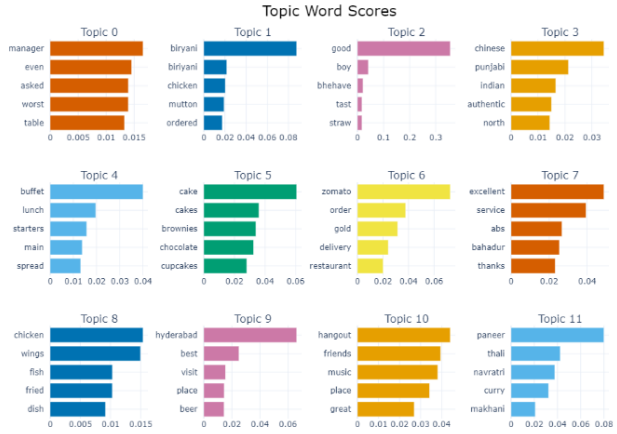

Figure 8: Top Words for Each Topic Identified in Topic Modelling

An important question to consider is: *what do the emergent topics mean*? Emergent topics represent clusters of similar words and documents identified by the model based on their co-occurrence patterns. They reflect underlying themes or concepts within the data that were not pre-defined. In this specific case, we can identify some emergent topics, such as: delivery time (topic 62), ambient quality (topic 65) and delivery quality (topics 19, 33 and 59). These were all topics derived from the Bert model, as it makes it easier to visualize them.

For our analysis of the reviews, we will classify them according to these emergent topics and utilize BERT for clarity. For example, in the review at index 2000, we can see that it encompasses five different topics: 26% of topic 31 (meaningless), 22% of topic 76 (awesome taste), 21% of topic 37 (not good taste), 21% of topic 34 (packaging), and 10% of topic 22 (received delivery). Thus, this review touches on topics related to taste, packaging, and deliveries.

From the topics we have gathered, we can deduce that fast delivery is a common theme across several models. Staff quality also appears frequently, with many reviews highlighting negative experiences with staff members and managers. Delivery quality is also a common topic however reviews under this topic are a bit more contradictory. Another common topic is food quality which has most reviews criticizing the food and saying things like "worst food", "few quantities of food despite tasting good", "too expensive for the quantity and quality". These are just a few of the many topics available; analysing each one individually is impractical, as some models contain up to 100 topics.

7. Conclusions

71. Sentiment Analysis

We efficiently addressed the question by using all four methods to prevent unnecessary limitations caused by the models (we employed TextBlob because it could surprisingly perform better, which was not the case). Furthermore, we tackled potential challenges associated with removing emojis by improving our preprocessor to encode them instead.

72. Multilabel Classification

Our model significantly outperforms a basic Dummy Classifier, indicating that we can classify a restaurant's cuisine type based on its reviews far more effectively than relying on a random model. The results achieved were commendable given the circumstances, as this could be considered a garbage-in-garbage-out model. As previously mentioned, reviews are not the most reliable for predicting cuisine types, and there were inherent limitations due to the quality of the data.

73. Co-occurrence analysis

We successfully addressed the requirements by creating well-defined clusters that represent different types of cuisine. However, there may have been some limitations due to the fact that the list of dishes was compiled manually, which means some dishes may have been overlooked.

74. Topic Modelling

In the Evaluation section, we have drawn several insightful and meaningful conclusions. However, there were some limitations to consider. The BERT model produced a lower coherence score compared to the LDA model using Gensim. This could be attributed to our efforts to identify the optimal number of components, which may not have sufficiently addressed other hyperparameters, such as language settings or n-gram ranges, that could also impact performance.

75. Final Conclusions

This project successfully tackled the essential information requirements by predicting Zomato scores, classifying cuisine types, and identifying significant topics and clusters. While challenges were encountered due to data quality limitations and imbalanced datasets, the chosen methods yielded valuable insights. These findings emphasize the potential of text mining for practical applications in the real world.

8. References

Le Q. V. and T. Mikolov (2014), “Distributed Representations of Sentences and Documents,” vol. 4, May 2014, Available: https://www.researchgate.net/publication/262416109_Distributed_Representations_of_Sentences_and_Documents

Joachims T. (1998), “Text categorization with Support Vector Machines: Learning with many relevant features,” *Machine Learning: ECML-98*, vol. 1398, pp. 137–142, Available: https://doi.org/10.1007/bfb0026683.

Zahoor, N., Z. Bawany, and S. Hamid (2020), “Sentiment Analysis and Classification of Restaurant Reviews using Machine Learning,” *21st International Arab Conference on Information Technology (ACIT)*, Available: https://doi.org/10.1109/acit50332.2020.9300098.