Erasmus University Rotterdam
Erasmus School of Economics
Bachelor Thesis Econometrics and Operations Research

Visualizing Customer Representations using P2V-MAP

Name student: Fenna ten Haaf
Student ID number: 450812fh

Supervisor: Luuk van Maasakkers MSc
Second assessor: dr. Flavius Frasincar

Date final version: July 5, 2020

Abstract

Analyzing the structure of markets and relations between products and customers is key to assisting managers when deciding about promotion strategies, pricing, and many other objectives. *product2vec-map* (P2V-MAP) is an effective method to represent products in a two-dimensional space by training embeddings and subsequently using *Barnes–Hut t-distributed stochastic neighbour embedding* (t-SNE) dimensionality reduction to plot a map. In this thesis, we replicate the methodology of P2V-MAP and show that it is effective at representing product relationships by applying it to simulated data and an empirical dataset. Additionally, we extend the method by pooling product vectors into customer vectors with various pooling methods and representing these customer vectors in a map. We show that these aggregation methods are effective at creating customer maps for a simulated dataset without needing to train new customer embeddings.

Contents

1 Introduction

The introduction of loyalty cards to major grocery retailers in the 1990s played a big role in what some authors have labelled a 'revolution' in marketing information (Blattberg, Glazer, & Little, 1994). Since then, retailers have had access to incredible volumes of high-quality data about not only customer characteristics but also the make-up of their purchases and all features and promotions that were in effect at the time of purchase (Bradlow, Gangwar, Kopalle, & Voleti, 2017). This wide availability of data has led to huge opportunities for the field of market structure analysis, which searches for quantitative frameworks that show the arrangements and interrelationships of various parts of a market (Myers & Tauber, 2011).

The primary point of interest in market structure analysis is the relationship between products. Products that are frequently bought together are described as *complementary goods*. When two products are completely replaceable in the eyes of the consumer, they are described as *substitute goods* (O'Sullivan & Sheffrin, 2003). Elrod et al. (2002) argue that understanding these product relationships is highly important to marketing research. Retailers can use the information for various applications, from guiding their pricing decisions to deciding which products to offer in each category (Urban, Johnson, & Hauser, 1984; Qian, Jiang, Du, Sun, & Liu, 2019). In addition, analysing marketing structures is closely related to market segmenting, which involves the grouping of customers into nonoverlapping segments (Wedel & Kamakura, 2012). Knowing more about relationships between products can provide information about the groups of customers that buy those products, helping managers to effectively customize their product offerings, promotions, and recommendations to the preferences of those groups (Smith, 1956).

One limitation of some market structure analyses is that they frequently focus on only a narrow category of products and they do not model product complementarity across categories (Elrod et al., 2002). Gabel, Guhl, and Klapper (2019) present a new approach to analyze market structures, described as *product2vec-map* (P2V-MAP). In this approach, they first extend the Skip-Gram model presented by Mikolov, Sutskever, Chen, Corrado, and Dean (2013) to model co-occurrences between products based on market basket data. Next, a two-dimensional map of the products is created using the *Barnes–Hut t-distributed stochastic neighbour embedding* (t-SNE) dimensionality reduction algorithm (Van der Maaten & Hinton, 2008). A key benefit of P2V-MAP is that the authors of (Gabel et al., 2019) actually consider a retailer's whole product assortment. In addition to providing a scalable and complete analysis, they also improve the visualization method compared to earlier work. Whereas other methods of market structure analysis tend to produce maps with results that are lumped together (Netzer, Feldman, Goldenberg, & Fresko, 2012), making it more difficult to interpret, P2V-MAP shows well-separated product clusters. This is very relevant for practical applications, as visualizations are often key in supporting management decisions.

In the same way that P2V-MAP helps managers by providing insight in the relationships between products, it can also be valuable to have insight into the groups of customers who buy those products. One

way to do this is to create *embeddings* (vector representations) of customers directly (Jagabathula, Subramanian, & Venkataraman, 2018), much like how P2V-MAP creates embeddings of products and uses those as a base for representing the products in a two-dimensional space. However, Baldassini and Serrano (2018) show that it is also possible to create customer embeddings from existing product embeddings by pooling the embeddings together. This could be an efficient way to gain extra information from market basket data for existing product-embedding frameworks like P2V-MAP.

In this thesis, we replicate the P2V-MAP methodology presented by Gabel et al. (2019) and we validate its effectiveness by applying it to simulated data. We also apply P2V-MAP to a large, real-world dataset containing customers' orders (Instacart, 2017). In addition, we aim to expand the P2V-MAP method by using pooling methods to get customer representations from the product representations. The main focus of this thesis is twofold:

Can we replicate the P2V-MAP methods and, subsequently, use the output to visualize customer representations?

This thesis has the following structure: in Sect. 2 we discuss related relevant literature that forms the background of our research problem. In Sect. 3, an overview of the empirical dataset is given. Moreover, we describe our methodology for the replication as well as our customer map extension in Sect. 4. Finally, we present our results in Sect. 5 and conclude in Sect. 6.

2 Related Works

Market structure analysis is an important subfield of *competitive intelligence* (CI), which aims to gather and analyze information to support executives and managers in strategic decision making (Bergeron & Hiller, 2002). The primary purpose of market structure analysis is to identify competitive submarkets, where a competitive submarket can be defined as a set of products judged to be substitutes (Srivastava, Alpert, & Shocker, 1984). Gabel et al. (2019) provide a scalable, comprehensive method to model and visualize products in relation to each other, referred to as *product2vec-map* (P2V-MAP). There are many different sources of data used for investigating market segmentation, including surveys, online customer reviews, and scanner-panel data (Qian et al., 2019). However, we focus on the analysis of market basket data, as done in the P2V-MAP method (Gabel et al., 2019). In this section, we go deeper into the background literature relevant for understanding the P2V-MAP model. In particular, we explain the specific machine learning methods used to create vector representations, or embeddings, of products in Sect. 2.1. In Sect. 2.2, we go into detail on how these vectors can be reduced to a two-dimensional representations for plotting on a product map. Lastly, we explain the relevant literature on pooling vector representations in Sect. 2.3, which we can use to create vector representations of customers out of product vectors.

2.1 Product Embeddings

The goal of P2V-MAP is to map products according to their co-occurrence patterns in a large number of shopping baskets. These co-occurrences are made by extending the Skip-Gram model presented by Mikolov, Chen, Corrado, and Dean (2013), which is commonly used in the Natural Language Processing (NLP) field for creating word embeddings. A word embedding is a method for mapping words to a single vector space. It creates unique vectors in a way that retains information about the word the vector represents, while at the same time having relatively low dimensionality when compared to the original dataset.

Some of the most well-known methods for word embeddings are proposed by Mikolov, Chen, et al. (2013), known as *local context window methods*. The methods are named this way because they primarily determine the vector of a word by looking at the neighbouring words in the sentence that the word appears in. There are two variations to this presented by Mikolov, Chen, et al. (2013). In the Continuous Bag Of Words (CBOW) model, the embedding is learned by predicting the current word based on its context (the surrounding words). The Skip-Gram model, on the other hand, learns by predicting the surrounding words given a current word. While there are some differences between the two methods, the performance is fairly similar in practical applications (Mikolov, Sutskever, et al., 2013).

Mikolov, Sutskever, et al. (2013) extend Skip-Gram by so-called *subsampling of frequent words* and *negative sampling*. The first extension reduces training time and improves the representation of uncommon words, whereas the second extension achieves a similar feat but for more frequently used words. It is shown that these local window context methods of CBOW and Skip-Gram outperform previous *global matrix factorization* methods like Latent Semantic Analysis (LSA) and Latent Dirichlet Allocation (LDA), which is a method that has also been used by some other authors in modelling product relations (Jacobs, Donkers, & Fok, 2016). Lastly, it is less computationally expensive compared to LDA (Mikolov, Sutskever, et al., 2013), which makes a method using word embeddings more scalable for large datasets.

As mentioned before, P2V-MAP adjusts the Skip-Gram model to obtain representations of products (as opposed to words) in latent attribute spaces. There are a few key differences when applying Skip-Gram to product basket data. The first difference is that there are some very frequent words (such as 'the') in NLP but there are generally no products that appear with such frequency relative to the remainder of products in a given market basket dataset. Therefore, there is no need to do any subsampling of frequent words as presented by Mikolov, Sutskever, et al. (2013). A second difference is that, in NLP, training samples are only constructed from terms within a context window w around a target. This makes sense for NLP, where words that are closer to each other are often related, but less so for the random order of products in shopping baskets. Therefore, Gabel et al. (2019) use all combinations of products with the remaining products in the shopping basket (context products), to maximize the information used. Lastly, P2V-MAP also applies negative sampling like Mikolov, Sutskever, et al. (2013) but the implementation suppresses training sample 'collisions'. For a given training sample, a product

pair can either be a positive or a negative sample but not both at the same time. A final modification is that the Skip-Gram score function is extended with a bias term that captures the base probability that a product-center pair occurs in the data. With these modifications, latent product vectors can be derived.

2.2 Product Mapping

As discussed earlier in this section, a clear and interpretable visualization of product relations is very important. Visualizations such as dashboards frequently play an important role in the decisions made by management (Wedel & Kannan, 2016). With P2V-MAP, all products in a retailer's assortment are visualized in a two-dimensional map according to their co-occurrence patterns (Gabel et al., 2019). This is achieved through *Barnes–Hut t-distributed stochastic neighbor embedding* (t-SNE) dimensionality reduction (Van der Maaten & Hinton, 2008), applied to the latent product vectors. Here, similarities between the L2-normalized product vectors are modeled by a normalized Gaussian kernel in the one-dimensional space and by a Student's t-distribution with one degree of freedom in the two-dimensional space. Since t-SNE evaluates the pairwise similarities of products, product proximities may be based on different characteristics (for instance, wines may form a cluster where proximity is mainly determined by price, while chocolate milk could have a cluster mainly determined on brand). Map overlays can be used to find out what the proximity of products in a cluster is based on.

Previous research often only focuses on a narrow set of products, which means only competitive relations can be mapped (Ringel & Skiera, 2016). When larger numbers of products are mapped, the visualizations tend to have dense clusters that are not easy to read. The heavy tails of the t-distribution used in t-SNE means that products with lower similarity are farther apart, allowing a better representation (Van der Maaten, 2014). Another advantage of the visualization aspect in P2V-MAP compared to earlier work is that it facilitates the analysis of product complementarity in addition to substitutability. It also does not require any assumptions in order to choose which products to model since it takes all products in an assortment into account. Furthermore, t-SNE has a low computational complexity of only $\mathcal{O}[n \log(n)]$ given n products, which makes it even better for larger datasets (Van der Maaten & Hinton, 2008).

2.3 Vector Pooling

There are various fields of machine learning that apply variations of vector pooling to obtain global representations from local ones. For example, in natural language processing, word embeddings can be pooled to represent phrases or sentences (Mitchell & Lapata, 2010). Pooling is also extensively used in image recognition algorithms to aggregate local image descriptors, preserving the information necessary to classify images while discarding unnecessary details (Jégou, Douze, Schmid, & Pérez, 2010).

In market analysis, the application of these techniques is less common. One application is presented by Baldassini and Serrano (2018), who propose a way to obtain client representations from banking

transaction data. One of the methods used to achieve this is by extracting vector representations of transactions by training on the Skip-Gram model described in Sect. 2.1 and subsequently pooling the resulting vectors. The transaction embeddings that Baldassini and Serrano (2018) pool are directly comparable to the product vector outputs with P2V-MAP, which we expect would also be able to represent customer information when pooled.

Some examples of simple methods to aggregate vectors that are used in the literature are max pooling (Jarrett, Kavukcuoglu, Ranzato, & LeCun, 2009) and average pooling (LeCun et al., 1990), which is simply taking the average of the individual vectors in order to aggregate them into a global feature. In some applications, like pooling word vectors to represent sentences, the disadvantage of these simple methods is that they do not preserve word order (Le & Mikolov, 2014). However, we believe that this is not an issue for the representation of customers from product embeddings, as there is no element of order in the market basket data that we use. Therefore, we opt to investigate the effectiveness of average pooling and max pooling to make our client vectors, as they are easy and efficient to apply.

The reason why we investigate pooling vectors into customer embeddings rather than training customer embeddings from purchasing data directly is that training embeddings can be time-intensive. By pooling the product vectors which are already trained, the method becomes more efficient and scalable for creating both product and customer maps.

3 Data

A benefit of P2V-MAP is that the only inputs needed to create product embeddings are market basket IDs and the IDs of the products that belong to those baskets. This means that we can easily simulate data to test the effectiveness of the method and we can use data that does not have any information about product characteristics. Using simulated data does have the disadvantage that it is harder to evaluate the effectiveness of P2V-MAP for gaining insight into the products, since the product IDs of simulated data do not necessarily represent a real-life situation. Therefore, we also attempt to visualize real-world data in a product map. In this section, we go into more detail for the empirical data that we use for this thesis.

We apply the P2V-MAP framework to the Instacart Online Grocery Shopping Dataset (Instacart, 2017). This dataset contains over 3 million orders from more than 200,000 customers. Every order is a market basket with information of the purchased products, the ID of the customer, the day and time at which the order is placed, whether the products have been ordered before, and a few other identifying variables such as aisle ID and department ID. Each product belongs to one of 134 aisles and multiple aisles together belong to one of 21 departments. For example, we have a product *whole milk* which belongs to the aisle *milk* and the department *dairy, eggs*.

In total, there are 49,677 unique products in the Instacart dataset. Training embeddings for each of

these products would be very time-consuming. Therefore, we take a subselection of 50 aisles, randomly selected from the full set of aisles (without the aisles 'missing' and 'other household'). For each of these aisles, we select the 10 products that appear in the most shopping baskets to make sure that we have enough training data for each product. This means that we end up with basket information for 500 products. Figure 1 gives an indication of how often the products appear in the dataset and how big the disparities are, as well as which product categories (aisles) we selected.

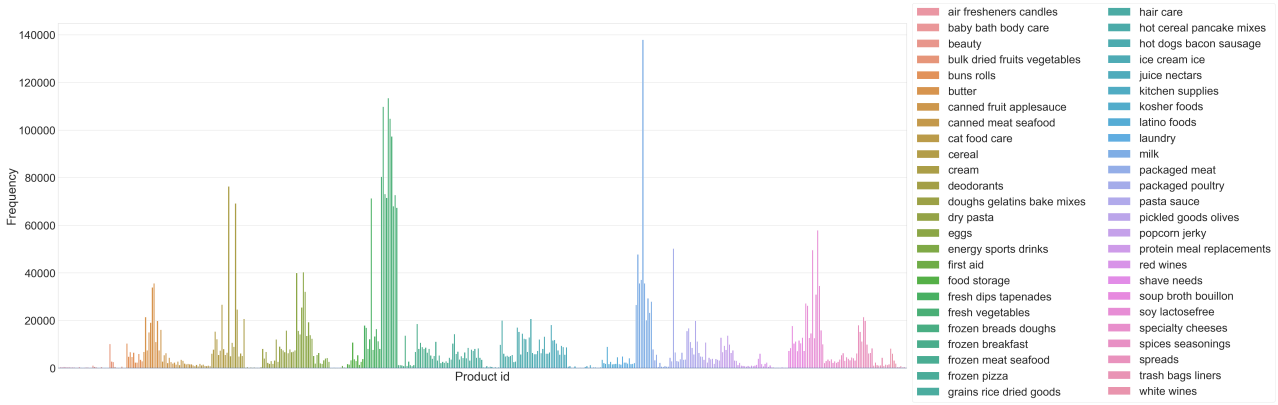


Figure 1. Frequency of products in the subsampled Instacart dataset

4 Methodology

In the following sections, we explain the methodology used to create and evaluate results for this thesis. First, we will explain the methods used to replicate the P2V-MAP implementation as proposed by Gabel (2019) in Sect. 4.1. We also explain in this section how we create customer maps out of the embeddings trained with P2V-MAP. Next, we explain in Sect. 4.2 how we replicate the methodology for creating simulated data which we use to create a product map with P2V-MAP. Additionally, we explain in this section how we extend the simulation to have distinct customer groups with different characteristics. Finally, the metrics by which we evaluate the product and customer maps are given in Sect. 4.3.

4.1 P2V-MAP

In this section, we describe the steps we take to replicate the P2V-MAP methodology as described by Gabel et al. (2019). There are three parts to this process: firstly, we need to do the data preparation, which is explained in Sect. 4.1.1. Then, we derive the latent product attributes by applying a modified version of Skip-Gram (Mikolov, Sutskever, et al., 2013), as explained in Sect 4.1.2. Finally, we process the resulting vectors such that they can be presented in a two-dimensional product map. The methodology for this is explained in Sect. 4.1.3. In Sect. 4.1.4, we explain how we then extend this three-step process by aggregating the products derived in step 2 to get customer maps.

4.1.1 Data Preparation

The first step in the P2V-MAP implementation is processing market basket data for use with the Skip-Gram model. For input data, only the basket IDs b and the set of n_b product IDs j that belong to the basket are required. Products that appear in fewer than n_{min} baskets are removed from the training set since very infrequent products only add noise. A common setting for this value in literature using word embeddings is 5 as this is a default value in some of the coding implementations of Skip-Gram (Rehurek & Sojka, 2010). However, there is no literature showing that any particular value is better to use than others. We choose instead to set n_{min} to 20, because it ensures for higher quality vectors while still not removing too many products from the dataset.

Training samples for the Skip-Gram model are constructed by creating combinations of each product in the basket with all the remaining products such that we have $n_b(n_b - 1)$ center-context pairs for each shopping basket b. As explained in Sect. 2.1, we make combinations with all of the other products in the basket (as opposed to just a selection of products) because the order of products in the basket is random. This means that the location of the product in the basket is not meaningful in the way that the location of words in a sentence is. Therefore, it is better to use all of the products because it gives us more training data.

The last step is to create negative samples, which are pairs that are not observed together in the shopping basket. For each positive training example, we sample n_{neg} products from the total product assortment J, using a probability distribution with density $P_n(j) = \frac{1}{Z} n_j^{pow}$ as sampling weights. Here, n_j is the number of times product j occurs in training data, Z is a normalization constant, and *pow* is a constant that allows for modifying the shape of the sampling distribution. The reason we take these negative samples is that we want to make sure that, in the end, the vectors of products that occur in similar contexts (baskets) are more akin than products that do not occur in the same contexts. Therefore, if we have a center product vector v_j, a context product vector $v_{c(i)}$ from the same basket, and context vectors $v_c(k)$ of products from different baskets, we want to maximize the ratio

$$\frac{v_j \cdot v_{c(i)}}{\sum_{k \neq i} (v_j \cdot v_{c(k)})}. \tag{4.1}$$

The numerator is the similarity between products i (the context) and j (the target), while the denominator computes the similarity of all other contexts k and the target product. Maximizing (4.1) ensures that products that appear together in baskets have more similar vectors than products that do not. However, computing this can be very slow because there are many context products k. Negative sampling is one of the ways of addressing this problem, as it selects some context products k at random. The value we use for n_{neg} is 20. The value for *pow* is commonly set to 0.75 (Mikolov, Sutskever, et al., 2013). Note that, for a given training sample, a product pair can be either a positive or a negative sample, but not both at the same time.

4.1.2 Modified Skip-Gram

Now that we have created combinations of center products with positive and negative samples, we use these as inputs to train embeddings for each of our center products using a modified implementation of Skip-Gram (Mikolov, Sutskever, et al., 2013). We implement our methods with the TensorFlow library in Python (Abadi et al., 2015).

The key idea behind the training of product embeddings with Skip-Gram is that products which appear together in mostly the same contexts (baskets) should have embedding vectors in the end that have a high similarity score. The loss function for Skip-Gram for a particular center product ce is defined as

$$\text{Loss}_{ce} = -\log \sigma(v_{ce} \cdot w_{co}) - \sum_{k \in n_{neg}} \log \sigma(-v_{ce} \cdot w_k), \tag{4.2}$$

where $\sigma(x) = (1 + e^{-x})^{-1}$ is the standard logistic function. This loss function can be explained by describing the two terms of its sum. The first term measures the loss for a center product occurring together with a positive sample context product. Here, v_{ce} is the vector for a particular center product and w_{co} the vector for a context product, while the dot product between the two $v_{ce} \cdot w_{co}$ gives the similarity score for them. The loss will, therefore, be lower if the vectors are more similar. The second part measures the loss for a center product *not* occurring together with a set of negative sample products. Due to the minus sign before the similarity score, the loss will be lower if the similarity between a center product and its negative samples is lower. Therefore, minimizing loss will mean that our final vectors v and w should be similar for products that appear in the same contexts together and less similar for products that do not often appear together.

One modification that Gabel et al. (2019) make to (4.2) is that the similarity score function $v_{ce} \cdot w_{co}$ is changed by adding a bias term β_0:

$$\text{similarity}_{ce,co} = v_{ce} \cdot w_{co} + \beta_0. \tag{4.3}$$

This bias term represents the base probability that a product-center pair occurs in the data. The result is that product vectors are scaled and centered, on average, around zero (Gabel et al., 2019).

Note that each product i gets two representations in Skip-Gram, v_i and w_i (also known as the input vector and the output vector, respectively), where v_i is has shape $1 \times L$ and w_i has shape $L \times 1$, such that the similarity between them is a scalar. The $1 \times L$ input vectors v are what we use in the end to plot our product map because the vector products for two items i and j have been shown to be higher for more similar products when calculating $v_i \cdot v_j$, whereas $v_i \cdot w_j$ is higher if i and j co-occur together a lot in the data (Mitra, Nalisnick, Craswell, & Caruana, 2016). Since we are more interested in representing similarity on our map, we use v as input.

In our application, we set the embedding dimension L to 30. The embeddings are trained by iterating through the data and calculating the loss function for each triple of center products, positive sample products, and negative sample products. The parameters are updated by stochastic gradient descent, which means that each parameter is changed by an amount proportional to the gradient of the loss for a set of randomly chosen samples (Bottou, 2010).

4.1.3 Product Mapping

As mentioned in the previous section, we use the input embeddings v that are trained with the adjusted Skip-Gram model to map products. We use the t-SNE dimensionality reduction presented by Van der Maaten and Hinton (2008) to reduce the embeddings from an L-dimensional to a two-dimensional space. t-SNE creates a probability distribution using the Gaussian distribution that defines the relationships between the points in high-dimensional space. The Student's t-distribution is used to recreate the probability distribution in low-dimensional space. Due to the large tails of the distribution, it is less likely that points get crowded together in low-dimensional space, which is a large benefit of t-SNE.

Before entering the vectors into the t-SNE model, we first perform L2-normalization on them. For larger embedding dimensions, it is recommended to apply principal component analysis (PCA) to reduce the dimensionality to 50 or lower (Van der Maaten & Hinton, 2008). This is because it speeds up the computation of pairwise distances between the data points and suppresses some noise without distorting the interpoint distances too much. However, in our case we set L to 30, which is already lower than 50, so it is not necessary to apply PCA.

4.1.4 Pooling Product Vectors

In order to create customer embeddings c of dimension $1 \times L$, we aim to aggregate the n $1 \times L$ center product vectors v belonging to the products that they buy. Once we have our customer vectors, we can plot it on a two-dimensional map using t-SNE dimensionality reduction (Van der Maaten & Hinton, 2008), just as described in Sect. 4.1.3.

Baldassini and Serrano (2018) use *average pooling* to get client embeddings from transactions, which involves simply taking the average of the vectors representing the products that each customer bought. We also apply average pooling, as it is simple and efficient to implement. In addition, we apply max pooling and min pooling for comparison which take the maximum or the minimum instead of the average. Lastly, we try our own method of pooling which is to take the average of the product vectors for only the top x most frequently bought products. The idea behind this is that the purchases that customers buy recurrently might be more important for describing the characteristics of the customers compared to purchases that they make only incidentally. Additionally, grouping customers based on these most important purchases may be more informative for managers, as the most frequent purchases are likely more relevant for devising promotion strategies. On the other hand, the average pooling method does already have an element of representing the most frequently purchased products more, since the products

that customers buy more often also have more weight when taking the average over all products.

4.2 Simulation

The first method used to validate and test P2V-MAP is by doing a simulation study. If we apply P2V-MAP to market basket data with a known market structure, it can then be compared with the results of the simulated market structure. The decision to purchase products is viewed as a sequential process, where the customer first chooses their consideration set or product category and then they choose the product (Neslin & Van Heerde, 2009). The model that we use is that consumer i chooses product j from category c in week t. Each category consists of $J^{(c)}$ products, and a product belongs to exactly one category. In this application, we use $c = 20$ categories and $J^{(c)} = 15$ products per category. We will first explain how we draw the categories that consumers purchase in a given week. After this, we elaborate on how the products that consumers select within those categories are sampled. Lastly, we explain a slight variation of the simulation process that we execute in order to model data where customers belong to distinct categories.

4.2.1 Category Choice

The category purchase incidence is modelled with the multivariate probit (MVP) model (Manchanda, Ansari, & Gupta, 1999), which allows for more than one category to be purchased simultaneously. We model a latent variable z_{ict}, defined as follows:

$$z_{ict} = \gamma_c + \epsilon_{ict}, \tag{4.4}$$

where γ_c is base utility and $\epsilon_{ict} \sim \text{MVN}(0, \Omega)$ is an uncorrelated error term. In our implementation, γ_c is -0.5 for all categories. Ω is a correlation matrix that captures (random) purchase incidence correlation across categories. It is manually specified by Gabel et al. (2019) with positive and negative correlations in varying magnitudes in order to capture a variety of category relationships. This MVP correlation matrix is given in Appendix A.1. After deriving the latent variable z_{ict}, the category purchase incidence is:

$$y_{ict} = 1 \text{ if } z_{ict} > 0,\ y_{ict} = 0 \text{ otherwise.} \tag{4.5}$$

where $y_{ict} \in \{0, 1\}$ describes the choice of customer i to buy from a certain category c in week t.

4.2.2 Product Choice

To model (discrete) product choice within each category, the multinomial probit (MNP) model (Chintagunta, 1992) is used. The utility of product j for consumer i in week t is given by

$$u_{ijt}^{(c)} = \alpha_{ij}^{(c)} - \beta^{(c)} p_j^{(c)} + \epsilon_{ijt}^{(c)}, \tag{4.6}$$

where $\alpha_{ij}^{(c)}$ is a base utility, $p_j^{(c)}$ is a price, $\beta^{(c)}$ is the price sensitivity, and $\epsilon_{ijt}^{(c)} \sim \text{N}(0, \sigma^{(c)})$ is a random error term. In our application, $\beta^{(c)}$ is 2 and $\sigma^{(c)}$ is 1 for every category, which are the same parameters

as used by Gabel et al. (2019). The product prices, $p_j^{(c)}$, are derived in two steps. First, the regular price of products in a category, $p^{(c)}$, is sampled from a log-normal distribution with a mean of 0.5 and a standard deviation of 0.3. Next, product prices $p_j^{(c)}$ are sampled from a U $\left[\frac{p^{(c)}}{2}, 2p^{(c)}\right]$ distribution. This way, the product prices stay within a certain price range based on the category prices, similar to empirical data. The base utility $\alpha_{ij}^{(c)}$ is drawn from a MVN $\left(0, \Sigma^{(c)}\right)$ distribution. Here, the category-specific MNP covariance matrix $\Sigma^{(c)}$ is computed as follows:

$$\Sigma^{(c)} = \left(\tau^{(c)} I^{(c)}\right) \Omega^{(c)} \left(\tau^{(c)} I^{(c)}\right) \tag{4.7}$$

for each category c. Here $I^{(c)}$ is the $J^{(c)} \times J^{(c)}$ identity matrix, $\Omega^{(c)}$ is a correlation matrix, and $\tau^{(c)}$ is the standard deviation of the product preference heterogeneity. In this simulation, we set $\tau^{(c)}$ to 2 for every category. The random correlation matrices $\Omega^{(c)}$ are made using the Vine method (Lewandowski, Kurowicka, & Joe, 2009), with the partial correlations being sampled from a Beta$(0.2, 1)$ distribution.

After the utility $u_{ijt}^{(c)}$ is computed for every product in a category, customers are simulated to buy exactly one of the $J^{(c)}$ products, namely the one which has the highest utility. The choice $y_{ijt}^{(c)} \in \{0, 1\}$ of customer i to buy product j in week t from category c is therefore determined by

$$y_{ijt}^{(c)} = \begin{cases} 1 & \text{if } u_{ijt}^{(c)} > u_{ikt}^{(c)} \quad \forall k \neq j \\ 0 & \text{otherwise.} \end{cases} \tag{4.8}$$

Since consumers often put more than one product per category in their shopping basket, we make it so the consumer buys two different products from a given category instead of just one for 50% of the times that a customer chooses to buy from a category.

4.2.3 Simulation with Distinct Customer Categories

In the simulation method described so far, the parameters for base utility and price sensitivity are constant for every category and every customer that we simulate. This is fine when using the simulated data for mapping products, as the products should still form distinct categories. However, we expect that pooling the product vectors into customer vectors and creating a customer map would likely show a fairly uniform distribution of points since the only aspect differentiating customers are random error terms. To be able to validate that we are able to plot a customer map that shows distinct groups of customers, some small adjustments are made.

We define 10 customer categories c_i for which we want to simulate clearly distinct data. The first adjustment we make regards the process of choosing a category. The latent variable z_{ict} described in (4.4) is so far dependent on base utility γ_c and an error term, where γ_c is constant over all categories. In the adjusted simulation, we instead define the latent variable as

$$z_{ict} = \gamma_{c,c_i} + \epsilon_{ict}, \tag{4.9}$$

where the base utility γ_{c,c_i} depends on the customer category c_i as well as the product category c. We manually construct γ_{c,c_i} which is provided as an input for the simulation. The error term $\epsilon_{ict} \sim \text{MVN}(0, \Omega)$ is an uncorrelated error term. When constructing the base utilities, we do not want these to be in conflict with the purchase incidence correlation indicated by the correlation matrix Ω. Therefore, we construct a 20×10 matrix Γ consisting of the individual γs by setting the standard utility in each element of the matrix to -0.5 and adding to each column (representing a customer category) two other columns (from different 'clusters' of product categories) from the correlation matrix Ω. This way, each customer category gets a base utility vector that has slight variations of -0.5 for some of the categories, but is the standard -0.5 for others, while at the same time being in line with the specifications of Ω. The exact specification of Γ can be found in Appendix A.2.

A last change made to the simulation process is by specifying different price sensitivities β in (4.6) per customer category, instead of setting it equal to 2 for everything, we define half of the customer categories to have a 'low' price sensitivity of 1.5 and the other half of the categories to have a 'high' price sensitivity of 2.5. We believe that these changes in price sensitivity are not so big that they greatly influences how often some products are bought but they do introduce a more realistic difference to the customer groups.

4.2.4 P2V-MAP Application

We simulate data for $T = 100$ weeks and 1000 customers, both with and without the varying parameters for different customer groups. For the normal simulation data, we expect that products on the product map within categories will be closer on the product map than compared to products from different categories, since the mapping is based on pairwise similarities and products from the same category are more similar. However, we expect that a customer map will not show any clear patterns. For the adjusted simulation data, we expect to see distinct groups in both the product map and the customer map.

In addition to making a product map, we will get the co-occurrence and similarity scores of average product vectors per category. We expect that these should be correlated to the purchase incidence matrix Ω, which we used to model which categories were more likely to be bought together.

4.3 Evaluation and Benchmarking

In this section, we explain the metrics we use to evaluate the performance of our P2V-MAP implementation. In particular, we want to evaluate how well the separation of points in the product or customer maps corresponds to what we know about how many *real* categories there are. We know these real categories for the simulation data since we modelled it ourselves to have 20 product categories and 10 customer categories. We know for the Instacart data how many real categories there are as well because we have the information of which aisle/department each product belongs to.

The first metric we use to evaluate the clusters is the silhouette score (Rousseeuw, 1987). Silhouette helps to evaluate the correctness of the assignment of a point in a particular cluster by measuring how similar it is to its own cluster compared to other clusters. For a product j, if we define $a(j)$ as the average distance to other points in the same category and $b(j)$ as the average distance to the category that the point is closest to on the map, then the silhouette score is given by

$$\text{SIL} = \frac{1}{J} \sum_{j} \frac{b(j) - a(j)}{\max\{b(j), a(j)\}}. \tag{4.10}$$

This score can range from 1 to -1, where 1 is the best and -1 is the worst. Values near 0 indicate overlapping clusters, which means the map is harder to read. Negative values generally mean that a point has been assigned to the wrong cluster, since it is closer on the map to points that are not in its actual category.

The second metric we use is the adjusted mutual information (AMI) score (Vinh, Epps, & Bailey, 2010), which is used for evaluating clustering as well. It measures the agreement between true category labels X and predicted category labels Y, where it corrects for agreement that could be solely due to chance. To predict cluster labels, we use k-means clustering on the product map coordinates, where we set k to the number of categories that we know the products belong to. Then, we can calculate the AMI score, which is defined as

$$\text{AMI} = \frac{\text{MI}(X, Y) - E[\text{MI}(X, Y)]}{\frac{\text{H}(X)+\text{H}(Y)}{2} - E[\text{MI}(X, Y)]}. \tag{4.11}$$

Here, $\text{MI}(X, Y)$ is the mutual information score

$$\text{MI} = \sum_{x \in X} \sum_{y \in Y} p(x, y) \log\left(\frac{p(x, y)}{p(x)p(y)}\right)$$

where $p(x, y)$ is the joint probability distribution of X and Y and $p(x)$ and $p(y)$ are the marginal probability distribution of X and Y. $E[\text{MI}(X, Y)]$ is the expected value of the mutual information score and $H(X)$ and $H(Y)$ are the label entropies. The AMI score can range from 0 to 1. If the true labels align with the predicted labels received through the k-means clustering, the AMI score will take a score of 1. A random clustering assignment would get a score 0.

The third metric that we use is the hit rate (HR), which measures for each product j in category c what fraction $\text{f}(j, c(j))$ of the $J^{(c)} - 1$ nearest neighbours belong to the same category. The best value for this would be 1, as there are $J^{(c)}$ products per category. The worst value is 0. The equation for the hit rate is given by

$$\text{HR} = \frac{1}{J} \sum_{j} \text{f}(j, c(j)). \tag{4.12}$$

In order to compare the metrics that we obtain for our product maps, we also generate a product map by drawing random product coordinates as a benchmark. We sample these coordinates (x, y) within a unit circle, where $x = r\cos(\alpha)$, $y = r\sin(\alpha)$ and $r^2 \sim \text{U}[0, 1]$ and $\alpha \sim \text{U}[0, 2\pi]$, as is also done by Gabel et al. (2019).

5 Results

In this section, we present the maps and the accompanying evaluation metrics that we created. In Sect. 5.1, we describe the simulated dataset that resulted from our simulation method. In Sect. 5.2, we present the product maps and benchmark metrics obtained after implementing our replication of the P2V-MAP method by Gabel et al. (2019). Finally, we present the customer maps that we obtained by pooling product vectors in Sect. 5.3.

5.1 Simulation Results

The final simulated dataset with no distinct customer groups contains 554,421 entries of products for 100 customers over 100 weeks. There are 9.3 products per basket on average, with a minimum of 1 and a maximum of 25 products. There are 8 products that occur fewer than $n_{min} = 20$ times in the dataset, which are removed before training the Skip-Gram model. The distribution of frequencies of products across baskets can be found in Appendix A.3. Figure 2 gives an additional indication of how prices were simulated and what impact it had on the purchase frequency. As expected, cheaper products appear in more baskets. This is because we set price sensitivity at 2, meaning that more expensive products are less likely to be chosen.

Figure 2. Prices of simulated products versus frequency across baskets

5.2 P2V-MAP Replication Results

The following sections present our results for the replication part of this thesis. A short description of the code written for this thesis can be found in Appendix B. All results were obtained on an Intel(R) Core(TM) i5-6200U CPU @ 2.30GHz, 2400 Mhz with 8.00GB RAM running 64-bit Windows 10.

5.2.1 Simulated Data Product Map

For the simulated data, more than 5 million training tuples consisting of a center product, a positive sample, and 20 negative samples were created. For the purpose of training, these tuples were split into around 5,000 batches of 1,000 training samples. Furthermore, the model is trained for multiple epochs, where one epoch is one pass over all the training batches. Figure 3 shows that the more often batches are used in training, the more the average batch loss decreases, especially towards the beginning. For this reason, we use multiple epochs to train the data. Each epoch takes around 50 seconds to train. Note that training for too many epochs causes overfitting on the training data, which is generally not desirable. However, we are not interested in out-of-sample prediction in this thesis but only in in-sample explanation. As such, overfitting is not necessarily a large issue.

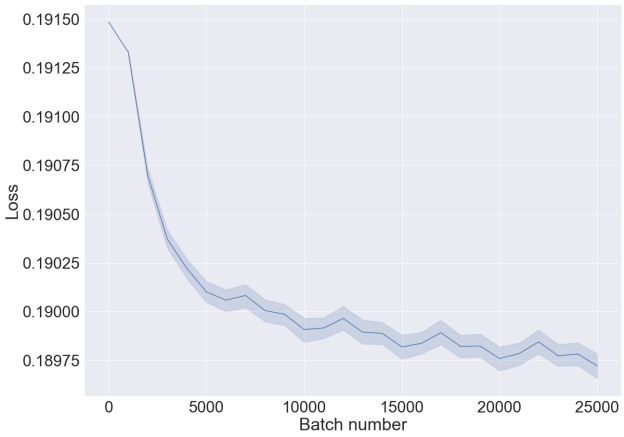
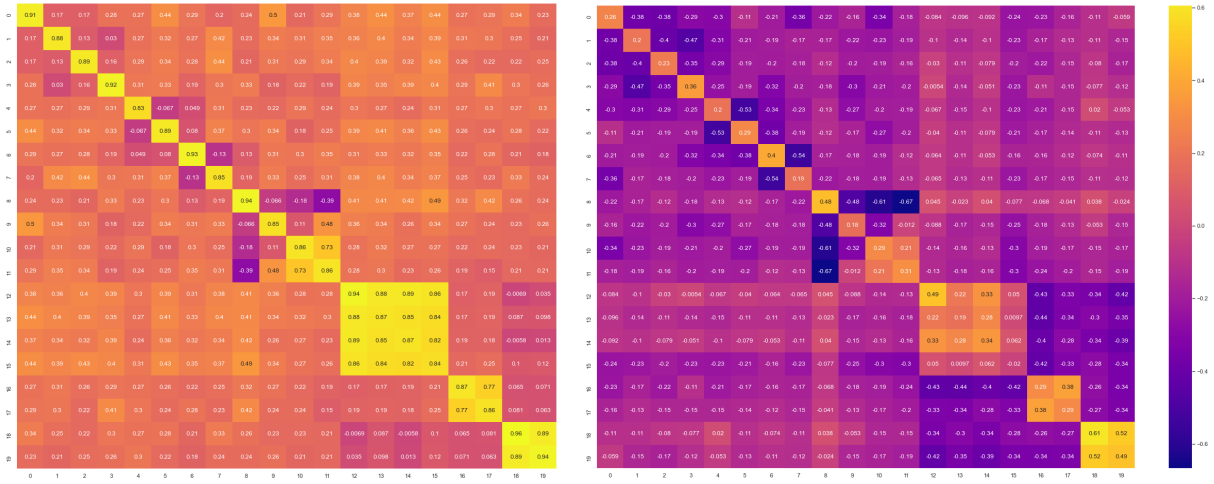


Figure 3. Batch loss when training the simulated dataset over 5 epochs (5 times 5,000 batches)

In order to evaluate the similarities and co-occurrences between the product categories that we simulated, we present annotated heatmaps of the scores calculated for the average vectors per category in Fig. 4.



(a) Similarity scores **(b)** Co-occurence scores

Figure 4. Similarity and co-occurrence scores for the average category vectors trained on the simulated data

The heatmaps shown in Fig. 4 are fairly similar in the sense that product categories that have high similarity scores with each other also have relatively higher co-occurrence scores. This is to be expected, since products that co-occur often are also more likely to be similar. The biggest difference between these two maps are that co-occurrence values of categories with themselves are lower compared to the similarity scores, but this is not surprising as products from the same category should have high similarities, whereas it is not guaranteed that they co-occur with products from the same category to the same extent. In addition, these results are very much in line with the heatmaps produced by Gabel et al. (2019).

The product map for the simulated data is presented in Fig. 5, with the product categories labelled on the map.

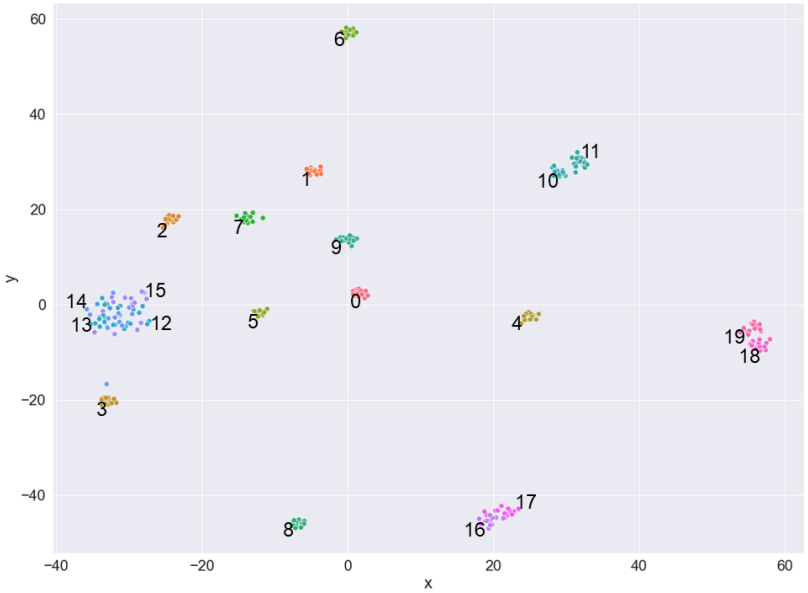


Figure 5. Simulated product map

The product map shows clear clusters that mostly adhere to the simulated product categories. Only categories 12, 13, 14, and 15, as well as categories 16 and 17, show points that overlap too much to be discernible as separate clusters. This is in line with the category similarity and co-occurrence scores presented in Fig. 4, which confirms that products and product categories with higher similarity also have a smaller distance to each other on the product map.

5.2.2 Instacart Product Map

For the Instacart data, more than 5.5 million training samples are created. We train the data for 50 epochs to ensure that the training loss would be low. A plot of the average batch loss is presented in Appendix A.4. Figure 6 shows the product map obtained for this data, which is coloured based on the departments that the products belong to. A few clusters of interest have been labelled for easier referencing.

Figure 6. Instacart product map

This product map is clearly not as easy to interpret as the map of simulated data that was presented in Fig. 5. Some of the more distinct clusters are of products that belong to different aisles but the same department. For example, cluster A has both white wines and red wines, which both belong to the department alcohol. Other clusters are of only a specific aisle within that department. For example, cluster E contains milk and cluster F contains eggs, both of which belong to the *dairy eggs* department. Cluster C consists of different ice creams while cluster J consists of frozen pizzas but both belong to the *frozen* department. This shows that some departments have stronger category (aisle) similarity within them than others, possibly in part because a category like *frozen* is very broad.

5.2.3 Benchmark Results

Table 1 presents the benchmark metrics discussed in Section 4.3 calculated for the product maps of the Instacart data, the simulated data, and a randomly generated product map. In addition, we added the benchmark results for the simulated product map presented by Gabel et al. (2019).

Table 1. Benchmark scores for product maps

	Silhouette	AMI	Hit rate
Simulated	0.614	0.869	0.808
Instacart	-0.320	0.364	0.236
Random	-0.287	-0.008	0.015
Simulated (Gabel et. al)	0.350	0.823	0.696

All three metrics are significantly better for the simulated dataset than for the Instacart dataset, which also corresponds to the differences in readability of the product maps that could be seen in Fig. 5 and Fig. 6. A potential explanation for this is that the product vectors for the Instacart dataset are of a lower quality because we took a subsample, which means that basket sizes were relatively smaller and, therefore, each center product had fewer context products to train with than they otherwise could have had. Consequently, this product map does not show a complete picture. In addition, the benchmarks

as well as the colours on the Instacart product map are based on certain categories given by the aisles and departments but in reality it is likely that product co-occurrences do not always follow these aisles very closely. For example, in Fig. 6, some *beauty and hair care products* (cluster H) are similar on the map to products from the aisle *air fresheners/candles* (cluster I). While this is bad for the benchmarking metrics, it is not too unexpected when considering that these products are likely to be bought by the same customers (most probably women) even if they are not in the same category.

The randomly generated product map, which can be found in Appendix A.5, has very low benchmarking scores as expected. The silhouette score is negative indicating that products are not mapped near their own categories and the AMI score is very near zero, which is the result we would expect for randomly generated data.

Finally, we see that compared to the benchmarking metrics presented by Gabel et al. (2019) for their simulated data, we have a similar AMI score for our simulated data and we even have a higher silhouette score and hit rate. This confirms that we have been generally successful at replicating the P2V-MAP methodology.

5.3 Customer Mapping Results

We pooled the product vectors that were used to plot the product maps presented in Sect. 5.2 in order to create customer maps. We first applied average pooling to the vectors created for the standard simulated data as well as vectors trained on the adjusted simulated data with different parameters per customer group. The resulting customer maps are presented in Fig. 7.

(a) Standard simulation customers

(b) Adjusted simulation customers

Figure 7. Customer maps for the standard simulated data and for the adjusted simulation

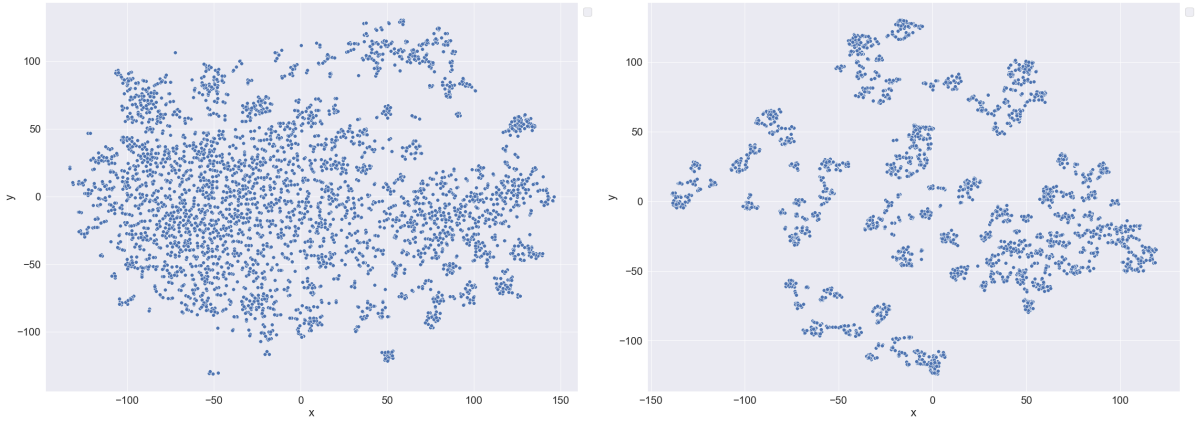
As expected, the customer map for the normal simulation data shows a uniform distribution of customers, which is also how they were simulated. However, the map for the adjusted simulation data clearly shows 10 distinct clusters which correspond to the 10 categories of customers that were simulated. This confirms that aggregating the vectors for products purchased by customers can successfully create a representation for those customers.

Because the adjusted simulation data has known categories for the customers, we were also able to calculate our benchmarking metrics for the customer maps made with this data. This way, we can compare different pooling methods. We implemented average pooling, max pooling and min pooling. In addition, we implemented 'top x' pooling where we take the average of only the top x most frequently bought products of a customer. For this, we tried different versions of x and measured the benchmark metrics to see what would result in better scores. The results for this can be found in Appendix A.7. In the end, we chose to use 10 for our x. The benchmarking metrics for our pooling methods can be found in Table 2.

Table 2. Benchmark metrics for different pooling methods

	Silhouette	AMI	Hit rate
Average	0.856	1	1
Top 10	0.323	0.723	0.647
Max	-0.103	0.118	0.146
Min	-0.118	0.139	0.160

The average pooling method, which is also the method used to create the attractive customer map in Fig. 7, has very high scores here, having the maximum score for both AMI and hit rate. The top 10 pooling method also seems to represent the customers relatively well according to these metrics, which makes sense since it also uses average pooling, just for fewer products. The min and max pooling methods are less successful at representing the simulated customer characteristics. However, as we have seen in Sect. 5.2, it is not necessarily the case that a method that works well for mapping simulated data will also immediately create a clear product map for real-life data. For this reason, we also apply the average and top 10 pooling to the vectors trained on the Instacart dataset. The results for this are presented in Fig. 8.



(a) Customer map by average pooling **(b)** Customer map by top-10 pooling

Figure 8. Customer maps created for the Instacart dataset

These figures are harder to evaluate, since we do not have any information about the 'real' customer groups found in the Instacart data. Consequently, we can not determine how well the data represents the different customer segments. However, we do see that both the map from average pooling and the map

from top 10 pooling have slightly more of a clustered structure than a random customer map like that of the standard simulation data in Fig. 7. Interestingly, customers appear to be clustered together much more with the top 10 aggregation method compared to the map created by average pooling. This may be because the customers' top 10 lists are more similar to each other, as there are likely a few basic staples that are very commonly bought by any customer. One last thing to note is that not all customers appear with the same frequency in the Instacart dataset, which means that some of the customer vectors are not created with as much information. Selecting data with more information for each customer may lead to more apparent patterns in the data. Overall, we believe that the results presented so far show that there is potential for the application of pooling methods to create customer maps out of vector maps.

6 Conclusion

In this thesis, we implemented a method to embed latent product attributes and represent them in a two-dimensional space as described by the P2V-MAP methodology (Gabel et al., 2019). In order to evaluate our implementation, we simulated product basket data and we sampled data from the Instacart dataset (Instacart, 2017) to create product maps from. Visual inspection of the product maps as well as benchmark metrics let us conclude that our implementation of the P2V-MAP methodology is able to successfully represent product characteristics in a two-dimensional space. Compared to the results presented by (Gabel et al., 2019), we have similar patterns and metrics for the simulation data which indicates that the replication part of this thesis has been succesful.

In an extension of the P2V-MAP methodology, we created customer maps by pooling the vector embeddings trained for the products that were bought. We applied average pooling, min pooling, max pooling, and top x pooling where we have taken the average of the x most frequently bought products of the customer. Average pooling resulted in high benchmark scores for adjusted simulation data which used different parameters per customer group to create 10 distinct groups. For the Instacart dataset, it is less conclusive whether pooling is able to create representative vectors for the customers in the dataset.

Future research could be conducted to look into different methods to pool the embeddings. In addition, it would be interesting to create customer maps for realistic data where customer characteristics are known and where there is enough data for each customer. This way, it can be evaluated how well the customer characteristics are represented for realistic data.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., ... Zheng, X. (2015). *TensorFlow: Large-scale machine learning on heterogeneous systems.* Retrieved from `https://www.tensorflow.org/` (Software available from tensorflow.org)

Baldassini, L., & Serrano, J. A. R. (2018). client2vec: towards systematic baselines for banking applications. *arXiv preprint arXiv:1802.04198*.

Bergeron, P., & Hiller, C. A. (2002). Competitive intelligence. *Annual Review of Information Science and Technology (Arist)*, *36*, 353–90.

Blattberg, R. C., Glazer, R., & Little, J. D. (1994). *The marketing information revolution.* Harvard Business School Press.

Bottou, L. (2010). Large-scale machine learning with stochastic gradient descent. In *Proceedings of compstat'2010* (pp. 177–186). Springer.

Bradlow, E. T., Gangwar, M., Kopalle, P., & Voleti, S. (2017). The role of big data and predictive analytics in retailing. *Journal of Retailing*, *93*(1), 79–95.

Chintagunta, P. K. (1992). Estimating a multinomial probit model of brand choice using the method of simulated moments. *Marketing Science*, *11*(4), 386–407.

Elrod, T., Russell, G. J., Shocker, A. D., Andrews, R. L., Bacon, L., Bayus, B. L., ... others (2002). Inferring market structure from customer response to competing and complementary products. *Marketing Letters*, *13*(3), 221–232.

Gabel, S. (2019). One-to-one marketing in grocery retailing.

Gabel, S., Guhl, D., & Klapper, D. (2019). P2v-map: mapping market structures for large retail assortments. *Journal of Marketing Research*, *56*(4), 557–580.

Instacart. (2017). https://www.instacart.com/datasets/grocery-shopping-2017.

Jacobs, B. J., Donkers, B., & Fok, D. (2016). Model-based purchase predictions for large assortments. *Marketing Science*, *35*(3), 389–404.

Jagabathula, S., Subramanian, L., & Venkataraman, A. (2018). A model-based embedding technique for segmenting customers. *Operations Research*, *66*(5), 1247–1267.

Jarrett, K., Kavukcuoglu, K., Ranzato, M., & LeCun, Y. (2009). What is the best multi-stage architecture for object recognition? In *2009 ieee 12th international conference on computer vision* (pp. 2146–2153).

Jégou, H., Douze, M., Schmid, C., & Pérez, P. (2010). Aggregating local descriptors into a compact image representation. In *2010 ieee computer society conference on computer vision and pattern recognition* (pp. 3304–3311).

Le, Q., & Mikolov, T. (2014). Distributed representations of sentences and documents. In *International conference on machine learning* (pp. 1188–1196).

LeCun, Y., Boser, B. E., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W. E., & Jackel, L. D. (1990). Handwritten digit recognition with a back-propagation network. In *Advances in neural information processing systems* (pp. 396–404).

Lewandowski, D., Kurowicka, D., & Joe, H. (2009). Generating random correlation matrices based on vines and extended onion method. *Journal of multivariate analysis*, *100*(9), 1989–2001.

Manchanda, P., Ansari, A., & Gupta, S. (1999). The "shopping basket": A model for multicategory purchase incidence decisions. *Marketing science*, *18*(2), 95–114.

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. *1st International Conference on Learning Representations (ICLR 2013)*.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *27th annual conference on neural information processing systems (nips 2013)* (pp. 3111–3119). Curran Associates.

Mitchell, J., & Lapata, M. (2010). Composition in distributional models of semantics. *Cognitive science*, *34*(8), 1388–1429.

Mitra, B., Nalisnick, E., Craswell, N., & Caruana, R. (2016). A dual embedding space model for document ranking. *arXiv preprint arXiv:1602.01137*.

Myers, J. H., & Tauber, E. (2011). *Market structure analysis*. Marketing Classics Press.

Neslin, S. A., & Van Heerde, H. J. (2009). *Promotion dynamics*. Now Publishers Inc.

Netzer, O., Feldman, R., Goldenberg, J., & Fresko, M. (2012). Mine your own business: Market-structure surveillance through text mining. *Marketing Science*, *31*(3), 521–543.

O'Sullivan, A., & Sheffrin, S. (2003). *Prentice hall economics: principles in action* (No. 18). Prentice Hall.

Qian, Y., Jiang, Y., Du, Y., Sun, J., & Liu, Y. (2019). Segmenting market structure from multi-channel clickstream data: a novel generative model. *Electronic Commerce Research*, 1–25.

Rehurek, R., & Sojka, P. (2010). Software framework for topic modelling with large corpora. In *In proceedings of the lrec 2010 workshop on new challenges for nlp frameworks.*

Ringel, D. M., & Skiera, B. (2016). Visualizing asymmetric competition among more than 1,000 products using big search data. *Marketing Science*, *35*(3), 511–534.

Rousseeuw, P. J. (1987). Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, *20*, 53–65.

Smith, W. R. (1956). Product differentiation and market segmentation as alternative marketing strategies. *Journal of marketing*, *21*(1), 3–8.

Srivastava, R. K., Alpert, M. I., & Shocker, A. D. (1984). A customer-oriented approach for determining market structures. *Journal of Marketing*, *48*(2), 32–45.

Urban, G. L., Johnson, P. L., & Hauser, J. R. (1984). Testing competitive market structures. *Marketing Science*, *3*(2), 83–112.

Van der Maaten, L. (2014). Accelerating t-sne using tree-based algorithms. *The Journal of Machine Learning Research*, *15*(1), 3221–3245.

Van der Maaten, L., & Hinton, G. (2008). Visualizing data using t-sne. *Journal of machine learning research*, *9*(Nov), 2579–2605.

Vinh, N. X., Epps, J., & Bailey, J. (2010). Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. *The Journal of Machine Learning*

Research, *11*, 2837–2854.

Wedel, M., & Kamakura, W. A. (2012). *Market segmentation: Conceptual and methodological foundations* (Vol. 8). Springer Science & Business Media.

Wedel, M., & Kannan, P. (2016). Marketing analytics for data-rich environments. *Journal of Marketing*, *80*(6), 97–121.

Appendices

A Figures

A.1 Manual Correlation Matrix Ω

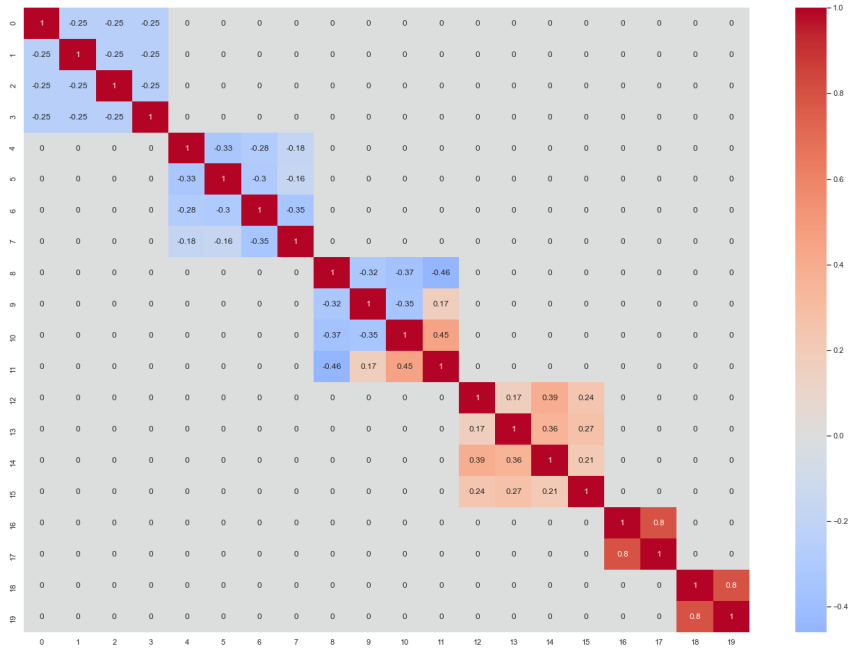


Figure 9. Manually made correlation matrix used for the multivariate probit model in the simulation study

A.2 Manual Base Utilities Matrix Γ, for Distinct Customer Categories

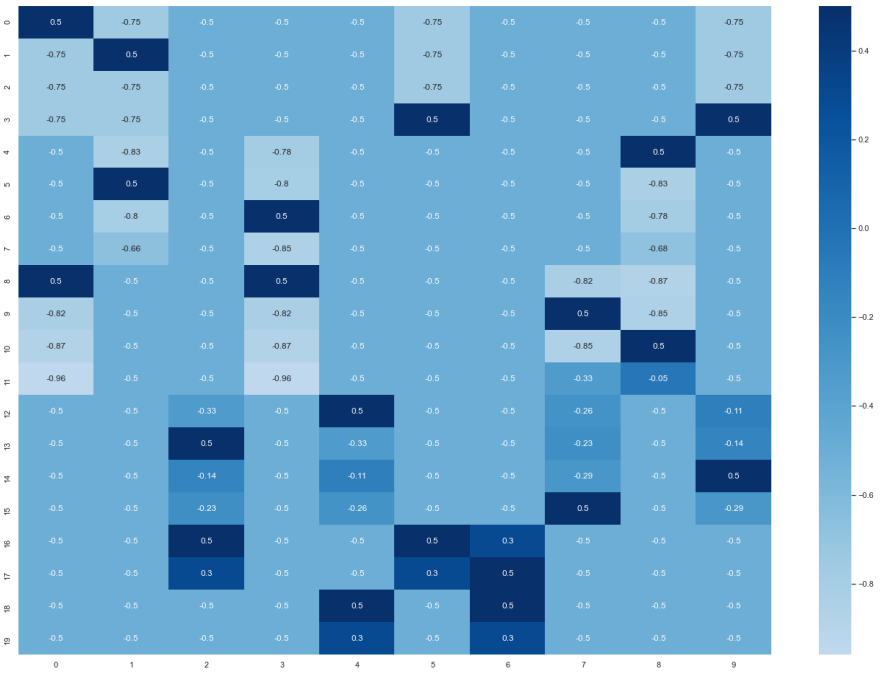


Figure 10. Manually made base utilities matrix, used for the adjusted simulation study with different parameters per customer category

A.3 Product Counts in Simulated Train Data

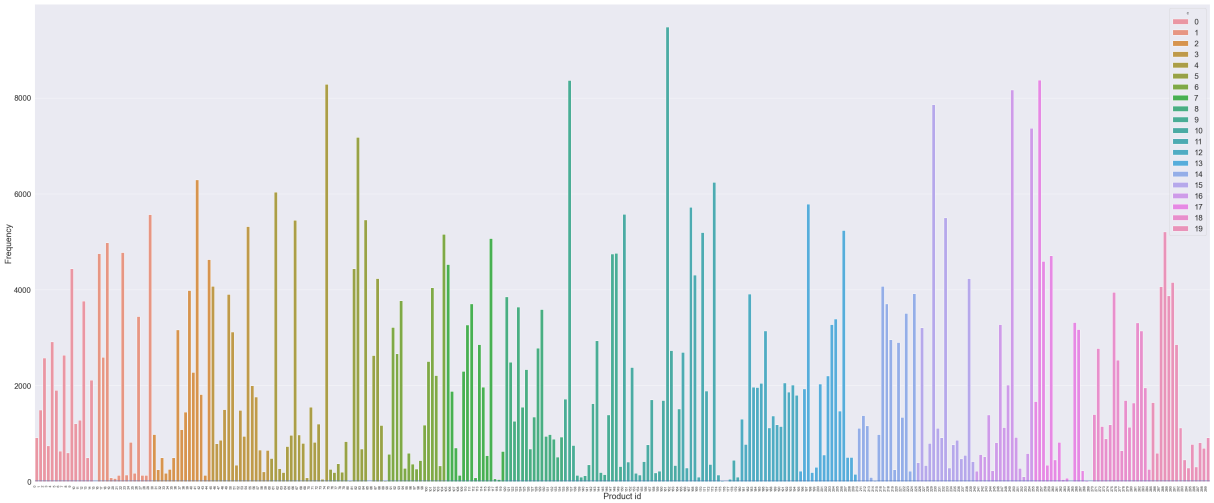


Figure 11. Frequency of products across baskets for the simulated training data

A.4 Instacart Training Loss

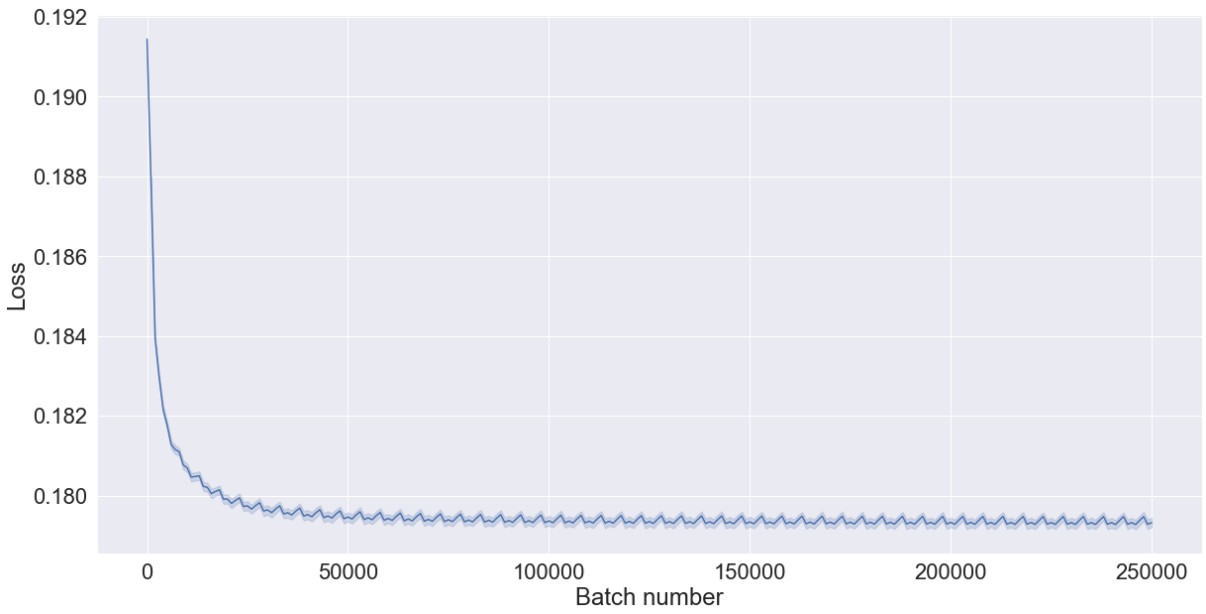


Figure 12. Plot of Instacart training loss, over 50 epochs

A.5 Randomly Generated Product Map

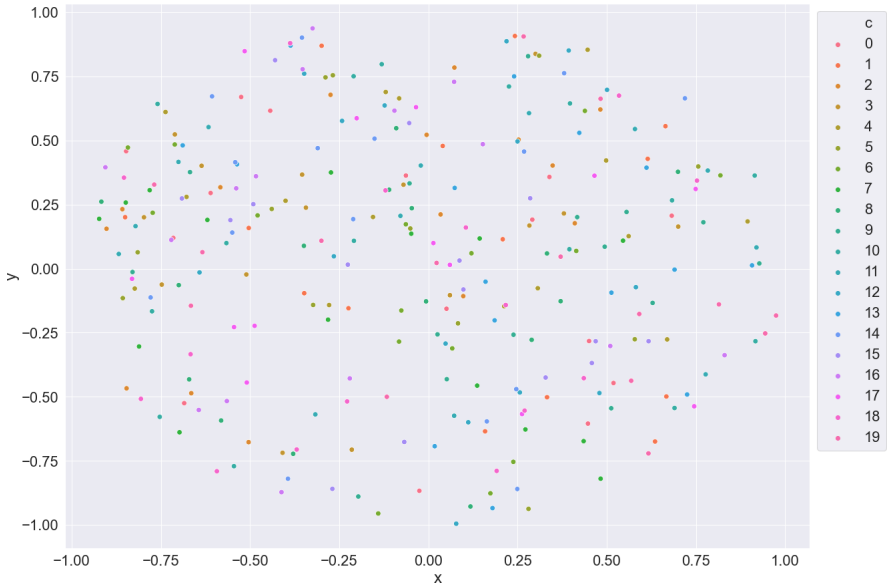
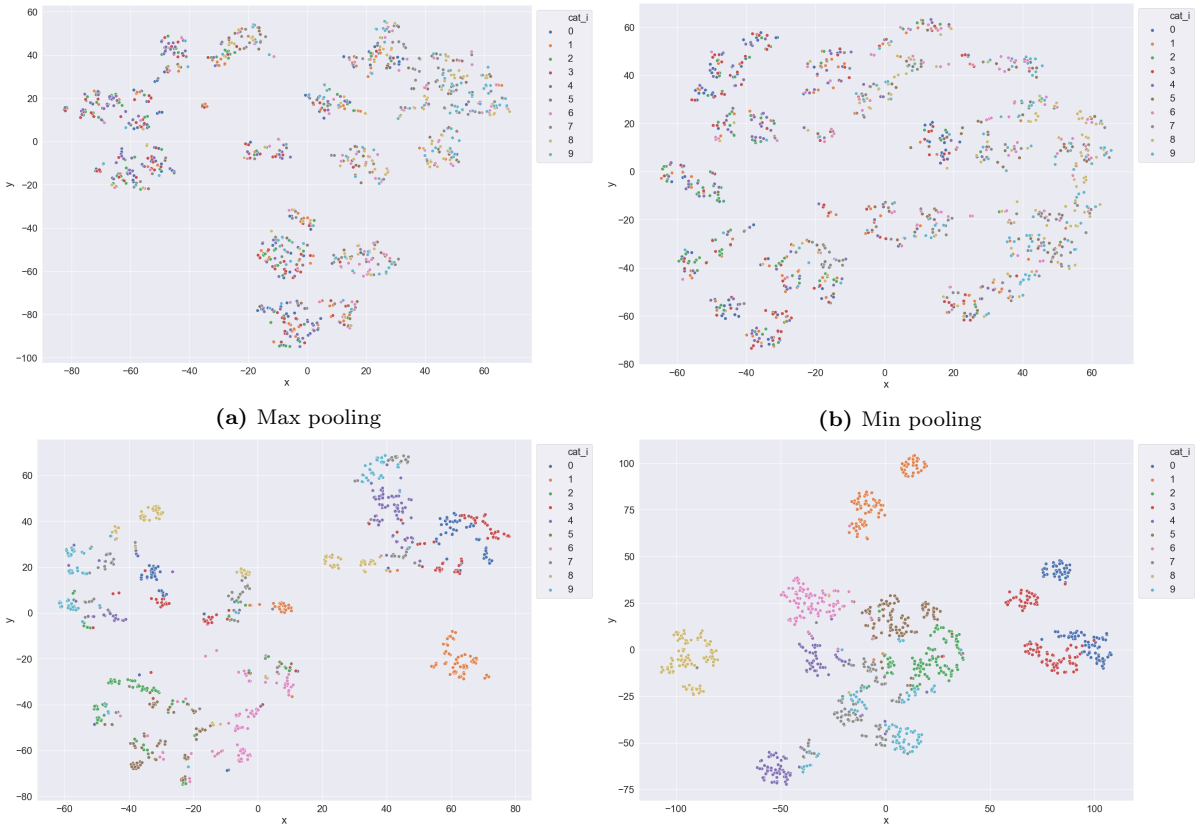


Figure 13. Plot of coordinates randomly sampled from a unit circle

A.6 Customer Maps from Adjusted Simulation Data



(a) Max pooling

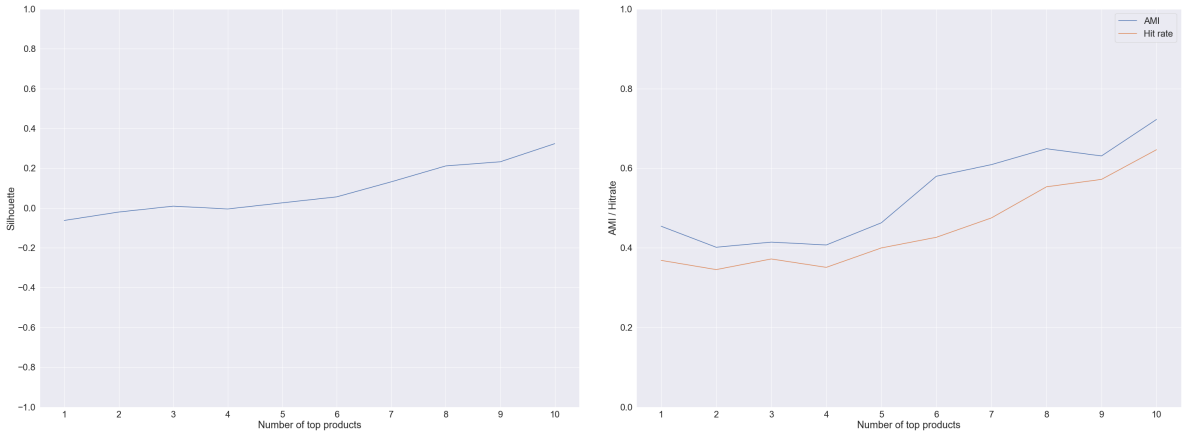
(b) Min pooling

(c) Top 5 pooling

(d) Top 10 pooling

Figure 14. Customer maps for the adjusted simulation data using four different pooling methods

A.7 Benchmark Metrics for Different Values of x in the Top x Pooling Method



(a) Silhouette score for different x

(b) AMI score and hit rate for different x

Figure 15. Silhouette scores and AMI hit rate for different x in the top x pooling method

B Code Description

`simulationExperiment.py` contains the code used to create our simulated data, both with and without varying customer parameters.

`createInstacartData.py` contains methods to process raw data from the Instacart dataset, merging various files so everything is available in one *pandas* DataFrame. Additionally, it provides methods to subsample the full dataset so that it is easier to use for training embeddings.

`dataPrep.py` contains classes that will prepare input data for being trained with the modified Skip-Gram model. It removes infrequent products and creates pairs of center products with positive samples and 20 negative training samples.

`productEmbed.py` implements the modified Skip-Gram model.

`productMap.py` implements t-SNE to create a product map out of the embedding outputs from `productEmbed.py` or out of pooled customer embeddings. It gives an option to create a product map that is shown in the console or an interactive product map that is labeled with product IDs (or in the case of the Instacart data, names and department names).

`benchmarkAndPool.py` implements ways to plot loss, get average category similarity and co-occurrences, calculate benchmarks, and to get a random benchmark for embedding outputs from `productEmbed.py`. Additionally, it provides a method to pool product vector embeddings and show them in a customer map using the methods from `productMap.py`.

`visualise.py` contains methods to visualize some of the datasets, such as visualizing correlations in a heatmap or visualizing the counts in the Instacart dataset.

`utils.py` contains a few helper methods, like saving DataFrames to CSV files and getting the current time.

`BachelorThesisMain.py` contains code to act as a central hub from which most of the methods in the other files can be run.