

A Work Project, presented as part of the requirements for the Award of a Master's degree in
Business Analytics from the Nova School of Business and Economics.

IMPROVING MERCHANT IDENTIFICATION IN TRANSACTION DATA USING
WORD2VEC AND GLOVE EMBEDDINGS

GILIAN DUSTIN WAGNER

Work project carried out under the supervision of:

Prof. Dr. Susana Lavado

Prof. Dr. Patricia Xufre

16/12/2024

Abstract

Merchant identification and aggregation in financial transaction data is crucial for ensuring fraud detection and seamless payment processing. Inconsistent merchant data often results in legitimate transactions being flagged as fraudulent or fraudulent activities going undetected. This thesis explores the use of NLP techniques, including Word2Vec and GloVe, to enhance merchant identification by clustering similar entities. A filtering process for dominant merchants was implemented, addressing clustering limitations. While Word2Vec showed potential for capturing contextual nuances, neither method outperformed an existing string-matching model. Future research should explore advanced NLP models, larger datasets, and address the scarcity of public data for merchant aggregation.

Keywords: NLP, Word Embeddings, Transaction Data, Merchant Identification, Word2Vec, GloVe

This work used infrastructure and resources funded by Fundação para a Ciência e a Tecnologia (UID/ECO/00124/2013, UID/ECO/00124/2019 and Social Sciences DataLab, Project 22209), POR Lisboa (LISBOA-01-0145-FEDER-007722 and Social Sciences DataLab, Project 22209) and POR Norte (Social Sciences DataLab, Project 22209)

1 Introduction

1.1 Problem Statement

The digitization of processes across all areas of life has driven an exponential increase in data generation, with global data volumes reaching 123 zettabytes in 2023 and projected to grow to 394 zettabytes by 2028 (Statista 2024). This digital transformation extends to shopping behavior, where e-commerce has become increasingly prevalent. In 2023, 70% of all EU citizens between 16 and 74 years old ordered goods or services online (Eurostat 2024). As a result, payment processing providers face growing challenges in managing complex transactions and improving fraud detection mechanisms.

This thesis is focused on one such provider, who plays a pivotal role in ensuring secure transactions. This company is a leading provider of payment processing and digital services, with operations spanning over 25 countries. The company supports more than 300 million users globally and handles over 4 billion transactions annually.

This company faces a critical business challenge related to the identification of merchants in card-based transactions. For merchants who do not process payments directly through them, each merchant defines their own name. If this name is not consistently the same, their transactions cannot be aggregated, which leads to the proliferation of distinct Merchant IDs within the system and to the detriment of effective fraud and anomaly detection. As of November 2023, there were around 400,000 different Merchant IDs in their database.

The inability to accurately identify merchants results in legitimate transactions being mistakenly flagged as fraudulent (“False Positives”) and actual fraud going undetected (“False Negatives”), affecting operational efficiency and customer trust.

1.2 Aim of the Work & Course of Investigation

Within the course of the Project-Based Learning (PBL) program at the Nova School of Business and Economics, a model for merchant identification and aggregation was developed using a string-matching approach. While effective, this model had limitations in handling the variety and complexity of merchant names in large transaction datasets. The goal of this study is to improve upon this model by leveraging state-of-the-art NLP techniques like Word2Vec and GloVe to better identify and aggregate merchant names.

This work begins with a Literature Review, examining the evolution of NLP techniques, including Word2Vec as well as GloVe and their applications in financial transaction processing. It also poses the research question as well as the hypotheses that should be examined in this study. Chapter 3 describes key characteristics of the dataset used and conducts an Exploratory Data Analysis (EDA), highlighting insights and patterns that informed the development of the proposed models. Finally, it outlines the modelling approach of the PBL project, which serves as a baseline model for comparison. Following this, chapter 4 details the steps taken to develop and evaluate different approaches for merchant identification and aggregation, along with their corresponding results and insights. Finally, chapter 5 provides a discussion of the results, evaluates the proposed methods against the original model and examines whether the posed hypotheses are supported. It also outlines key limitations and future research opportunities in this field.

2 Literature Review

In recent decades, artificial intelligence (AI) has undergone a rapid evolution, driving transformative advancements across numerous industries. Originally grounded in rule-based systems and statistical models, AI has since evolved with significant breakthroughs in machine learning (ML) and, more recently, deep learning (Ertel 2024). These advancements have

enabled AI systems to process vast amounts of data with increasing accuracy and speed, making them invaluable in fields such as image and speech recognition and natural language understanding, among others (Lecun, Bengio and Hinton 2015). Within the financial services industry, AI applications have been pivotal in enhancing fraud detection, risk management, and customer service by analyzing large-scale transaction data in real-time (Heaton, Polson and Witte 2016).

One of the critical areas in AI's evolution has been Natural Language Processing (NLP), which focuses on enabling machines to understand, interpret, and generate human language to retrieve insights. Early NLP methods relied heavily on statistical techniques, which enabled foundational tasks such as part-of-speech tagging and parsing by leveraging probabilistic models and statistical patterns in language (Manning and Schütze 1999). These methods, however, failed to capture semantic relationships between words, limiting their effectiveness in capturing contextual meaning and therefore hindering their ability to perform more advanced tasks such as meaningful text generation and entity recognition.”

This changed with the introduction of Artificial Neural Networks (ANNs), where deep learning enabled models to outperform former approaches, as first demonstrated by Yoshua Bengio and colleagues in 2003 (Bengio et al. 2003) and later by Tomas Mikolov and others in 2010 with the use of Recurrent Neural Networks (Mikolov et al. 2010).

The development of “Word2Vec” marked a significant milestone in NLP, introducing a method to embed words in a continuous vector space based on their contextual relationships. In their pioneering work, Mikolov and colleagues (2013a) proposed two neural network architectures, Skip-Gram and Continuous Bag of Words (CBOW), which predict word context or target words based on surrounding words in a corpus. This approach enabled the model to capture both semantic and syntactic relationships between words efficiently. Shortly after, Mikolov

expanded on this work by demonstrating the model's ability to learn compositional relationships between words and phrases. They showed that Word2Vec could perform linear algebra operations on word vectors, revealing latent relationships (e.g., $\text{vec}(\text{"Berlin"}) - \text{vec}(\text{"Germany"}) + \text{vec}(\text{"France"}) \approx \text{vec}(\text{"Paris"})$) (Mikolov et al. 2013b). This capability revolutionized NLP applications such as sentiment analysis, machine translation, and named entity recognition, where contextual word representations greatly improved model performance.

Following Word2Vec, the introduction of “Global Vectors for Word Representation” (GloVe) by Pennington advanced the field further by incorporating both local context and global statistical information from the corpus. GloVe’s matrix factorization approach allowed it to capture co-occurrence probabilities, effectively representing the relationships between words based on their shared contexts. This enables the model to create embeddings that reflect word associations across the dataset and makes it more adept at capturing global patterns, such as word analogies and relationships. It was designed as a competitor to Word2Vec, and the paper noted several improvements of GloVe over Word2Vec (Pennington, Socher and Manning 2014).

The next significant milestone was the introduction of “Bidirectional Encoder Representations from Transformers” (BERT). Unlike Word2Vec and GloVe, which are unidirectional, BERT uses a bidirectional approach, enabling it to capture context from both preceding and following words (Devlin et al. 2019). It is built on the transformer architecture, which employs a self-attention mechanism. This allows the model to weigh the importance of each word in a sentence relative to every other word (Vaswani et al. 2017). This mechanism enables BERT to understand the full context of a word within a sentence, capturing complex dependencies across the text (Devlin et al. 2019).

Building on the success of transformers, the Generative Pre-trained Transformer (GPT) series has emerged as the state-of-the-art technique in NLP. GPT utilizes a transformer-based

architecture similar to BERT, but focuses on unidirectional autoregressive training, where the model predicts the next word in a sequence based only on the preceding context, making it highly effective for generative tasks like text completion, summarization, and conversation (Radford et al. 2018). GPT models now set the benchmark for NLP performance, with applications spanning search engines, virtual assistants, and more (Bommasani et al. 2022).

Despite the advancements in NLP, the application of these techniques in the financial transaction industry faces significant challenges, primarily due to the limited availability of public datasets. Most financial transaction data reside within private companies, restricting the development and evaluation of NLP models in this domain (Hilal, Gadsden and Yawney 2022; Boulieris et al. 2023). Nevertheless, NLP has still been successfully used in financial transaction processing, with models like Word2Vec and BERT demonstrating their potential in fraud detection and classification tasks. For instance, Chang et al. proposed an NLP-powered anti-fraud chatbot that utilizes Word2Vec and BERT for fraud detection, illustrating how these models can be applied in financial systems. Another notable example is FinChain-BERT, a model designed for fraud detection by handling complex financial text information, which further underscores the versatility of NLP in this field.

Efforts to create public datasets for NLP-driven fraud detection also highlight the significance of this area. Recent studies have introduced datasets designed for Natural Language Processing in fraud detection, providing benchmarks for evaluating NLP techniques (Boulieris et al. 2023).

However, advanced models like BERT often require substantial computational resources, which can be a limiting factor when working with large datasets. Fine-tuning and training BERT, for example, is computationally intensive, as noted by Chang, Yen and Hung (2022) as well as Vucetic et al. (2022), who identified resource constraints as a key hurdle for applying BERT models to extensive datasets. Given the limited computing resources available for this

research project - where GPU access is not available, and resources must be shared across teams at the company - less resource-intensive methods are more suitable for addressing the merchant identification problem. Therefore, this work project explores whether merchant identification and aggregation can be improved using the NLP techniques Word2Vec and GloVe. Hence, the research question for this thesis is: "Do Word2Vec and GloVe word embeddings enhance the accuracy of merchant identification and aggregation in transaction data, compared to traditional string-matching techniques?".

This work aims to examine the following hypotheses:

1. Utilizing word embeddings to represent merchant names will improve the accuracy of merchant identification and aggregation compared to traditional string-matching techniques.
2. The Word2Vec model will demonstrate superior accuracy compared to GloVe for merchant name aggregation within the Spark infrastructure.
3. Incorporating additional contextual features - such as transaction location, the Merchant Category Code and the Acquirer Code - with name-based embeddings will improve the accuracy of merchant identification over using name-based embeddings alone.

3 Data & Context

3.1 Data Description

The dataset provided included transaction data from a period of 93 days and contains 28,603,301 transactions, which are stored in an Apache Hive database, while the underlying files reside in a Hadoop Distributed File System (HDFS). The dataset encompasses a wide range of transactions, featuring over 2.5 million distinct merchant names and transaction amounts varying from 0€ to a maximum of 101,582.44€.

It includes variables such as a transaction identifier, which uniquely identifies each transaction, the name of the merchant, the transaction country as well as the operation amount. It also entails the *Merchant Category Code (MCC)* of each transaction, which describes the merchant’s business category, such as *Fast-Food Restaurants* or *Digital Goods*. This variable may assist in classifying merchants according to their industry, helping to aggregate them correctly. Another feature is the *Payment System Code*, which differentiates transactions that are processed through either Visa or Mastercard.

Additionally, the dataset includes the variable *Acquirer Code*, which refers to the code of the financial institution that processes the transaction, as well as the *Merchant External Code*, which uniquely identifies a merchant across different acquirers and systems. Both features are useful to determine if transactions belong to the same company. All variables in the dataset are listed in Appendix 1.

The data was anonymized to ensure compliance with the General Data Protection Regulation (GDPR) in the European Union (EU). For that, the original transaction IDs, which are unique identifiers for each transaction, have been processed through a one-way hash function. In addition, the transaction dates were encrypted by not stating the exact date but only the relative time difference in seconds between the first timestamp in the dataset and each transaction timestamp. Both measures comply with the “Data minimization” principle of the GDPR, which states that personal data should be “adequate, relevant and limited to what is necessary in relation to the purposes for which they are processed” (Article 5(1)(c)) (GDPR 2016a). It also applies an appropriate measure to ensure security of processing according to Article 32(1)(a) (GDPR 2016b).

3.2 Exploratory Data Analysis

Before conducting the Exploratory Data Analysis (EDA), it is important to note that the dataset was already pre-processed by the company itself. As a result, no additional data cleaning was required. Instead, the focus of the EDA was on uncovering trends and patterns within the data that could be helpful for building the final model.

The *Merchant Name* variable is the most critical variable to analyze, as it contains numerous inconsistencies and variations that need to be standardized. Upon examining the most frequently occurring merchant names (Appendix 2), it is evident that transactions from a few large, globally recognized companies dominate the dataset. Well-known corporations such as Apple, Uber, and Netflix account for a significant portion of the transactions, highlighting the importance of correctly identifying and aggregating these merchants.

To illustrate this, the merchant *Netflix* is considered as an example. In total, there are 850,875 transactions associated with 67 different merchant names containing the word "Netflix". However, the 5 most prevalent names make up 99.23% of all transactions containing "Netflix", with "NETFLIX.COM", "NETFLIX INTERNATIONAL" and "PAYPAL *NETFLIX COM" being the most prevalent (Appendix 3).

As described in chapter 3.1, the feature Merchant Category Code (MCC) is important for understanding the distribution of transactions across different industry sectors and identifying patterns that may aid in merchant classification and fraud detection. A small number of the 558 unique MCC's account for most of the merchant names and transactions. For instance, the MCC's "Fast Food Restaurants", "Limousines and Taxicabs" and "Digital Goods and Computer Programming" collectively represent 66% of all merchant names (Appendix 4). When analyzing transaction volume, a slightly different set of MCCs dominates. The MCCs "Limousines and Taxicabs", "Eating Places, Restaurants" and "Digital Goods" account for approximately 50% of the total transactions processed (Appendix 5).

Netflix provides a clear example of this concentration. While its transactions are spread across 12 different MCCs, 99.96% of all Netflix-related transactions fall under just two categories: “Cable, Satellite, and Other Pay Television and Radio Service” as well as “Digital Goods”. This highlights the importance of accurately grouping merchants by their MCCs, as for some companies, the majority of transactions may be concentrated within only a few categories.

When analyzing the total transaction amount per MCC, we observe that most of the transaction value comes from “Money Transfers”, which dominates the dataset by a significant margin (Appendix 6). In contrast, examining the average transaction amount per MCC reveals that many individual hotels and airlines are categorized as separate MCCs, which represent specific companies rather than broader industry categories, such as “Soho Grand Hotel” or “Japan Air Lines” (Appendix 7). This indicates that, while this variable can help in aggregating merchant names, it must be used with caution.

When examining the most frequently occurring merchant names within each MCC, we again observe the dominance of well-known corporations. For instance, in the “Limousines and Taxicabs” category, companies like Uber, Bolt, and FreeNow represent the majority of transactions, while the “Digital Goods” segment is dominated by Netflix, Apple, and HBO. This confirms the previous finding about big companies being the most prevalent merchants in the dataset.

The dataset contains merchants from a total of 191 countries, though most transactions are concentrated in just a few (Appendix 8). The Netherlands, Estonia, Ireland, and Portugal account for a significant portion of the transactions, with the Netherlands leading at 20.5%. In contrast, 95.29% of the countries have an insignificant number of transactions, which reflects the influence of e-commerce platforms that operate across a limited number of key regions.

When examining the transaction status, most transactions were accepted (79.4%), while 20.2% were rejected (Appendix 9), showing a significant number of transactions being rejected. This

highlights the need for better fraud detection, which may be achieved through better merchant aggregation.

Lastly, an example of a falsely rejected transaction due to inconsistent merchant names is presented (see Appendix 10). All five transactions in the table share the same Acquirer Code and Merchant External Code, indicating that they originate from the same merchant. The first two entries represent account verifications. Transactions 3, 4, and 5 are from the same merchant but have different variations of the Merchant Name. Transactions 3 and 4 were successfully aggregated and processed, despite the slight differences in their merchant names. However, transaction 5 was incorrectly rejected because the merchant name “BOLT.EU” was not recognized as the same entity. This demonstrates how inconsistent merchant naming can lead to false rejections, highlighting the importance of correctly aggregating merchant names to improve customer experience.

3.3 Preceding Work

To address the challenges posed by inconsistent merchant names, the respective financial service provider has implemented several approaches aimed at improving the accuracy of transaction aggregation.

One strategy that the company currently employs for merchant identification is “Hashing”. Here, unique hash values are generated from merchant data elements like names or IDs. This allows transactions from the same merchant to be grouped and analyzed efficiently. However, this strategy proves ineffective when merchants provide inconsistent information as disparate hashes are generated for the same merchant.

Another existing strategy is the use of Regular Expressions (RegEx) which consists of identifying and aggregating transactions for the 20 most common merchants, even with variations in

merchant names. Nevertheless, this manual approach is resource-intensive and updated only when major changes occur in merchant names. Attempts to achieve near-complete coverage (95-100% of transactions) lead to costly resource consumption and extended processing times, making the approach unsuitable for real-time decision-making.

Within the course of the Project-Based Learning (PBL) program, the goal was to build a model that achieves better results than the current approaches in terms of accuracy and efficiency. Given that the dataset was heavily dominated by a few large merchants and knowing that efficiency is crucial for real-time decision-making, the initial focus was on correctly aggregating the transactions of these major merchants.

Initially, a pre-defined list of major merchants was created, and the model was designed to determine whether each transaction corresponded to one of these companies. By focusing on these corporations, the model was able to cover the majority of transactions in the dataset. Merchants were chosen based on how often they appear in the dataset and on the ease of cleaning their names (see Appendix 11).

Then, as illustrated in Figure 1, the similarity scores were calculated between the merchant name value and each name of the pre-defined list. Here, a character-level bi-gram similarity technique is used to compare the similarity between two strings. N-gram similarity is a method that generalizes the concept of the longest common subsequence by comparing sequences of characters or tokens, known as n-grams (Kondrak 2005). This means that, the more similar the sequences of consecutive characters, the higher the similarity score. This approach is particularly effective for detecting subtle variations between strings, such as typos or abbreviations, making it useful for identifying matching merchants across different transaction datasets.

Additionally, the Levenshtein Distance between the merchants and each element of the merchant list was calculated in case there are several merchants with the same similarity score for

a given transaction. This metric indicates the number of edits required to change one string to another (Po 2020). In the end, the results were ordered by the similarity score as the primary and the Levenshtein distance as the secondary ranking criteria.

Before calculating similarity scores, a filtering process was applied using MCC, Country, and the Acquirer Code as key features to help identify merchants. For each merchant in the predefined list, the most common values from these variables were assigned based on the merchant's historical data. If a transaction's values for MCC, Country, and Acquirer Code matched those of a merchant in the predefined list, that merchant was considered a potential match, and similarity scores were then calculated for all potential matches. If no match was found, the transaction was categorized under "small merchant" and no further categorization was performed.

Prior to the steps above, the merchant name column was cleaned by removing special characters such as commas, asterisks, and underscores, as well as common words like "paypal," "crv," and ".com." This pre-processing step was included because it led to a slight improvement in the model's performance.

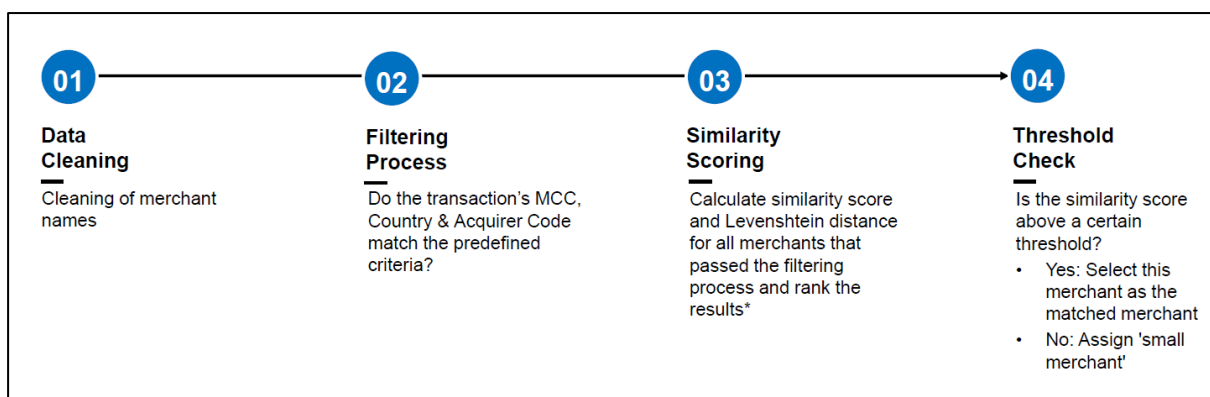


Figure 1 – PBL Modelling Approach

To create a labeled test dataset for evaluating the model's performance, regex expressions were applied to extract and standardize merchant names from the dataset. These extracted names were further cleaned to remove noise, such as "paypal," "crv," and numerical characters.

Additionally, high-frequency merchants like "Bolt," which account for a significant proportion of transactions, were explicitly added to the dataset to ensure proper representation. This resulted in a labeled test dataset containing 2.8 million rows of transaction data from major merchants, providing a reliable benchmark to measure the model's ability to aggregate and identify merchant names.

The final model achieved an accuracy of 99.81% in the test dataset, while 3.66% of transactions were classified as "small merchant" as they could not be attributed to any of the names in the pre-defined list. These metrics will serve as benchmarks for comparing the model developed in this work project with the existing model described.

4 Methods & Results

4.1 Word2Vec

The first approach explored in this work combines Word2Vec embeddings with clustering techniques. Clustering was chosen as a complementary method to Word2Vec for its ability to group similar merchant names without the need for labeled data. The property of Word2Vec embeddings to position semantically similar words close to each other in a high-dimensional vector space can be leveraged to form meaningful groups of merchant names.

The first clustering algorithm explored is k-means, chosen for its straightforward and computationally efficient nature, with a time complexity of $O(knt)$ (as detailed in Appendix 12) (Yu and Tian 2015). This makes it a practical starting point for clustering tasks on large datasets.

To enhance the clustering process, the features Merchant Name, MCC, Country, and Acquirer Code were selected, as these had proven valuable in separating merchant names during the exploratory data analysis (EDA) and in the model developed during the PBL project.

Preprocessing was conducted to clean, standardize, and prepare these features for efficient processing. For the Merchant Name feature, noise such as "www," "crv," and numerical characters was removed, as these expressions were identified as irrelevant for distinguishing merchant names (see Appendix 13). For the MCC feature, textual descriptions from financial service provider websites were used as input, allowing the model to group merchants by category and leverage semantic information. For instance, MCC codes 1–1500 correspond to "Agricultural Services," while 4000–4800 represent "Transportation Services" (Citibank, 2015). Additionally, the Country feature was converted to lowercase and the Acquirer Code was transformed into a string format to ensure compatibility with the clustering model.

After preprocessing, these features were combined into a single vector format for clustering. First, cleaned merchant names were tokenized, and Word2Vec embeddings were generated using a vector size of 1000 to capture the semantic complexity of merchant names. A minimum sample threshold of 1 was applied to include even rare tokens, ensuring that less frequent but valid merchant names were represented. The remaining features - MCC, Country, and Acquirer Code - were indexed and one-hot encoded before being integrated into the final vector. This unified format allowed clustering algorithms to efficiently process relationships between merchants based on these combined attributes.

The dataset was downsized to 286,033 rows (1% of the original dataset) due to the high dimensionality of the resulting vectors (1600 dimensions) and system resource limitations, which frequently led to kernel failures. The downsizing was performed using a random sampling technique, which preserved the statistical distributions of features.

The elbow graph, which was created to determine the optimal number of clusters (Appendix 14), suggested that the model becomes more accurate the higher the number of clusters.

However, this also bears the risk of aggregating the same merchants into different clusters. Therefore, several numbers of k were tried.

When $k=50$, some clusters accurately grouped merchants together, such as *Uber* and *Glovo* (Appendix 15). However, merchants like *Google* appeared in several clusters, suggesting that this number of clusters was insufficient for separating all unique merchants effectively. Setting the number of clusters to 200 resulted in merchants from the same industry being grouped together, likely because they share the same MCC, causing their embeddings to be positioned closer in the vector space. For instance, *Spotify* and *HBO Max* were placed in the same cluster, likely due to their shared MCC as streaming services. The same was true for the airlines *Ryanair* and *TAP* (Appendix 16). This pattern persisted with the larger k values 500 and 1000.

To address this observed limitation, k -means was retried using Word2Vec embeddings derived solely from merchant names to reduce the influence of non-distinctive features like MCC or country. Focusing solely on merchant name embeddings seemed to improve the ability to separate distinct merchants into different clusters for all k 's. For example, one cluster contained only "wise", while another grouped solely "applecombill" (see Appendix 19).

However, this approach also introduced new challenges. Merchants with slight variations in their names were often assigned to different clusters, even when they referred to the same company. For instance, "uberpending" and "ubereatspending" appeared in separate clusters, as did "bolteuo" and "bolteuovan," regardless of the number of clusters. This sensitivity to slight changes in merchant names likely arises from the model placing greater emphasis on the name embeddings alone, which can exaggerate minor differences.

When visualizing the name-based embeddings (Appendix 22), it can be observed that most points are concentrated within a very small region close to the origin, suggesting minimal differentiation in the embeddings across merchant names as well as insufficient contextual

diversity for Word2Vec to generate meaningful differences. On the other hand, the embeddings of the combined features (Appendix 23) show a more noticeable spread of data points, indicating that combining merchant names with other features captures greater variability. However, the example of "ubereatspending" shows that the same merchants can also be located relatively far away from each other. This is likely because the other features MCC, Country or Acquirer Code can be different throughout several transactions.

A limitation of k-means is the requirement to pre-define the number of clusters, which is challenging to optimize, considering that the exact number of distinct merchants in the dataset is not known. To address these issues, the next step will involve testing the Density-Based Spatial Clustering of Applications with Noise (DBSCAN) technique, which does not require the number of clusters to be specified beforehand and can detect outliers as "noise".

The application of DBSCAN presented a significant challenge as it is not natively supported in Spark and had to be implemented in Pandas, a widely used Python library for manipulating data, where the dataset with 1600-dimensional Word2Vec embeddings was too large to handle efficiently due to Pandas' in-memory processing limitations. To address this issue, a Principal Component Analysis (PCA) was chosen as a straightforward and efficient method (Ayesha, Hanif and Talib, 2020) to reduce the dimensionality of the dataset from 1600 to 10 dimensions, as this number was identified to retain enough distinctive characteristics of the embeddings while ensuring kernel stability. In addition, Dask ML, a Python library designed for scalable Machine Learning, was used as it enables distributed processing by splitting datasets into smaller chunks and processing them across multiple cores or nodes (Daniel 2019).

DBSCAN was tested with both combined features and Word2Vec name embeddings. Initial tests with "epsilon" values (maximum distance between points in a cluster) of 0.5, 0.1, and 0.01 and "minimum samples" (minimum number of points required to form a cluster) set to 20

(Appendix 24– 26) grouped unrelated merchants like "Comboios Portugal," "Spotify," and "Playstation" into the same cluster, indicating poor accuracy. The model created only 70–156 clusters, which is significantly lower than the expected number given the high variety of distinct merchants in the dataset. Reducing the minimum samples parameter to 5 increased the clusters to 344, but the aggregations still lacked meaningful structure (Appendix 27).

Following this, the DBSCAN technique was applied again using only the merchant name embeddings (Appendix 28). While this adjustment aimed to focus on merchant-specific features, it led to a large number of transactions being labeled as "noise" (cluster -1). For example, when "epsilon" was set to 1 and "minimum samples" to 20, 151,877 points were classified as noise, which is 53.1% of all transactions.

These results suggest two primary issues. First, the merchant name embeddings alone seem to lack sufficient variety to differentiate between distinct merchants effectively. Second, including additional features such as MCC, Country, and Acquirer Code introduces another challenge, which was also observed in the k-means approach: when transactions from the same merchant have different values for these features, the model incorrectly treats them as separate entities.

To address these problems, a greater emphasis was put on the merchant name embeddings while still including other features by scaling the Word2Vec embeddings by a factor of 2 first (Appendix 29) and 5 second (Appendix 30).

All in all, using the scaled embeddings led to a slight increase in the number of clusters (89 and 87 compared to 84 before). Apart from that, the structure of clusters remained relatively similar, containing several different merchants in the same cluster. Despite these adjustments, the DBSCAN approach, even with scaled features, failed to produce satisfactory results.

4.2 GloVe

As a next step, the clustering technique will be retried using GloVe embeddings, to evaluate whether this alternative NLP approach can better capture the semantic and contextual variety needed for effective merchant clustering.

Due to GloVe's higher memory requirements, the dataset was downsized further to 0.3% of the original size, resulting in 13,668 distinct merchant names. The same preprocessing steps as in the Word2Vec model were applied, including the removal of irrelevant expressions like "www" and numerical characters.

After the preprocessing, a text corpus was prepared, containing all unique merchant name tokens. Using the cleaned name corpus, co-occurrence statistics were generated by computing a co-occurrence matrix that captures how often word pairs appear within a pre-defined context window. The window size was set to 10, meaning each word's context was defined by the 10 words appearing before and after it in the corpus. This window size was chosen to strike a balance between capturing both local (nearby tokens) and global relationships (overall distribution) between words.

The GloVe model was trained with key hyperparameters to balance performance and efficiency. The embedding dimensionality was set to 100 to capture semantic details while remaining computationally efficient, with a learning rate of 0.05 for stable updates. The model trained for 25 iterations to allow adequate convergence while managing computational cost. The embeddings were saved as word-vector mappings, and the final merchant name embeddings were computed by averaging the vectors of the words in each name.

The DBSCAN model was now applied to the GloVe embeddings, using both the merchant name embeddings alone and the combined features. When using only the GloVe embeddings of the merchant names, DBSCAN struggled to form any clusters. For a range of "epsilon

values” (0.01, 0.5 and 1) and varying “minimum samples” parameters, the model consistently failed, assigning all points to a cluster 0 (Appendix 31– 33). In a setup with “epsilon” is 0.02 and “minimum samples” is 2, only two merchants were grouped into a single cluster, while the remaining points were classified as noise (cluster -1).

Across most hyperparameter settings, DBSCAN either resulted in a single large cluster or treated the majority of points as noise, failing to capture meaningful groupings or structure in the data. This is because the embeddings are scattered across the feature space, failing to form meaningful aggregations (see Appendix 34).

When using the combined features (merchant name embeddings along with MCC, Country, and Acquirer Code), the dataset had to be scaled down further to 0.15% of the original size due to the increased memory requirements for processing multiple features. Here, the clusters were more distributed which could also be observed in the visualization of the features in Appendix 35. However, the resulting clusters still did not indicate any logical aggregation, with many clusters containing different merchants (see Appendix 36 and 37).

Overall, the results indicate that Word2Vec embeddings perform better at clustering than GloVe embeddings, likely due to Word2Vec's ability to capture contextual nuances in sequential data, whereas GloVe relies solely on co-occurrence statistics.

In the end, the findings suggest that clustering all data points in the dataset is not an ideal approach due to the significant variety and complexity of merchant names, which make it difficult to form consistent and meaningful groupings.

4.3 Word2Vec & GloVe with Filtering Logic

As the objective to aggregate all merchants in the dataset proved to be unsuccessful, the next step was to refocus on the dominant merchants in the dataset, as these account for the majority

of transactions. To achieve this, the filtering process developed during the PBL project was applied again (see chapter 3.3), but with a key modification in how similarity is calculated.

After the Data Cleaning, a pre-defined list of the most common merchants in the dataset was created, and transactions were filtered using the key features MCC, Country, and Acquirer Code, assigning the most common values for these features to each merchant in the list. Only transactions matching these assigned values were retained for further processing.

Instead of calculating the Similarity Score and Levenshtein Distance as in the PBL project, this approach now leverages Word2Vec and GloVe embeddings of the cleaned merchant names. The similarity is determined by calculating the cosine similarity between the name embeddings of each transaction and the embeddings of the predefined merchant names.

In this approach, it was not feasible to combine the merchant name embeddings with additional features like MCC, Country, and Acquirer Code as the predefined merchants are operating across multiple countries or are associated with multiple MCC's. Including all possible feature combinations in the embedding space would have diluted the embeddings and weakened the results. Therefore, the approach focused exclusively on merchant name embeddings created using Word2Vec and GloVe.

The Word2Vec model correctly mapped merchant names such as "bolteuo" and "bolteud" to "bolt" (Appendix 38). This accurate mapping stems from the effectiveness of the filtering process, where "Bolt" was the only merchant with matching MCC, country, and Acquirer Code values (see Table 1). However, despite the correct match, the cosine similarity score for this pairing was slightly negative (-0.012), indicating a low similarity between the embeddings of the transaction and the predefined merchant. Additionally, the correct matching of "Revolut" showcased the effectiveness of the model if the semantics are perfectly similar. In this case, the embedding of "Revolut" in the dataset ([0.002, -0.0016, -0.0041, -0.0006, 0.0038, ...])

perfectly matched the embedding of "Revolut" in the predefined list, resulting in a perfect cosine similarity of 1.

Merchant Name	Mapped Merchants	Cosine Similarities	Results
bolteuo	['bolt']	[-0.012]	bolt
revolut	['revolut']	[1]	revolut
googleads	['facebook', 'google', 'apple', 'meta']	[-0.013, -0.031, -0.032, -0.124]	facebook
applecombill	['facebook', 'google', 'apple']	[0.019, 0.001, -0.031]	facebook

Table 1 – Example Outputs from Word2Vec Model with Merchant Mapping

On the other hand, the model struggled with merchants like "googleads," which was mapped to multiple pre-defined merchants, including "Facebook" and "Google." All cosine similarity scores for "googleads" were negative, indicating weak alignment with the embeddings of pre-defined merchants, and the transactions was incorrectly assigned to "Facebook". Similarly, the merchant "applecombill" was incorrectly mapped to "Facebook" instead of "Apple," as the noisy attachment "combill" disrupted the similarity between the embeddings. This highlights Word2Vec's sensitivity to slight variations in merchant names.

The GloVe model (Appendix 39) performed worse than Word2Vec. For instance, the similarity score between "bolteuo" and "Bolt" was more negative (-0.2976) compared to the Word2Vec score, despite the correct mapping.

Merchant Name	Mapped Merchants	Cosine Similarities	Results
bolteuo	['bolt']	[-0.2976]	bolt
revolut	['revolut']	[1]	revolut
googleads	['facebook', 'google', 'apple', 'meta']	[0.0911, 0.0514, -0.0644, -0.1360]	facebook
applecombill	['facebook', 'google', 'apple']	[0.019, 0.001, -0.031]	facebook

Table 2 – Example Outputs from GloVe Model with Merchant Mapping

Another issue was observed with "amazonprimebr," which was incorrectly mapped to "Netflix" instead of "Amazon." This mismatch can be attributed to GloVe's reliance on co-occurrence

statistics, where the frequent co-occurrence of terms like "prime" and "Netflix" in contexts such as "streaming services" likely influenced the embeddings.

Taking a closer look at the embeddings (Appendix 40) highlights the nature of these mismatches. The embedding for "amazonprimebr" was relatively neutral, with most values close to zero ([0.0017, 0.0090, 0.0156, -0.0010, ...]), indicating a weak directionality in the embedding space. In contrast, "Netflix" had more prominent pronounced values in its embedding ([-0.9772, 0.0188, 0.1838, 1.4762, ...]), reflecting a well-defined concept such as "streaming services." While "Amazon" also showed string variations ([0.6767, 0.1799, -0.5816, -1.2919, ...]), its directionality likely diverged from "amazonprimebr" due to contextual differences or noise introduced by the "prime" and "br" tokens. The negative similarity score with "Amazon" suggests greater divergence compared to "Netflix."

The results of the Word2Vec and GloVe models were evaluated using the same test dataset as in the PBL model to ensure comparability. Word2Vec achieved an accuracy of 89.93%, with 2.81% of transactions classified as "small merchants." In contrast, GloVe performed significantly worse, achieving an accuracy of only 63.54%, while the percentage of "small merchants" remained the same at 2.81%.

Model	Accuracy	% of "small merchant"
PBL Model (Baseline)	99.81%	3.66%
Word2Vec	89.93%	2.81%
GloVe	63.54%	2.81%

Table 3 – Performance of Word2Vec & GloVe Model with Filtering Logic

5 Discussion

This work aimed to explore how Word2Vec and GloVe word embeddings could enhance the accuracy of merchant identification and aggregation in transaction data, addressing the research question: “Do Word2Vec and GloVe word embeddings enhance the accuracy of merchant

identification and aggregation in transaction data, compared to traditional string-matching techniques?” Several approaches were explored, combining Word2Vec and GloVe embeddings with clustering algorithms such as k-means and DBSCAN. While some progress was made in enhancing merchant name aggregation, the results revealed several limitations. The study tested three hypotheses by applying these embeddings with clustering techniques and filtering logic to aggregate merchants in the dataset.

H1: Utilizing word embeddings to represent merchant names will improve the accuracy of merchant identification and aggregation compared to traditional string-matching techniques.

This hypothesis was not supported. While the number of transactions classified as "small merchants" decreased, the overall performance of both Word2Vec and GloVe models was worse compared to the 99.81% accuracy achieved by the string-matching approach used in the PBL project. Both models also failed to aggregate more merchants than the PBL model, as the attempt to cluster all merchant names - rather than focusing solely on the dominant merchants - proved unsuccessful due to the variety in merchant names and feature values.

H2: The Word2Vec model will demonstrate superior performance compared to GloVe for merchant name aggregation within the Spark infrastructure.

This hypothesis was supported. Word2Vec demonstrated superior performance, achieving an accuracy of 89.93% compared to 63.54% for GloVe. Word2Vec’s ability to capture contextual nuances made it more robust for representing merchant names, whereas GloVe struggled with semantic diversity and noise due to its reliance on co-occurrence statistics. Additionally, Word2Vec’s lower computational demands and native compatibility with Spark allowed the inclusion of a larger dataset and thus more data to produce meaningful results.

H3: *Incorporating additional contextual features - such as transaction location, the Merchant Category Code and the Acquirer Code - with name-based embeddings will improve the accuracy of merchant identification over using name-based embeddings alone.*

This hypothesis was not supported. While incorporating features such as MCC, Country, and Acquirer Code enhanced clustering variability by distributing merchant names more broadly across clusters, it introduced significant inconsistencies. Merchants with the same names were often split into different clusters due to variations in feature values across transactions.

All in all, clustering approaches revealed critical limitations in handling the dataset's diversity and high dimensionality. K-means, while computationally efficient, struggled with defining the optimal number of clusters and often aggregated unrelated merchants. DBSCAN, despite its adaptability, treated many transactions as noise or formed overly broad clusters. These results underscore the inherent challenges of clustering methods when applied to noisy, high-dimensional datasets.

Focusing on dominant merchants through a filtering process significantly improved results, particularly for the Word2Vec model, which highlighted the importance of manual filtering and feature engineering in transaction data, as generic machine learning techniques fail to achieve meaningful aggregations. However, even with these improvements, neither Word2Vec nor GloVe outperformed the string-matching approach of the PBL project, which achieved a nearly perfect accuracy of 99.81% in the test dataset while classifying only 3.66% of transactions as “small merchant”, indicating that this method is better suited for aggregating the major companies in financial transaction data.

One possible explanation is that both Word2Vec and GloVe are optimized for large text corpora where words have rich semantic relationships within a context, making them highly effective for tasks like text generation, completion or summarization. In contrast, this study

focused on aggregating merchants based on specific attributes like their name, shared MCCs, countries or Acquirer Codes. Unlike text-based contexts, merchant names often lack sufficient internal relationships, reducing the effectiveness of embedding-based approaches in capturing meaningful similarities.

Subsequent studies should explore the development of enhanced filtering and data cleaning measures. As shown in this work, manual interventions such as noise removal and feature filtering were the most effective methods for improving merchant aggregation. Exploring alternative similarity measures (e.g. Euclidean, Jaccard distance, etc.) could also yield improvements, depending on the structure of the embeddings and the nature of the dataset.

Resource limitations were a recurring challenge in this work as GPU access was not available and resources had to be shared across teams at the company. This required significant downsizing of the dataset, which reduced the variability of the merchant names and limited the models in their ability to capture subtle variations or rare patterns within the data. Future research with access to larger computational resources could test these methods on the full dataset or leverage more sophisticated NLP models like BERT or GPT. These advanced models could address the limitations of Word2Vec and GloVe by enabling a deeper understanding of merchant names through contextual embeddings, handling variations in naming conventions, and potentially improving clustering performance by more accurately aggregating similar entities based on nuanced attributes.

Finally, the absence of public datasets for merchant identification and aggregation remains a significant barrier; developing and sharing such datasets would enable broader research, support the testing of advanced techniques, and foster collaboration within the research community.

References

- Ayesha, Shaeela, Muhammad Kashif Hanif and Ramzan Talib. 2020. "Overview and comparative study of dimensionality reduction techniques for high dimensional data". *Information Fusion* 59 (July): 44-58. <https://doi.org/10.1016/j.inffus.2020.01.005>.
- Bommasani, Rishi, Drew A. Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S. Bernstein et al. 2022. "On the Opportunities and Risks of Foundation Models." arXiv:2108.07258.
- Boulieris, Petros, John Pavlopoulos,, Alexandros Xenos and Vasilis Vassalos. "Fraud detection with natural language processing." *Mach Learn* 113 (July): 5087–5108. <https://doi.org/10.1007/s10994-023-06354-5>.
- Chang, Jia-Wei, Neil Yen and Jason C. Hung. "Design of a NLP-empowered finance fraud awareness model: the anti-fraud chatbot for fraud detection and fraud classification as an instance." *J Ambient Intell Human Comput* 13 (March): 4663–4679. <https://doi.org/10.1007/s12652-021-03512-2>.
- Citibank. 2015. "Merchant Category Codes". Accessed December 8, 2024. <https://www.citibank.com/tts/solutions/commercial-cards/assets/docs/govt/Merchant-Category-Codes.pdf>.
- Daniel, Jesse. 2019. *Data Science with Python and Dask*. 1st ed. New York: Simon and Schuster. 6-7.
- Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." *Proceedings of the 2019 Conference of the North American Chapter of the Association for*

Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers). 4171–4186. 10.18653/v1/N19-1423.

Ertel, Wolfgang. 2024. *Introduction to Artificial Intelligence*. 3rd ed. Wiesbaden: Springer.

Eurostat. 2024. “E-commerce statistics for individuals.” Accessed December 8, 2024.

https://ec.europa.eu/eurostat/statistics-explained/index.php?title=E-commerce_statistics_for_individuals#of_EU_citizens_aged_16-74_years_bought_or_ordered_goods_or_services_online_in_2023.

GDPR. 2016a. “Art. 5 GDPR Principles relating to processing of personal data.” Accessed December 8, 2024. <https://gdpr-info.eu/art-5-gdpr/>.

GDPR. 2016b. “Art. 32 GDPR Security of processing.” Accessed December 8, 2024. <https://gdpr-info.eu/art-32-gdpr/>.

Heaton, J.B., Nick Polson and Jan Witte. 2016. “Deep Learning for Finance: Deep Portfolios.” *Applied Stochastic Models in Business and Industry* 33 (1): 3-12.
<http://dx.doi.org/10.2139/ssrn.2838013>.

Hilal, Waleed, S. Andrew Gadsden and John Yawney. 2022. “Financial Fraud: A Review of Anomaly Detection Techniques and Recent Advances.” *Expert Systems with Applications* 193 (May): 1-34.

Kondrak Grzegorz. 2005. “N-Gram Similarity and Distance”. In *String Processing and Information Retrieval*, edited by Mariano Consens and Gonzalo Navarro, 115-126. Berlin: Springer.

Lecun, Yann, Y. Bengio and Geoffrey Hinton. 2015. "Deep Learning." *Nature* 521 (May): 436-444. 10.1038/nature14539.

Manning, Christopher D. and Hinrich Schütze. *Foundations of Statistical Natural Language Processing*. 1st ed. Massachusetts: The MIT Press. 447.

Mikolov, Tomas, Martin Karafiat, Lukas Burger, Jan Cernocky and Sanjeev Khudanpur. 2010. "Recurrent neural network based language model." *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association*. 1045-1048. 10.21437/Interspeech.2010-343.

Mikolov, Tomas, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. "Efficient Estimation of Word Representations in Vector Space." *Proceedings of the 1st International Conference on Learning Representations*. 1-12. arXiv:1301.3781.

Mikolov, Tomas, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013b. "Distributed Representations of Words and Phrases and Their Compositionality." *Advances in Neural Information Processing Systems 26 (NIPS 2013)*. 1-9. arXiv:1310.4546.

Pennington, Jeffrey, Richard Socher, and Christopher Manning. 2014. "GloVe: Global Vectors for Word Representation." *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 1532–1543. 10.3115/v1/D14-1162.

Po, Daw Khin. 2020. "Similarity Based Information Retrieval Using Levenshtein Distance Algorithm." *International Journal of Advances in Scientific Research and Engineering* 6 (4): 6-17. 10.31695/IJASRE.2020.33780.

- Radford, Alec, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. "Improving Language Understanding by Generative Pre-Training." Accessed December 8, 2024. <https://www.mikecaptain.com/resources/pdf/GPT-1.pdf>.
- Statista. 2024. "Volume of data/information created, captured, copied, and consumed worldwide from 2010 to 2023, with forecasts from 2024 to 2028." Accessed December 8, 2024. <https://www.statista.com/statistics/871513/worldwide-data-created/>.
- Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. "Attention Is All You Need." *Advances in Neural Information Processing Systems* 30. 5998-6008. arXiv:1706.03762.
- Vucetic, Danilo, Mohammadreza Tayaranian, Maryam Ziaeeefard, James J. Clark, Brett H. Meyer and Warren J. Gross. "Efficient Fine-Tuning of BERT Models on the Edge." *2022 IEEE International Symposium on Circuits and Systems (ISCAS)*: 1838-1842, 10.1109/ISCAS48785.2022.9937567.
- Xu, Dongkuan and Yingjie Tian. 2015. "A Comprehensive Survey of Clustering Algorithms." *Annals of Data Science* 2 (2): 165-193. 10.1007/s40745-015-0040-1.

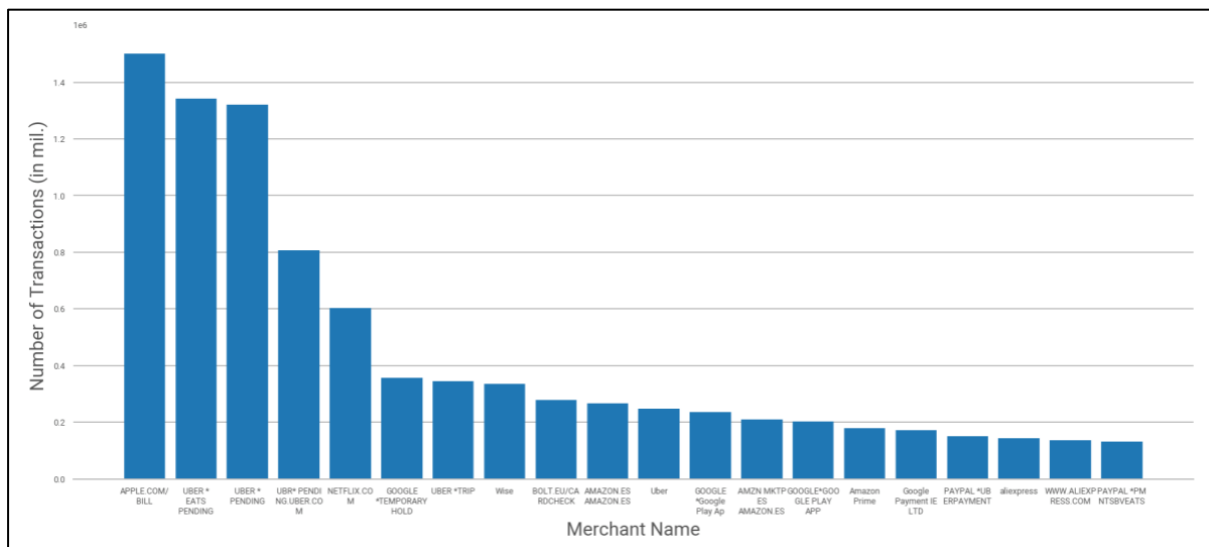
Appendix

Appendix 1 – Data Dictionary

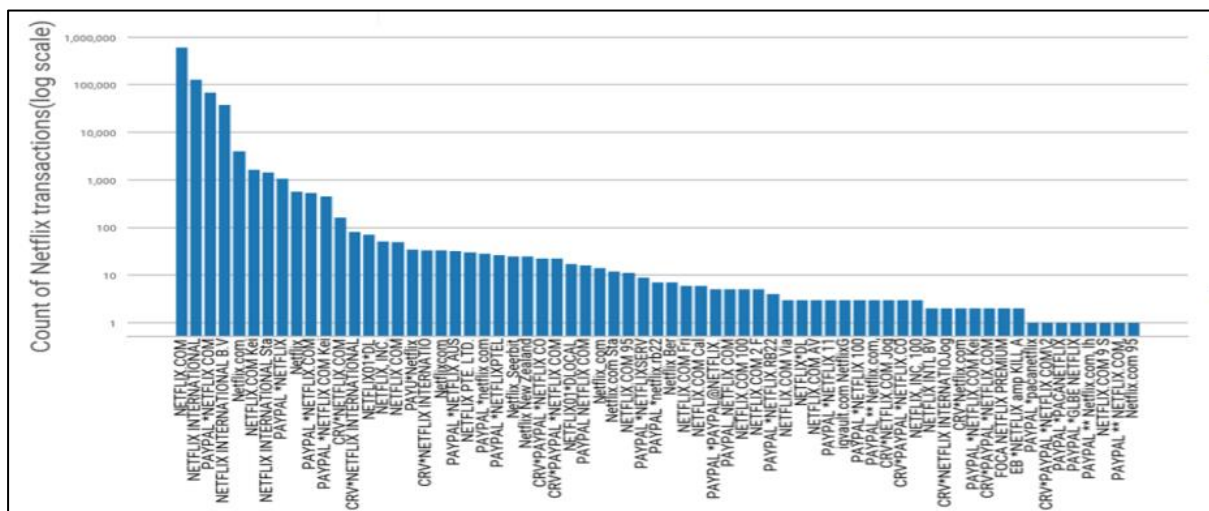
Variable Name	Type	Description	Example	Distinct Values	% Missing
HashedID	string	hashed identifiers corresponding to each transaction	a5ce-bcb8834489c09ca1996122...	28,603,301	0,0%
MCC	Int	Code description for a group of merchants	4112	558	0,0%
MERCHANT_NAME	string	Merchant name	CP COMBOIOS PORTUGAL	2,543,892	0,0%
FRAUD_ACCEPTOR_CDE	string	Company's aggregation for merchants	MULTIBZ00000530626	3,138,276	0,0%
ACQUIRER_CDE	bigint	Number of the acquirer merchant	299073	3,671	0,0%
MERCHANT_EXTERNAL_CDE	string	Code Number of the Merchant	204187450003	444,527	0,0%
TOKEN_REQUESTOR_CDE	string	Code for Token Merchant/Wallet	M50154810714	2,877	77,8%
TOKEN_PAN_ENCRYPTED	string	Token created by the card to be used in several merchants and wallets such as Apple or Uber	6CACBB3C42720A50503853...	2,189,655	65,6%
CARD_RANGE	bigint	First 8 digits of a card	52364170	71	0,0%
OPERATION_CDE	string	10 different codes which correspond to different operations	15	10	0,0%
TRANSACTION_COUNTRY_ISO_ALPHA3	string	Transaction country correspondent to the merchant	PRT	191	0,0%
RESPONSE_STATUS_CDE	string	Final outcome of the transactions	ACCP	3	0,4%
OPERATION_AMOUNT	double	Amount of the transaction	96,4	124,957	0,0%
PAYMENT_SYSTEM_CDE	string	"VIS" if the transaction was accomplished using Visa and "MCC" for MasterCard.	VIS	3	0,0%

PAN_EN-CRYPTED	string	Card number encrypted	2F17E15B4A9CC654C7ACD...	1,681,414	0,0%
RELATIVE_DATETIME	bigint	Relative time difference in seconds between the first seen timestamp in the dataset and each transaction timestamp	5392384	1,681,414	0,0%
WEEK_DAY	string	Day of the week for the current transaction	Thu	7	0,0%
HOUR	int	Hour of the current transaction	13	24	0,0%
MINUTES	int	Minutes of the current transaction	2	60	0,0%
SECONDS	int	Seconds of the current transaction	32	60	0,0%

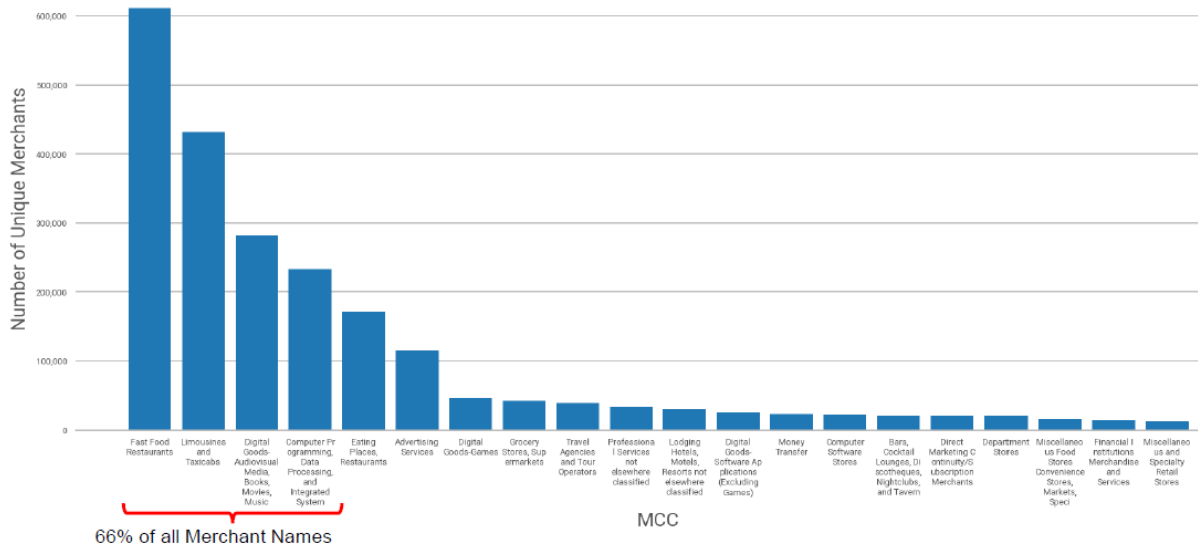
Appendix 2 – Distribution of Merchant Names



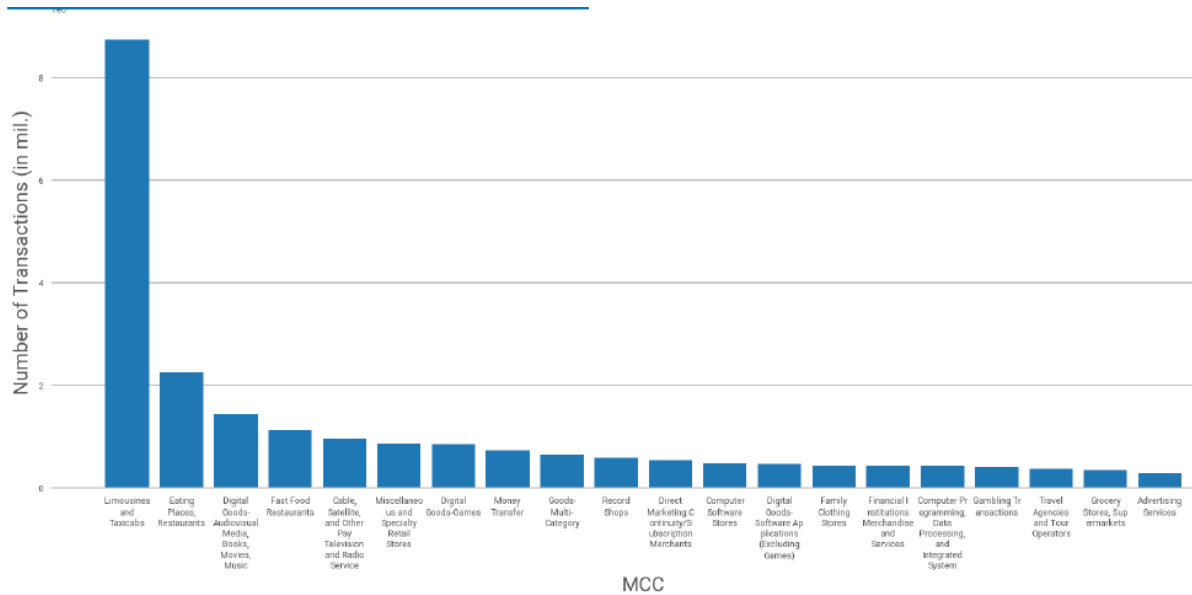
Appendix 3 – Distribution of Merchant Names in Netflix transactions



Appendix 4 – Number of unique merchants per MCC

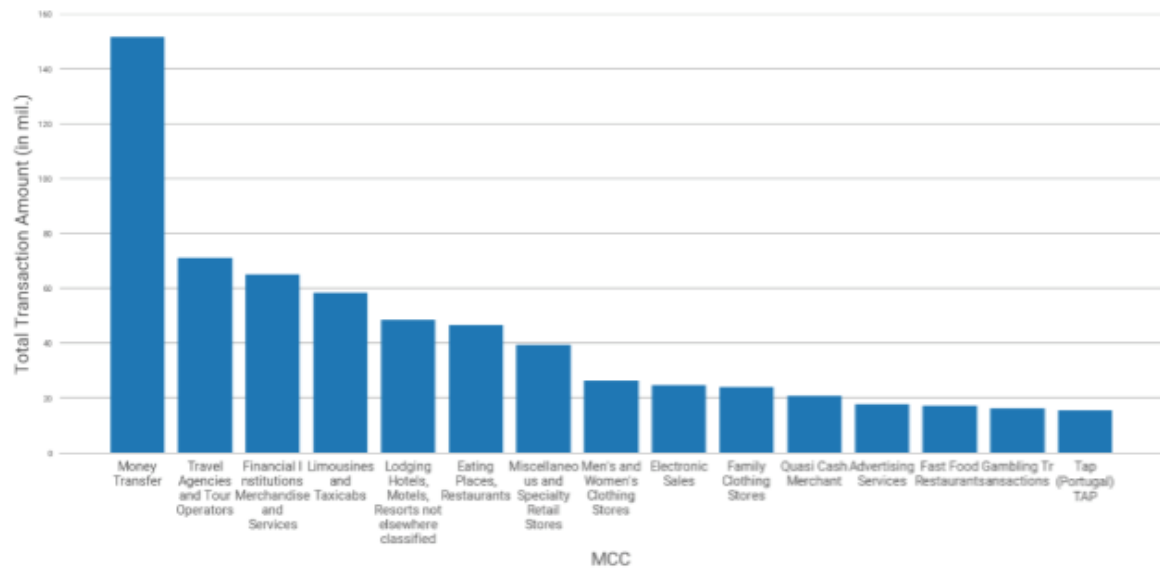


Appendix 5 – Number of transactions per MCC



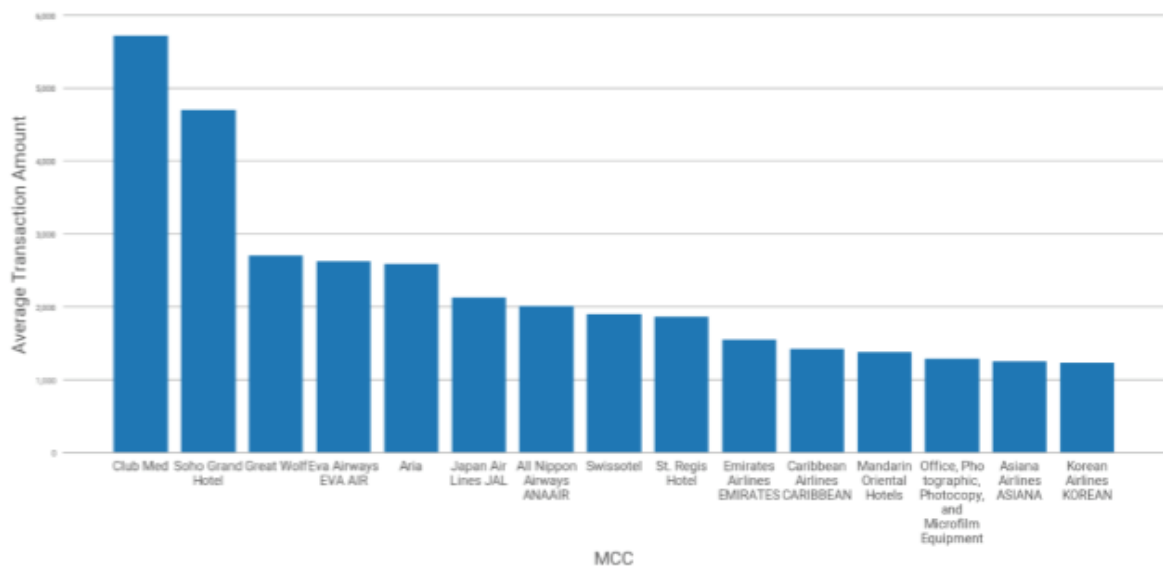
Appendix 6 – Total Transaction Amount per MCC

Total Transaction Amount per MCC (in €):



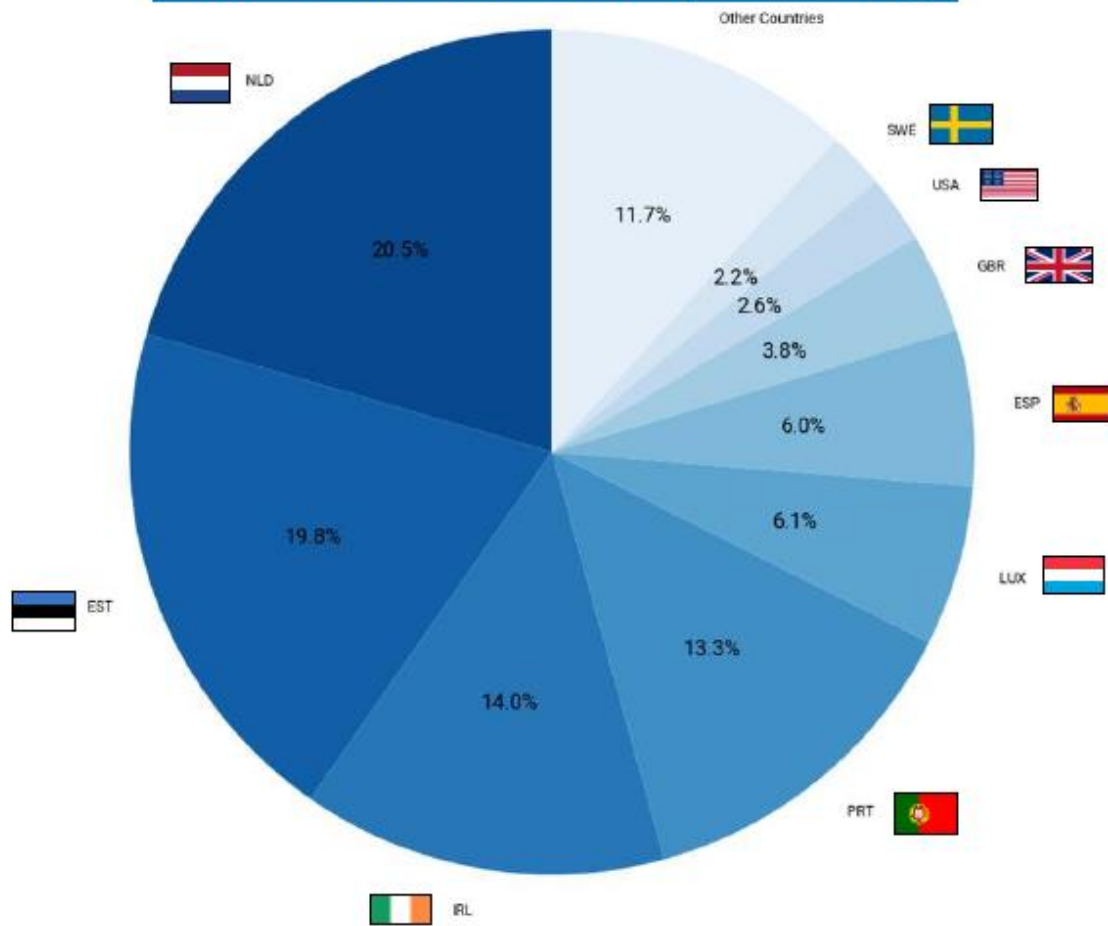
Appendix 7 – Average Transaction Amount per MCC

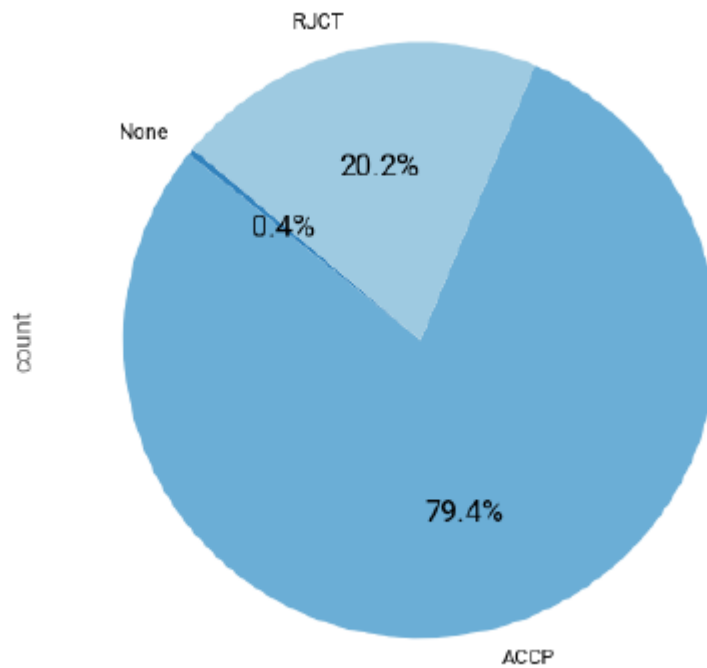
Average Transaction Amount per MCC (in €):



Appendix 8 – Country Distribution

Number of transactions per Country:



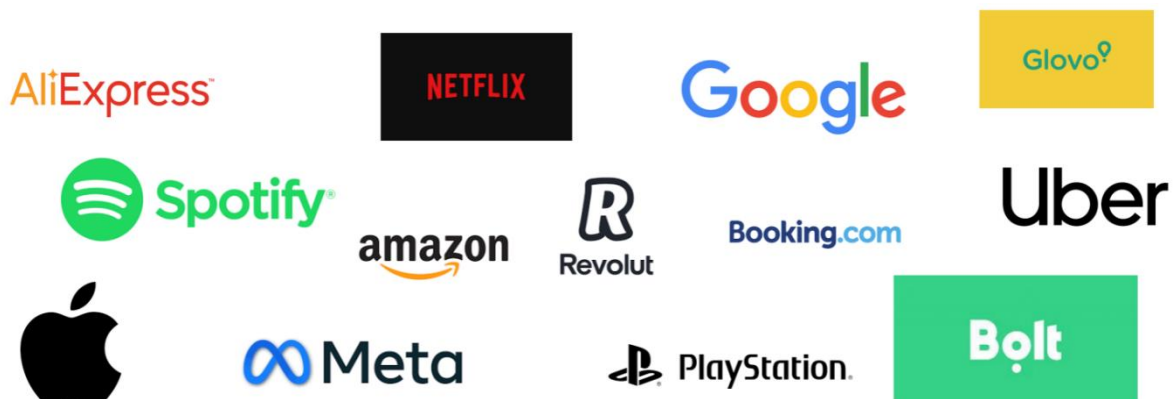


ACCP= Accepted; RJCT= Rejected; None= Null values;

Appendix 10 – Example of a falsely rejected transaction

	MERCHANT_NAME	MCC	ACQUIRER_CDE	MERCHANT_EXTERNAL_CDE	TOKEN_REQUESTOR_CDE	TOKEN_PAN_ENCRYPTED	OPERATION_CDE	RESPONSE_STATUS_CDE	OPERATION_AMOUNT	PAYMENT_SYSTEM_CDE	DAY	TIME
1	card verification	4121	900000013445	526567008630027	None	722850C1F2B7AA7DF77914CC D3FC1352A9588EAE935E68...	ACV	RJCT	0.00	MCC	39	14:06:14
2	card verification	4121	900000013445	526567008630027	None	722850C1F2B7AA7DF77914CC D3FC1352A9588EAE935E68...	ACV	ACCP	0.00	MCC	39	14:06:34
3	BOLT.EU/O/2 211081408	4121	900000013445	526567008630027	None	722850C1F2B7AA7DF77914CC D3FC1352A9588EAE935E68...	012	ACCP	6.06	MCC	39	14:08:06
4	BOLT.EU/O/2 211081425	4121	900000013445	526567008630027	None	722850C1F2B7AA7DF77914CC D3FC1352A9588EAE935E68...	012	ACCP	3.73	MCC	39	14:25:36
5	BOLT.EU	4121	900000020948	000000000210966	None	722850C1F2B7AA7DF77914CC D3FC1352A9588EAE935E68...	012	RJCT	3.47	MCC	45	17:23:27

Appendix 11 – Pre-defined Merchants



Appendix 12 – K-Means Time Complexity

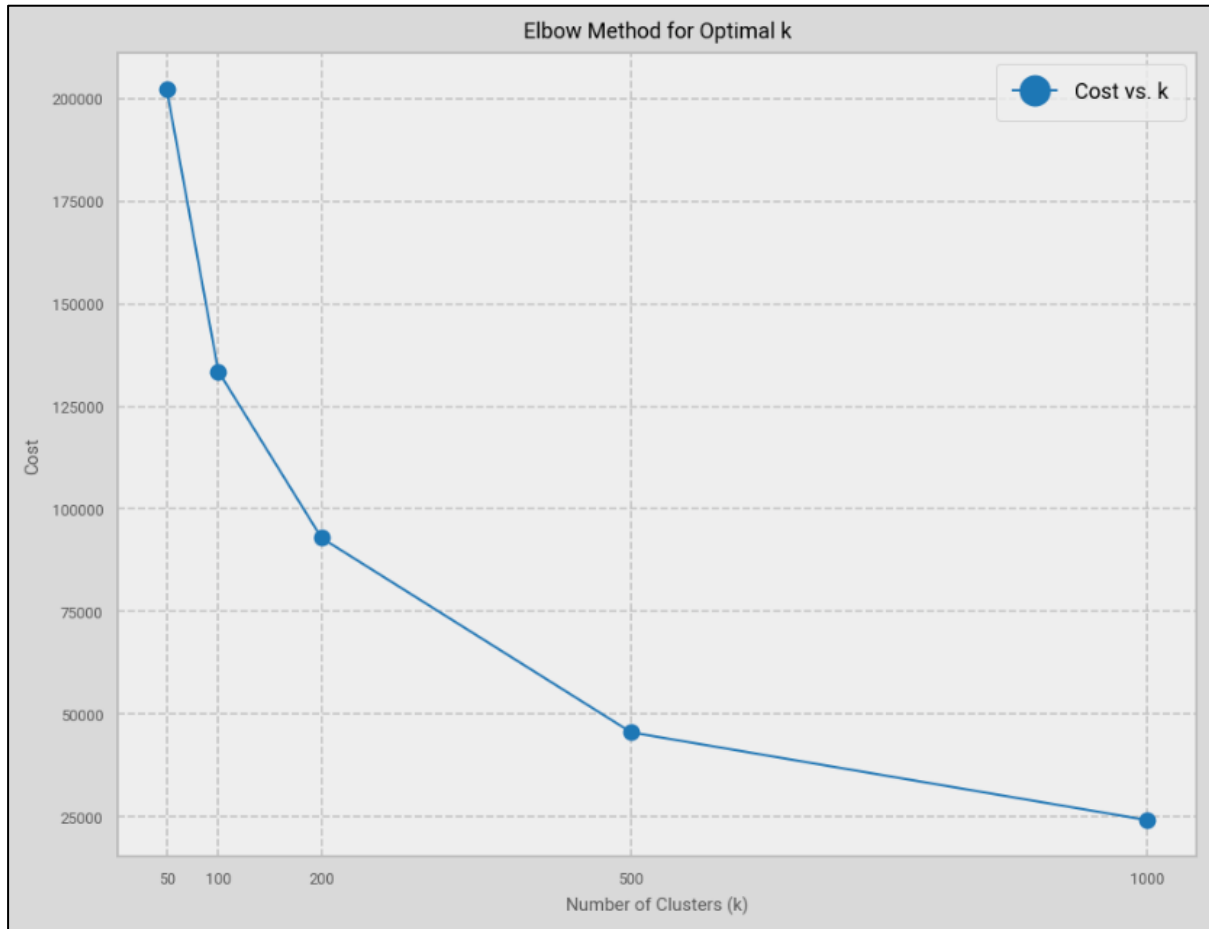
K-means	K-medoids	PAM	CLARA	CLARANS
$O(knt)$	$O(k(n-k)^2)$	$O(k^3n^2)$	$O(ks^2+k(n-k))$	$O(n^2)$

N = total number of data sets
 K = total number of partitions
 T = number of iterations in the clustering process

Appendix 13 – Examples of Cleaned Merchant Names

Original Merchant Name	Cleaned Merchant Name
bolt.eu/o/2212010134	bolteuo
cp comboios portugal	cpcomboiosportugal
alibaba.com	alibabacom
WWW.CONTINENTE.PT	continentept
uber * eats pending	ubereatspending

Appendix 14 – Elbow Graph of Word2Vec & K-Means Model



Appendix 15 – Word2Vec & K-Means Clustering Results (Combined Features) – k=50

Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5
glovocardverificatio	ubereatspending	revolut	naturitas	lookfantastic
glovodecopgbkv	ubereatspending	googlegooglestorage	leroym MerlinBadajoz	googletemporaryhold
glovodecsejznzxs	ubereatspending	revolut	ikeagoyahfb	sundaybarremo
glovodeclisenhyx	ubereatspending	itunesappstap	balkanica distral	microsoftstore
glovodecopvazcjala	ubereatspending	udemyonlinecourses	impextrom	yayemco
glovodeclikbse	ubereatspending	googlepaymentieltldgoo	el corte Ingles departa	onlyfans
glovodecliyygzdhu	ubereats	googlegoogleplayapp	maksu	googletemporaryhold
glovodecliqhs	ubereatspending	itunesappstapp	orodetede chus	tflunpaidfares
glovodeclidmlwd	ubereatspending	googleyoutube	sentimental journey	ramp(svwfctvfqnxyu)
glovodeclikghlu	ubereatspending	googlegooglestorage	expjabasc	ricssubseurcybs
glovodecliprybelb	ubereatspending	revolut	motocardcom	googletemporaryhold
glovodecammxuadw	ubereatspending	revolut	el corte Ingles sportown	googleyoutube premium

glovodecliztfuue	ubereatspending	applecombill	caramelospaco	amazonprime
glovoglovoprime	ubereatspending	googlepaymentieltldgoo	wdfgbarcelona	googletemporaryhold
glovodecmgbhlmq	ubereatspending	googlegoogleplayapp	golosinasmeloso	ehostingtcom
glovodeclidmeb	ubereatspending	googlegooglestorage	techeducationrights	googletemporaryhold
glovocardverificatio	ubereats	revolut	mialcampoprincipecarlos	googletemporaryhold
worten	ubereatspending	microsoftxbox	labotigadelpoble	gamerdircom
glovodecljcccl	ubereatspending	googlegoogleplayapp	franciscolazaromartin	winterwonderland
glovodecbrpjlxrt	ubereatspending	googlegoogleplayapp	supermercadosemirsl	viatorit
glovodecvimfxprk	ubereatspending	googlesamfina	nutritienda	activtradescom
glovodecopnnhcvaj	ubereatspending	googlepaymentieltldgoo	adenosasturias	quirkoscloud
glovodeclefbxdbg	ubereatspending	badoohelp@badoocom	costaluzh	googletemporaryhold
glovodecliqusfz	ubereatspending	segpaycomsepg	superasia	googletemporaryhold
glovodeccoqteyu	ubereatspending	googlegoogleplayapp	sentimentaljourney	of
glovodecopzqlzs	ubereatspending	itunesappstapp	carrefrescantos	skrill
glovodeclidytgwwala	ubereatspending	badoohelp@badoocom	sentimentaljourney	googletemporaryhold
glovodecagxmhhxn	ubereatspending	patreonmembership	ikeaibericaweb	of
glovocardverificatio	ubereatspending	itunesappstapple	hotelmeliabarajas	taskerontaskrabit
mcdgondomar	ubereatspending	googlegoogles	expjuansebastianelcano	googletemporaryhold
glovodecopdwzvt	ubereatspending	googledisney	sentimentaljourney	onlyfans
glovodecvcsqgqycp	ubereatspending	googlegoogleplayapp	spainlatamservicios	googletemporaryhold
glovodeclittlgwn	ubereatspending	googleyoutube	cedipsacruceesp	howbecome
glovodeccbujbnbp	ubereatspending	revolut	maksu	googletemporaryhold
glovodecpoqvwfwgtala	ubereatspending	googlegoogleplayapp	eurobikesstoresl	onlyfanscom
glovodecliwaecn	ubereatspending	googlegoogleplayapp	supermercadofroiz	paddlenet
glovodecoprplkx	ubereatspending	revolut	repsolwaylet	mubicom
glovodeclidcvlq	ubereats	googlepaymentieltldgoo	sentimentaljourney	googletemporaryhold
worten	ubereatspending	linkedin	lidltudela	playstationnetwork
glovodecliffta	ubereatspending	microsoftxbox	gadislavictoria	paddlenetpaddlenet
glovodeclidmhxt	ubereatspending	revolut	oxinity	onlyfanscom
glovodeclinrxjrbmd	ubereatspending	googlegoogleplayapp	sentimentaljourney	googletemporaryhold
glovodeclifyaqsb	ubereatspending	googlepaymentieltldgoo	depositohidrograficosl	googlegooglep
glovodeclidkcyjcala	pntvorbantotal	googlelivetec	esdomaio	googletemporaryhold
glovodecopmjjb	ubereats	googlepaymentieltldgoo	lavaturnoatendido	googletemporaryhold
glovodecvipqtrqy	ubereatspending	googletiktok	bonarea	runwildportcom
glovodeclibxldjala	ubereatspending	applecombill	monesterioautovia	of

glovodecepcyjegy	ubereatspending	googlegoogleplayapp	cedipsaesraigada	wayaheadgroofei
glovodeclirlnvsh	ubereatspending	googlepaymentieltldgoo	esbtp	googletemporaryhold
glovodecpdvcvh	ubereatspending	googlegoogle	elcorteinglespintorso	taskerontaskrabbit

Appendix 16 – Word2Vec & K-Means Clustering Results – k=200

Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5
tapwebunitedkingdom	hbomax	mgpvinted	ubrpendingubercom	zarainternetportugal
ryanair	tidalmusic	mgpvinted	ubrpendingubercom	calzedoniaccfaro
ryanair	spotifypfcaf	mgpvinted	ubrpendingubercom	zaracomworldwide
ryanair	hbomax	mgpvinted	ubertrip	zarainternetportugal
tapwebportugalflyt	spotifypfcf	mgpvinted	ubrpendingubercom	fashionreta
ryanair	spotifypfe	mgpvinted	ubrpendingubercom	okaaspain
tapairportugal	spotifypfcc	mgpvinted	ubrpendingubercom	leftiescom
ryanair	spotifypfcbbbb	googleyoutubeme	ubrpendingubercom	dospieselfaro
ryanair	hbomax	mgpvinted	ubrpendingubercom	lefties
tapccenterportugal	spotifypfebe	mgpvinted	ubrpendingubercom	payesdemarca
ryanair	hbomax	mgpvinted	ubrpendingubercom	zarainternetportugal
tapairportugal	spotifypfe	mgpvinted	ubertrip	brildorsl
ryanair	spotifypfebe	mgpvinted	ubrpendingubercom	krackzapateriasl
tapwebitaly	spotifypf	mgpvinted	ubertrip	massimoduttiportugal
ryanair	spotifypfefce	mgpvinted	uber	zara
tapwebportugalflytap	spotifypffa	mgpvinted	ubrpendingubercom	zara
ryanair	spotifypfcc	mgpvinted	ubrpendingubercom	buganco
ryanair	spotifypfbaaf	mgpvinted	ubrpendingubercom	leftiescom
britishairwayspteco	spotifypfbb	mgpvinted	ubrpendingubercom	bershkaportugal
ryanair	spotifypfed	mgpvinted	ubrpendingubercom	puntofaslma
tapwebportugalplusgrad	spotifypfad	mgpvinted	ubrpendingubercom	zara
ryanair	spotifypfbac	mgpvinted	ubrpendingubercom	dakondacom
tapwebportugalflytap	spotifypfbed	mgpvinted	ubrpendingubercom	spohlabela
tapwebportugalflytap	spotifypfbc	mgpvinted	ubertrip	puntofaslma
tapwebportugalflyt	hbomax	mgpvinted	ubrpendingubercom	primarkvigo
ryanair	spotifypfdeda	mgpvinted	ubrpendingubercom	fashionreta
tapwebportugalflyt	spotifypfa	mgpvinted	ubrpendingubercom	babaadesign
tapwebportugalflytap	spotifypfaea	mgpvinted	ubrpendingubercom	dfranklin

tapwebportugalflytap	tidalmusic	mgpvinted	ubrpendingubercom	zarabarcelonadiagonal
ryanair	spotifypfeef	mgpvinted	ubertrip	zara
ryanair	spotifypfab	mgpvinted	ubrpendingubercom	vans
ryanair	spotifypcf	mgpvinted	ubrpendingubercom	beurbanaddon
tapwebportugalflyt	hbomax	mgpvinted	ubrpendingubercom	thedesireshop
tapdcspportugal	spotifypfce	mgpvinted	ubrpendingubercom	dectejodolcevita
tapwebportugalflyt	hbomax	mgpvinted	ubrpendingubercom	calzedoniaccfaro
finnair	spotifypfdcbdd	mgpvinted	uber	toieraslisboa
ryanair	spotifypfec	mgpvinted	ubrpendingubercom	restodefabrica
tapwebportugalflytap	hbomax	mgpvinted	ubertrip	browniespaincom
tapwebportugalflytap	spotifypfbff	mgpvintedexs	ubertrip	primarkmarbella
tapairline	spotifypfceb	mgpvinted	ubrpendingubercom	zarainternetportugal
ryanair	hbomax	mgpvinted	ubrpendingubercom	adolfodominguez
ryanair	spotifypfebb	mgpvinted	ubrpendingubercom	zarainternetportugal
tapwebportugalflytap	spotifypfebdf	mgpvinted	ubrpendingubercom	zara
ryanair	spotifypfc	mgpvinted	ubrpendingubercom	fashionreta
ryanair	spotifypfbccc	mgpvinted	ubrpendingubercom	calzedonia
ryanair	spotifypfafa	mgpvinted	ubrpendingubercom	zarainternetportugal
ryanair	spotifypfedc	mgpvinted	ubertrip	svetistefan
tapcccenterportugalf	spotifypfdfb	mgpvinted	ubrpendingubercom	fashionreta
iberiaoperadoraadpav	spotifypfece	mgpvinted	ubertrip	fashionreta
ryanair	spotifypfdc	mgpvinted	ubrpendingubercom	martasoto

Appendix 17 – Word2Vec & K-Means Clustering Results – k=500

Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5
servicodetransporte	applecombill	googletemporaryhold	primarkbrageparque	amazonprime
servicodetransporte	applecombill	googletemporaryhold	primarksintraforum	amazonprime
servicodetransporte	applecombill	googletemporaryhold	primarkbrageparque	amazonprime
servicodetransporte	applecombill	suaacademia	primarkportonorteshopp	amazonprime
servicodetransporte	applecombill	googletemporaryhold	primarkalmada	amznmktpes
servicodetransporte	googlegoogleplayap	assinet	primarkbrageparque	amznmktpe
servicodetransporte	googlegoogleplayap	googlegarena	primarklisboncolombo	amznmktpes
servicodetransporte	applecombill	googledisneymobile	primarklisboncolombo	amznmktpes

servicodetransporte	googlegoogleplayap	iugusarahbeatriz	primarkportonorteshopp	amazonmktplceeu
servicodetransporte	googlegoogleplayap	googletemporaryhold	primarklisboncolombo	amazonprime
servicodetransporte	applecombill	sapark	primarkportonorteshopp	amazonprime
servicodetransporte	googlegoogleplayap	googlestorage	primarkbrageparque	amazonprime
servicodetransporte	applecombill	googletemporaryhold	primarksintraforum	amznmktpes
servicodetransporte	applecombill	facebookrtztungpt	primarksintraforum	businessprime
servicodetransporte	applecombill	mpbuser	primarkportonorteshopp	amazonprime
servicodetransporte	applecombill	googletiktoklive	mangoalmadaforumalmada	amazonprime
servicodetransporte	applecombill	pgastrolink	primarklisboncolombo	amazonprime
servicodetransporte	applecombill	googleartsiomkozel	primarklisboncolombo	amazonprime
servicodetransporte	applecombill	movidalocacaodeveicu	lisboaarmazensdochiado	amznmktpes
servicodetransporte	applecombill	googlenianticinc	primarklisboncolombo	amazoncombe
servicodetransporte	applecombill	facebookads	primarkcoimbra	amznmktpes
servicodetransporte	applecombill	pgifckopenhagenifck	primarkportonorteshopp	amazonneu
servicodetransporte	googlegoogleplayap	googlenianticinc	lojasprimarklisbondvi	amznmktpes
servicodetransporte	googlegoogleplayap	wixcom	lojasprimarklisbondvi	amazonprime
servicodetransporte	applecombill	heavyloveeventose	primarklisboncolombo	amznmktpes
servicodetransporte	applecombill	googletemporaryhold	mangomatosinhosikeaopo	amznmktpes
servicodetransporte	applecombill	googledisneymobile	mangosetubalalegrosetu	amazonneu
servicodetransporte	applecombill	dmhelphbomaxcom	primarkalmada	amznmktpes
servicodetransporte	applecombill	googletemporaryhold	primarkportonorteshopp	amazonprime
servicodetransporte	applecombill	raquelbritzenutriti	primarkalmada	amznmktpes
servicodetransporte	applecombill	hulkcopa	primarksintraforum	amazonprime
servicodetransporte	applecombill	googletemporaryhold	lojasprimarklisbondvi	amznmktpes
servicodetransporte	applecombill	googlegamebasics	primarkalmada	amznmktpes
servicodetransporte	applecombill	googlenexters	lojasprimarkportopna	amznmktpes
servicodetransporte	googlegoogleplayap	facebookpjufja	primarkalmada	amazonit
servicodetransporte	applecombill	digipare	primarkportonorteshopp	amazonprime
servicodetransporte	applecombill	googletemporaryhold	primarklisboncolombo	amazonprime
servicodetransporte	applecombill	facebookads	primarklisboncolombo	amznmktpes
servicodetransporte	googlegoogleplayap	facebooktldm	primarklisboncolombo	amazonprime
servicodetransporte	applecombill	googletiktoklive	lojasprimarkportopna	amazonprime
servicodetransporte	applecombill	googletemporaryhold	primarklisboncolombo	amznmktpes
servicodetransporte	googlegoogleplayap	facebookads	primarkportonorteshopp	amazonprime
servicodetransporte	applecombill	googletemporaryhold	primarkportonorteshopp	amznmktpr

servicodetransporte	applecombill	ozielribeirodasilva	primarklisboncolombo	amazonprime
servicodetransporte	applecombill	googletemporaryhold	lojasprimarklisbondvi	amazonprime
servicodetransporte	googlegoogleplayap	facebookkqnpkxj	lojasprimarkportopna	amznmktpes
servicodetransporte	googlegoogleplayap	organizzeteorganizze	primarksintraforum	amznmktpes
servicodetransporte	googlegoogleplayap	googletemporaryhold	primarksintraforum	amazonneu
servicodetransporte	applecombill	googlestorage	lisboaamoreirassc	amazonprime
servicodetransporte	applecombill	googlecanvptyltd	primarkportonorteshopp	amznmktpes

Appendix 18 – Word2Vec & K-Means Clustering Results – k=1000

Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5
glovocardverificatio	ubereatspending	amznmktpesamazones	ms	hbomax
glovodecopgbkv	ubereatspending	amznmktpesamazones	scalapaysrl	tidalmusic
glovodecsejznzxs	ubereatspending	amazonesamazones	scalapaysrl	spotifypfcaf
glovodeclisenhyx	ubereatshelpercom	amazonitamazoni	scalapaysrl	hbomax
glovodecopvazcjala	playstationnetwork	amznmktpesamazones	scalapaysrl	spotifypfcf
glovodeclikbse	ubereatspending	amznmktpesamazones	scalapaysrl	spotifypfe
glovodecliyygzdhu	ubereatshelpercom	amazonesamazones	scalapaysrl	spotifypfcc
glovodecliqhs	playstationnetwork	amazonesamazones	scalapaysrl	spotifypfcbbbb
glovodeclidmld	ubereatspending	amznmktpesamazones	scalapaysrl	hbomax
glovodeclikghlu	ubereatspending	amazonesamazones	scalapaysrl	spotifypfebe
glovodecliprybelb	ubereatspending	amznmktpesamazones	scalapaysrl	hbomax
glovodecammxuadw	hotelonbookingcom	amazondemarketplace	scalapaysrl	spotifypfe
glovodecliztfuue	playstationnetwork	amazonesamazones	scalapaysrl	spotifypfceb
glovoglovoprime	ubereatspending	amznmktpesamazones	scalapaysrl	spotifypf
glovodecmgblhmq	playstationnetwork	amazonesamazones	scalapaysrl	spotifypfefce
glovodeclidmeh	ubereatshelpercom	amznmktpesamazones	scalapaysrl	spotifypfpa
glovocardverificatio	playstationnetwork	amznmktpesamazones	scalapaysrl	spotifypfcc
worten	ubereatshelpercom	amazonesamazones	scalapaysrl	spotifypfbaaf
glovodecljcccl	ubereatspending	amazonesamazones	deliveroo	spotifypfbb
glovodecbrjlxrt	ubereatspending	amazonesamazones	scalapaysrl	spotifypfed
glovodecvimfxprk	digitalproductsplanoda	amznmktpesamazones	scalapaysrl	spotifypfad
glovodecopnnhcvaj	workshopmetaswc	amznmktpesamazones	dimorrasrl	spotifypfbac
glovodeclefbxdbg	ubereatspending	amazonesamazones	scalapaysrl	spotifypfbed

glovodeclisqsfz	ubereatspending	amznmktpesamazones	scalapaysrl	spotifypfbc
glovodeccoqteyu	ubereatspending	amazonesamazones	scalapaysrl	hbomax
glovodecopzqlzs	ubereatspending	amznmktpesamazones	scalapaysrl	spotifypfeda
glovodeclidygtwwala	ubereatspending	amznmktpesamazones	scalapaysrl	spotifypfa
glovodecagxmhhxn	ubereatspending	amazonesamazones	scalapaysrl	spotifypfaea
glovocardverificatio	ubereatspending	amznmktpesamazones	scalapaysrl	tidalmusic
mcdgondomar	hotelatbookingcom	amznmktpesamazones	scalapaysrl	spotifypfeef
glovodecopdwzvt	playstationnetwork	amazoncoukamazonco	scalapaysrl	spotifypfab
glovodecvcqsgqycp	ubereatspending	amazonesamazones	scalapaysrl	spotifypfcf
glovodeclittlgwn	ubereatspending	amznmktpesamazones	scalapaysrl	hbomax
glovodeccbujbnbp	ubereatspending	amznmktpesamazones	scalapaysrl	landings
glovodecpoqvfwgtala	ubereatspending	amznmktpesamazones	ludwigpremiumsubscri	spotifypfce
glovodecliwaecn	hotelatbookingcom	amznmktpesamazones	scalapaysrl	hbomax
glovodecoprplkx	bookingcom	amznmktpesamazones	scalapaysrl	spotifypfcbdd
glovodeclidcvlq	ubereatspending	amznmktpesamazones	scalapaysrl	spotifypfec
worten	ubereatspending	amznmktpesamazones	scalapaysrl	hbomax
glovodecliffta	ubereatspending	amznmktpesamazones	scalapaysrl	spotifypfbff
glovodeclidmhxt	hotelatbookingcom	amazonesamazones	deliveroo	spotifypfceb
glovodeclinrxrbmd	ubereatspending	amznmktpesamazones	scalapaysrl	hbomax
glovodeclifyaqsb	playstationnetwork	amazonesamazones	scalapaysrl	spotifypfebb
glovodeclidkcyjcala	ubereatspending	amznmktpesamazones	scalapaysrl	spotifypfedf
glovodecopmijb	playstationnetwork	amznmktpesamazones	scalapaysrl	spotifypfc
glovodecvipqtrqy	ubereatspending	amznmktpesamazones	scalapaysrl	spotifypfccc
glovodeclibxldjala	ubereatspending	amazonesamazones	scalapaysrl	spotifypfafa
glovodecepchjegy	ubereatspending	amznmktpesamazones	scalapaysrl	spotifypfedc
glovodeclirlnvlsh	ubereatspending	amznmktpesamazones	scalapaysrl	spotifypfdfb
glovodecpdvcvh	ubereatspending	amznmktpesamazones	scalapaysrl	spotifypfec

Appendix 19 – Word2Vec & K-Means Clustering Results (Name Embeddings) – k=50

Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5
wise	applecombill	uberpending	ubereatspending	amznmktpesamazones
wise	applecombill	uberpending	ubereatspending	spotify
wise	applecombill	uberpending	ubereatspending	spotify
wise	applecombill	uberpending	ubereatspending	amznmktpesamazones

[illegible]

wise	applecombill	uberpending	ubereatspending	amznmktpesamazones
wise	applecombill	uberpending	ubereatspending	amznmktpesamazones
wise	applecombill	uberpending	ubereatspending	amznmktpesamazones
wise	applecombill	uberpending	ubereatspending	amznmktpesamazones
wise	applecombill	uberpending	ubereatspending	spotify
wise	applecombill	uberpending	ubereatspending	spotify
wise	applecombill	uberpending	ubereatspending	spotify
wise	applecombill	uberpending	ubereatspending	amznmktpesamazones
wise	applecombill	uberpending	ubereatspending	amznmktpesamazones
wise	applecombill	uberpending	ubereatspending	amznmktpesamazones
wise	applecombill	uberpending	ubereatspending	spotify

Appendix 20 – Word2Vec & K-Means Clustering Results (Name Embeddings) – k=100

[illegible]

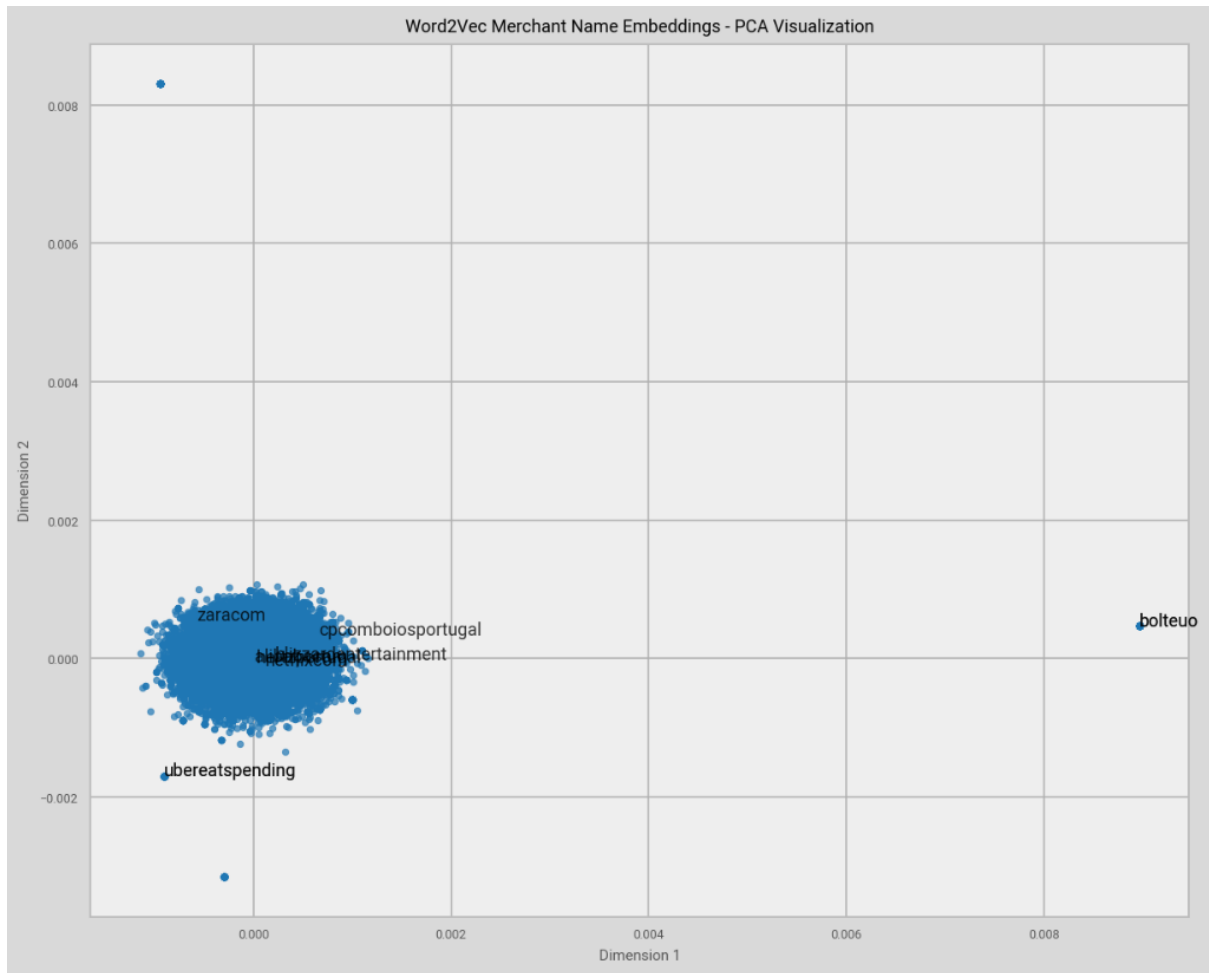
[illegible]

Appendix 21 – Word2Vec & K-Means Clustering Results (Name Embeddings) – k=500

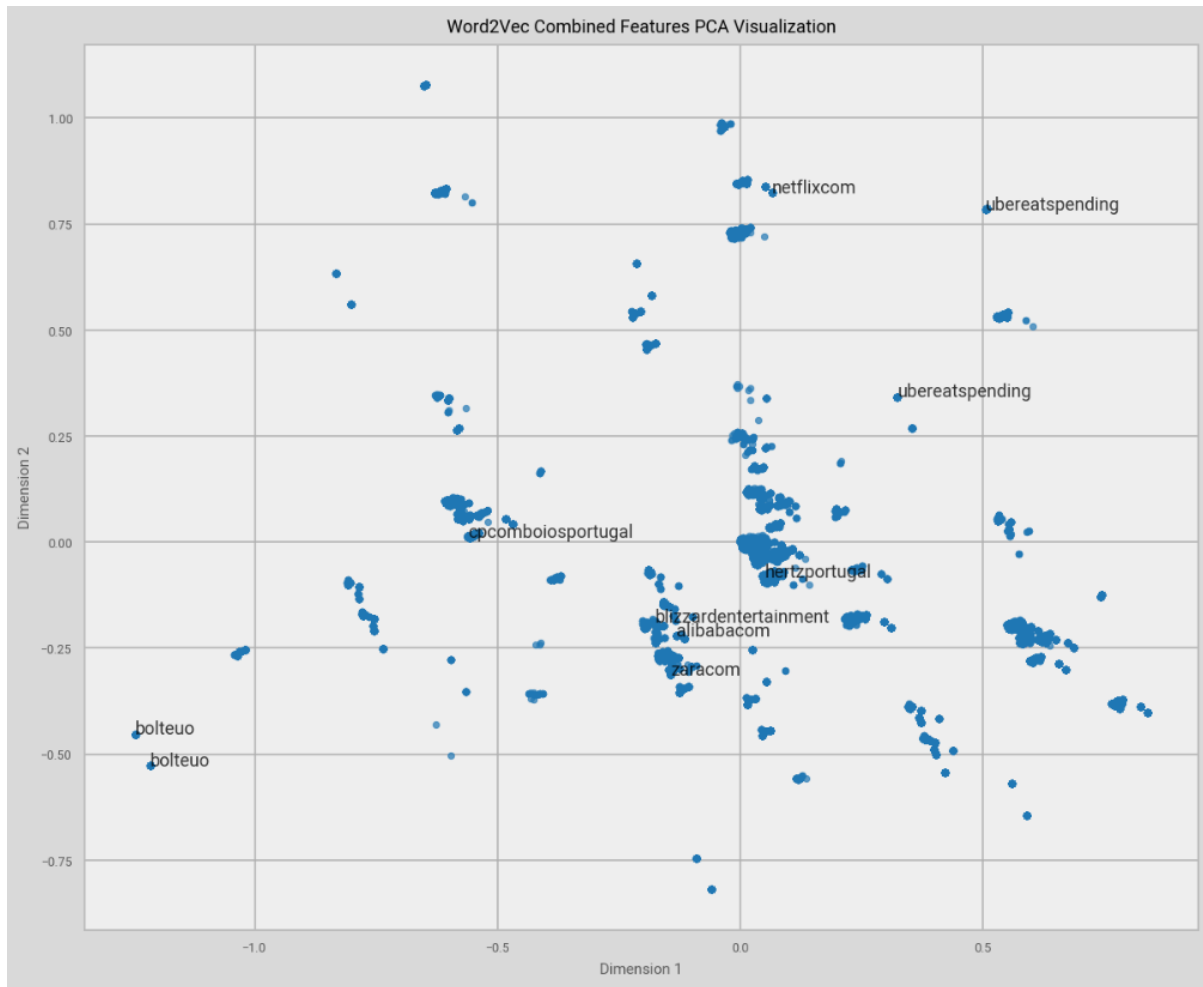
[illegible]

[illegible]

Appendix 22 – Visualization of Word2Vec Merchant Name Embeddings



Appendix 23 – Visualization of Word2Vec Combined Features Embeddings



Appendix 24 - Word2Vec & DBSCAN Clustering Results (Combined Features) – eps=0.5, min_samples=20

Number of Clusters:		Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5
70		cpcomboiosportugal	bolteuo	alibabacom	netflixcom	bolteuo
		cpcomboiosportugal	bolteuo	aliexpress	netflixcom	bolteud
Cluster Sizes:		transpintermodaisporto	bolteuo	ebayo	netflixcom	bolteuo
3	13266	cpcomboiosportugal	bolteuo	aliexpress	netflixcom	bolteur
1	12999	wptredeexpressos	bolteuo	aliexpress	netflixcom	bolteuo
2	12326	wptredeexpressos	bitsaarey	bolteud	blinkist	ubereatspending
4	10420	spotifypfbac	sppetprintedeu	bolteuo	mgpvinted	ubereatspending

6	9965	hbomaxbir	machersrl	bolteuo	googletempo- raryhold	ubereatspend- ing
10	9893	playstationnet- work	zettleamadis- tribu	bolteuovan	flowspectacles	ubereatspend- ing
7	9870	toogoodtogguz- twaryy	praische	bolteuovan	googletempo- raryhold	ubereatspend- ing
8	9760					
5	9573					
16	9404					
12	8979					
17	8025					
11	7435					
13	7154					
15	6769					
9	6571					
0	6401					
33	5554					
14	5442					
-1	5148					

Appendix 25 - Word2Vec & DBSCAN Clustering Results (Combined Features) – eps=0.1, min_samples=20

Number of Clusters:		Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5
89		cpcomboiospor- tugal	bolteuo	alibabacom	netflixcom	bolteuo
		cpcomboiospor- tugal	bolteuo	aliexpress	netflixcom	bolteud
Cluster Sizes:		transpinter- modaisporto	bolteuo	ebayo	netflixcom	bolteuo
2	11364	cpcomboiospor- tugal	bolteuo	aliexpress	netflixcom	bolteur
3	10981	wptredeex- pressos	bolteuo	aliexpress	netflixcom	bolteuo

-1	10180	too-goodtogaquntlg vr	googleyoutube- music	bolteud	techsolutionscy	ubereatspend- ing
8	9997	spotifypeefafb	googletempo- raryhold	bolteuo	mjpgvinted	ubereatspend- ing
1	9084	spotifypfbac	googletempo- raryhold	bolteuo	flowspectacles	ubereatspend- ing
10	9081	hbomaxbir	googletempo- raryhold	bolteuovan	blinkist	ubereatspend- ing
4	8709	toogoodtogguz- twaryy	googletempo- raryhold	bolteuovan	googletempo- raryhold	ubereatspend- ing
17	8187					
15	8026					
5	7996					
11	7671					
7	6755					
16	6449					
0	6358					
12	6129					
14	5570					
22	5538					
20	5468					
27	5285					
6	5115					

Appendix 26 - Word2Vec & DBSCAN Clustering Results (Combined Features) – eps=0.01, min_samples=20

Number of Clusters:	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5
156	cpcomboiospor- tugal	bolteuo	alibabacom	netflixcom	bolteuo
	cpcomboiospor- tugal	bolteuo	aliexpress	netflixcom	bolteud

Cluster Sizes:			transpinter- modaisporto	bolteuo	ebayo	netflixcom	bolteuo
-1	40370		cpcomboiospor- tugal	bolteuo	aliexpress	netflixcom	bolteur
5	9964		cpcomboiospor- tugal	bolteuo	aliexpress	netflixcom	bolteuo
1	8833		spotifypecde	googleyoutube- music	bolteud	shopify- chargecom	mgpvinted
8	8557		spotifypfdd	googletempo- raryhold	bolteuo	lingguiqi	mgpvinted
3	8007		spotifypeefab	googletempo- raryhold	bolteuo	birdapp	mgpvinted
4	6969		spotifypfbac	googletempo- raryhold	bolteuovan	fbpaypayment	mgpvinted
2	6690		hbomaxbir	googletempo- raryhold	bolteuovan	googletempo- raryhold	mgpvinted
11	6665						
10	5850						
13	5584						
6	4726						
7	4575						
14	4455						
31	4264						
0	4178						
19	3827						
22	3709						
12	3586						
32	3378						
17	3260						

Appendix 27 - Word2Vec & DBSCAN Clustering Results (Combined Features) – eps=0.01, min_samples=5

Number of Clusters:		Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5
---------------------	--	-----------	-----------	-----------	-----------	-----------

344		cpcomboiospor-tugal	bolteuo	alibabacom	netflixcom	bolteuo
		cpcomboiospor-tugal	bolteuo	aliexpress	netflixcom	bolteud
Cluster Sizes:		transpinter-modaisporto	bolteuo	ebayo	netflixcom	bolteuo
-1	12438	cpcomboiospor-tugal	bolteuo	aliexpress	netflixcom	bolteur
2	8224	cpcomboiospor-tugal	bolteuo	aliexpress	netflixcom	bolteuo
12	7850	spotifypecde	googleyoutube-music	bolteud	shopify-chargecom	mgpvinted
3	7365	spotifypfdd	googletempo-raryhold	bolteuo	lingguiqi	mgpvinted
1	7359	spotifypeefafb	googletempo-raryhold	bolteuo	birdapp	mgpvinted
9	6428	spotifypfbac	googletempo-raryhold	bolteuovan	fbpaypayment	mgpvinted
6	6371	hbomaxbir	googletempo-raryhold	bolteuovan	googletempo-raryhold	mgpvinted
5	6323					
4	6075					
13	4746					
8	4705					
15	4651					
14	4470					
10	4398					
0	4049					
18	4036					
19	3304					
37	3264					
7	3203					
20	3187					

Appendix 28 - Word2Vec & DBSCAN Clustering Results (Name Embeddings) – eps=1, min_samples=20

Number of Clusters:			Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5
87			netflixcom	zaracom	ubereatspend- ing	applecombill	amazonesama- zones
			netflixcom	zaracom	ubereatspend- ing	applecombill	amazonesama- zones
Cluster Sizes:			netflixcom	zaracom	ubereatspend- ing	applecombill	amazonesama- zones
-1	151877		netflixcom	zaracom	ubereatspend- ing	applecombill	amazonesama- zones
0	8096		netflixcom	zaracom	ubereatspend- ing	applecombill	amazonesama- zones
3	6258		googletempo- raryhold	sheincom	ubereatspend- ing	applecombill	itunesappstap
13	6093		googletempo- raryhold	sheincom	ubereatspend- ing	applecombill	itunesappstap
2	5121		googletempo- raryhold	sheincom	ubereatspend- ing	applecombill	itunesappstap
9	4895		googletempo- raryhold	sheincom	ubereatspend- ing	applecombill	itunesappstap
1	4098		googletempo- raryhold	sheincom	ubereatspend- ing	applecombill	itunesappstap
14	3946						
16	3794						
12	3433						
6	3377						
4	3232						
8	3191						
21	3178						
7	3018						
10	3016						
18	2819						
5	2604						
23	2353						
20	2339						

Appendix 29 - Word2Vec & DBSCAN Clustering Results (Combined Features with Scaled Word2Vec Name Embeddings by a Factor of 2) – eps=0.1, min_samples=20

Number of Clusters:		Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5
89		cpcomboiospor-tugal	bolteuo	alibabacom	netflixcom	bolteuo
		cpcomboiospor-tugal	bolteuo	aliexpress	netflixcom	bolteud
Cluster Sizes:		transpinter-modaisporto	bolteuo	ebayo	netflixcom	bolteuo
2	11364	cpcomboiospor-tugal	bolteuo	aliexpress	netflixcom	bolteur
3	10981	wptredeexpres-sos	bolteuo	aliexpress	netflixcom	bolteuo
-1	10180	too-goodtogaquntlg-vr	googleyoutube-music	bolteud	techsolutionscy	ubereatspend-ing
8	9997	spotifypeefafb	googletempo-raryhold	bolteuo	blinkist	ubereatspend-ing
1	9084	spotifypfbac	googletempo-raryhold	bolteuo	mgpvinted	ubereatspend-ing
10	9081	hbomaxbir	googletempo-raryhold	bolteuovan	flowspectacles	ubereatspend-ing
4	8709	toogoodtogguz-twaryy	googletempo-raryhold	bolteuovan	googletempo-raryhold	ubereatspend-ing
17	8187					
15	8026					
5	7996					
11	7671					
7	6755					
16	6449					
0	6358					
12	6129					
14	5570					
22	5538					
20	5468					
27	5285					
6	5115					

Number of Clusters:		Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5
87		cpcomboiospor-tugal	bolteuo	alibabacom	netflixcom	bolteuo
		cpcomboiospor-tugal	bolteuo	aliexpress	netflixcom	bolteud
Cluster Sizes:		transpinter-modaisporto	bolteuo	ebayo	netflixcom	bolteuo
2	11889	cpcomboiospor-tugal	bolteuo	aliexpress	netflixcom	bolteur
3	10981	wptredeex-pressos	bolteuo	aliexpress	netflixcom	bolteuo
-1	9955	too-goodtogaqiuntlg-vr	googletempo-raryhold	bolteud	techsolutionscy	ubereatspend-ing
4	9775	spotifypeefafb	googletempo-raryhold	bolteuo	blinkist	ubereatspend-ing
1	9655	spotifypfbac	googletempo-raryhold	bolteuo	mjpgvinted	ubereatspend-ing
8	8928	hbomaxbir	googletempo-raryhold	bolteuovan	flowspectacles	ubereatspend-ing
11	8280	toogoodtogguz-twaryy	googletempo-raryhold	bolteuovan	googletempo-raryhold	ubereatspend-ing
7	8244					
17	8216					
10	7550					
5	7530					
6	7167					
15	6854					
12	6649					
16	6602					
0	6192					
14	6115					

Number of Clusters:		Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5
87		cpcomboiospor-tugal	bolteuo	alibabacom	netflixcom	bolteuo
		cpcomboiospor-tugal	bolteuo	aliexpress	netflixcom	bolteud
Cluster Sizes:		transpinter-modaisporto	bolteuo	ebayo	netflixcom	bolteuo
2	11889	cpcomboiospor-tugal	bolteuo	aliexpress	netflixcom	bolteur
3	10981	wptredeex-pressos	bolteuo	aliexpress	netflixcom	bolteuo
-1	9955	too-goodtogaqiuntlg-vr	googletempo-raryhold	bolteud	techsolutionscy	ubereatspend-ing
4	9775	spotifypeefafb	googletempo-raryhold	bolteuo	blinkist	ubereatspend-ing
1	9655	spotifypfbac	googletempo-raryhold	bolteuo	mjpgvinted	ubereatspend-ing
8	8928	hbomaxbir	googletempo-raryhold	bolteuovan	flowspectacles	ubereatspend-ing
11	8280	toogoodtogguz-twaryy	googletempo-raryhold	bolteuovan	googletempo-raryhold	ubereatspend-ing
7	8244					
17	8216					
10	7550					
5	7530					
6	7167					
15	6854					
12	6649					
16	6602					
0	6192					
14	6115					

26	5819						
9	5812						
22	5523						

Appendix 31 – GloVe & DBSCAN Clustering Results (Name Embeddings) – eps=0.5, min_samples=20

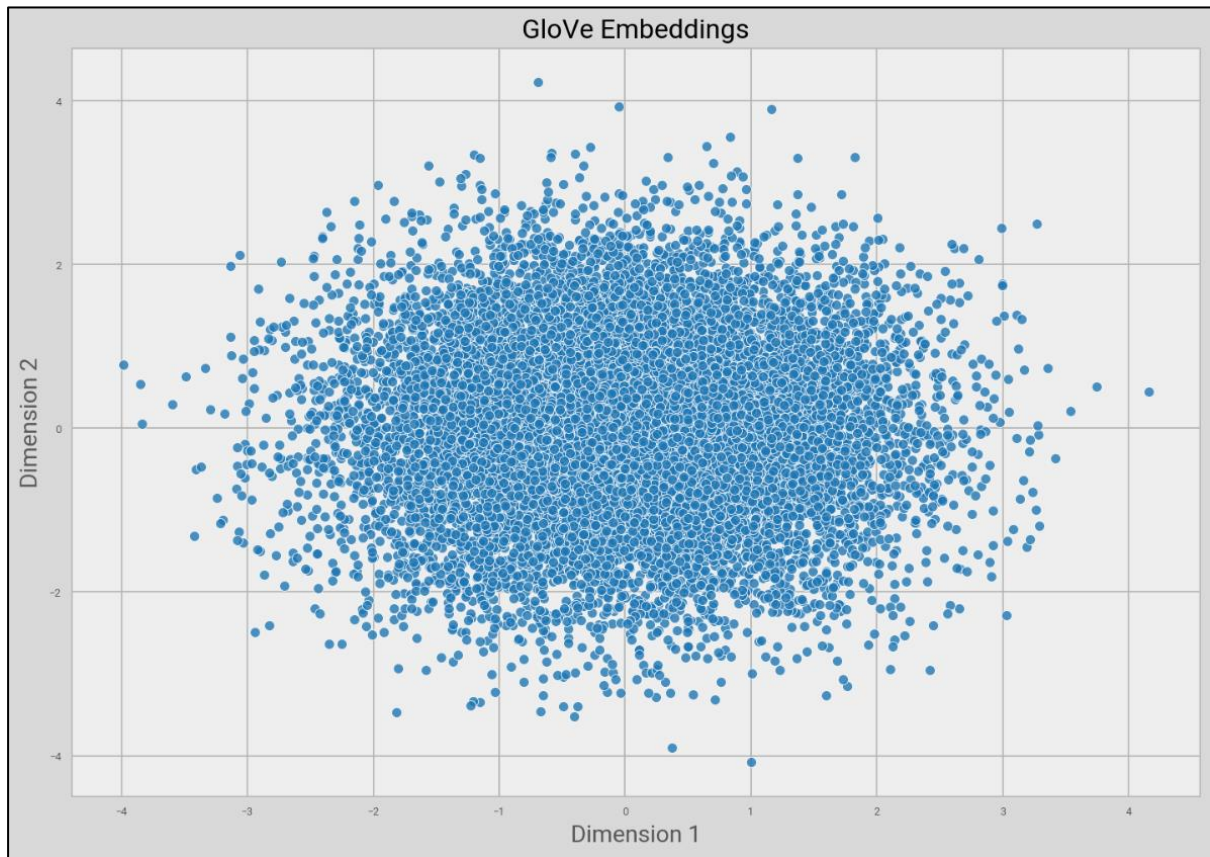
Number of Clusters:		Cluster 0	Cluster 1	Cluster 2	Cluster 3	Cluster 4
1		ticketweezevent				
		glovooctovhlnkr				
Cluster Sizes:		alipayeur				
0	13668	glovooc-toptpqqyv				
		jechura-campana				
		farmacianicolau				
		glovonovseullpk				
		glovonovop-bwchms				
		sunsama				
		flirtmecom				

Appendix 32 – GloVe & DBSCAN Clustering Results (Name Embeddings) – eps=1, min_samples=3

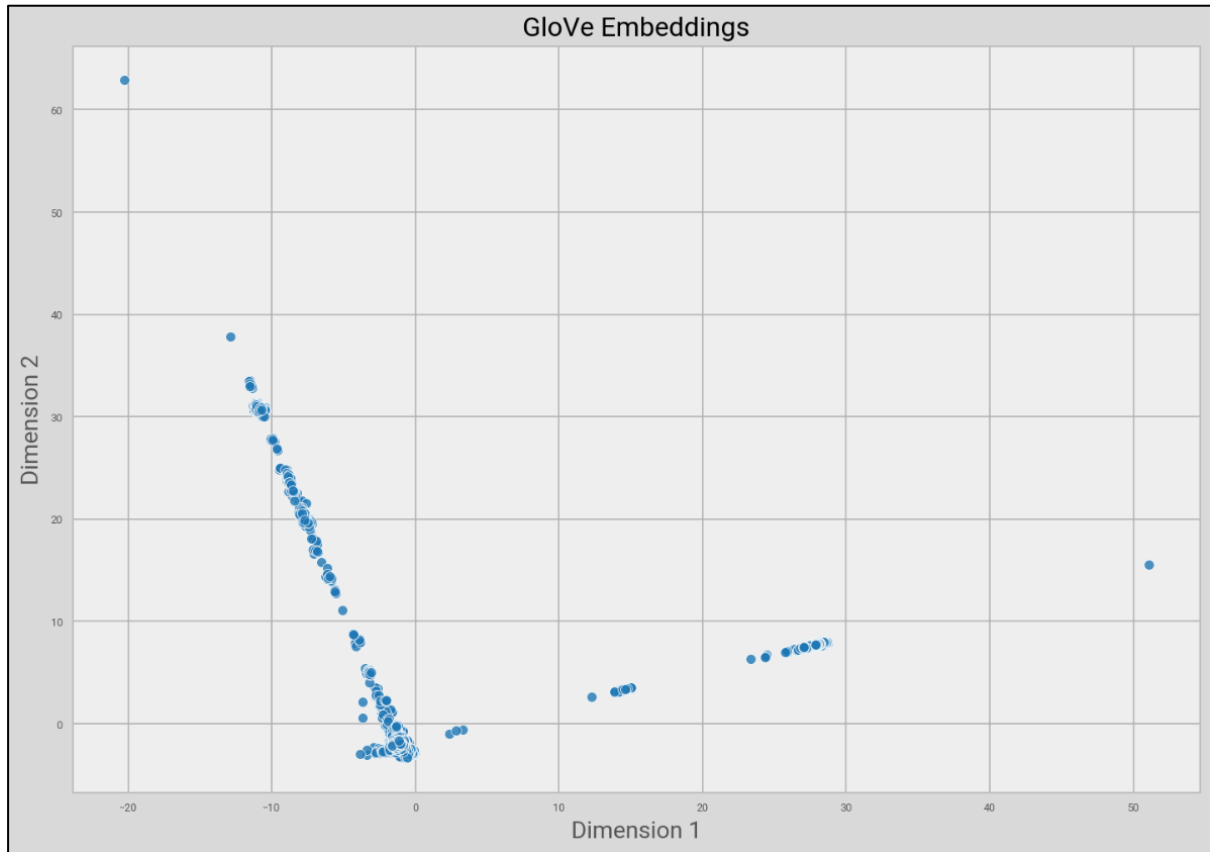
Number of Clusters:		Cluster 0	Cluster 1	Cluster 2	Cluster 3	Cluster 4
1		ticketweezevent				

[illegible]

Appendix 34 – Visualization of GloVe Merchant Name Embeddings



Appendix 35 – Visualization of GloVe Combined Features Embeddings



Appendix 36 – GloVe & DBSCAN Clustering Results (Combined Features) – $\text{eps}=0.5$, $\text{min_samples}=5$

Number of Clusters:		Cluster 0	Cluster 1	Cluster 2	Cluster 3	Cluster 4
3		airbnbhmnmkx	bolteuo			
		airbnbhmnrkdzz	bolteuo			
Cluster Sizes:		airbnbhmrschw qz	bolteuo			
-1	16065	airbnbhmxfcapf	bolteuo			
1	15	airbnb- mybekwpk	bolteuo			
0	5		bolteuo			
			bolteuo			
			bolteuo			
			bolteuo			

				bolteuo			

Appendix 37 – GloVe & DBSCAN Clustering Results (Combined Features) – eps=1, min_samples=5

Number of Clusters:		Cluster 0	Cluster 1	Cluster 2	Cluster 3	Cluster 4
98		cipampalfonos-beor	airbnbhmprdnj	airbnbhmkzr-pwh	alladoba	
		centerribadaviai	airbnbhmcxpzx	airbnbhjndran	alditiarguelles	
Cluster Sizes:		lidlalmoradi	airbnbhmjhttrq	airbnbhmanydchf	carrefaznar-farche	
-1	12993	lidl bilbaoboluea	airbnbhmxdsxeb	airbnbhmasftdqf	carrefrescantos	
35	470	lidlvcavpuerto	airbnbhmnmtkx	airbnbhmdaysmqf	juandiegogonzalezlechu	
36	437	spartcambrilscolo	airbnbhmthxyc	airbnbhmfzhmta	qzbernabeu	
90	220	supermercadosanpablo	airbnbhmjayhdd	airbnbhmfkhkfcq		
89	212	supermercados trevias	airbnbhmafmyhkc	airbnbhmnjprdj		
47	198		airbnbhmayyzd	airbnbhmpyykkm		
48	174		airbnbhmacxhcx	airbnbhmrstzst		
10	148					
25	78					
32	70					
12	47					
37	47					
64	43					
7	38					
72	33					
94	32					
1	29					

24	27						
54	27						
40	26						

Appendix 38 – Word2Vec with Filtering Logic – Results

Word2Vec	
Merchant Name	Cleaned Merchant Name
cpcomboiosportugal	small merchant
bolteuo	bolt
alibabacom	small merchant
netflixcom	netflix
bolteuo	bolt
blizzardentertainmentinc	small merchant
zaracom	small merchant
ubereatspending	uber
hertzportugal	small merchant
ubereatspending	uber
tripcom	booking
revolut	revolut
googleads	facebook
applecombill	facebook
bookingcom	booking
amazonesamazones	amazon
revolut	revolut
bolteud	bolt
paycasinoestoril	small merchant
bolteuo	bolt
tradeinnret	small merchant
applecombill	google
ubrpendingubercom	booking

Appendix 39 – GloVe with Filtering Logic – Results

GloVe	
Merchant Name	Cleaned Merchant Name
cpcomboiosportugal	small merchant
bolteuo	bolt
alibabacom	small merchant
netflixcom	netflix
bolteuo	bolt
blizzardentertainmentinc	small merchant
zaracom	small merchant
ubereatspending	booking
hertzportugal	small merchant
ubereatspending	booking
tripcom	booking
revolut	revolut
googleads	meta
applecombill	google
bookingcom	booking
amazonesamazones	amazon
revolut	revolut
bolteud	bolt
paycasinoestoril	small merchant
amazonprimebr	netflix
ubrpendingubercom	booking

Appendix 40 – GloVe Embeddings of Mapped Merchants for “amazonprimebr”

Embeddings of chosen Merchant Name	
Merchant Name	GloVe Embeddings
amazonprimebr	[0.0018, 0.009, 0.0156, -0.001, -0.0026, 0.0105, 0.0021, -0.0269, 0.0207, -0.0309, 0.0361, -0.0021, 0.0127, -0.0092, -0.0012, -0.0322, -0.0097, -0.0018, 0.0009, -0.0172, 0.0094, -0.006, 0.0065, 0.0248, -0.0077, -0.0008, 0.0073, -0.0216, 0.0036, -0.024, -0.011, -0.0146, 0.0111, -0.0058, -0.0009, 0.0189, 0.0119, 0.0055, 0.0116, -0.0105, -0.0277, 0.0059, 0.0033, 0.0407, -0.0008, -0.0135, -0.0224, -0.0116, 0.0162, -0.0037, -0.0044, -0.0014, -0.0034, -0.0148, 0.007, 0.0206, 0.0044, 0.0225, -0.0017, -0.0049, 0.0059, 0.0086, 0.0011, 0.0121, 0.020, 0.0043, -0.0114, -0.0285, 0.0017, -0.0011, 0.0066, -0.011, 0.0276, 0.0088, -0.0318, -0.0016, -0.0143, -0.0065, 0.0018, -0.0073,

	0.0076, 0.0199, -0.0251, -0.0013, -0.0215, 0.0172, -0.0017, 0.0096, 0.0095, -0.0207, 0.0115, 0.0013, 0.0105, 0.0102, -0.0002, 0.0183, -0.0151, -0.0063, 0.0251, -0.0216]	
Embeddings of Mapped Merchants		
Mapped Merchants	Cosine Similarity	GloVe Embeddings
netflix	0.0168	[-0.977231 0.018826 0.183785 1.476241 -0.282574 -1.021442 0.30329 -0.102016 0.228777 0.892349 0.138403 -0.018828 -0.363061 0.700926 -0.123419 -0.773919 0.329876 -0.57854 -0.178922 -0.177938 0.038803 0.395301 -0.011484 -0.678183 1.113502 0.854679 0.439204 0.334285 -0.75911 1.172213 0.269956 0.62797 0.037629 -1.034676 0.756428 -0.75611 -0.287206 -0.406188 0.095439 0.666651 0.363311 -0.71501 -0.771697 0.560297 0.266413 -1.011022 0.67634 -0.546475 -0.802144 0.319818 -0.605276 0.474562 -0.050406 0.436049 -0.835619 0.302417 -0.341037 0.08222 -0.215993 -0.391721 -0.34176 -0.535434 -0.096211 -0.785738 -1.594155 0.93796 0.252921 0.302149 -0.615484 0.654771 0.632255 -0.351613 -0.307643 1.723995 0.950621 0.148205 -0.139542 0.078833 -0.223434 -0.314806 -0.36984 -0.726733 0.310571 -0.67495 -0.019242 -0.057904 0.789759 0.025094 0.291195 -0.965959 -0.715629 0.326553 1.051728 0.80391 -0.365815 -0.36616 1.037483 -0.204284 -0.839829 0.107156]
google	0.0093	[-7.500e-04 1.657e-03 -6.488e-03 -4.056e-03 1.996e-03 2.226e-03 -4.183e-03 -9.143e-03 -6.068e-03 3.944e-03 -1.687e-03 1.205e-03 -1.777e-03 -1.438e-03 -3.183e-03 -1.161e-03 3.958e-03 7.540e-04 5.720e-04 -4.663e-03 -5.650e-04 -2.886e-03 -1.502e-03 6.722e-03 -1.578e-03 5.422e-03 2.936e-03 -3.019e-03 2.239e-03 1.012e-03 4.180e-03 1.490e-03 2.669e-03 7.692e-03 -2.566e-03 -5.336e-03 -8.200e-05 3.250e-03 -4.479e-03 -6.151e-03 -2.806e-03 -6.165e-03 -4.946e-03 5.417e-03 2.397e-03 1.871e-03 4.257e-03 6.355e-03 2.625e-03 4.828e-03 1.692e-03 2.060e-03 1.943e-03 1.910e-04 -1.218e-03 3.640e-04 -4.388e-03 1.718e-03 -2.655e-03 -2.149e-03]

		2.730e-03 1.525e-03 -6.590e-04 -4.602e-03 -7.830e-04 -3.225e-03 6.200e-05 -8.660e-04 2.500e-05 -4.417e-03 -7.016e-03 -2.781e-03 -5.820e-04 -1.962e-03 2.636e-03 1.815e-03 -9.100e-05 -3.107e-03 -1.830e-03 -7.466e-03 -8.279e-03 -1.370e-04 -5.406e-03 -6.336e-03 5.300e-05 3.376e-03 4.028e-03 -4.334e-03 -4.906e-03 1.374e-03 3.517e-03 -2.177e-03 -7.101e-03 2.858e-03 3.221e-03 2.116e-03 -3.670e-04 3.283e-03 1.250e-03 -3.430e-04]
amazon	-0.2338	[0.676704 0.179892 -0.581564 -1.291859 1.21451 0.914511 -0.017124 -0.020467 0.125982 -1.178556 1.030682 1.714268 1.490543 0.033365 -0.155588 -0.221323 -0.441022 -0.709255 0.247546 0.12088 -0.219666 -0.370279 -0.214432 -0.963824 -0.245806 1.324578 0.676391 -0.422245 0.200348 -0.490532 -1.390922 -0.591626 2.325126 0.920226 -0.296156 -0.730598 -0.262879 0.369742 -0.069898 -0.065564 -0.633603 0.410101 -0.382056 -0.115663 -0.82172 -0.05879 0.68978 0.772926 0.417367 1.096248 1.400883 -0.592858 -0.600666 -0.908381 1.083797 0.894654 -0.49422 -0.235571 1.137727 -0.127395 -1.532662 0.161803 1.697625 -0.174605 0.363386 0.211323 0.16157 1.32547 -0.348945 0.741471 -0.734474 -0.32637 -1.15185 -0.795748 -1.138422 -1.919566 0.987124 1.440816 0.908897 0.409879 -1.075788 0.820145 0.845127 0.780199 -0.487003 0.259983 -0.190179 -1.307441 0.416001 0.995603 1.511112 -0.927674 -1.48144 -1.739705 0.841321 0.360231 1.734274 -0.790827 -0.14855 0.729893]