UNIVERSITÀ
DEGLI STUDI
DI MILANO

Algorithms for Massive dataset

Yelp Porjects

Project Supervisor: Prof. Dario Malchiodi

Reza Ghahremani

February 2024

Abstract

Nowadays, with the improvement of online review platforms, consumers can share their opinions and experiences about various products and services. Businesses, through analyzing the reviews, can obtain valuable insights to understand customer preferences and improve their offerings. This project focuses on utilizing text mining techniques to obtain valuable information from user-generated reviews. The Yelp dataset is a rich source of real-world data that helps us explore various text mining tasks such as recommendation systems, review clustering, sentiment analysis, review similarity analysis, information extraction, and artificial neural network modeling. In this project, we are going to examine three important tasks that have been implemented: finding similarities between reviews, creating a content-based recommendation system, and building an artificial neural network model.

Contents

Chapter 1

Introduction to the Yelp dataset

1.1 Yelp dataset structure

Services like TripAdvisor, Amazon, Epinions, and Yelp provide reviews as an integral part of these services, where users can post their experiences about businesses, products, and services through reviews. These reviews consist of free-form text and a numeric star rating, usually out of 5.[1]

In this study, we use the Yelp dataset[1], which is a subset of the businesses, reviews, and user data, for use in connection with academic research. This dataset, which is available in JSON format and contains 6,990,280 reviews, 150,346 businesses, and 1,987,897 users, is prepared in five files: business, user, review, check-in, and tips.

Consequently, within the 'business.json' file, a business contains details such as business ID, name, location, star rating, review count, opening hours, etc. Similarly, a textual review is represented in the 'review.json' file, containing information like the business ID, user ID, star rating (ranging from 1 to 5), review text, date, and more. The business.json and review.json files play a crucial role in our project.

In the Yelp dataset, the businesses described belong to different categories, such as restaurants, shopping, hotels, travel, car rentals, shipping centers, etc. The text reviews for different business categories may be very different. For instance, a typical restaurant review may contain the words or phrases 'food', 'pizza', and 'delivery', but these words would not occur in a restaurant review.

[1] `https://www.yelp.com/dataset`

1.2 Data preparation

To initiate the project, the first step is loading the Yelp dataset files. For this purpose, we used Kaggle's API[2] to upload files to Google Colab. To initiate the project, the first step is loading the Yelp dataset files. For this purpose, we used Kaggle's API to upload files to Google Colab. Then, it's crucial to set up the necessary prerequisites for the project, including libraries. Among these essential prerequisites is PySpark, which plays an important role in this project.

PySpark[3] is the Python API for Apache Spark, which enables us to perform real-time, large-scale data processing in a distributed environment using Python. In addition, it provides a PySpark shell for interactively analyzing your data. Spark Session[4] is the entry point to programming Spark with the Dataset and DataFrame APIs.

Given the substantial volume of data we're dealing with, it's important to align our project goals with the specific components we require. For example, we select columns business_id, name, categories, city, latitude, logitude, and is_open from the business file, review_id, user_id, business_is, star, text from the review file, and user_id and name from the user file. Also, to have integer IDs, we appended a column to each of these files, denoting the numeric ID of each line.

1.3 Data preprocessing

After preparing the data, we will focus on the various projects that we have done with the help of the Yelp dataset. This project includes finding similarities in the reviews that users had, creating a recommender system based on content, clustering based on text, and implementing an artificial neural network model to predict stars based on user reviews. In these tasks, reviews are very important and each of the mentioned tasks will be done based on the reviews. As a result, it is necessary to pre-process the text of the reviews.

Text data originating from natural language tends to be unstructured and noisy. Therefore, text preprocessing plays a crucial role in converting messy, unstructured text data into a format suitable for training machine learning models. This preprocessing step ultimately enhances the quality of results and insights derived from the data.

[2] `https://github.com/Kaggle/kaggle-api`
[3] `https://spark.apache.org/docs/latest/api/python/index.html`
[4] `https://spark.apache.org/docs/latest/api/python/reference/pyspark.sql/spark_session.html`

During this step, we leverage the functions provided by PySpark to preprocess our text data. This involves utilizing various techniques such as RegexTokenizer, StopWordsRemover, CountVectorizer, IDF (Inverse Document Frequency), Word2Vec, VectorAssembler, Normalizer, and Pipeline to prepare the text.

1.3.1 RegexTokenizer

Using RegexTokenizer[5], which is based on regular expressions, tokens are extracted either by using the provided regex pattern to split the text or by repeatedly matching the regex if gaps are disabled. The gaps=False parameter specifies whether to include gaps between tokens, and the pattern='\w+' parameter defines the regular expression pattern for tokenization.

1.3.2 StopWordsRemover

A set of commonly used words in a language are called stop words. Examples of stop words in English are "a," "the," "is," "are," etc. The StopWordsRemover[6] filters out stop words from input.

1.3.3 CountVectorizer

The CountVectorizer [7] extracts vocabulary from document collections and generates a CountVectorizerModel. In other words, it converts the tokenized text into a numerical vector representation by counting the frequency of each word.

1.3.4 IDF (Inverse Document Frequency)

We apply the IDF[8] transformation to the raw feature vectors to reduce the importance of commonly occurring words. The formula for IDF (Inverse Document Frequency) is typically represented as:

$$\mathrm{IDF}(t, D) = \log\left(\frac{N}{\mathrm{df}(t, D)}\right)$$

[5] https://spark.apache.org/docs/3.1.3/api/python/reference/api/pyspark.ml.feature.RegexTokenizer.html

[6] https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.ml.feature.StopWordsRemover.html

[7] https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.ml.feature.CountVectorizer.html

[8] https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.ml.feature.IDF.html

where: N is the total number of documents in the corpus, and df(t,D) is the number of documents containing term t in the corpus D.[2]

1.3.5 Word2Vec

Word2vec[9] generates word embeddings by mapping words to high-dimensional vectors. The vectorSize parameter specifies the dimensionality of the word vectors (i.e., the length of the vectors representing words). vectorSize = 100 means that each word will be represented as a vector of 100 dimensions. The minCount parameter sets the minimum number of times a word must appear in the corpus (collection of documents) to be included in the word vector model. Words that occur fewer times than this threshold will be ignored and not included in the model. minCount = 5 indicates that a word must appear at least 5 times in the corpus to be considered for inclusion in the word vectors model.

1.3.6 Pipeline

A pipeline[10] is defined as a series of stages, where each stage is either a Transformer or an Estimator. These stages are executed sequentially, and the input DataFrame is modified as it progresses through each step. In Transformer stages, the transform() method is applied to the DataFrame. However, in Estimator stages, the fit() method is invoked to generate a Transformer, which becomes part of the PipelineModel or the fitted Pipeline. Subsequently, the transform() method of the generated Transformer is applied to the DataFrame.

[9] `https://spark.apache.org/docs/3.1.2/api/python/reference/api/pyspark.ml.feature.Word2Vec.html`

[10] `https://spark.apache.org/docs/latest/ml-pipeline.html`

Chapter 2

Exploratory Data Analysis

2.1 Business categories

Understanding the types and quantity of available categories can guide us in selecting the categories to focus on for our project. For instance, we might develop a recommender system that aims to introduce new restaurants to users. The top five categories of this dataset are restaurants, food, shopping, home services, and beauty and spas, which are 52264, 27781, 24395, 14356, and 14292, respectively.

2.2 Analyzing rating distribution

The score that each business receives from the user is a numerical value between 1 and 5. It can be seen that the number of "1 stars" is 1069561, "2 stars" are 544240, "3 stars" are 691934, "4 stars" are 1452918, and "5 stars" are 3231627.

2.3 Top 50 Most Reviewed Businesses

The number of feedbacks a business receives can indicate the importance of that business and its serious interaction with users. As you can see in Figure 6.1, the business with ID "_ab50qdWOk0DdB6XOrBitw" called "Acme Oyster House" has received the most number of feedbacks.

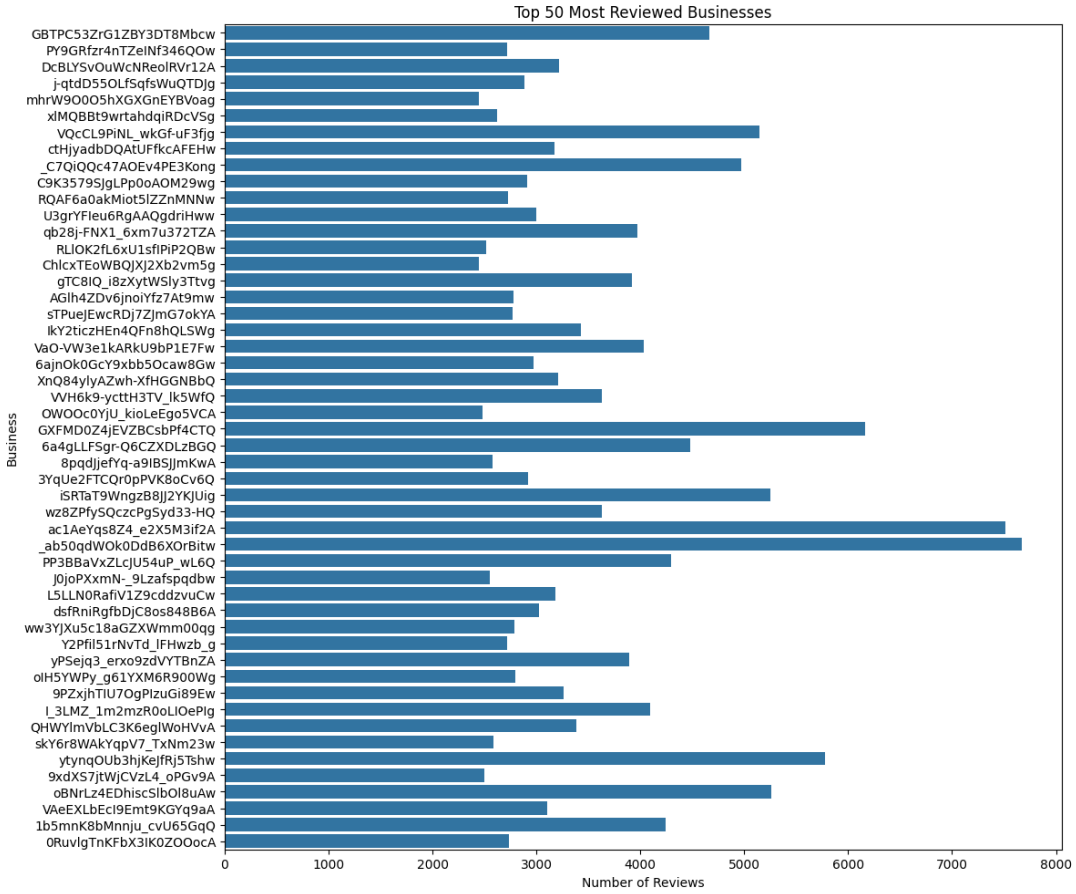

Figure 2.1: Top 50 Most Reviewed Businesses

2.4 Display cities based on the number of businesses

Philadelphia leads with the highest number of businesses, totaling 14,560. Following closely are Tucson, with 9,244 businesses, and Tampa, with 9,042 businesses, securing second and third place, respectively.

Chapter 3

Finding similar items

3.1 Introduction

In the modern world, we often depend on websites like Yelp to help us make decisions, whether it's choosing a restaurant for dinner, deciding on a movie to watch, or buying new clothes. Imagine you love a particular pizza place because of its crispy crust and gooey cheese. Wouldn't it be great to find other restaurants with similar qualities?

To find similarity between two items or multiple items, it is important that the nature of the items align. In this study, for example, we can measure similarity between two users or compute it between multiple businesses. However, our focus will be on finding the similarities between various texts.

In this project, we are diving into the world of finding similar items, focusing on the Yelp dataset and we will explore how we preprocess the Yelp dataset, the methods employed to analyze reviews, and how we can find similar businesses based on reviews.

3.2 Data preprocessing

From the Yelp dataset's five files, we extract the review_id and text columns from the review dataframe. Our project objective is to analyze the text similarity. The text preprocessing procedure is detailed in Chapter 1.

3.3 Methodology

This section outlines the methodology we've adopted and explains the rationale behind each step.

3.3.1 Word Vector Representation (Word2Vec)

Word vectors, generated using Word2Vec, represent words in a continuous vector space where words with similar meanings are mapped closer together. This technique transforms words into high-dimensional vectors, where each dimension captures different aspects of the word's meaning. For example, words like "restaurant" and "food" would have similar vectors because they often appear in similar contexts and share semantic relationships.

In our project, we leverage Word2Vec to generate word vectors for the text data in the Yelp dataset. By doing so, we aim to capture the nuanced semantic relationships between words and phrases, enabling more accurate and meaningful comparisons between texts.

3.3.2 Cosine similarity

Cosine similarity is a metric used to measure the similarity between two vectors in a vector space. It calculates the cosine of the angle between the two vectors, with values closer to 1 indicating higher similarity. In our project, we use cosine similarity to quantify the similarity between word vectors representing different texts.

To implement cosine similarity, we calculate the dot product of the two vectors and normalize it by the product of their magnitudes. This results in a value between -1 and 1, where 1 indicates perfect similarity and -1 indicates complete dissimilarity.

The cosine similarity between two vectors A and B is calculated using the formula:

$$S_c(A, B) := \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

Here, A_i and B_i are the components of vectors A and B respectively, and $\|A\|$ and $\|B\|$ represent their magnitudes or norms.

These vectors encode semantic information about words, such that words with similar meanings are mapped closer together in the vector space. For instance, let's consider the following two

reviews from your dataset:

1. "Great place for a drink, nice friendly staff. Didn't eat there though and would be hesitant to as it doesn't look to be the cleanest place."
2. "A good neighborhood Ritas stand, no indoor seating, more of a grab and go place. Staff is friendly and willing to concoct any combo can you can think up."

Although the reviews use slightly different wording, they both express similar sentiments about the establishments being reviewed. Both mention positive features like friendly staff and a welcoming atmosphere, while also expressing minor concerns about cleanliness or the absence of indoor seating. As a result, these reviews share similar characteristics and sentiments, even though they may use different words or sentence structures.

Cosine similarity is robust to changes in the length of vectors. This means that even if the texts have different lengths or dimensions, cosine similarity can still effectively measure their similarity. In addition, cosine similarity normalizes the vectors before computing the similarity, ensuring that the similarity score is not affected by the magnitude of the vectors. This normalization enables fair comparisons between texts of different lengths.[1]

3.3.3 Locality-Sensitive Hashing (LSH)

LSH is a technique utilized to efficiently approximate nearest neighbors in high-dimensional spaces. It works by hashing input data points into buckets in such a way that similar points are more likely to be hashed into the same or nearby buckets. In our project, LSH plays a crucial role in efficiently finding approximate nearest neighbors based on cosine similarity. By employing LSH, we can significantly reduce the computational complexity associated with searching for similar items in large datasets, making the process more scalable and efficient.

3.3.4 Euclidean Distance

Euclidean distance serves as a measure of the straight-line distance between two points in Euclidean space. In our project, we leverage Euclidean distance as a metric to quantify the dissimilarity between data points, particularly when performing approximate similarity joins. By computing the Euclidean distance between vectors representing different texts, we can determine the extent of dissimilarity between them.

[1] `https://aitechtrend.com/how-cosine-similarity-can-improve-your-machine-learning-models/`

The Euclidean distance between two vectors A and B is defined as:

$$d(A, B) = \sqrt{\sum_{i=1}^{n} (A_i - B_i)^2}$$

After applying LSH (Locality-Sensitive Hashing) to hash the data points into buckets, we compute the Euclidean distance between vectors representing different texts to determine their dissimilarity. By comparing the Euclidean distances between pairs of text vectors, we can identify the extent of dissimilarity between them.

3.4 Future work

In the future, we can explore implementing various methods for calculating text similarity. For instance, we can utilize the Jaccard approach to determine similarities and compare the outcomes with other methods.

Chapter 4

Recommender systems

4.1 Introduction

A recommender system is a software tool or algorithm designed to suggest items or content to users based on their preferences, behavior, or past interactions. The primary goal of a recommender system is to enhance the user experience by providing personalized recommendations that are relevant and likely to be of interest to the user. Recommender systems can be categorized into two types based on the underlying algorithms and techniques used for generating recommendations: Collaborative filtering and Content-based filtering.

Collaborative filtering methods analyze user-item interactions and identify patterns of preferences among users or items. Content-based filtering methods recommend items to users based on the attributes or features of the items and the user's preferences. These systems analyze the content of items (e.g., textual descriptions, metadata) to find similarities and suggest items that match the user's profile. Hybrid recommender systems combine multiple recommendation techniques to leverage the strengths of each approach. For example, a hybrid system may integrate collaborative filtering with content-based filtering to provide more accurate and diverse recommendations.

Suppose we have a user, let's call her "Alice," who enjoys exploring new restaurants in her city. The Yelp dataset contains information about users, businesses, and reviews, which can be leveraged to build a recommender system tailored to Alice's preferences. The system can analyze the attributes and categories of businesses that Alice has previously visited and rated

highly. Using content-based filtering, the system can recommend similar businesses that share similar attributes or categories. For instance, if Alice has visited and rated highly several Italian restaurants in her city, the system can recommend other Italian restaurants in the same area based on the similarity of their cuisine, ambiance, or menu offerings.

4.2 Data preprocessing

The complete details of this step are provided in Chapter 1. Moreover, the data was filtered to focus on businesses located in specific target cities, such as Edmonton. Additionally, only businesses that are marked as open were selected, ensuring that users receive recommendations for establishments that are currently operational and accessible.

4.3 Methodology

As mentioned earlier, text from reviews forms the core of our project. Next, we'll explore how text data assists us in building a recommender system.

4.3.1 Cosine similarity

Finding similarities between texts was discussed in the previous chapter. Cosine similarity is utilized as a measure of similarity between businesses based on their feature vectors derived from textual content. The calculated cosine similarity scores are used to rank the businesses in descending order of similarity. Businesses with higher cosine similarity scores are considered more similar to the input business.

4.3.2 Content-based recommender system

The recommender system implemented in this project can be categorized as a content-based filtering system. This classification is based on the fact that the recommendations are generated by analyzing the content (text) of the items (businesses) and the preferences of the users. It allows for the generation of personalized recommendations based on the specific interests and preferences of individual users. By analyzing the content of the items and matching them with the user's profile, the system can recommend businesses that are likely to be of interest to the user.

Recommendations for users are generated by identifying businesses that are similar to those reviewed by users. Initially, the system calculates the similarity between the input business and other businesses in the dataset. This is done by comparing the feature vectors of businesses, which typically represent their attributes or characteristics. In this case, the feature vectors are derived from textual content. Once the similarities are calculated, the system filters out businesses that the user has already reviewed. This ensures that the recommendations are relevant and novel to the user.

After filtering, the system selects the top recommendations based on their relevance and similarity to the user's preferences. These recommendations are typically presented to the user in descending order of their similarity scores. Finally, the system visualizes the recommended businesses on a map to provide a more intuitive representation of the recommendations.

For example, we have a user with the user_id "-6bkYGD5J7szG0JAblLmIA" who has reviewed "bf34LpGej_MZbOlAKB1beQ", "Esf3-D_44pArPd9GqysoCg", "Ga2GyMUjrN7V0vhJVA-wBA", "kA-yHIy_W_O3n5tMhW1zyA", "KkIHmOm9rl-2sQ63Ixdwhg" businesses. The task is to generate recommendations for this user based on the businesses they have previously reviewed. For each business reviewed by the user, the system calculates the similarity scores with all other businesses in the dataset. The similarity score is calculated using cosine similarity, which measures the cosine of the angle between two vectors representing the textual content of businesses. Higher cosine similarity indicates greater textual similarity between two businesses.

Once similarity scores are calculated for each reviewed business, the system selects the top five most similar businesses for each reviewed business based on their similarity scores. These similar businesses are potential candidates for recommendations because they share similar textual content with the businesses already reviewed by the user. After selecting the top similar businesses, the system filters out any businesses that the user has already reviewed. This step ensures that the system does not recommend businesses that the user has already experienced. Finally, the system presents the top recommendations to the user. Each recommendation includes details such as the name, categories, and location of the recommended business. These details help the user assess the relevance and suitability of the recommended businesses.

In the provided example, the final recommendations for the user "-6bkYGD5J7szG0JAblLmIA" include businesses such as "Lego Store," "Gemini Bridal," and "The Wish List." These recommendations are selected based on the user's previous reviews and the textual similarity between

businesses in the dataset.

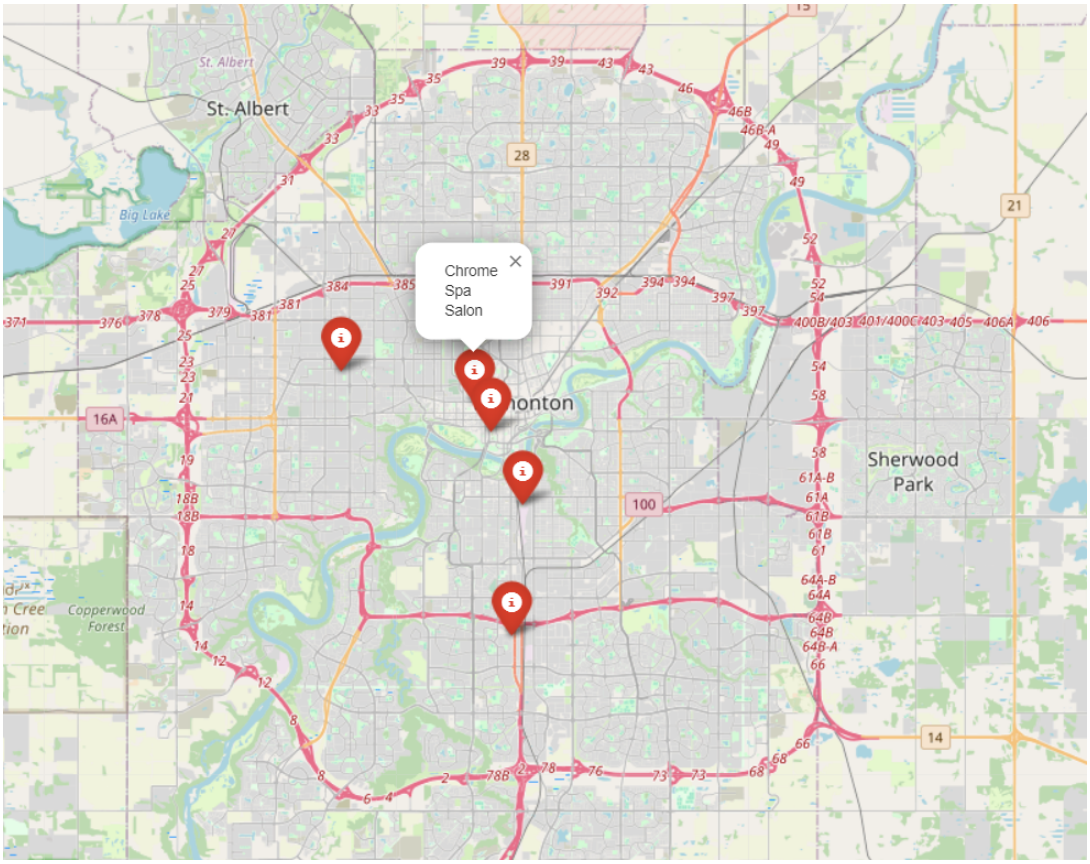

Figure 4.1: Sample visualization of recommendations

4.3.3 Cold start problem

The cold start problem refers to the challenge of providing accurate recommendations for new users or items with limited historical data. To address the cold start problem and ensure that recommendations are still relevant and effective, the Bayesian average rating approach has been implemented.

The Bayesian average rating is a statistical method that combines the average rating of an item with a prior expectation based on the overall distribution of ratings in the dataset. In this approach, each item's average rating is adjusted by considering both the average rating across all items and the number of ratings received by the item. This adjustment helps mitigate the impact of limited data by providing a more stable estimate of the item's quality, even when there are few ratings available.

The Bayesian average rating is calculated for each item in the dataset using the formula:

$$\text{Bayesian Average} = \frac{\text{prior\_weight} \times \text{prior\_average} + \text{data\_weight} \times \text{data\_average}}{\text{prior\_weight} + \text{data\_weight}}$$

where C is the average count of ratings across all items, m is the average mean rating across all items, and "ratings" represents the ratings received by the item.

Using the Bayesian average, we identify the businesses with the highest ratings, which are 43284, 120472, 77980, 542, and 39699. These businesses are likely to be recommended to users due to their high Bayesian average ratings, indicating their popularity and quality in the dataset.

4.3.4 Hybrid recommender system

Given that we have already implemented the content-based recommender system method and discussed the cold start problem approach, we can develop a hybrid recommender system by combining these two approaches. For instance, we can recommend businesses obtained via the cold start problem method for users with the fewest reviews, while for other users, businesses generated by the content-based recommender system method should be recommended.

Chapter 5

Deep Learning

5.1 Introduction

This section presents an analysis of the Yelp dataset using Apache Spark and a Multilayer Perceptron Classifier (MLP). The objective is to build a classification model to predict star ratings for businesses based on reviews. The analysis includes data preprocessing, feature extraction using Word2Vec, and training a neural network classifier.

5.2 Data preprocessing

The Yelp dataset consists of three main tables: businesses, reviews, and users. We selected relevant columns from each table. Data cleaning steps were performed to remove null values and ensure data integrity. Also, We focused on businesses located in the city of Edmonton and filtered out closed businesses.

Word vectors were generated using Word2Vec to represent the textual content of reviews. This process captures the semantic meaning of words and phrases. The word vectors were used as features for training the classification model.

5.3 Methodology

MLP stands for Multilayer Perceptron, which is a type of artificial neural network composed of multiple layers of nodes or neurons. It is a feedforward neural network, meaning that information flows in one direction, from input to output.

In the provided project, an MLP classifier is built using the MultilayerPerceptronClassifier class from the PySpark ML library. The size of the input layer is determined by the number of features in the input data. In this case, the input layer size is equal to the feature vector size obtained from the data. The architecture of the MLP includes one or more hidden layers, each consisting of a specified number of neurons. These hidden layers help the model learn complex patterns in the data. In the project, the hidden_layer_sizes parameter specifies the number of neurons in each hidden layer. For example, hidden_layer_sizes = [64, 32] indicates that there are two hidden layers with 64 and 32 neurons, respectively.

The output layer size is determined by the number of classes or categories in the target variable. In this case, the output layer size is set to 5, corresponding to the five star rating categories. The activation function determines the output of each neuron in the network. In the MLP classifier, the activation function used is the rectified linear unit (ReLU) by default, although other activation functions can be specified.

The data is split into training and test sets to evaluate the performance of the MLP classifier. This splitting process involves randomly dividing the dataset into two subsets: one for training the model and the other for evaluating its performance. In the provided code, the data splitting is performed using the randomSplit function from PySpark ML. The dataset is divided into two portions: 80% for training (train_data) and 20% for testing (test_data).

5.4 Model Evaluation

The performance of the trained Multilayer Perceptron Classifier was evaluated using a test dataset. The accuracy of the classifier on the test dataset was found to be 0.5846. This indicates that the model correctly predicted the star ratings for approximately 58.46% of the reviews. The confusion matrix provides a breakdown of the classifier's predictions versus the actual star ratings. Each row represents the actual class, while each column represents the predicted class.

	Predicted 0	Predicted 1	Predicted 2	Predicted 3	Predicted 4
Actual 0	2107	124	109	73	99
Actual 1	547	237	336	231	78
Actual 2	199	158	680	921	214
Actual 3	104	51	468	2490	1616
Actual 4	140	15	86	1239	4066

The classification report provides a summary of the precision, recall, and F1-score for each class, along with support (the number of instances for each class) and averages. The classification report for our classifier is as follows:

Class	**Precision**	**Recall**	**F1-score**	**Support**
0.0	0.68	0.84	0.75	2512
1.0	0.41	0.17	0.24	1429
2.0	0.41	0.31	0.35	2172
3.0	0.50	0.53	0.51	4729
4.0	0.67	0.73	0.70	5546
Accuracy		0.58		
Macro avg	0.53	0.52	0.51	16388
Weighted avg	0.56	0.58	0.57	16388

Table 5.1: Classification Report

Chapter 6

Clustering

6.1 Introduction

Clustering is a fundamental unsupervised learning technique used to group similar data points together based on their features or characteristics. The primary goal of clustering is to identify natural groupings or patterns within a dataset without prior knowledge of the group labels.

In the Yelp dataset, clustering can be used to group similar reviews based on their content and sentiment. For instance, clustering restaurant reviews based on the words used to describe food quality, service, ambiance, etc., can reveal different categories of dining experiences such as fine dining, casual dining, fast food, etc. This segmentation can help restaurant owners understand customer preferences and tailor their offerings accordingly.

6.2 Date preprocessing

In addition to the preprocessing steps mentioned earlier, we also sampled our data to reduce the computational complexity and processing time. Sampling involves selecting a subset of the data points from the original dataset for analysis. By sampling the data, we can work with a manageable subset while still retaining the essential characteristics of the original data.

6.3 Methodology

In the clustering phase of the project, we employed the K-means algorithm to group similar text data points into clusters. This approach helps us identify patterns and relationships within

the dataset, allowing for better understanding and analysis of the underlying structure.

K-means is an iterative algorithm that partitions data points into K clusters based on their features. It begins by randomly initializing K centroids, which represent the centers of the clusters. Then, in each iteration, the algorithm assigns each data point to the nearest centroid and recalculates the centroids based on the mean of the data points assigned to each cluster. This process continues until the centroids converge, resulting in stable cluster assignments.

6.4 Silhouette Score

To evaluate the quality of the clustering results, we computed the silhouette score. The silhouette score measures how similar an object is to its own cluster compared to other clusters. It ranges from -1 to 1, where a higher score indicates that the object is well-matched to its own cluster and poorly matched to neighboring clusters.

6.5 Visualization

We visualized the clustering results using PCA (Principal Component Analysis) for dimensionality reduction. PCA helps visualize high-dimensional data in a lower-dimensional space, making it easier to observe clusters and their relationships. The scatter plot shows the data points colored by their assigned clusters, with centroids marked in red.

6.6 Conclusion

In this section, we explore the impact of varying the number of clusters (k) on the K-means clustering results. We assess the quality of clustering using silhouette scores and examine the distribution of tweets across different clusters for each value of k.

Based on the experimentation results, we observe that the silhouette scores decrease as the number of clusters increases, indicating diminishing clustering quality. However, the number of tweets in each cluster decreases as well, suggesting more fine-grained clustering with higher k values. Considering both the silhouette scores and the distribution of tweets, we find that k = 3 yields a relatively high silhouette score and a balanced distribution of tweets across clusters, making it a reasonable choice for clustering our dataset. However, further analysis and domain expertise may be necessary to determine the optimal number of clusters for specific applications.

K Value	Number of Tweets in Each Cluster	Silhouette Score
3	0: 3598 1: 2851 2: 2303	0.2327
4	2: 2763 0: 2689 3: 1815 1: 1485	0.1585
5	0: 2325 4: 2283 2: 1581 3: 1299 1: 1264	0.1399
10	0: 1542 3: 1423 2: 1314 9: 808 7: 739 1: 735 4: 658 5: 628 8: 571 6: 334	0.1163

Table 6.1: Cluster Information and Silhouette Scores for Different K Values

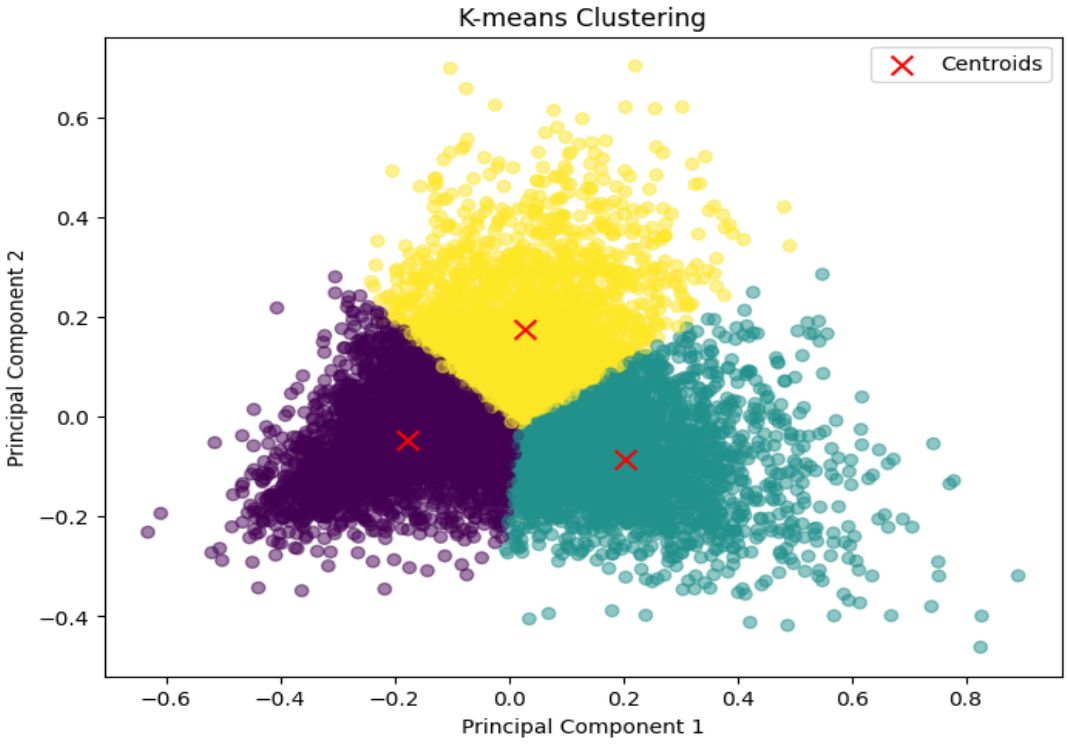

Figure 6.1: K = 3 clustering

1

[1]I declare that this material, which I now submit for assessment, is entirely my own work and has not been taken from the work of others, save and to the extent that such work has been cited and acknowledged within the text of my work. I understand that plagiarism, collusion, and copying are grave and serious offences in the university and accept the penalties that would be imposed should I engage in plagiarism, collusion or copying. This assignment, or any part of it, has not been previously submitted by me or any other person for assessment on this or any other course of study.

Bibliography

[1] Nabiha Asghar. *Yelp Dataset Challenge: Review Rating Prediction*. 2016. arXiv: 1605.05362 [cs.CL].

[2] Jure Leskovec, Anand Rajaraman, and Jeffrey D. Ullman. *Mining of Massive Datasets*. Second. Cambridge University Press, 2014. URL: http://mmds.org.