ProdBERT: Shopping Basket Completion Using Bidirectional Encoder Representations from Transformers

ERASMUS UNIVERSITY ROTTERDAM
Erasmus School of Economics

Bachelor Thesis Business Analytics and Quantitative Marketing

Ruben Eschauzier
480900re
July 2020

Supervisor: Luuk van Maasakkers
Second assessor: Flavius Frasincar

Abstract

Market analysis allows a company improve its understanding of the market it operates in and creates an opportunity to increase profits. This thesis aims to first replicate P2V-Map as a springboard to improved market analysis. Secondly, by using a state-of-the-art NLP model on product baskets this thesis aims to improve unsupervised learning of market structure. For this purpose this thesis introduces ProdBERT. ProdBERT learns market structures by pre-training the BERT architecture using product baskets as 'sentences' and product IDs as 'words' for masked language modeling. ProdBERT also trains itself by splitting baskets into two parts for next sentence prediction. To compare the performance of ProdBERT and P2V-Map in modeling market structure this thesis evaluates the performance of both models in a basket completion task. The results shows that BERT significantly outperforms P2V-Map in the basket completion task.

Contents

1 Introduction

To properly market products one must understand the structure of the market, this is typically done through market analysis. An example of structure in a market is complementary and subsidiary relationships between products. Creating insight into these relationships aids greatly in deciding how to market and position a company. Online retailers want to be able to group products based on attributes quickly and correctly. Another application of market analysis is product recommendation based on a given shopping basket. As seen in [Hinz and Eckert, 2010] recommendation systems decrease search costs for niche products. This reduction in search costs leads to niche products being purchased more often. Depending on the margins on niche products, a good recommendation system might increase profit for an online retailer. The direct impact of product recommendations is also outlined in [Hinz and Eckert, 2010]. The paper finds that a recommended product is purchased up to two times as much as non-recommended products. Finally, brick-and-mortar stores can optimize the way the shelves are arranged using market structure analysis.

This thesis first replicates an existing method of mapping market structure called P2V-Map [Gabel et al., 2019]. This is an unsupervised machine learning method that uses a skip-gram model to create product vectors containing latent attributes of the products. These product vectors are then mapped to a 2-dimensional space to create a visual mapping of market structure that puts similar products close to each other. The main advantage of P2V-Map is that it makes no assumptions about the data used and only uses shopping basket data, which is easy to gather. This thesis extends this method by applying a Transformer based language model named BERT (**B**idirectional **E**ncoder **R**epresentations from **T**ransformers) to learn market structure. This method was introduced in [Devlin et al., 2018] as an improvement to existing language models, such as the skip-gram model. The input for BERT is a sequence of tokens, most commonly sentences and trains itself to be able to 'understand' language. For market analysis application, This thesis introduces ProdBERT, which will use baskets of products instead of sentences to train the model on market structure. Its main advantage being that it is able to look at the context of a basket to represent a product. An example of this is tomatoes bought with lettuce and tomatoes bought with pasta. Although tomatoes are in both baskets the logical recommendation for these two baskets are in completely different categories. Using this advantage of ProdBERT over P2V-Map this thesis aims to model more complex dependencies within baskets and allow for more in-depth market structure analysis. This thesis will test the ability of this new language model by making both P2V-Map and ProdBERT do product recommendations and comparing the accuracy.

The aim of this thesis is to advance deep-learning application in the field of marketing and implement a more sophisticated way of learning market structure using baskets of products. More formally goal of our research is to compare the performance of P2V-Map and ProdBERT on basket completion tasks. Additionally this thesis proposes an empirical application of the ProdBERT model in product recommendation.

First we will go into literature in the fields of NLP (Natural Language Processing) and basket completion in Section 2. This Section will also highlight the gap in literature basket completion that this thesis attempt to fill. Section 3 describes the data used to replicate P2V-Map and the empirical data used to train and validate ProdBERT with.Section 4 explains the P2V-Map method and how it is implemented, this Section also introduces the BERT model as proposed in [Devlin et al., 2018] and proposes a new model ProdBERT. Section 5 presents and discusses the results of the implementation of both P2V-Map and ProdBERT proposed in this thesis. Finally Section 6 presents the conclusion of our research and outlines possible avenues for further work.

2 Literature review

In literature there are two fields that are of interest for this thesis: Natural Language Processing (NLP) and basket completion tasks. Traditionally researchers in NLP used one-hot encoded vectors of size V where V is the number of words in the vocabulary. Each word vector is a binary vector with the value one being in a unique index for each word within the vocabulary. This suffers from high dimensionality when the vocabulary is large and it captures no semantic meaning of a word. As a solution a model that can learn distributed representations of words was introduced in [Mikolov et al., 2013] and named Word2Vec. These representations are low-dimensional vectors that contain semantic meaning of the word it represents. This solved both the high dimensionality problem and allowed researchers to use these vectors for analysis of text.

Word2Vec is used to perform basket completion tasks in marketing. In [Grbovic et al., 2015] a method of basket completion using skip-gram models is proposed. It does so by treating products as 'words' and purchase sequences as 'sentences', thus allowing the model to learn low-dimensional vectors that represent products and its latent attributes. These product vectors are then used in two algorithms that are able to predict the next product to be purchased. The first is **prod2vec-topK**, where given a purchased product the cosine similarity between that product and all other products available is calculated. The similarity is defined as

$$similarity = \frac{A \cdot B}{\|A\|\|B\|}, \tag{1}$$

where A and B are product vectors. This similarity is high if the products to which the vectors belong as similar. The product with the highest cosine similarity is then used to complete the basket of products. The second is **prod2vec-cluster** where similar products are clustered using K-means clustering based on product vectors. Then given a purchased product p_i in cluster c_i a product p_j is recommended from a related cluster c_j. The product recommended within cluster c_j is found by calculating the cosine similarity between every product vector within c_j and the purchased product's vector p_j.

A disadvantage of Word2Vec in NLP is that its vectors are not context dependent. This means that a given word has the same word vector regardless of what context this word was used. This does not allow the word vector to capture the different meanings of a word based on the sentence it is used in. The same drawback applies to products, where prod2vec is unable to capture the different uses and attributes of products that depend on the basket it is in. The model proposed in [Radford et al., 2018] can look at the left context of a sentence to model language. This means that every word in a different context will have an unique word vector representing the meaning of that word within the context. The left context of a word is the sentence before the word to predict. For example in the sentence 'I want to buy a car and drive it' if the word to predict is car, the left-context is 'I want to buy a'. This concept was further improved in [Devlin et al., 2018]. This paper proposes BERT, which is able to take into account both the right and left context of a word and use that to create word vectors. This approach is able to achieve state-of-the-art results in eleven NLP tasks.

To the knowledge of the author, the BERT model has not been applied to product vector learning yet. Using this improved model could bring better performance in basket completion tasks and extend the state-of-the-art in basket completion modeling using deep learning models from created for NLP.

3 Data

To evaluate the proposed methods in this paper, two different datasets are used. One is simulated with a known data generating process (DGP) and another is the Instacart dataset [Instacart, 2017] to test the proposed methods on empirical data. Both datasets contain information on what products are bought in a single transaction, these products bought are also referred to as a basket of products. P2V-Map only uses data on baskets of products to model market structure and does not require any customer specific data.

3.1 Simulated data

Deriving statistically sound metrics to compare different methods of analysing market structure is challenging. In empirical data the underlying market structure that is being mapped is unknown. This leads to researchers not having a proper way of identifying if the market structure mapped by a model is performing better or worse than other already existing models. Since P2V-Map only uses basket data, it is possible to simulate data in a semi-realistic way. Using models that describe category and product choice, where category is a set of products that are used in the same way, such as ice cream or coffee. This thesis uses a simulation that models a sequential process as proposed in [Neslin and Van Heerde, 2009] where first a consumer chooses a category they want to purchase a product in and then choose products to buy in that category.

This is done as follows: consumer i chooses a product j that falls into category c. This choice is made for every week t. In a category there are $J^{(c)}$ products and every product can only belong in one category. To model what categories products are purchased in by consumer i, we use the multivariate probit model as proposed in [Manchanda et al., 1999]. The model uses a latent variable $z_{i,c,t} = \gamma_c + \epsilon_{i,c,t}$ where γ is the base utility of a given category and epsilon is the error term $\epsilon_{i,c,t} \sim \text{MVN}(0, \Omega)$. Using this latent variable this thesis models category purchase $y_{i,c,t}$, where $y_{i,c,t} = 1$ if category c is chosen by consumer i in week t process as follows:

$$\begin{aligned} y_{i,c,t} &= 1 \text{ if } z_{i,c,t} > 0 \\ y_{i,c,t} &= 0 \text{ if } z_{i,c,t} \leq 0. \end{aligned} \tag{2}$$

In this model Ω denotes the correlations of all categories we simulate which, if related to consumer behavior, captures the co-occurrence of categories. So if the correlation for two categories in Ω is negative,the products within the two categories are not often bought together. Positive correlation means the opposite happens. Product choice is modeled using a multinomial probit model as described in [Chintagunta, 1992]. The utility for product j for consumer i in week t is given by $u_{i,j,t} = \alpha_{i,j}^{(c)} - \beta^{(c)} p_j^{(c)} + \epsilon_{i,j,t}^{(c)}$, where $\alpha_{i,j}^{(c)} \sim \text{MVN}(0, \Sigma^{(c)})$ denotes the base utility of product j for consumer i, $\beta^{(c)} p_j^{(c)}$ is the utility lost from paying price $p_j^{(c)}$ multiplied by a price sensitivity $\beta^{(c)}$ and finally the random error term $\epsilon_{i,j,t}^{(c)} \sim \text{MVN}(0, \sigma^{(c)})$ captures the random factors that influence purchases. $y_{i,j,t}^{(c)}$ is modelled according to the following process:

$$y_{i,j,t}^{(c)} = 1 \text{ if } u_{i,j,t}^{(c)} > u_{i,k,t}^{(c)} \; \forall \; k \neq j, \tag{3}$$

where $y_{i,j,t}^{(c)} = 1$ means that product j from category c is purchased by consumer i in week t.

Using the above model formulation the dataset simulates 20,000 consumers over a time period of 20 weeks. Each consumer chooses categories to buy products from out of 20 possible categories according to equation 1. Within the category chosen the consumer purchases the product with

the highest utility and with a 50% chance the product with the second highest utility is also purchased. Each category contains 15 products, this means a total of 300 products are simulated. Furthermore, the correlation matrix for choosing categories Ω is taken from [Gabel et al., 2019] which represents substitution and complementary category relationships. $\Sigma^{(c)}$ is calculated using $\Sigma^{(c)} = (\tau^{(c)} I^{(c)}) \Omega^{(c)} (\tau^{(c)} I^{(c)})$, where $\tau^{(c)}$ is the standard deviation of the product preferences of consumers, $I^{(c)}$ is a $J^{(c)}$ by $J^{(c)}$ identity matrix. To simulate $\Omega^{(c)}$ the simulation draws the partial correlations from a Beta(.2, 1) distribution, which gives a median partial correlation of 0.11 while maintaining 10% of the partial correlations above 0.54.

Using this simulation technique, a total of 400,000 baskets are simulated with a total of 3.7 million products, which gives an average basket size of 9.25 products.

3.2 Empirical (Instacart) data

For the empirical application of both P2V-Map and ProdBERT this thesis uses the Instacart dataset [Instacart, 2017]. This dataset contains a total of 99,574 baskets that contain 1,048,575 products in total. This means that an average of 10.5 products are purchased within a basket. The dataset contains 36,864 unique products that are purchased at least once. The majority of products (90%) are bought less than 50 times and 8,392 different products are bought only once in the dataset. The most often bought product is a banana, this product is bought 14,136 times. The distribution of categories bought in the data is displayed in Figure 1. Produce is purchased often compared to other categories, and alcohol, bulk, other, international, baby and pets products are purchased very little.

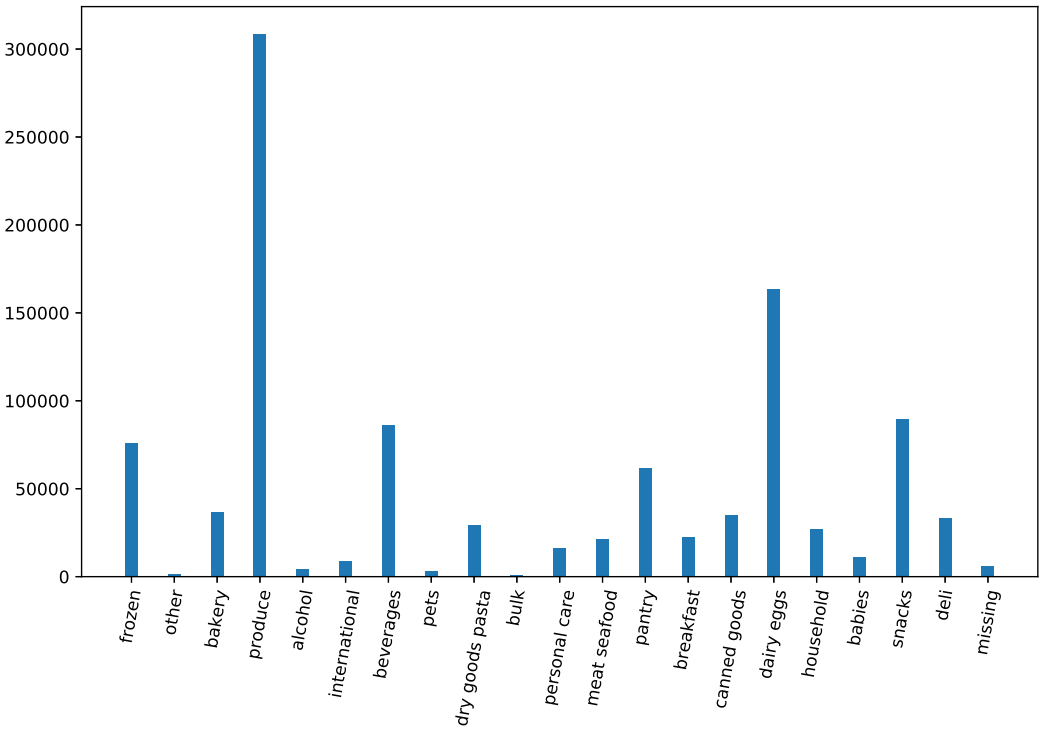


Figure 1: Distribution of number of times a category is bought.

4 Methodology

4.1 Replication of P2V-Map

P2V-Map is a method that attempts to learn product co-occurrences within baskets by using a skip-gram model and creates vectors that contain latent attributes of products. Using these product vectors, P2V-map maps these products to a 2-dimensional space. The P2V-Map mapping process consists of three steps, which are described below.

4.1.1 Data processing

To perform data processing this thesis creates a data pipeline that first reads the data given to the model and creates shopping baskets b. Shopping baskets are a collection of n_b products that are bought together in one transaction. The shopping baskets and product IDs of the products in the baskets are the only data P2V-Map needs to be able to map market structures. These baskets are then processed in three steps to make the data usable for our model.

First, the data pipeline filters rarely seen products in our data. The skip-gram cannot map proper vectors to these products because if they do not occur enough the model is unable to properly learn latent attributes of the product, thus mainly leaving it in the state it was initialised in. The minimum occurrence of a product is set to 100. Then the data pipeline builds the training samples that will be fed into the skip-gram model. Here the training samples use a format often used in NLP, center-context pairs. For a product in a basket (center product) the data pipeline creates center-context pairs with every other product in that same basket. Thus if a basket b contains 3 products: $b = \{1, 2, 3\}$ the data pipeline then creates the following center-context pairs: {{1,2}, {1,3}, {2,3}}. These center-context pairs are called positive training samples and for a basket with n_b products the data pipeline generates $n_b(n_b - 1)$ positive samples. Finally the data pipeline creates negative samples, which are center-context pairs where the context product is not in the same basket as the center product. By doing so the data pipeline created all data needed to train the skip-gram model to accurately predict if a certain product does or does not occur in the same basket. The negative samples are drawn from all products sold using a probability function with the following density: $P_n(j) = (1/Z)n_j^{pow}$ where Z is a normalization constant, n_j is the number of times a product occurs in our data and pow is a parameter that changes the shape of the distribution. The implementation of sampling proposed in [Gabel et al., 2019] additionally does not allow products within a basket to be drawn as a negative sample for any product in that basket. In total the data pipeline draws 20 negative samples per positive training sample.

4.1.2 Learning product vectors

Vectors containing values for latent attributes are learned using a skip-gram model. This model is trained under the assumption that products that are often bought together share certain characteristics and reveal information about the underlying market structure. This is done by using a skip-gram neural network, where both the center(ce) and contex(co) product are multiplied with an embedding look-up layer, which maps the products to a vector denoting latent attributes. These vectors are called v_{ce} and w_{co} with size (1xL) and (Lx1) respectively. In this thesis L = 30 for both the empirical and simulated dataset. Using these vectors the skip-gram model calculates the similarity score as follows:

$$score_{ce,co} = v_{ce} \cdot w_{co} + \beta_{ce} + \beta_{co}. \tag{4}$$

The added variables β_{ce} and β_{co} are product specific bias terms which are trained together with the embedding look-up layers. These bias terms are added to account for product frequency in the training data. By adding these bias terms an additional 2L parameters are trained. The similarity score is then passed into the sigmoid function which maps the similarity score to a probability of the center context pair occurring in a basket together. This is done as follows:

$$probability_{ce,co} = \frac{1}{1 + \exp(score_{ce,co})}. \tag{5}$$

To determine how well our neural network performs, we calculate the loss of a certain probability prediction. The loss used is binary cross-entropy loss

$$loss = -t_1 \log(\sigma(score_{ce,co})) - (1 - t_1) \log(1 - \sigma(score_{ce,co})), \tag{6}$$

where $\sigma(score_{ce,co}) = probability_{ce,co}$. t_1 is the true label of our center context pair, which is 1 if the pair is a positive sample and 0 if the pair is a negative sample.

The model is then trained on hundreds of millions of center context pairs by minimizing the loss across all training samples. This is done using the Adam optimizer [Kingma and Ba, 2014] to change the values of the embedding layers in the network in the direction that decreases the loss function the most. In doing so the model learns to predict what products are often bought together, thus learning attributes of products that are often bought together. After training v_{ce} and w_{co} become the two possible vector representations of products. [Mitra et al., 2016] proposes that the dot product of vectors of the same embedding layer ($v_{w0} \times v_{w1}$) is high for similar products and that the vector product of vectors from different embedding layers ($v_{w0} \times w_{w1}$) is high if the products are used together often. In the implementation this thesis wants to model and map products with similar attributes rather than merely modeling co-occurrences. So for the mapping of product vectors this thesis uses v_w.

4.1.3 Mapping product vectors

Using the vectors trained by our skip-gram model P2V-Map visualizes and map the products into a two-dimensional space using pairwise similarity. The mapping of products to a two-dimensional space is done using the t-SNE dimensionality reduction technique proposed in [Maaten and Hinton, 2008]. t-SNE models the relationship between points in the high dimensional space using the Gaussian distribution. In the two-dimensional space a Student-t distribution with one degree of freedom is used. These resulting two-dimensional mappings are then trained by minimizing the Kullback-Leibler divergence between the two probability distributions. One downside to t-SNE is that the optimization is non-convex, thus is not guaranteed to converge to a global minimum. To counteract this this thesis runs t-SNE 50 times and choose the run with the lowest Kullback-Leibler divergence [van der Maaten, 2014]. Before running t-SNE first the product vectors are normalized using L2-normalization.

This thesis chooses t-SNE to map product vectors to a two-dimensional space, because it solves the crowding problem observed in other dimensionality reduction techniques. The crowding problem basically means that because there is more 'space' for points to exist in higher dimensions, when you reduce the dimensionality to two it causes the points to crowd together. This crowding of points makes it hard the read the mapping and understand the learned market structure. To alleviate this, t-SNE uses the Student-t distribution to describe the points in the two-dimensional space, which has fatter tails than the Gaussian distribution.

4.1.4 Performance metrics

To analyse the performance of p2v this thesis utilises 3 different performance metrics. The first metric is the silhouette score(SIS) [Rousseeuw, 1987], which uses distances within a given cluster or category C_j:

$$\text{SIS} = \frac{1}{J}\sum_j \frac{b(j) - a(j)}{\max(b(j),\ a(j))}. \tag{7}$$

In this equation $\text{a(j)} = \frac{1}{J-1}\sum_{j\in C_j, i\neq j} d(i,j)$ which is a number that represents the average distance of product j to all other products within the same category. The second variable b(j) represents the average distance of product j to the category closest to C_j, thus $b(j) = \min_{k\neq j}\left(\frac{1}{J}\sum_{j\in C_k} d(i,j)\right)$. The SIS roughly represents how readable the map is for humans, because it is higher when the clusters are more tightly formed. The second metric used is the adjusted mutual information score (AMI) [Vinh et al., 2010].

$$AMI = \frac{MI(X,Y) - E[MI(X,Y)]}{[H(X) + H(Y)]/2 - E[MI(X,Y)]}, \tag{8}$$

where MI is the mutual information score $MI = \sum_{x\in X}\sum_{y\in Y} p(x,y)log[\frac{p(x,y)}{p(x)p(y)}]$, this indicates how correct the predictions of the categories predicted Y are compared to the true category values X. H(X) and H(y) denote the label entropies and p(x,y) is the joint probability distribution of X and Y, p(x) and p(y) are the marginal distributions of X and Y respectively. AMI has a value between 0 and 1, where if the predicted clusters align perfectly with the actual clusters the AMI is 1. The predicted clusters are found using k-means clustering. Finally the Hit rate (HR) is used

$$HR = \frac{1}{J}\sum_j f[j, c(j)], \tag{9}$$

it represents the average of what part f[j, c(j)] of the J - 1 nearest neighbours of product j belong to the same category as product j. This metric is similar to what the AMI tries to score, but does not require the mapping to be clustered using k-means.

4.2 Basket Completion task using ProdBERT

P2V-Map uses skip-gram to learn product attributes, which is a NLP method that is no longer state of the art. In NLP new deep learning methods based on Transformer models [Vaswani et al., 2017] are achieving state-of-the-art results. We will use the BERT model proposed in [Devlin et al., 2018] to learn product vectors. Although BERT is not state-of-the-art, its code is well documented and maintained better than the more advanced alternatives. Furthermore the architecture driving BERT and other state-of-the-art models is the same, thus making BERT a valuable candidate to demonstrate the usefulness of this new architecture in market analysis. Using BERT architecture we propose a new model: ProdBERT, which is trained to understand market structure and do basket completion.

4.2.1 ProdBERT input

BERT as a model requires the text to be formatted in a specific way. It uses the special tokens [CLS], [SEP] and [MASK]. [CLS] is added at the beginning of a sentence and [SEP] is added to

denote the ending of a sentence. To perform Language modeling (LM), which will be explained later, BERT randomly replaces 15% of the input products with a [MASK] token. Additionally 15 % of the tokens selected for masking are replaced with a random token within the vocabulary of products and 10% are unchanged. This is because in fine-tuning the [MASK] token is never used again, thus if ProdBERT is only trained using the [MASK] token, a disparity would appear between pre-training and fine-tuning. To create input for ProdBERT this thesis creates 'sentences' of products that represent baskets. Thus an example input for the model is: '[CLS] 24852 46667 41570 41787 27521 [MASK] 18656 [SEP]'.

4.2.2 Transformers

ProdBERT uses multiple encoder Transformer models in its architecture. The Transformer model uses multi-head scaled dot-product attention [Vaswani et al., 2017]. First, 'single-head' scaled dot-product attention is explained. Intuitively scaled dot-product attention attempts to learn what products are most important for prediction and modeling in a given basket, thus putting more weight on the products that have the most predictive power. To do so, an input sequence of products $(x_1, ..., x_n)$ is given to the model. Then for each product in the input sequence we calculate an embedding, which is a vector representation of that product with size d_e. The attention mechanism packs the embedding for each product into matrix $X_{n \times d_e}$. This matrix is then multiplied by three different weight matrices: WQ, WK and WV, with WQ $\in \mathbb{R}^{d_e \times d_k}$, WK $\in \mathbb{R}^{d_e \times d_k}$ and WV $\in \mathbb{R}^{d_e \times d_v}$. The outputs are called Query ($Q_{n \times d_k}$), Key ($K_{n \times d_k}$) and Value ($Q_{n \times d_v}$) respectively, where d_k and d_v are hyper parameters that determine the size of the Key and Value vector for one input symbol. The weight matrices that create these Query, Key and Value matrices are parameters that can be trained. Using Q, K and V, we calculate the attention score

$$\text{Attention}(Q, K, V)_{n \times d_v} = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V. \tag{10}$$

Thus for each input product x_i we have query $Q(x_i)$ and we test the compatibility with key $K(x_j)$ for every product j using the dot product of QK^T in Equation 10. If the dot product between $Q(x_i)$ and $K(x_j)$ is high we 'look up' the value of x_j, $V(x_j)$. Doing so allows the model to learn which products are important to the input basket (compatible), thus learning which products should be focused on for prediction and modeling.

In multi-head attention h different learned weight matrices WQ, WK and WV are used. For each of these weight matrices the attention function is calculated in parallel. These outputs are then concatenated into $Z_{n \times d_v \cdot h}$. This result matrix is then multiplied by a learned weight matrix W_o, where $W_o \in \mathbb{R}^{h \cdot d_v \times d_e}$, resulting in a hidden state output vector $R_{n \times d_e}$. This vector can then be passed onto another Transformer model or be fed into a language modeling layer, which will be discussed later.

4.2.3 BERT architecture

ProdBERT uses the BERT architecture. BERT uses multiple bidirectional Transformer encoders. In the implementation in [Vaswani et al., 2017] the Transformer uses both an encoder and decoder. The encoder takes an input and maps this into a higher dimensional space (vector). This vector is then fed into the decoder which uses this vector to create an output. BERT uses only the encoder output and a task specific network to function, so it only needs the encoder part of Transformers within its architecture.

In [Devlin et al., 2018] two different model sizes are proposed with different number of Transformer blocks L, self-attention heads A and size of embeddings H. $\text{BERT}_{\text{BASE}}$ with L = 12, H = 768,

A = 12, totaling 110 million parameters and $\text{BERT}_{\text{LARGE}}$ with L = 24, H = 1024, A = 16, totaling 340 million parameters. For ProdBERT this thesis uses $\text{BERT}_{\text{BASE}}$, because of the relatively small amount of empirical data available and the possible computational costs involved in training a large model like $\text{BERT}_{\text{LARGE}}$. The computational cost is not an issue for this thesis given our current (small) dataset, but in applications with more data it might prove problematic.

4.2.4 Pre-training ProdBERT

For NLP, BERT is trained in two phases, pre-training and fine-tuning. ProdBERT only uses the pre-training part of the BERT architecture. This is by simultaneously training ProdBERT on two tasks: Masked Language Modeling (MLM) and Next Sentence Prediction (NSP).

Language modeling in ProdBERT is trying to learn a probability distribution of products given a certain context. Traditionally language models use AR language modeling. Using a given product sequence of products x = $[x_1, ..., x_T]$ AR modeling performs pre-training by maximizing the likelihood of a certain product given all predecessors:

$$\max_{\theta} \ \log p_\theta(\text{x}) = \sum_{t=1}^{T} \log p_\theta(x_t|x_{<t}) = \sum_{t=1}^{T} \log \frac{\exp(h_\theta(x_{1:t-1})^T e(x_t))}{\sum_{x'} exp(h_\theta(x_{1:t-1})^T e(x'))}, \tag{11}$$

where $e(x)$ is the embedding for input x and $h_\theta(x_{1:t-1})$ is the context representation produced by neural models. In our case if ProdBERT used AR language modeling, this would be the output of the last transformer encoder in ProdBERT. However this problem definition does not allow for bidirectional language modeling, since if for the first product the likelihood is calculated given all other products in the basket, the model can basically 'see' what the next product should be if the likelihood for the second product is calculated.

ProdBERT is able to use bidirectional language modeling, because it is based on denoising auto-encoding. It does so by creating a corrupted input $\hat{x}$ from input x by replacing 15% of the input products with the special symbol [MASK]. We will denote these masked products as $\bar{x}$. The goal in training is then to reconstruct $\bar{x}$ from the corrupted input $\hat{x}$:

$$\max_{\theta} \ \log p_\theta(\bar{x}|\hat{x}) = \sum_{t=1}^{T} m_t \log \frac{\exp(H_\theta(\hat{x}_t)^T e(x_t))}{\sum_{x'} exp(H_\theta(\hat{x})_t^T e(x'))}, \tag{12}$$

where H_θ is the output Transformer that maps an input sequence x with length T to hidden state vectors $H_\theta(x_t) = [H_\theta(x)_1, H_\theta(x)_2, \ldots, H_\theta(x)_T]$ and $m_t = 1$ if x_t is masked. This problem definition allows ProdBERT to use the full basket of a given product to predict what product was originally masked, achieving bidirectionality.

The next step in pre-training is NSP. To allow ProdBERT to better understand basket structure ProdBERT splits every basket in two. These two parts of the basket acts as sentences, thus a given part of a basket is only the next basket part to the input basket if they together form the original basket that was split in two. In training, 50% of the time a random basket part is given and should be identified as not the next sentence. The other 50 % of the time the actual next basket part is given, which ProdBERT has to correctly identify as such.

4.2.5 Comparing P2V-Map to ProdBERT

The product vectors obtained from ProdBERT can not easily be visualized, since for each basket x and corrupted input $\hat{x}$ the hidden state vectors created by the Transformer encoders depend on the products in the entire basket. This is useful for NLP where the word 'bank' has a different meaning

in: 'I am going to rob a bank' and 'You should fish on the river bank'. This context dependence allows ProdBERT to understand the context a product is used in and what function that product has in a given basket. The downside is that extracting a product vector representing latent attributes is difficult since that would involve pooling all different vectors for the same product. This thesis attempted to averaging these different vectors for the same product, but in doing so information is lost. This caused the vectors to be just marginally better than random vectors. To still enable comparison between the learned product vectors this thesis will make both models complete baskets where one product is removed or for ProdBERT masked. Completing baskets requires knowledge of what products go together well and what kind of products are already in the basket that needs to be completed. This makes basket completion a viable method for comparing results between two models that learn market structure and product attributes.

Since standard P2V-Map as a method does not do basket completion, we use both the average and maximum co-occurrence scores between the products in the basket and a given product within our vocabulary of products. Then for each product within our vocabulary we compare the maximum and average co-occurrence to a given basket and select the product within our vocabulary with the highest score. The accuracy is then calculated for both BERT and P2V-Map on 10% of the original dataset that was not used to train both models. The accuracy is used as an indication of how well the market structure is learned by both models.

4.2.6 Empirical application basket completion

An empirical application for basket completion is product recommendation for online retailers based on products already put in the shopping cart. For both P2V-Map and ProdBERT we will select the 5 products with the highest prediction scores of all products to fill the basket with. Although the quality of these recommendations aren't easily measured, this thesis will compare P2V-Map and ProdBERT recommendations made for a select few baskets by hand.

5 Results

This Section will first discuss the results obtained from our replication of P2V-Map as proposed in [Gabel et al., 2019], then the results for basket completion using ProdBERT are presented and discussed.

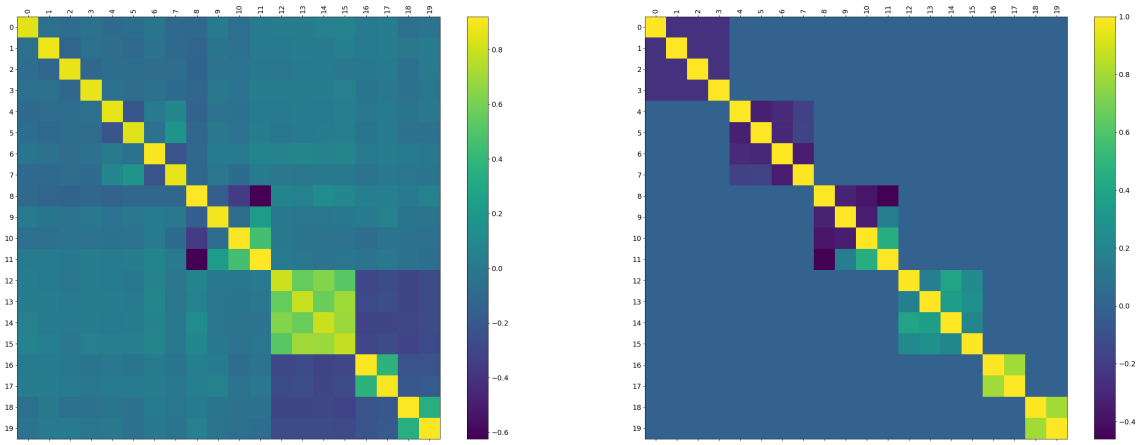
5.1 P2V-Map replication

For the replication of P2V-Map the 2 datasets as described in Section 3 are used. We will first discuss the performance of our implementation of P2V-Map in mapping known data generating processes using performance metrics and comparing correlation matrices. Then we will evaluate the performance of our implementation of P2V-Map on empirical data using the Instacart dataset.

5.1.1 Simulated data

After processing the simulated data, there are over 730 million center context pairs generated. We use a batch size of 2016 center context pairs and train the model on the whole dataset. We do not split the dataset into test and validation datasets, because p2v is a method where the prediction performance of the model is not of importance. The only important part of the network is its learned product vectors extracted from the embedding layer. The model trained for 5 epochs in 15 hours on an Intel i5-4690K.

To analyze how well the product vectors map similarity and co-occurrence between clusters, we first calculate the average vector within each cluster. We then find the dot product between all of the category pairs. Doing so allows us to create a correlation like matrix learned by the p2v model, which can be found in Figure 2a. The similarity matrix matches our category level correlation matrix in 2b reasonably well, which seems to indicate that the p2v implementation is able to capture correlations between categories.



(a) (b)

Figure 2: Comparison between the matrix of dot products between the average product vector in each category and the actual correlation matrix used to simulate the data.

The performance metrics SIS, AMI and HR are calculated on 3 bootstraps of simulated data, for both the context and center product vectors and random vectors initialized using a random normal variable with the same mean and standard deviation as the vectors produced by the simulated data. These results and the results reported in the original P2V-Map paper are found in Table 1 below. Our implementation seems to perform much better, we hypothesise that this difference is caused by our implementation only using one center context pair at the time. In the original implementation the model processes 1 positive center context pairs and 20 negative center context pairs simultaneously. Another possible explanation is that the simulated data we use is not simulated in the same way as the original paper, because of slight errors in code.

Table 1: P2V-Map metrics.

Method	Mapping Input	SIL	AMI	HR
P2V-Map	Center Embedding	0.894 (0.017)	0.995 (0.007)	0.983 (0.011)
P2V-Map	Context Embedding	0.887 (0.023)	0.987 (0.012)	0.979 (0.014)
P2V-Map(original)	Center Embedding	0.350 (0.004)	0.823 (0.003)	0.696 (0.003)
Random	-	-0.192 (0.011)	0.000 (0.008)	0.051 (0.002)

In Figure 3 the simulated product mappings of our first bootstrap are shown. As seen in the figure our model is able to very accurately capture the category of a product within its product vectors. Furthermore, the products within category 13, 14 and 15 are put close to each other, this

matches with the correlation matrix where these products are positively correlated. In the top right corner the categories 18 and 19 are found close to each other. However since the correlation between these categories are 0.8 it would be expected that they are closer to each other. This same phenomenon is also seen for product categories 16 and 17 in the middle. As also demonstrated by the very high performance metrics, the mapping of the simulated data is almost 'too' perfect. Our implementation seems to perform much better than the paper this method was introduced in. This might be caused by an error in the simulation of data that causes our product vector learning task to be trivial.

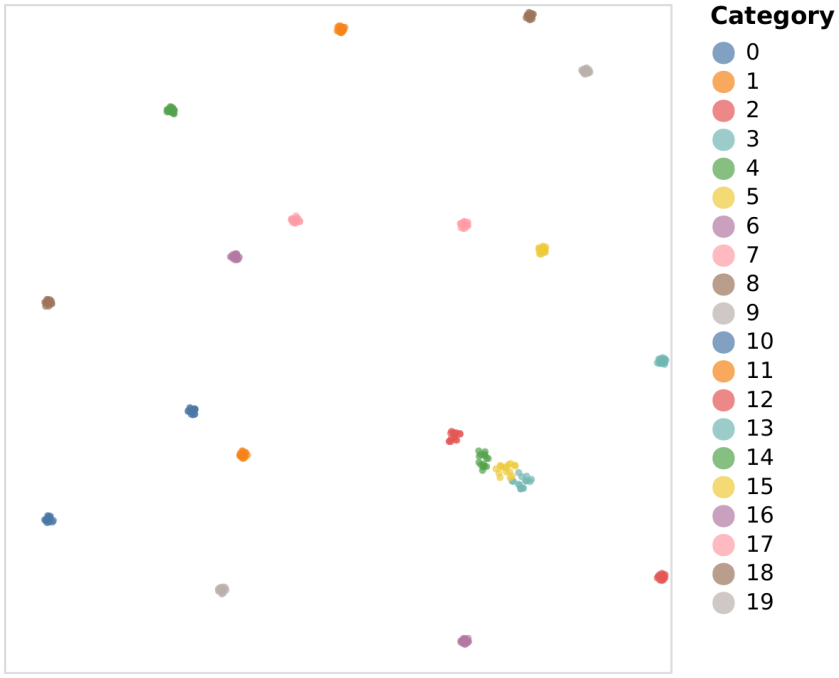


Figure 3: Product mapping of simulated data.

5.1.2 Instacart data

After processing the data from the Instacart dataset we create 244 million center context pairs.The model is trained on the entire Instacart dataset with a batch size of 2016, for the same reason as mentioned in the simulated data results. Training was done over 5 epochs and took around 8 hours on an Intel i5-4690K.

After mapping the products to a two dimensional space, the product map seen in Figure 4 is generated. Each colour of the dots corresponds to a category in the data. There is some clear clustering seen in the plot, but because there are so many products and limited space true clusters are hard to see. We will elaborate on the products within the clusters marked by a black and green ellipses.

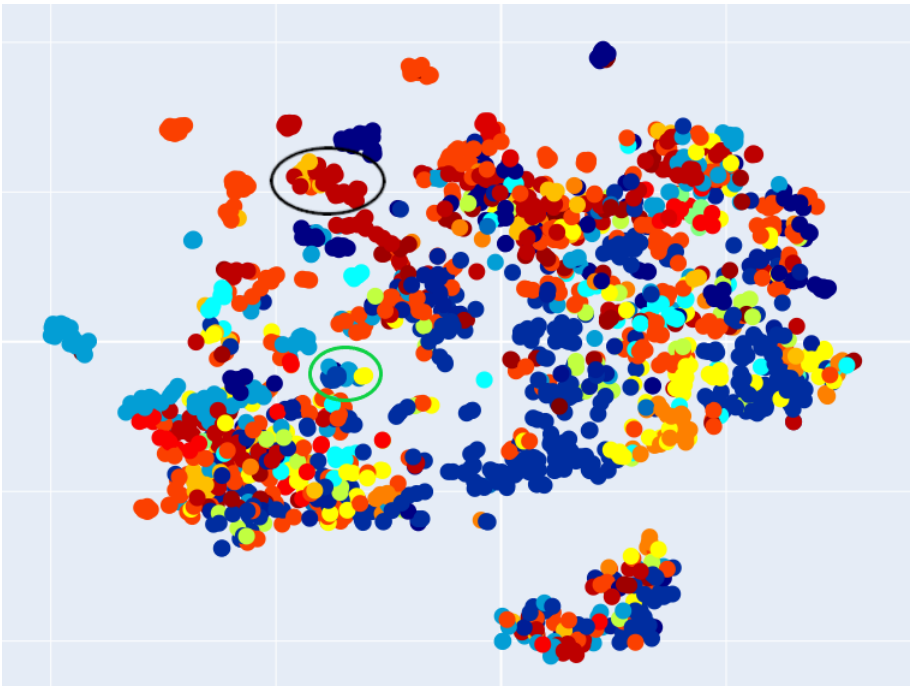
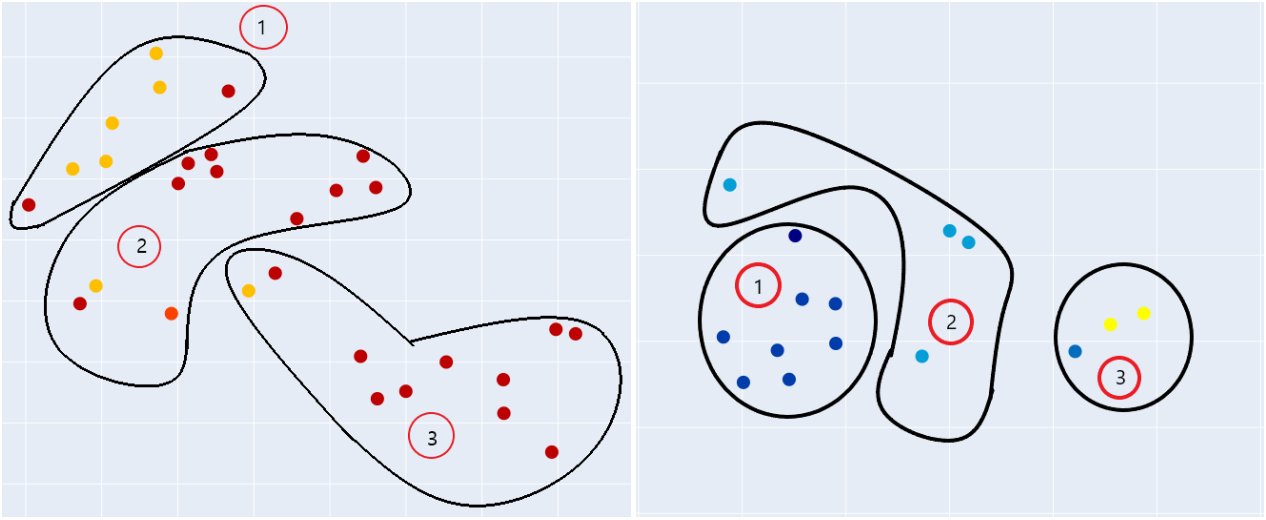


Figure 4: Product mapping of Instacart data

First we examine the clusters and products within the black circle. In figure 5a we see that within the black circle 3 sub clusters can be made. We do so by analysing the products by hand and putting products that belong together in one cluster regardless of shape. Cluster 1 mainly contains various cereals and granola's, products that are within this cluster are for example Organic Cinnamon Harvest Cereal and Vanilla Blueberry Clusters With Flax Seeds Granola. The second cluster mainly contains various granola and protein bars. While the content of the product is very similar, our model identified that there is a slight difference between them. Examples of products that are in this cluster are: Gluten Free Dark Chocolate Chunk Chewy with a Crunch Granola Bars and Oats Honey Gluten Free Granola. Finally the third sub-cluster contains mainly savory snacks like Boomchickapop Sea Salt Popcorn , Pita Chips Simply Naked and Gluten Free Pretzel Sticks. Using these sub clusters we find that our implementation of P2V-MAP is able to identify similar products and can differentiate between different forms of similar products. The cluster within the green circle can be found in figure 5b. In this cluster there are also 3 sub clusters found, the first sub cluster contains mostly wines and some miscellaneous alcoholic beverages, such as Chardonnay and India Pale Ale. Cluster 2 contains non-alcoholic beverages such as Club Soda. Finally, cluster 3 is filled

with various hot sauces, one example is chili sauce. Although the cluster in the green circle is less defined in terms of the kinds of products that are in it, still P2V-Map is able to separate non similar products reasonably well.



(a) Cluster within the black circle

(b) Cluster within the green circle

Figure 5: Analysing small parts of p2v mapping using clusters made by analysing the product within each cluster by hand.

5.2 ProdBERT basket completion

The ProdBERT model was trained on 75,954 baskets, creating a total of 151,908 basket halves. These baskets contain 589,234 products in total. During training a maximum sequence length of 28 is used with a batch size of 64, thus per training step 28 * 64 = 1,792 tokens are processed. The model performs 20,000 training steps where 1,792 tokens are processed per training step, which is roughly equal to 60 epochs. Training was done on one Cloud TPU and took 10 minutes.

5.2.1 Prediction results in basket completion task

The results are found in Table 2. This includes a baseline prediction metric, this method only predicts the product used most often within the data. In the Instacart dataset this is a banana. The accuracy and standard deviation of the validation data were obtained by running the validation ten times. This is not done for training, because training performance is just an indication of model functioning and potential overfitting. We find that ProdBERT performs significantly better than both P2V-Map and the baseline. P2V-Map performs worse than the baseline. One explanation for this is that although the vectors should contain information on market structure, the model is not explicitly trained for basket completion. Another possible explanation is that the P2V-Map vectors are not able to capture the context of a basket and how certain items being in a basket affect what product is most likely to be purchased with it. The accuracy of all methods could be seen as low, but considering there are over 3,000 products within our dataset ProdBERT shows it is able to understand baskets of products well.

Table 2: Comparing accuracy of P2V-Map to ProdBERT in the basket completion task.

Method	Training data	Validation data
P2V-Map (Average)	0.011	0.010 (0.0011)
P2V-Map (Max)	0.011	0.011 (0.0007)
BERT	0.137	0.135 (0.0055)
Baseline	0.023	0.021 (0.0014)

5.2.2 Empirical application

For our empirical application we will look at three different baskets and compare the BERT predictions with the predictions made by P2V-Map using the 'max' method. The first basket we highlight contains the following items: {Organic Ginger Root, Orange Bell Pepper, Organic Bell Pepper, Organic Red Bell Pepper, Organic Coconut Milk, Organic Kale Greens, Limes, Organic Black Beans, Large Lemon} with missing product {Organic Red Onion }. This basket is categorised by its many vegetables and organic products. The second basket contains {Root Beer, Cream Top Smooth & Creamy Maple Yogurt, 2% Reduced Fat DHA Omega-3 Reduced Fat Milk} with missing product {Spring Water} , this contains drinks and dairy. The final basket we will look at contains {Penne Rigate, Marinara Pasta Sauce, Organic Shredded Mozzarella,Organic Red Bell Pepper, Grated Parmesan} with missing product {Organic Yellow Onion}, this basket is clearly purchased to make pasta with tomato sauce.

The predictions made by ProdBERT for the organic vegetable basket are {Organic Red Onion, Organic Cilantro, Green Bell Pepper, Organic Garlic, Yellow Bell Pepper} and by P2V-Map {Butternut Squash, Brussels Sprouts, Bunched Cilantro, Jalapeno Peppers, Baby Bok Choy}. ProdBERT performs better here, identifying both the type of vegetables in this basket and suggesting

mainly organic products. P2V-Map's predictions seem off, because the type of vegetable does not seem to match and there are no organic products recommended.

For the basket containing drinks and dairy the predictions by ProdBERT were {Banana, Bag of Organic Bananas, Organic Whole Milk, Half & Half, Organic Avocado} and the predictions by P2V-Map are {Pulp Free Orange Juice, Natural Spring Water, Vanilla Ice Cream, Sugar Free Energy Drink, Light Semisoft Cheese}. For this basket P2V-Map seems to better capture both the dairy and drinks aspect of the basket, even suggesting a product very similar to the missing product. ProdBERT on the other hand seems lost in what it should recommend and just recommended the product most often purchased.

The predictions made by ProdBERT for the pasta basket are as follows: [Organic Bell Pepper, Green Bell Pepper, Organic Strawberries, Organic Yellow Onion, Organic Baby Spinach}, the predictions by P2V-Map are {Organic Green Beans, Bag of Organic Bananas, Brussels Sprouts, Canola Oil, Taboule Salad}. Here ProdBERT is able to identify that this basket requires vegetables often used in pasta and one of the suggestions is the missing product. P2V-Map's recommendations are very generic and seem to be unable to understand what these products mean together. This supports this thesis's theory that the product vectors are unable to capture relationships between products within a given basket.

6 Conclusion

This thesis first replicates the methods proposed in [Gabel et al., 2019] to map products to a 2-dimensional space for market analysis. This thesis found that the implementation works well on the simulated data, but produces significantly different results than the paper this thesis replicates. A potential explanation is that this might be due to difference in the implementation of P2V-Map between the original paper and this thesis. In the empirical application we show that this thesis's P2V-Map implementation is able to capture different clusters of products well and in an intuitive way. This thesis extends the literature on product mapping by introducing ProdBERT, a model that uses BERT to learn product representations. This thesis uses ProdBERT to model products, because of its performance in NLP and its ease of use. ProdBERT is not suited for product mapping as described in the P2V-Map paper, so to still compare the performance of P2V-Map and ProdBERT this thesis uses basket completion as the evaluation task. This thesis shows that ProdBERT significantly outperforms both the baseline of choosing the most frequently occurring product and an algorithm based on P2V-Map vectors.

For further research, several avenues can be explored. First of all, BERT is no longer considered fully state-of-the-art, a model called XLNet proposed in [Yang et al., 2019] outperforms BERT in most tasks and is able to remedy certain disadvantages of BERT's masking of input, while retaining the deep bi-directionality. Besides its better performance, it also has another significant advantage for this task. The model is able to do AR language modeling as explained in the methodology. This means that it is more suited for generating product recommendations based on products already added to a given basket. Another avenue for improvement of the results in this thesis is to use a larger dataset. The current dataset size pales in comparison to the size of the BERT model's trainable parameters. Finally, more personal recommendations could be achieved by also feeding consumer specific data into the model, like purchase records, age and gender.

In conclusion, this thesis added the application of more sophisticated deep-learning methods to the field of marketing. In doing so it improved on existing results and paved the way to using more fitting models on the same task.

References

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

[Devlin et al., 2018] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

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

[Grbovic et al., 2015] Grbovic, M., Radosavljevic, V., Djuric, N., Bhamidipati, N., Savla, J., Bhagwan, V., and Sharp, D. (2015). E-commerce in your inbox: Product recommendations at scale. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1809–1818.

[Hinz and Eckert, 2010] Hinz, O. and Eckert, J. (2010). The impact of search and recommendation systems on sales in electronic commerce. *Business & Information Systems Engineering*, 2(2):67–77.

[Instacart, 2017] Instacart (2017). The Instacart Online Grocery Shopping Dataset. https://www.instacart.com/datasets/grocery-shopping-2017.

[Kingma and Ba, 2014] Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[Maaten and Hinton, 2008] Maaten, L. v. d. and Hinton, G. (2008). Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605.

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

[Mikolov et al., 2013] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

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

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

[Radford et al., 2018] Radford, A., Narasimhan, K., Salimans, T., and Sutskever, I. (2018). Improving language understanding by generative pre-training.

[Rousseeuw, 1987] 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.

[van der Maaten, 2014] van der Maaten, L. (2014). Accelerating t-sne using tree-based algorithms. *Journal of Machine Learning Research*, 15(93):3221–3245.

[Vaswani et al., 2017] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

[Vinh et al., 2010] Vinh, N. X., Epps, J., and 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.

[Yang et al., 2019] Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R. R., and Le, Q. V. (2019). Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in neural information processing systems*, pages 5753–5763.

A Description of code

A.1 Basket_completion.py

Requires: Trained BERT model, vocab.txt file, bertconfig.json file and text file containing baskets to test the model on.
Execution: This script loads a trained ProdBERT model and iteratively load one basket of products and masks a random product within that basket. It then prints the five products with the highest probability of being the masked product. It also performs predictions using P2V-Map and the baseline strategy of predicting the most frequent product.

A.2 data_pipeline.py

Requires: Raw data in excel file.
Execution: Processes the data in the three steps outlined in Section 4. It then saves the processed data to disk so other scripts can use the processed data.

A.3 data_simulation

Requires: A correlation matrix for selection of categories.
Execution: Creates 20,000 consumer objects and simulates their behavior as described in Section 3. It then saves the data in excel files.

A.4 helper_functions.py

Requires: -
Execution: Not executed, it just contains various functions used throughout the project to perform certain tasks.

A.5 input_bert

Requires: Raw data in excel file.
Execution: Processes the data using the class in data_pipeline.py and shapes it into basket halves and writes it to a text file that can be used to train ProdBERT with. It also writes the vocabulary file needed for training.

A.6 performancemetrics.py

Requires: Trained P2V-Map weights, (optional) empirical product information and processed basket data.
Execution: Calculates performance metrics for P2V-Map.

A.7 skipgram_model.py

Requires: Processed basket data.
Execution: Performs training of the skip-gram model within P2V-Map and saves the weights to disk to perform analysis on. This takes between eight and fifteen hours based on what dataset is used (Instacart vs Simulated).

A.8 vector_mapping.py

Requires: Trained P2V-Map weights, (optional) empirical product information and processed basket data.
Execution: Calculates the tsne-embeddings and creates an html file that can be used to view the plot interactively.

A.9 Prodbert.ipynb

Requires: Files created by input_bert.py and access to google cloud services.
Execution: First converts the raw text input to data in a format that ProdBERT can use for training. It then pre-trains the model and returns: training loss, accuracy on language modeling and accuracy on next sentence prediction. It finally also validates the model on validation data that was not used to train it, thus doing basket completion.