TI 2025-071/XII
Tinbergen Institute Discussion Paper

What Did I Forget? Basket Analysis for Large Assortments Using Transformers

Luuk van Maasakkers[1]
Bas Donkers[2]
Dennis Fok[3]

1 Erasmus University Rotterdam

2 Erasmus University Rotterdam, Tinbergen Insitute

3 Erasmus University Rotterdam, Tinbergen Insitute

What Did I Forget? Basket Analysis for Large Assortments Using Transformers

Luuk van Maasakkers, Bas Donkers, Dennis Fok
Erasmus University Rotterdam

December 10, 2025

Abstract

We propose a new method for learning product complementarity patterns in shopping baskets, inspired by Google's Bidirectional Encoder Representations from Transformers (BERT) for natural language processing. We reformulate BERT's masked learning task in a marketing context and learn to accurately identify missing products from a real-life grocery shopping basket based on the other products purchased in that same basket. The resulting model, which we call BaskERT, can be used by retailers for personalized product recommendations and for analyzing product complementarity patterns across the assortment. BaskERT outperforms several state-of-the-art benchmarks in a basket completion task. Different procedures for sampling the missing product during training impact the variety of recommendations returned by the model. This enables marketers to steer their recommendations away from the most popular products. The model is easily scalable to large assortments. As our model only requires basket data from the current shopping trip, it is applicable in many situations, also when customer information and purchase history data are not available, for example because of privacy regulations.

Acknowledgements
This work was carried out on the Dutch national e-infrastructure with the support of SURF Cooperative.

Contents

1 Introduction

Retailers nowadays have access to large amounts of purchase data of their customers. Sometimes this data contains customer-specific identifiers allowing one to match purchases over time to the same customer, but often such identifiers are not available. With the rise of data availability, the demand for automated models that can translate these data into meaningful marketing insights has also risen (Wedel & Kannan, 2016). For example, complementarity patterns across products and product categories can be used by retailers to optimize their assortment layout and serve customers with meaningful recommendations, thereby increasing sales and customer satisfaction.

In this paper, we propose a model that is designed for basket completion in the context where one only observes 'single baskets', that is, customer identifiers are not given. Given the set of products a customer has already added to their shopping basket, the model aims to predict which other product would also belong to the basket. Consequently, our model can be used to recommend products to users that would fit well within their basket, and that they might have forgotten during their shopping trip. For the model to be successful, it needs to learn the purchase patterns underlying purchase behavior. To do so, the model describes each product in the assortment through a vector of latent attributes, also known as an embedding. In the grocery shopping setting that we study, these embeddings will hence need to contain information on relevant product features, e.g. whether the product is organic or not, but also substitution and complementarity patterns among the products in the assortment need to be learned.

We call our model Basket Encoder Representations from Transformers (BaskERT), as it is inspired by Google's Bidirectional Encoder Representations from Transformers (BERT, Devlin et al. (2018)). Originally, BERT was developed for natural language processing (NLP) tasks such as question answering and text classification. We use the analogy between sentences comprised of words and baskets comprised of products to transfer the architecture and learning task of BERT to a marketing context. This analogy has been succesfully used in the past to transfer other methods from NLP to marketing (Jacobs et al., 2016; Gabel et al., 2019).

The strong performance of BERT (and BaskERT) can be attributed to its unique attention mechanism (Vaswani et al., 2017). This allows the model to adaptively put more weight on certain informative words (products) in a sentence (basket) when predicting a missing word (product). This is a desirable feature, as purchased products do not need to be complementary to all other products in the same basket, but are likely to match at least a few. The attention mechanism gives BaskERT the flexibility to deal with these cases and provide accurate recommendations accordingly.

Another major advantage of using BERT in marketing is that its original training task is extremely similar to the task of predicting the missing product from a basket: one or multiple words from a text are left out ("masked") and the model's task is to accurately recover these masked words based on the remainder of the sentence. We translate this task into masked product modelling (MPM), where the model's task is to recover a product that is left out from an originally observed basket. The predictions from BaskERT can hence be directly used to rank products based on their fit with the other products in the basket. The selection of products to be predicted in the training task provides opportunities for marketers to influence the model's predictions and subsequent recommendations. While popular products are often accurate predictions, these products might be perceived as somewhat boring recommendations. By undersampling the frequently purchased products and oversampling products that are purchased less often, BaskERT can generate predictions that are more balanced across all products, which can create value for retailers and consumers (Brynjolfsson et al., 2006).

In contrast to the original BERT framework (Devlin et al., 2018), we do not take any sequential ordering in the basket into account, as we believe that the order in which products are added to a basket is strongly influenced by the assortment layout and not as much by complementarity and substitution patterns across products. Taking order into account would favor recommending products from the same aisle or subpage, which could bias predictions towards the original (web) store layout of the retailer.

The proposed model only uses information from the current shopping trip to generate basket predictions. No past purchase data or personal customer information is required, which makes the model very generally applicable. Although past purchase behavior of a customer could contain useful information to make more accurate product recommendations, this information is not always available for retailers. For example, a customer could make an offline purchase without needing to identify him-/herself, or make an online purchase as an anonymous guest. Besides that, privacy regulations could restrict a retailer from storing or using personal information, even when it is available. However, BaskERT provides enough flexibility to be extended with additional input variables, if available. Such additional information could also be whether a product was recommended or not. While accurate predictions are helpful in making relevant recommendations, they are not necessarily the most impactful recommendations (Bodapati, 2008). In this paper we will show the predictive accuracy of BaskERT, but cannot establish the impact of recommending the products that fit well with the basket. In case such data is available, BaskERT can also be extended to capture the impact of recommending a product on purchase behavior.

In the next section, we review related literature. Next, we introduce our model in detail. We then use data from an online retailer (*The Instacart Online Grocery Shopping Dataset*,

2017) to evaluate our model's performance on a basket completion task and we contrast it with several benchmarks. We also investigate the impact of model hyperparameters and specific model components on predictive performance. Finally, we demonstrate how the learned embeddings from the model can be used to further analyze the retailer's assortment.

2 Literature review

Several models have been proposed to learn latent product attributes that capture co-occurrence patterns between products, and make purchase predictions accordingly. Yu et al. (2016) proposed a recurrent neural network for next basket prediction (DREAM). Based on the entire purchase history of a customer, the authors aim to predict the full content of the customer's next basket. Ruiz et al. (2020) proposed SHOPPER, a model designed to predict the next item in a basket. Although these two models are different in aim and design, they are similar in the sense that products are represented by learnable, latent embeddings. Next to that, products in a basket are pooled by taking the average or maximum of their embeddings. This simple aggregation implies that all products in a basket are equally important for the downstream task, i.e. predicting the next product or basket. In practice however, a well matching product does not have to complement all of the products in a basket equally. Instead, it often strongly complements a few products and might not be related to others. Therefore, our proposed method replaces these simple aggregation methods with richer representations of interactions via multi-headed attention mechanisms (Vaswani et al., 2017). We also demonstrate that the flexible aggregation operations of BERT lead to better performance than simply averaging product embeddings as in DREAM or SHOPPER.

Gabel et al. (2019) show that learned product embeddings can be used to visualize assortment structures in a low-dimensional map. First, high-dimensional product embeddings are learned with a skip-gram network called Product2Vec (P2V), inspired on Word2Vec (Mikolov et al., 2013). Although skip-gram is traditionally used in natural language processing to learning word representations, Gabel et al. (2019) adapt it to distinguish co-occurring pairs of products in a basket from pairs of products that are not purchased together. Next, the resulting embeddings are reduced into a two-dimensional space using the t-distributed stochastic neighbor embedding (t-SNE) algorithm (Van der Maaten & Hinton, 2008). BaskERT generalizes P2V, as it considers the whole set of co-purchased items, not just product pairs. We demonstrate that BaskERT's predictions outperform predictions resulting from P2V's embeddings. We also find that a t-SNE-based visualization of the BaskERT embeddings is similar to the one based on P2V, but that it better aligns with observed product characteristics.

The BERT model has been applied before by Sun et al. (2019) and Bianchi et al. (2020) to predict purchases, with promising results. Both papers demonstrate the strong performance of BERT in a sequential prediction task. They explicitly take the purchase order of products into account, while this is more informative about store layout than customer preferences. In contrast, we adapt the BERT architecture to not use this order information and learn product attributes solely based on co-occurrence patterns. This makes it easier to extract and visualize complementarity patterns from the learned embeddings. Besides, we do not only focus on BaskERT's performance, but also on the interpretation of its predictions and product embeddings. Moreover, we extend the sampling procedure in the masking task in order to shift predictions across the accuracy-variety domain. Finally, we visualize the assortment structure by mapping product embeddings in a low-dimensional space.

3 Model

The BaskERT model is a variant of Google's BERT model (Devlin et al., 2018) trained on a basket completion task. A basket is defined as a set of products, purchased by a customer in a single shopping trip out of an assortment of N products. In our application, N is in the order of several thousands, but the model is expected to scale well to larger assortments. Basket i can be represented by a set of product id's, $\mathcal{B}_i = \{b_{i1}, b_{i2}, ..., b_{in_i}\}$, where n_i is the number of products in basket i. For example, $\mathcal{B}_i = \{123, 4022, 6791\}$ represents a basket containing three products, with respective (unique) product id's 123, 4022 and 6791. If the same product is purchased multiple times in one basket, we include it only once. In the next section, we describe the preprocessing step of masking products. Afterwards, we explain the details of BaskERT's model architecture.

3.1 Creating the prediction task

The aim of our model is to predict a product in a basket, given the other products in the basket. For this purpose we randomly remove one product from a basket which we aim to predict based on the remaining content of the basket. This removal process is called masking. Once we have selected a product to be masked, we replace its product id by a [MASK] token. The resulting masked basket is denoted by $\tilde{\mathcal{B}}_i$. A potential masked version of the basket above is $\tilde{\mathcal{B}}_i = \{123, [\text{MASK}], 6791\}$. The original identifier of the masked product is represented by m_i ($m_i = 4022$ in the example above). The model's task is to recover the original item m_i given the masked basket $\tilde{\mathcal{B}}_i$.

The way in which we sample the masked product influences the type of patterns the model will learn. If products in a given basket are masked with equal probability, frequently occurring products will be masked more often in total. This results in a model

that has a tendency to predict highly popular products. For a marketer, this means that the recommendation is often 'right', but not always because the product is the best to fit in the basket. Instead, the recommendation is often 'right' because it recommends a generally popular product. This is not always desirable, as it harms the variation in recommendations to customers, as the same, popular products are frequently predicted. To steer the model's predictions more towards complementarity patterns, instead of favoring popular items, we implement a second sampling procedure for the masking task that samples popular products less often. Marketers can select either sampling procedure, based on the context and purpose of the recommendation system.

To be precise, we consider the following two sampling procedures for selecting the masked product:

1. A uniform sampling procedure, where each product j in the basket has an equal probability of being masked. Formally,

$$\Pr[m_i = j] = \begin{cases} \frac{1}{|\mathcal{B}_i|} & \text{if } j \in \mathcal{B}_i, \\ 0 & \text{if } j \notin \mathcal{B}_i. \end{cases} \tag{1}$$

2. A weighted sampling procedure, where each product in the basket has a probability of being masked inversely proportional to its purchase frequency in the training set. Formally,

$$\Pr[m_i = j] \propto \begin{cases} \frac{1}{f_j} & \text{if } j \in \mathcal{B}_i, \\ 0 & \text{if } j \notin \mathcal{B}_i, \end{cases} \tag{2}$$

   where f_j is the number of times product j is purchased in the training set. In this second procedure, the expected number of times a product is masked over all baskets is equal for all products. As a result, the model can only learn from product co-occurrence patterns and can not rely on product popularity when trying to recover a masked product. As we will demonstrate with our results, this approach makes the model less accurate, but leads to more variety in the recommendations. Besides that, it ensures that all learned latent product attributes represent complementarity and substitution patterns (and not the product's popularity), making it easier to isolate and analyze these patterns.

Generally, we can say that the masked product is selected with probabilities proportional to $f_j^{-\alpha}$, with $\alpha = 0$ denoting the first sampling procedure and $\alpha = 1$ denoting the second sampling procedure. Values of α between 0 and 1 result in a trade-off between accuracy of and variety in the model's predictions.

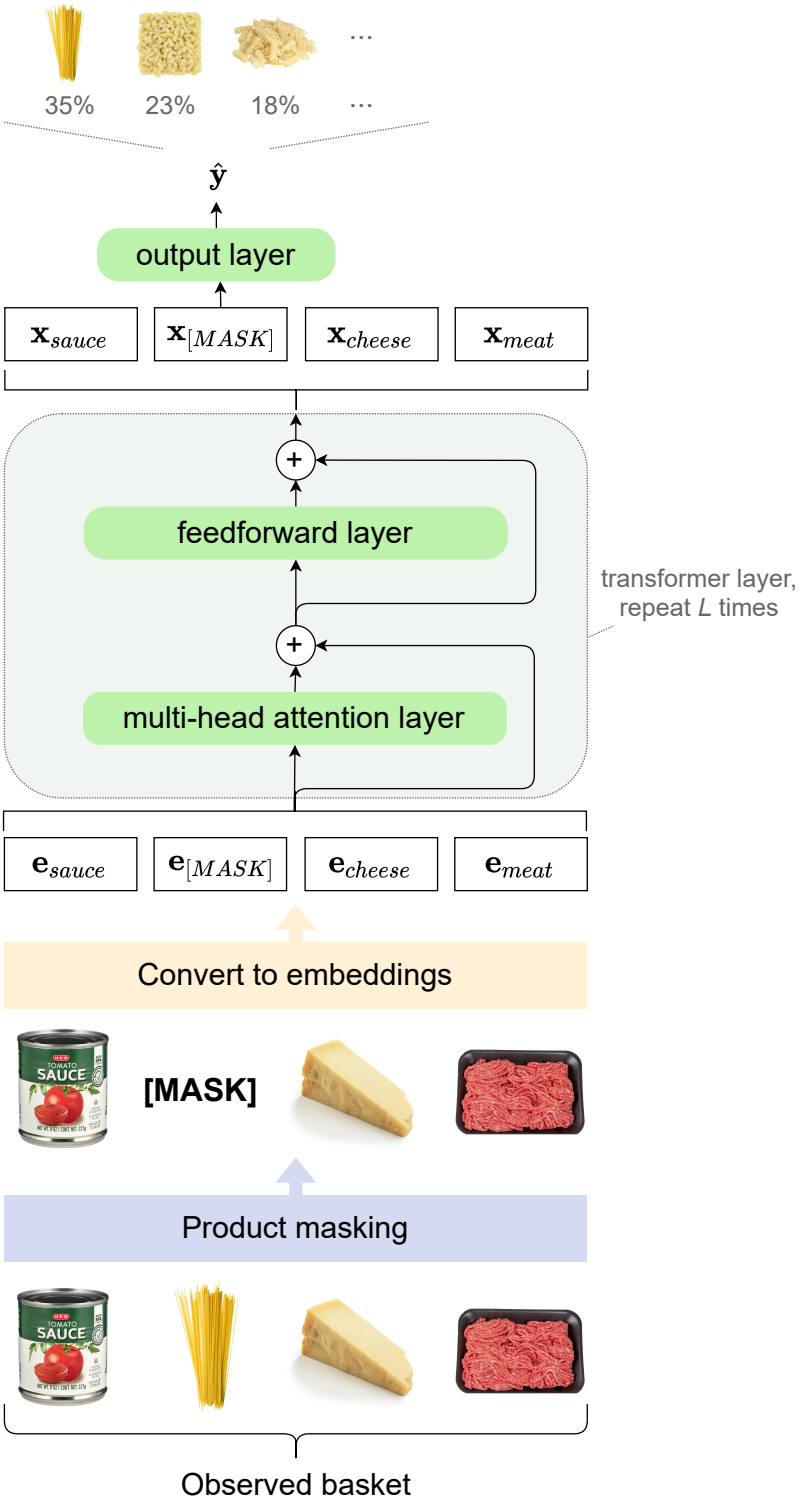

Figure 1: Outline of the BaskERT model with an example of a basket with four products. First, one product is masked, after which the products in the masked basket are translated into embeddings. After passing the embeddings through L transformer layers, the model outputs an estimated probability distribution for the masked product over the entire assortment.

3.2 Product embeddings

To be able to find complementarity patterns between products and make predictions accordingly, each product is described by a latent parameter vector, known as embedding. This latent vector can implicitly capture product attributes that are reflected by co-purchase patterns in basket data, such as whether a product is organic or not, vegan or not, healthy or unhealthy, etc. Together, the product embeddings form the core of BaskERT. The BaskERT model architecture is inspired by and very similar to the BERT model (Devlin et al., 2018) for natural language processing. An overview of the model is shown in Figure 1. The details of the model will be explained in the upcoming sections.

As a starting point, every product j in the assortment is represented by a unique embedding $\mathbf{e}_j$, which is a continuous parameter vector of dimension d. Usually, d is considerably lower than the assortment size N. The embedding parameters are learned while the model is trained, in such a way that the values in the embedding represent information about the product's latent attributes. Similar products tend to end up with similar embeddings. The embeddings of all products together form the $N \times d$ embedding matrix $\mathbf{E}$. Apart from the product embeddings, one additional embedding $\mathbf{e}_{[MASK]}$ is learned for the [MASK] token. This embedding, once learned, is likely to have neutral values, as it appears in all sorts of basket contexts.

3.3 Putting products into context

Although products are assigned to one unique embedding, they might serve different purposes for different customers. For example, the product cheese by itself can fulfill different needs: for one customer, the cheese may serve as a snack, whereas for another it may serve as an ingredient for a hamburger, pasta dish or sandwich. Depending on its purpose, good complements would be red wine, hamburger buns, pasta sauce or lettuce, for example. Because the product cheese is only assigned to one embedding, $\mathbf{e}_{cheese}$, this embedding includes information about all these possible purposes (and potential complements). However, for most purchase occasions the product cheese only fulfills one specific type of need, and potential complements belonging to other needs are not relevant to recommend. For example, if a basket already contains both cheese and spaghetti, it is highly likely that the cheese serves as a pasta-dish ingredient, and lettuce or hamburger buns would no longer be good recommendations.

Our model is able to discriminate between cheese purchased as a snack and cheese purchased as a pasta dish ingredient by adding contextual information to the product embeddings. This is achieved by the attention mechanism in BERT: the $|\tilde{\mathcal{B}}_i| \times d$ embedding matrix is modified based on contextual information, stemming from the other products purchased in the same basket. There is a strong analogy here with the use of attention

mechanisms in natural language processing: there, individual words may have different semantic meanings, which usually become clear from the rest of the sentence. Similarly, in a marketing setting the purpose of a product purchase often becomes clear from other products in the basket.

In more technical language, for a given masked basket $\tilde{\mathcal{B}}_i$, the embeddings of the products $j \in \tilde{\mathcal{B}}_i$ serve as input for the model, together with the embedding for the [MASK] token. As a result, each masked basket is represented by a $|\tilde{\mathcal{B}}_i| \times d$ input matrix. Next, it is important to share information in these embeddings across the products in the basket. This adds context information from the products in the basket to the embeddings. We achieve this by passing the embeddings through a set of L transformer layers, see Figure 1 and also Vaswani et al. (2017). Each of the L layers consists of both a multi-headed attention layer and a feedforward layer. The first makes it possible to put more weight on some products while sharing information, whereas the second allows for more interactions between the embeddings. We describe these two sublayers in more detail below. They are also the core difference between our approach and the simple pooling operations used in DREAM (Yu et al., 2016) and SHOPPER (Ruiz et al., 2020).

To further explain how context is added to the embedding of each product, we distinguish between the input and the output embedding of each layer. In general, we denote the input embedding of transformer layer l for basket i by $\mathbf{X}_{il}$. For the first layer, $\mathbf{X}_{i1}$ equals the set of unadjusted product embeddings $\mathbf{E}_i$. For other layers, the inputs are embeddings where some context has already been added in earlier layers. Specifically, each attention layer adds a d-dimensional context-based vector to each input embedding. The resulting *contextualized* output embeddings of attention layer l, denoted by $\mathbf{Y}_{il}$, are given by

$$\mathbf{Y}_{il} = A(\mathbf{X}_{il}) + \mathbf{X}_{il}, \tag{3}$$

where A denotes the scaled-dot attention function (Vaswani et al., 2017).

The scaled-dot attention function is based on so-called query, key and value matrices. These are linear transformations of the layer inputs $\mathbf{X}_{il}$ that can highlight some characteristics of the products (e.g. whether they are vegan, organic or belong to the Italian cuisine) by giving a large weight to that part of the embedding that captures this information. The query ($\mathbf{Q}_{il}$), key ($\mathbf{K}_{il}$) and value ($\mathbf{V}_{il}$) matrices are defined as

$$\mathbf{Q}_{il} = \mathbf{X}_{il}\mathbf{W}_l^Q \tag{4}$$

$$\mathbf{K}_{il} = \mathbf{X}_{il}\mathbf{W}_l^K \tag{5}$$

$$\mathbf{V}_{il} = \mathbf{X}_{il}\mathbf{W}_l^V, \tag{6}$$

where $\mathbf{W}_l^Q$, $\mathbf{W}_l^K$ and $\mathbf{W}_l^V$ are $d \times d$ matrices of model parameters that are shared across all baskets. The resulting $\mathbf{Q_{il}}$, $\mathbf{K_{il}}$, $\mathbf{V_{il}}$ contain $|\tilde{\mathcal{B}}_i|$ queries, keys and values, respectively, where a single query/key/value is a d-dimensional vector corresponding to one of the tokens in the masked basket. Given all queries, keys and values, the scaled-dot attention function A (Vaswani et al., 2017) is defined as

$$A(\mathbf{X}_{il}) = \text{softmax}\left(\frac{\mathbf{Q}_{il}\mathbf{K}_{il}^T}{\sqrt{d}}\right)\mathbf{V}_{il}, \tag{7}$$

where *softmax* denotes a rowwise normalized exponential function, which transforms its inputs into non-negative weights such that each row adds up to 1. The scaling by $\sqrt{d}$ is used to control the variance of the softmax input. The output of the attention function is therefore a weighted combination of the rows in $\mathbf{V}_{il}$, where weights are dependent on the queries and keys of all products in the basket. If the query of product A is similar to the key of product B, their dot product is high and the value of product B receives a relatively large weight in the update of product A's embedding. In that case, we say that A pays strong attention to B. For example, to predict the type of pizza that best fits a set of other products in a basket, a query could be whether other products are vegetarian or not. The query will then "ask" all other products whether they are informative about this and, if so, what information they have. Cheese, nuts, tofu, but also meatballs would "answer" that they are informative with a key that matches the query, resulting in large values for the corresponding elements in $\mathbf{Q}_{il}\mathbf{K}_{il}^T$. This results in attention being given to the "answers" of these products, which are provided by their values. These values would, in this case, differ a lot between cheese, nuts and tofu on the one hand and meatballs on the other hand. Where the latter suggests that the customer is not strictly vegetarian, the former products might suggest that a pizza pepperoni has a poor fit compared to a pizza quattro stagioni. Note that other products, e.g. washing detergent, might indicate through their key that they contain little information about this query, resulting in negative values of $\mathbf{Q}_{il}\mathbf{K}_{il}^T$ and little weight being given to their values.

3.4 Multi-headed attention

To add even more flexibility to the attention mechanism, Vaswani et al. (2017) introduced the so-called multi-headed attention layer. The attention mechanism in equation 7 has queries, keys and values that match the full embedding dimension d. This means that each product pays either high or low attention to the full value of another product. Multi-head attention splits up the dimension d in H equally sized parts and computes H attention functions in parallel, one on each part. These H parts are called attention *heads* and they allow for more queries to be made in the same computational effort. These queries are shorter and can be more focused, as they only have to pay attention to a particular sub-

space of a product's embedding. For example, one head could pay attention to vegetarian products and one could pay attention to Italian cuisine. Each head has its own queries, keys and values, defined by:

$$\mathbf{Q}_{ilh} = \mathbf{X}_{il}\mathbf{W}_{lh}^{Q} \tag{8}$$

$$\mathbf{K}_{ilh} = \mathbf{X}_{il}\mathbf{W}_{lh}^{K} \tag{9}$$

$$\mathbf{V}_{ilh} = \mathbf{X}_{il}\mathbf{W}_{lh}^{V}, \tag{10}$$

where $\mathbf{W}_{lh}^{Q}$, $\mathbf{W}_{lh}^{K}$ and $\mathbf{W}_{lh}^{V}$ are now $d \times d_h$ parameter matrices, with $d_h = d/H$. On each head, a separate scaled-dot attention function is computed, resulting in $A_h(\mathbf{X}_{il})$, a $|\tilde{\mathcal{B}}_i| \times d_h$ partial embedding update. Finally, the columns of the H partial embedding updates are concatenated to restore the original dimension d. The resulting multi-head attention layer output becomes

$$A(\mathbf{X}_{il}) = [A_1(\mathbf{X}_{il})\ A_2(\mathbf{X}_{il})\ \ldots\ A_H(\mathbf{X}_{il})]\,\mathbf{W}_{l}^{O}, \tag{11}$$

where $\mathbf{W}_{l}^{O}$ is a $d \times d$ parameter matrix that transforms the updated embedding matrix one final time.

3.5 Feedforward layer

Next to the attention layer that is aimed at adding context, the model also allows to add context through a more traditional feedforward mechanism. To enable a very rich representation of the possible context effect, the intermediate layer in the feedforward layer is of higher dimension than the embeddings themselves, following the structure proposed by Vaswani et al. (2017). Similar to Devlin et al. (2018), we set the dimension of the intermediate layer, d_{FF}, to four times the embedding size d and use Gaussian Error Linear Unit (GELU) activation. The GELU activation function equals $x\Phi(x)$ with Φ the Gaussian cumulative distribution function. The output of transformer layer l, which provides the input for layer $l+1$, then becomes

$$\mathbf{X}_{i,l+1} = FF(\mathbf{Y}_{il}) + \mathbf{Y}_{il}, \tag{12}$$

where FF denotes the feedforward operation (Mitchell, 1997).

3.6 Predicting the masked product

After the initial embeddings have been updated with contextual information through L transformer layers, the model needs to predict the missing, masked product. Besides the fact that the embeddings of already purchased products have been adapted with

information from other products, the embedding of the [MASK] token now also contains context information. If one of the original embeddings is similar to this contextualized [MASK] embedding, it means that it fits well in the context of the basket and we assign a high probability to it. We measure the similarity between product j and the contextualized [MASK] embedding by multiplying this embedding with the original product embedding of product j, $\mathbf{e}_j$. We rescale the resulting similarities for all products using a softmax function.

In mathematical terms, the predicted probabilities for basket i are given by

$$\hat{\mathbf{y}}_i = \text{softmax}(\mathbf{E}\mathbf{x}^{L+1}_{i,[MASK]}), \tag{13}$$

where $\mathbf{x}^{L+1}_{i,[MASK]}$ denotes the contextualized [MASK] embedding following from (12) and $\mathbf{E}$ the full set of product embeddings. For products that were already purchased in the basket, we set the assigned probability to zero and the remaining probabilities are rescaled to sum to 1. If one would like to account for repeat purchases within a basket, this step could be skipped.

3.7 Alternative model specifications

Although our model is very similar to the BERT model for natural language processing (Devlin et al., 2018), there are some important differences that we list here.

In the original BERT implementation, a fixed percentage of words in a sentence is masked. For long sentences, multiple words are masked. In our case, we limit ourselves to one masked product per basket. Recommending multiple products with BaskERT in a real-life setting is still possible though in an iterative way. Also, in BERT's original pre-training task, some masked products are not replaced by a [MASK] token, but by a fake product or simply by itself (leaving the original basket unchanged). Doing this results in a model that generally performs better in a second fine tuning task. As predicting the masked product is already our only and final training task, we do not adopt this approach.

Next, in the original BERT model, an additional positional embedding is added to each of the word embeddings, representing its position in the sentence. In BaskERT, we do not include such embeddings, as they require a natural ordering in the basket contents. Although the add-to-cart order (i.e. the order in which products are added to the basket by the customer) could be used to determine the natural position of each product, we believe this order is more driven by assortment layout than by complementarity and substitution patterns. For example, if the snacks aisle (or web page) is located next to the pasta aisle (or web page), the data might contain many baskets in which snacks are added directly after pasta, even though they do not have to be strong complements. Including the add-to-cart

order would bias the learned patterns towards the current assortment layout.

Another difference is that in the original architecture, the transformer layers in BERT contain two additional components. First, after each sublayer, layer outputs are normalized with a layer normalization operation (Ba et al., 2016). The reason for this is to control the distribution of each layer's inputs, making training faster. Secondly, each operation is followed by dropout regularization to avoid overfitting (Srivastava et al., 2014). However, after initial experimentation, both of these components turn out to have a small negative impact on out-of-sample performance in our application. Overfitting is not an issue in our application, probably because of the large number of baskets considered relative to the complexity of our model. Therefore, we leave out the layer normalization and dropout regularization from the model. For applications with smaller datasets, overfitting becomes a risk and applying dropout is recommended.

Finally, some techniques (such as P2V by Gabel et al. (2019)) use a different product embedding in the input and output layer, such that each product is represented by two different embeddings. Press & Wolf (2016) suggest to tie embeddings in the in- and output layer, as we do in our network. We experimented with using a different embedding matrix in the output layer, but this leads to worse out-of-sample performance. We also experimented with adding a bias vector before applying softmax, but this also did not result in any out-of-sample performance improvement.

3.8 Model training

The aim of our model is to predict the product that was masked. For this purpose, we define the loss function contribution of basket i as

$$L_i = -\log(\hat{y}_{i,m_i}), \tag{14}$$

with $\hat{y}_{i,m_i}$ denoting the estimated probability that the masked product equals m_i (the actual masked product). In practice, a batch of B baskets is passed through the network in parallel and their loss contributions are summed. Afterwards, the gradient of this summed loss is backpropagated through the network. Parameters are updated with Adam (Kingma & Ba, 2014), a gradient descent algorithm with adaptable, parameter-specific learning rates. Training is stopped once the loss calculated on a validation set no longer decreases. This is a common practice stopping rule (Mitchell, 1997).

4 Application

In this section, we apply BaskERT to the setting of an online grocery delivery service. We first describe the dataset, the preprocessing steps, and the selection of model hyperparam-

eters. We then introduce our benchmark methods and performance measures. Next, we report our results.

4.1 The Instacart dataset

We use data from the American grocery delivery service Instacart (*The Instacart Online Grocery Shopping Dataset*, 2017). It contains information on more than 3.3 million orders made by over 200,000 customers in 2017. The company offers 49,688 different products, organized in 134 aisles, which are in turn organized in 21 departments.

Figure 2 shows the purchase frequencies of products in the assortment, ranked from highest to lowest popularity. For visibility, the plot is vertically capped at 10,000 purchases. The most popular product, (*Bananas*), is bought nearly 500,000 times in total, whereas the vast majority of products is bought no more than 200 times. The data is hence very sparse with many product being purchased very infrequently.

To avoid the risk of overfitting by estimating a model based on very few purchases for many products, we first cluster the products in the dataset to guarantee sufficient support for each (clustered) product. For products that appear extremely infrequently in the training data, it will be difficult for any model to learn general purchase patterns. We avoid this problem by clustering infrequently bought products into a product group that is bought at least 500 times in total. This results in an assortment of $9,407$ (clustered) products with at least 500 purchases, i.e. a purchase is observed in at least 0.02% of all baskets. Details of this product aggregation procedure are provided in Appendix A.

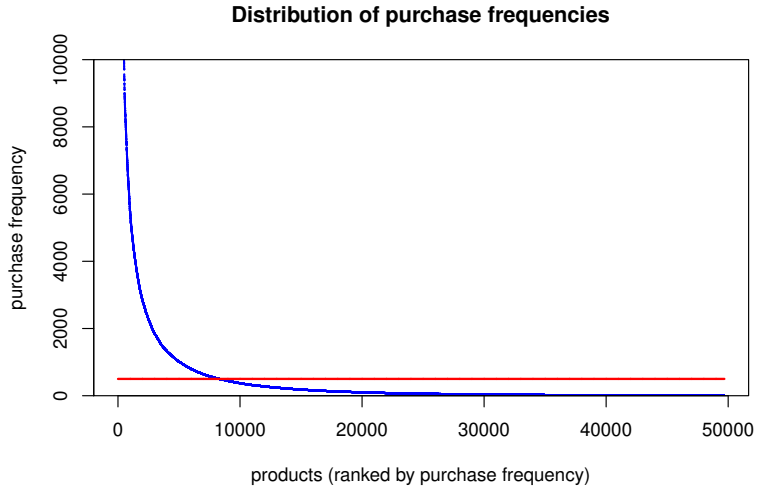

Figure 2: Purchase frequencies of products in the Instacart dataset. Products are horizontally ranked from highest to lowest popularity. For visibility, the vertical axis is capped at $10,000$. The horizontal line indicates 500 purchases, which we use as the minimum support during the pre-processing phase.

We randomly split the $3,346,083$ observed baskets into a training set of $3,200,000$ baskets, a validation set of $20,000$ and a test set of $126,083$ baskets. We use the validation set to track out-of-sample performance during training and to determine the optimal hyperparameter values. The test set is used for the final evaluation of the model. The size of the validation set is sufficient as loss values based on different random samples are not significantly different. For each of the three sets, baskets with only one product are ignored, as masking this single product would leave no further information about the basket. For computational efficiency, we also removed baskets with more than 50 items (0.09% of all baskets), to cap the size of the input baskets at a reasonable level. Characteristics of the resulting datasets after removing these baskets are summarized in Table 1.

Table 1: Characteristics of the datasets. A purchase is defined as a certain customer buying a certain product at a certain time.

dataset	# baskets	# purchases	avg. basket size
training set	3,040,206	31,896,729	10.49
validation set	19,004	199,875	10.52
test set	119,873	1,257,771	10.49
all	3,179,083	33,354,375	10.49

4.2 Performance measures

To evaluate the performance of the model in predicting the masked product, we randomly mask a product from each of the test or validation baskets and study the model's predictive performance. Again, we consider two approaches for sampling the masked product, as described in section 3.1. In the first approach, all products in a basket have an equal probability of being masked, so products that are purchased more often will naturally be selected more often. Always predicting the most popular product in the assortment would do relatively well on this task (as we will demonstrate). With the second, a weighted sampling approach is implemented such that each product is expected to be masked the same number of times across all baskets. To be able to perform well on this task, a method should also be able to accurately predict less popular products, when necessary.

In the prediction task, we use a consideration set of 100 products for each test basket. This set includes the truly masked product and 99 randomly sampled products from the entire assortment (apart from the products that were already in the basket). The reason we only consider a subset of the products for the prediction task and not the entire assortment is twofold: first, although BaskERT can quickly output a probability distribution over all products in the assortment, this does not hold for some of our benchmark methods (see Section 4.4). Hence, predicting the masked product for over $100,000$ test baskets would be time consuming if we would apply all (benchmark) methods to the entire assortment. Secondly, even the strongest predictive model would have a hard time predicting the

missing product from a set of nearly $10,000$ products. As a result, the model and all benchmarks would have a relatively low accuracy, and differences in performance would not always be very clear. By considering only a subset of 100 products, the performance differences between models becomes easier to see.

For each model, we report the accuracy, defined as the proportion of test baskets for which the product with the highest estimated probability, from the consideration set of 100 products, is indeed the truly masked product. Next to that, we report the average rank of the truly masked product if we would rank the 100 products from highest to lowest estimated probability. Even for more varied recommendations, we prefer low ranks as this implies that infrequently purchased product are positioned high in the ranking. We report these two measures for both the uniform and weighted sampling procedure, resulting in four performance measures.

4.3 Training and hyperparameter tuning

Model training is done using batches of 128 baskets. Training is stopped once the validation loss no longer decreases. For the final model, validation loss stabilizes after approximately $300,000$ training batches. By that time, each training basket had been passed through the network 12 or 13 times (each time with a differently sampled masked product). This takes slightly less than three hours on a single GTX-1080Ti GPU, for the optimal hyperparameter values given in this section.

When training BaskERT, values for the hyperparameters d (embedding size), L (number of layers), H (number of attention heads) and d_{FF} (dimension of feedforward layer) need to be set. Instead of tuning each of these hyperparameters separately, we chose to keep their *relative* size the same as in the original version of $\text{BERT}_{\text{BASE}}$ (Devlin et al., 2018), where $d = 768$, $L = 12$, $H = 12$, and $d_{FF} = 4d = 3072$ are used. We only tune the level of complexity by multiplying each of these hyperparameter values by the same constant. The constants we consider are $\frac{1}{2}, \frac{1}{3}, \frac{1}{4}, \frac{1}{6}$ and $\frac{1}{12}$, such that each of the hyperparameters still attains an integer value after multiplication. We only consider smaller versions of $\text{BERT}_{\text{BASE}}$ as our product assortment is smaller than the word vocabulary in BERT (9,407 versus 30,000) and the maximum basket size we consider is much smaller than the maximum sentence length considered in BERT (50 versus 512). Consequently, we believe a network with fewer parameters should be able to accurately describe the purchase patterns in the data. As shown in Table 2, the validation loss and accuracy on our basket completion tasks attained an optimum with 1/3 of BERT's original hyperparameter values. Therefore, we choose $d = 256$, $L = 4$, $H = 4$ and $d_{FF} = 1024$ in our final BaskERT implementation that we evaluate on the test set.

Table 2: Performance of several model configurations on the validation set. The presented performances are for baskets masked with $\alpha = 0$, but similar patterns are found for other values of α.

BaskERT configuration	d	d_{FF}	L	H	val. loss	val. accuracy (%)
$\frac{1}{12}$	64	256	1	1	7.05	33.27
$\frac{1}{6}$	128	512	2	2	6.92	35.75
$\frac{1}{4}$	192	768	3	3	6.86	36.56
$\frac{1}{3}$	256	1024	4	4	6.84	37.01
$\frac{1}{2}$	384	1536	6	6	6.84	36.90

4.4 Benchmarks

To assess the relative performance of BaskERT compared to state-of-the-art methods that could be applied in the same scenario, we compare it to several benchmarks. We also compare the mapped embeddings of BaskERT to those from P2V, to illustrate how insights into the assortment structure differ. In the following paragraphs, the benchmarks are separately discussed.

4.4.1 Overall popularity heuristic

The popularity heuristic simply ranks the 100 products in the consideration set based on their purchase frequency in the training set, from high to low. Therefore, it always predicts the most popular product from the consideration set for the masked product.

4.4.2 Memory-based collaborative filtering

As a second benchmark, we apply a memory-based collaborative filtering approach (Breese et al., 2013). For a given test basket (exluding the masked product), cosine similarities are computed with 320,000 randomly sampled training baskets (10% of the baskets used for training BaskERT), as computing similarities with all training baskets for each test basket would be heavily time consuming. Next, the k most similar training baskets are selected. Products from the consideration set are then ranked based on their purchase frequency in these k nearest neighbors, and the most frequently purchased product is predicted as the masked product. We considered values of k ranging from 30 to 3000, and for $k = 300$, out-of-sample performance turns out to be the best. We therefore only report the performance for $k = 300$.

4.4.3 P2V embeddings based predictions

Product2Vec (P2V) (Gabel et al., 2019) is a technique based on the skip-gram Word2Vec model used in natural language processing (Mikolov et al., 2013). Its main goal is to obtain product embeddings that accurately represent mutual complementarity and substitution

patterns in a retailer's assortment. This model does not output a probability distribution over the entire assortment based on a set of products. Instead, it gives a probability for a pair of products indicating the likelihood of the two products being purchased together in one basket.

The resulting model cannot directly be used for basket completion, as it always takes a pair of two products as input. In order to complete a given test basket, we pair up each of the products in the consideration set of 100 products with each of the products in the test basket that needs completion. For a given product in the consideration set, we pool the probabilities for each possible pairing with a product from the test basket, either by max pooling or by mean pooling. In case of max pooling, the best fitting product is a product that fits well with (at least) one of the products in the basket. In case of mean pooling, the best fitting product must fit reasonably well with all products in the basket. We evaluate and report the performance measures on both types of pooling.

P2V builds on a prediction task that requires sampling "negative" pairs, i.e. pairs of products for a basket that are not purchased, which the model needs to distinguish from "positive" pairs, i.e. pairs of products in the basket that are purchased. To construct the negative pairs, we apply a weighting scheme to match the evaluation criteria similar to the weighting scheme for BaskERT (see 3.1). Specifically, we construct product pairs with a non-purchased product by sampling a product with probabilities proportional to f_j^α, where f_j is the purchase frequency of product j and α is a constant. Gabel et al. (2019) suggested a value of 0.75 for α. We also applied P2V with values of 0 and 1 for α to match the evaluation tasks. The model was trained until the validation loss no longer decreased. After tuning, we end up with an optimal embedding size of 1000. We also added a bias in the final layer, as proposed by Gabel et al. (2019).

4.4.4 Continuous Bag-of-Words

Next to the skip-gram based model, we also apply a prediction model based on the continuous bag-of-words (CBOW) Word2Vec model (Mikolov et al., 2013). To do so, the embeddings of products in the basket are averaged during training, before being paired up with a product that potentially completes the basket. In other words, we move the mean pooling step in P2V from the test stage to the training stage. This model is almost equivalent to a vanilla SHOPPER model as discussed by Ruiz et al. (2020), but without the user-specific term. Similar to P2V, we used $\alpha = 0$, 0.75 and 1 to sample negative sets of products and an embedding size of 1000.

4.5 Predictive performance

The performance of BaskERT in a basket completion task is reported in Table 3, together with the performance of the benchmarks. First of all, doing well on the two different completion tasks simultaneously turns out to be difficult. For the uniformly sampled target, models that favor popular products perform well. BaskERT with $\alpha = 0$ clearly outperforms each of the benchmarks on the first training task with an accuracy of 36.8% and average rank of 6.9, but the other models trained with $\alpha = 0$, collaborative filtering and the popularity heuristic all reach relatively high accuracy and low average rank. However, each of these models predict a lot worse for products in the tail of the assortment. This is indicated by their considerably worse performance in the second training task, where each of the products contributes equally to the accuracy.

The completion task with weighted target sampling is more difficult than the first training task, considering the generally lower accuracies and higher average ranks. Models that are less geared towards the prediction of popular products during training (high α) do relatively better here. BaskERT with $\alpha = 1$ outperforms the other methods in terms of both accuracy and average rank. Even BaskERT trained with $\alpha = 0$ does relatively well here, given that none of the benchmarks outperforms its average rank and only one benchmark (CBOW with $\alpha = 1$) does so in terms of accuracy.

Table 3: Predictive performance of BaskERT compared to several benchmarks. See sections 4.2 and 4.4 for a detailed description of the performance measures and benchmarks.

		uniform target sampling		weighted target sampling	
		accuracy (%)	av. rank	accuracy (%)	av. rank
Most popular (POP)		19.7	19.6	3.0	39.3
Collaborative filtering (CF)		27.2	23.3	14.0	34.7
P2V + mean pool.	$\alpha = 0$	27.4	9.6	7.9	19.6
	$\alpha = 0.75$	29.8	9.2	19.1	14.5
	$\alpha = 1$	15.3	15.4	19.9	14.7
P2V + max pool.	$\alpha = 0$	30.5	9.9	14.1	19.5
	$\alpha = 0.75$	19.0	12.7	19.4	16.7
	$\alpha = 1$	12.7	22.1	17.8	18.4
CBOW	$\alpha = 0$	31.2	9.8	14.6	18.2
	$\alpha = 0.75$	24.6	11.8	20.2	14.6
	$\alpha = 1$	20.6	14.2	22.4	14.4
BaskERT	$\alpha = 0$	**37.0**	**6.9**	20.3	12.2
	$\alpha = 0.75$	27.7	8.5	25.9	10.9
	$\alpha = 1$	22.6	10.4	**26.4**	**10.8**

To gain more insight into the quality of the predictions, we further investigate the predictive performance of the models by dividing the test baskets in different groups based on their characteristics. First, we divide test baskets in 10 approximately equal groups based on their size and report the performance on each of these subgroups. Figure 3 shows the

accuracy and average rank for models trained with $\alpha = 1$ on the weighted prediction task, but similar results are obtained for models with lower α and on the unweighted prediction task. BaskERT performs relatively stably across groups and uniformly outperforms all benchmarks. A slight decrease in accuracy is visible for larger baskets, probably because more products fit well in large baskets and identifying the correct product becomes more difficult.

We repeat the same procedure by splitting test baskets based on the popularity of the target product (Figure 4) and based on the average popularity of the products already in the baskets (Figure 5). In both cases, we split the test baskets into 10 equally sized groups, based on their quantiles. Again, we observe that BaskERT uniformly outperforms all benchmarks on all subgroups and measures. CBOW and P2V perform very similar in terms of average rank, but CBOW slightly outperforms P2V in accuracy.

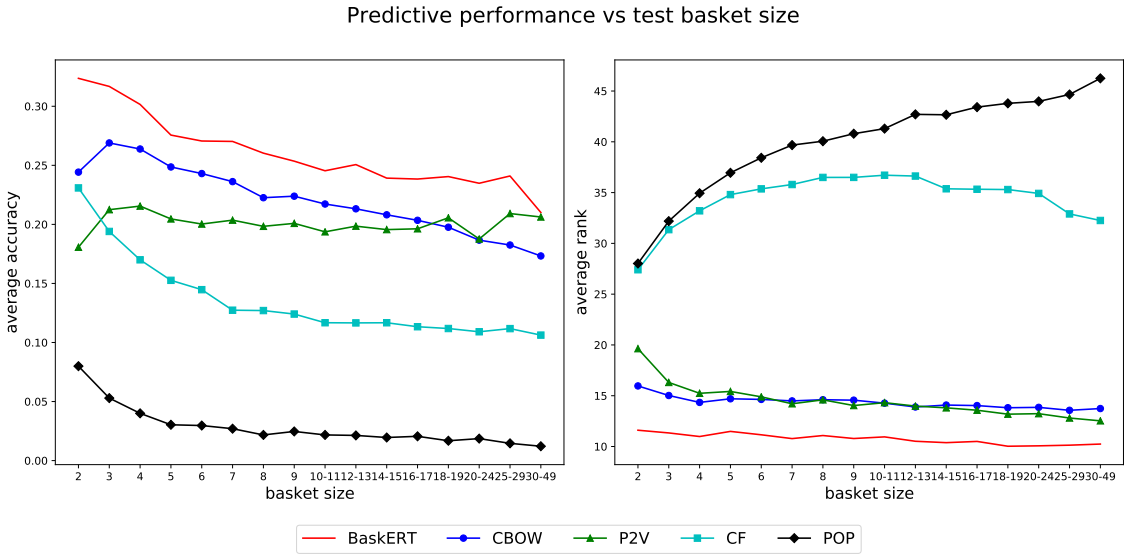

Figure 3: Accuracy (left) and average rank (right) of BaskERT versus benchmarks on differently sized test baskets.

4.6 Illustrative example

To illustrate the type of predictions that BaskERT generates, we provide some examples in Table 4. For five manually composed baskets, we present the top 8 predictions. The base product in each basket is ground beef. If no other product is provided, BaskERT predicts various types of products to fit in this basket. Hamburger buns and lettuce (to make a hamburger), pasta sauce (to make a pasta dish), milk (to make meatballs) and other sorts of meat all appear in the top 8. Once we add marinara sauce, mostly Italian cuisine products are predicted, especially spaghetti types. When we replace marinara sauce by an organic variant, it becomes clear that BaskERT encoded "organic" in the embeddings as we see more organic products appearing in the top 8, that did not appear for the non-

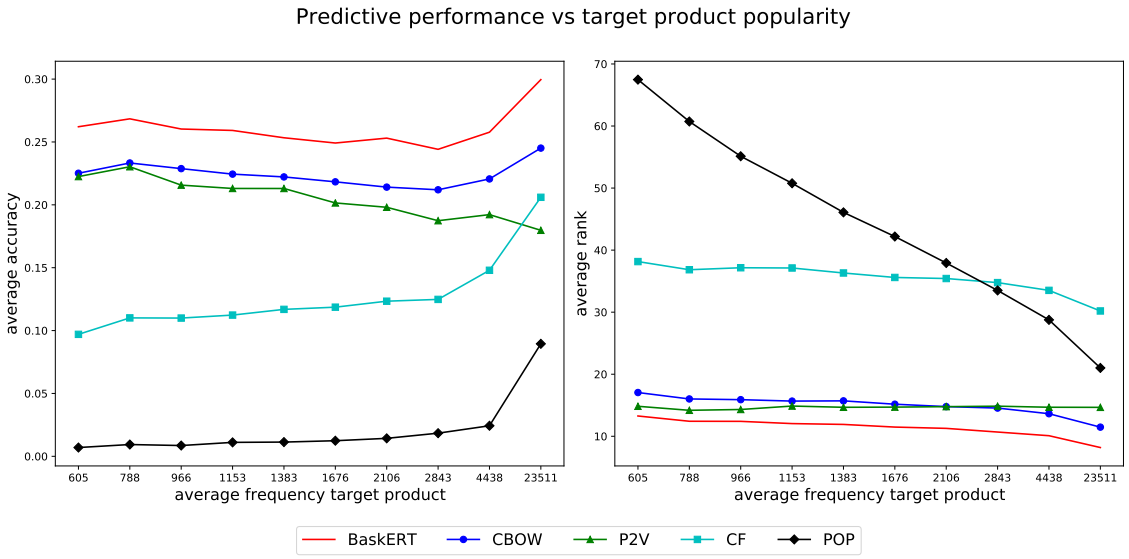

Figure 4: Accuracy (left) and average rank (right) of BaskERT versus benchmarks for different popularity of the target product in the training set.

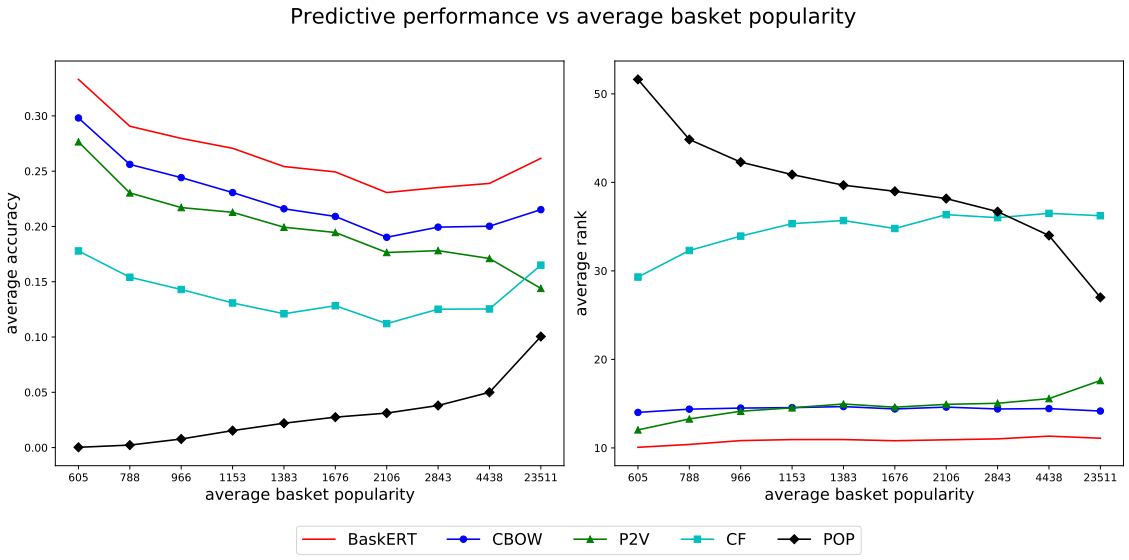

Figure 5: Accuracy (left) and average rank (right) of BaskERT versus benchmarks for different average popularity of the basket contents.

Table 4: Top 8 predictions for given sets of products in a test basket. Predictions are generated with BaskERT trained with $\alpha = 1$.

Basket	**Top 8 predictions**
85% Lean Ground Beef	Boneless Chicken Breasts Authentic French Brioche Hamburger Buns Spaghetti Squash Organic Whole Milk Organic Tomato Basil Pasta Sauce Tilapia Filet Romaine Lettuce Vitamin D Milk
85% Lean Ground Beef Marinara Sauce	Spaghetti Spaghetti No 12 Shredded Mozzarella Spaghetti Squash 93% Ground Beef Orecchiete Dry Pasta Homemade Hot Arrabiata Diavolo Sauce Boneless Skinless Chicken Breasts
85% Lean Ground Beef Organic Marinara Pasta Sauce	Shredded Mozzarella Spaghetti Organic Spaghetti Pasta Spaghetti Squash Organic Tomato Basil Pasta Sauce Organic Whole Wheat Penne Rigate Roasted Garlic Pasta Sauce Organic Whole Wheat Fusilli
85% Lean Ground Beef Spaghetti No. 12	Ground Veal Crushed Tomatoes Marinara Sauce All Natural Marinara Pasta Sauce Organic Tomato Basil Pasta Sauce Marinara Pasta Sauce Garlic Marinara Cooking Sauce Buffalo Chicken Wings
85% Lean Ground Beef Marinara Sauce Spaghetti No. 12	Ground Veal Crushed Tomatoes Homemade Hot Arrabiata Diavolo Sauce Penne Rigate #41 Pasta Rigatoni Pasta Organic Tomato Basil Pasta Sauce Bucatini No. 15 Fusilli No. 34

organic pasta sauce. Note that the product descriptions were not used to generate the predictions, only the identifiers of each product in each basket were used. When we include spaghetti instead of marinara sauce, many pasta sauces are predicted and spaghetti itself disappears from the recommendations. If we add both marinara sauce and spaghetti to the basket, we observe that mostly other types of pasta and pasta sauces are predicted. These predictions show that BaskERT can easily adapt its predictions to basket contents, taking into account both similarities and complementarities across products.

4.7 Embedding map

Retailers regularly need to make choices that require understanding of the relationships between products in their assortment. For example, they have to choose how to categorize products, how to arrange them in aisles, shelves or web pages, or which products to promote simultaneously (Gabel et al., 2019). Insights into these product relationships can be derived from our learned product embeddings. Figure 6 shows a map with all the products based on the similarity of the product embeddings $\mathbf{E}$ learned by BaskERT with $\alpha = 1$. Each dot represents a product and its color shows the department it was assigned to by Instacart. The map is constructed using t-SNE, which maps the high-dimensional embeddings to the two-dimensional space in such a way that distances between similar embeddings are preserved as accurately as possible. t-SNE focuses mostly on capturing local distances between very similar products, larger distances between product with dissimilar embeddings are less well preserved.

The map shows that products placed close to each other are often products from the same department, even though Instacart's department allocation was not used by the model in any way. This indicates that the learned embeddings capture similarity patterns between products well. Some departments, such as *babies*, *alcohol*, *personal care* and *household* cover a very specific region of the map, indicating that these departments are strongly homogeneous in terms of their purchase patterns. Other departments, such as *snacks* and *pantry*, can be found in several regions of the map, indicating that these departments are more heterogeneous.

For comparison, the same t-SNE map created with the P2V embeddings is displayed in Figure 7. The maps share some similarities. For example, bot identify clear clusters for the departments *alcohol*, *frozen* and *babies*. In general, however, the BaskERT map exposes more homogeneous groups of products. For example, the departments *beverages* and *canned goods* have clear clusters in the BaskERT map but not in the P2V map. This indicates that BaskERT's embeddings contain more accurate information about the relationships between products in the assortment.

For two of the more diffuse departments, *dry goods pasta* and *pantry*, we show the

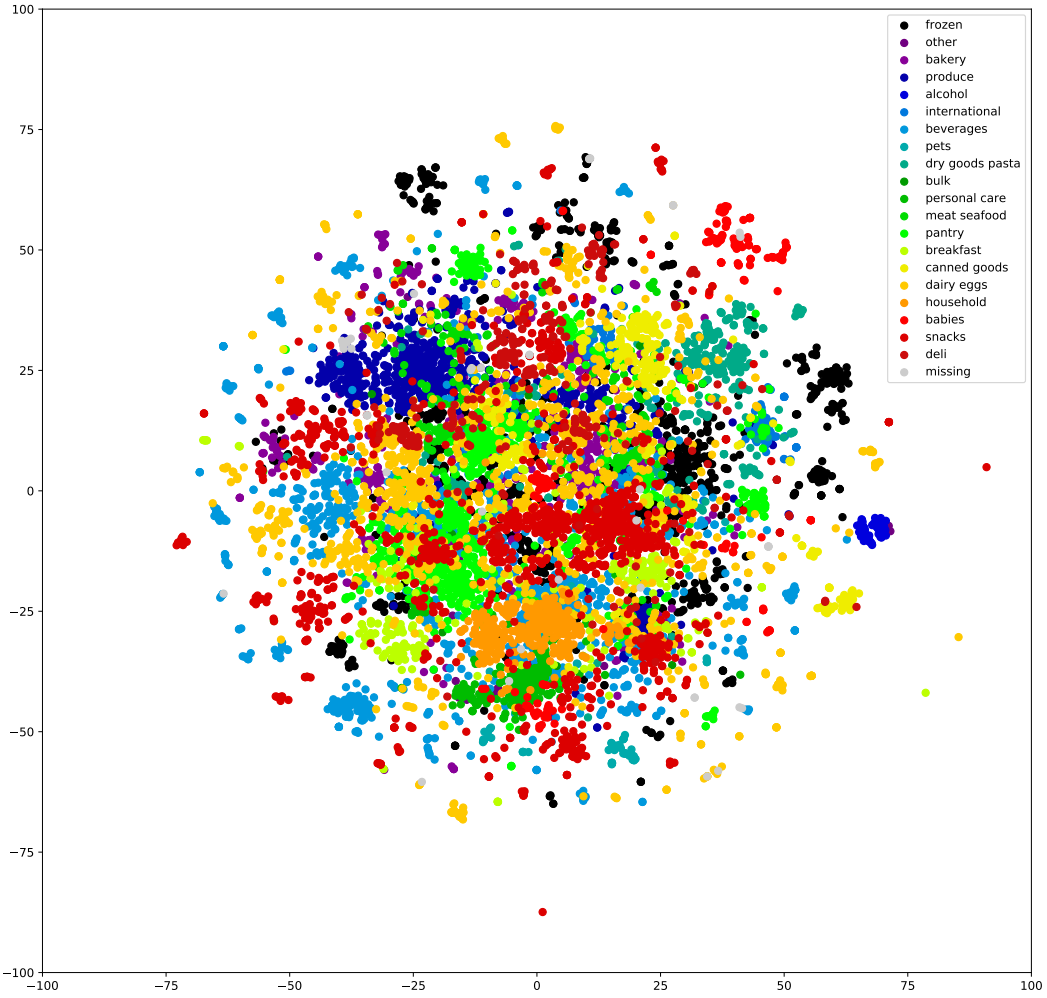

Figure 6: t-SNE map of the product embeddings learned by BaskERT. Each dot represents a product, its color denotes the department it is assigned to by Instacart.

BaskERT maps in Figure 8 and 9, respectively. This time, colors represent the subdepartments or aisles a product was assigned to by Instacart. For P2V, maps of the same departments are presented in Appendix B. The department maps yield similar conclusions as the full map about the differences between P2V and BaskERT. In the *dry goods pasta* department, products from the *pasta sauce* aisle and *dry pasta* aisle are clearly separated, but also very close to each other. Apparently, the learned embeddings of two complementary goods are very similar in BaskERT. The *pantry* department map shows that some aisles cover a very specific region of the map, such as *salad dressing topping*, *preserved dips spreads* and *spices seasonings*. Retail managers can use these visualizations to reallocate their products to other or new departments, to end up with more homogeneous categories.

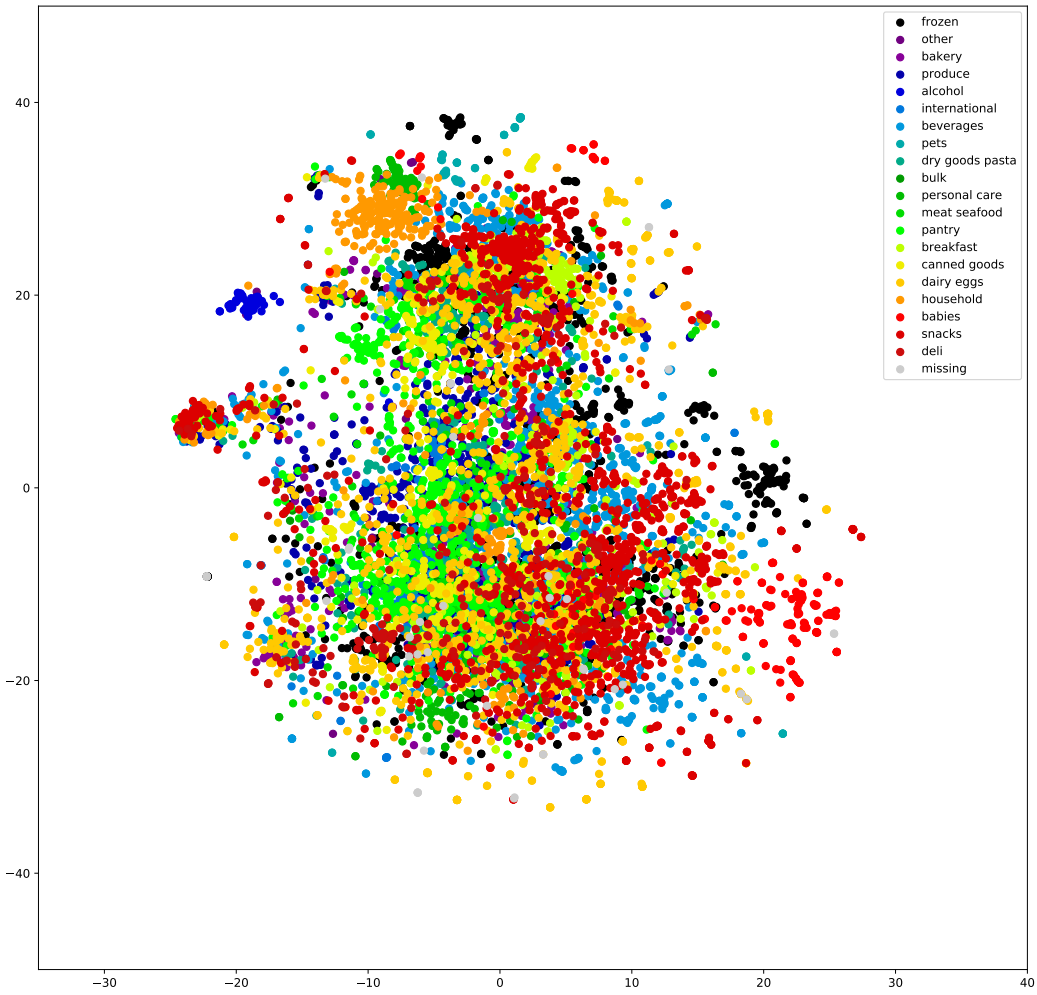

Figure 7: t-SNE map of the product embeddings learned by P2V. Each dot represents a product, its color denotes the department it is assigned to by Instacart.

4.8 Robustness to random starts

At the start of the training of BaskERT, its parameters are randomly initialized. Such random starts could influence the optimal parameters obtained by the training algorithm. We experimented with several random starts and found that basket completion performance was generally not affected by it. Both accuracies and average ranks changed with only 0 to 0.2 points in either direction, depending on the random start.

We also assess whether the insights encoded in the embeddings are stable across random starts, where we focus on the relative distance between embeddings. To be precise, we test whether products have the same set of "neighbors" in the embedding space. For each random start, we train the model and calculate an $N \times N$ Euclidean distance matrix from

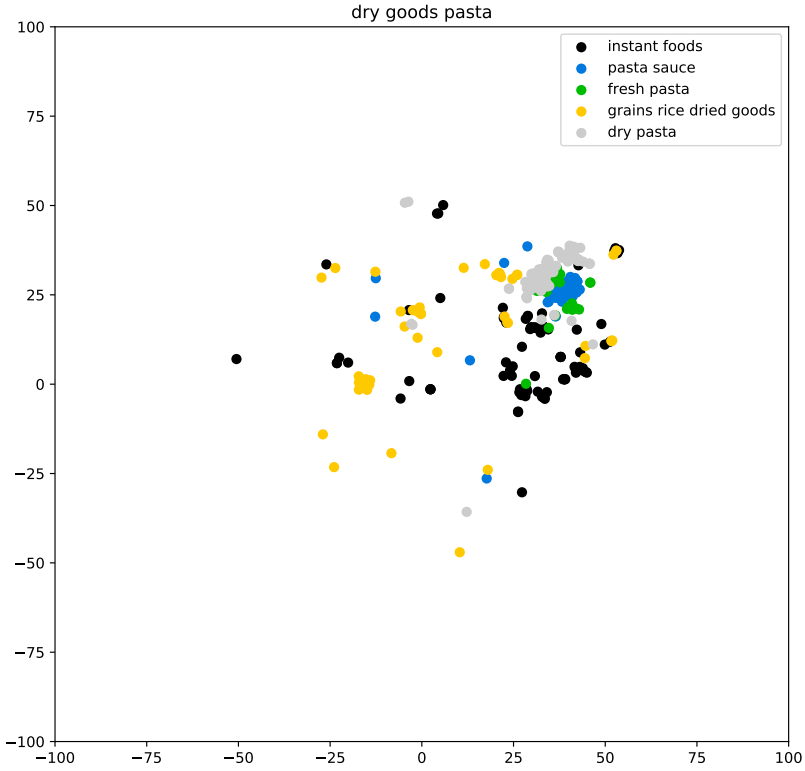

Figure 8: t-SNE map of products from the *dry goods pasta* department. Colors represent the aisles the product are assigned to by Instacart. The axis ranges match the axis ranges of Figure 6, for easy identification of the products appearing in both plots.

the learned product embeddings. For a given random start A and a focal product X, we assign 'scores' to the $1,000$ products with the smallest distance to product X's embedding. The most similar product is assigned a score of $1,000$, the second most similar 999, and so on. For another random start B and the same product X, we also order all other products based on their similarity to X. For the top k most similar products, based on the ranking that results from start B, we evaluate the summed score of these products in the similarity ranking based on start A, and vice versa.

Instead of doing this for product X only, we repeat it for all products in the assortment. This yields N different scores, each as a function of k, the number of most similar products considered. If the summed score increases rapidly for all focal products X as a function of k, the rankings of random starts A and B are very similar and the embedding is considered stable across both starts.

Figure 10 shows the cumulative number of points, as a proportion of the maximum possible number of points and averaged across all N focal products. We repeated the entire procedure for all pairs out of 10 random starts, but found very similar plots each time. We therefore continue with the results of one pair of random starts. The gray area in Figure 10 shows the interval containing 95% of the N score curves. The black line shows the average

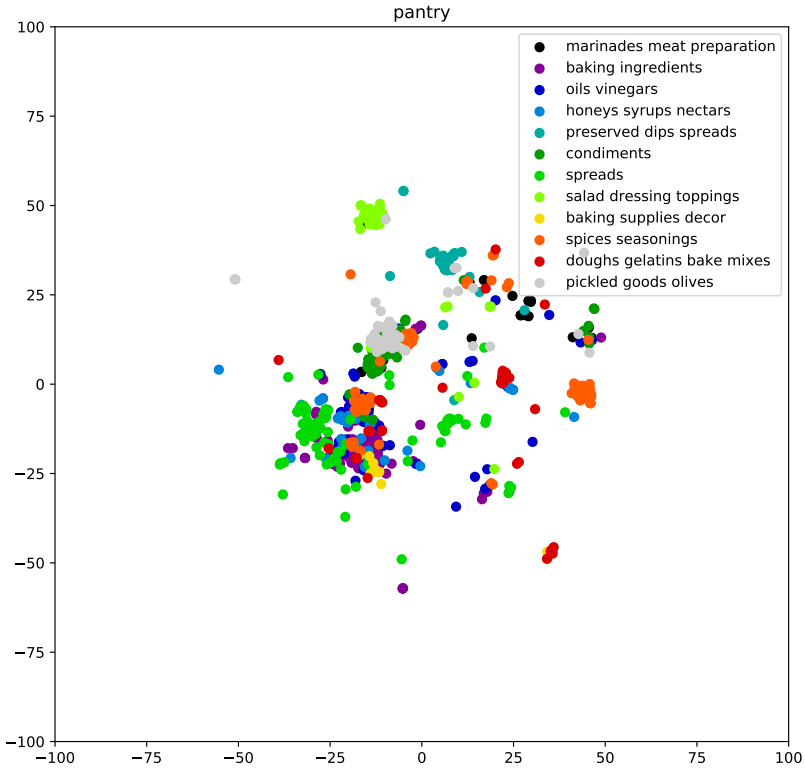

Figure 9: t-SNE map of products from the *pantry* department. Colors represent the aisles the product are assigned to by Instacart. The axis ranges match the axis ranges of Figure 6, for easy identification of the products appearing in both plots.

score curve. The larger the area under this curve, the larger the similarity between the two rankings. The red dashed line shows the maximum achievable number of points for each value of k. This curve reaches a maximum after $1,000$ products, as products lower in the ranking receive no points. The blue dotted line shows the expected score curve of a random ranking. As the average score curve is relatively close to the upper bound, we conclude that relative embedding distances are very similar across random starts, although they are not identical.

4.9 Ablation study

Finally, we assess the contribution of each model component in the predictive performance of BaskERT. To do so, we leave out the components one by one, train the resulting model and evaluate its performance. We also report the performance when dropout or layer normalization is added to model. As detailed in 3.7, these latter two are not adopted in BaskERT. The performance decreases are given in Table 5. The base model here is BaskERT trained with $\alpha = 0$. The attention layers clearly are the most important component of the model. Without those, it barely outperforms the popularity heuristic (see Table 3). Leaving out the feedforward layers does not really hurt the predictive

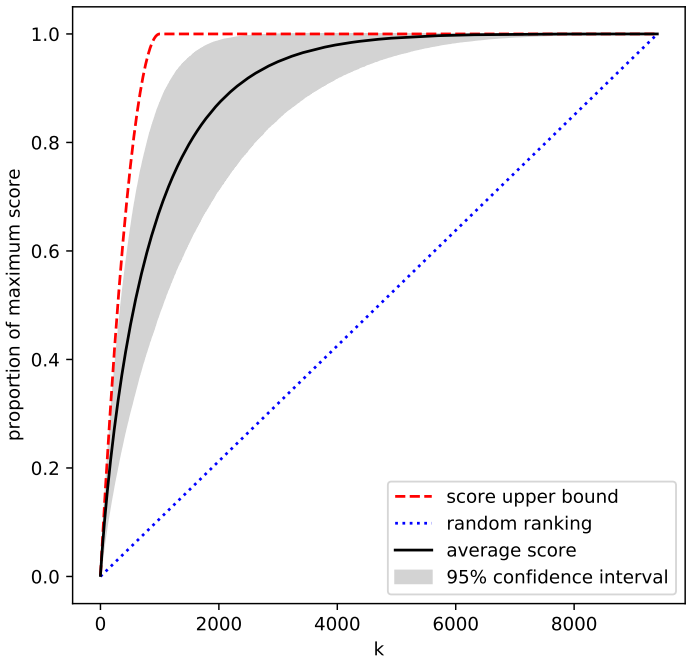

Figure 10: Summed scores for the top-k products most similar to a focal product in one random start, when scores are assigned based on another random start. The black line denotes the summed score of the k most similar products, averaged across N products, as a function of k. 95% of the N product-specific summed scores fall within the gray area. The blue dotted line gives the expected score of a random ranking and the red dashed line shows the maximum achievable summed score for all values of k.

performance. Finally, adding dropout or layer normalization decreases the accuracy on both tasks. The average rank slightly improves, but less strongly than the decrease in accuracy. These results demonstrate that the attention mechanism is the core of our model and essential to capturing the complex purchase patterns in our shopping data.

Table 5: Predictive performance of BaskERT ($\alpha = 0$) when model components are separately left out or added

	uniform target sampling		weighted target sampling	
	accuracy (%)	av. rank	accuracy (%)	av. rank
base model	37.0	6.9	20.3	12.2
- attention layers	19.7	19.4	3.1	38.7
- feedforward layers	35.4	7.1	18.7	12.7
+ dropout	36.7	6.7	19.8	12.1
+ layer normalization	36.1	6.8	19.3	12.0

5 Conclusion

In this paper, we introduced BaskERT, a basket completion model inspired by Google's BERT model for natural language processing. We demonstrated that BaskERT is able to outperform state-of-the art benchmarks in two different tasks, both aimed at identifying

a product that would fit an observed shopping basket well. In one task the target product is removed from the basket with uniform probability, which puts more weight on popular products ; in the other task, the target product is removed with an inverse-frequency weighted probability which requires more balanced predictions that are not driven by popularity. A retailer might not always opt for a recommender with maximum accuracy, as this might lead to only a limited set of (popular) products being recommended. A recommender that recommends also less popular products might not reach the same accuracy, but its recommendations are more varied. As a result, its recommendations could be a better fit with the products already in the basket, and are more likely to have been 'forgotten' during the customer's shopping trip. Moreover, the larger variety of recommended products might have the additional benefit that customers learn about more new products compared to a recommender that only predicts popular products. BaskERT's predictions provide flexibility to a retailer in constructing a recommender system by allowing the system to be trained with or without favoring the prediction of less frequently purchased products.

BaskERT's performance is to a great extent driven by the attention mechanism in its architecture. BaskERT's performance is also robust, as it does not require dropout regularization or layer normalization - which both aim at enhancing stability - and outcomes are also insensitive to random starts.

The t-SNE visualizations of the learned product embeddings showed that products from the same department and aisles are generally clustered closely together. Also, complementary goods tend to end up near each other. This indicates that the learned embeddings contain meaningful information about the retailer's assortment structure. The distances between embeddings could also be used by retailers to optimize the assortment layout and (online) shelf allocation of their (web) shop. Compared to the P2V map, the BaskERT embeddings separate the assortments into clearer, more homogeneous groups.

Future research could alleviate one of the main limitations of our investigation into the use of BaskERT for product recommendations. While we present clear evidence that the model identifies products the customer would purchase, we have no data and hence no evidence that recommending these products would actuallly be beneficial to the retailer. Investigating extensions of BaskERT with information about prices, promotions or the fact that a product was recommended, would be an important direction for future research. In addition, adding information about customers and their purchase histories might further improve the performance of BaskERT.

References

Ba, J. L., Kiros, J. R., & Hinton, G. E. (2016). Layer normalization. *arXiv preprint arXiv:1607.06450*.

Bianchi, F., Yu, B., & Tagliabue, J. (2020). BERT goes shopping: Comparing distributional models for product representations. *arXiv preprint arXiv:2012.09807*.

Bodapati, A. V. (2008). Recommendation systems with purchase data. *Journal of marketing research*, *45*(1), 77–93.

Breese, J. S., Heckerman, D., & Kadie, C. (2013). Empirical analysis of predictive algorithms for collaborative filtering. *arXiv preprint arXiv:1301.7363*.

Brynjolfsson, E., Hu, Y. J., & Smith, M. D. (2006). From niches to riches: Anatomy of the long tail. *Sloan management review*, *47*(4), 67–71.

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Gabel, S., Guhl, D., & Klapper, D. (2019). P2V-MAP: Mapping market structures for large retail assortments. *Journal of Marketing Research*, *56*(4), 557–580.

The Instacart Online Grocery Shopping Dataset. (2017). (Accessed from `https://www.instacart.com/datasets/grocery-shopping-2017` on March 26, 2018)

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

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

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

Mitchell, T. M. (1997). Machine learning. In (Vol. 1, p. 81-127). McGraw-hill New York.

Press, O., & Wolf, L. (2016). Using the output embedding to improve language models. *arXiv preprint arXiv:1608.05859*.

Ruiz, F. J., Athey, S., & Blei, D. M. (2020). SHOPPER: A probabilistic model of consumer choice with substitutes and complements. *The Annals of Applied Statistics*, *14*(1), 1–27.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, *15*(1), 1929–1958.

Sun, F., Liu, J., Wu, J., Pei, C., Lin, X., Ou, W., & Jiang, P. (2019). BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer. In *Proceedings of the 28th acm international conference on information and knowledge management* (pp. 1441–1450).

Van der Maaten, L., & Hinton, G. (2008). Visualizing data using t-SNE. *Journal of machine learning research*, *9*(11).

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems* (pp. 5998–6008).

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

Yu, F., Liu, Q., Wu, S., Wang, L., & Tan, T. (2016). A dynamic recurrent model for next basket recommendation. In *Proceedings of the 39th international acm sigir conference on research and development in information retrieval* (pp. 729–732).

A Pre-clustering of products

In this section, we describe the pre-clustering of products in the Instacart dataset. We restrict the clusters to contain only products that belong to the same aisle (as defined by Instacart). In this way, we ensure that clusters consist of similar products. Within an aisle, two products are assumed to be similar when they are bought together with similar sets of products from that same aisle. For example, if *Braeburn Apples* and *Fireside Apples* would both appear often in baskets together with *Bananas*, *Kiwis* and *Strawberries* (which are all from the *Fresh fruits* aisle), the two types of apples are considered similar. Note that *Braeburn Apples* and *Fireside Apples* do not necessarily appear often together themselves to be considered similar.

Formally, for each aisle a, an $N_a \times N_a$ co-occurrence matrix $\mathbf{C}_a$ can be constructed, where N_a denotes the number of items in aisle a. Let $\mathcal{P}_i$ denote the set of baskets in which product i appears. The off-diagonal elements of a matrix $\mathbf{C}$ are then defined by

$$c_{ij} = \frac{|\mathcal{P}_i \cap \mathcal{P}_j|}{|\mathcal{P}_i|} \quad \text{for } i, j = 1, ..., N_a, \quad i \neq j. \tag{15}$$

The values of the diagonal elements c_{ii} are debatable. To make sure that its value is not zero, but also not too high (and influential) compared to other values in the same row, it is set to $\frac{1}{|\mathcal{P}_i|}$. Next, an $N_a \times N_a$ cosine similarity matrix $\mathbf{S}$ is constructed over the rows of $\mathbf{C}$, where the elements of $\mathbf{S}$ are given by

$$s_{ij} = \frac{\mathbf{c}_i \cdot \mathbf{c}_j}{||\mathbf{c}_i|| ||\mathbf{c}_j||}, \tag{16}$$

where $\mathbf{c}_i$ is the ith row of $\mathbf{C}$. Finally, for each aisle separately, products are clustered in groups as follows:

1. Select the set of products corresponding to (clusters of) products with a purchase frequency lower than 500.
2. The product from this set of infrequently purchased products with the highest similarity with any other products is grouped with that product. This could be a product that is already purchased more than 500 times.
3. For the newly clustered products, the corresponding rows and columns in $\mathbf{S}$ are replaced by the similarities that correspond to the newly created cluster of products. As the purchase frequency of the new cluster is the sum of the two single purchase frequencies, the purchase frequency of this product has increased.
4. The first three steps are repeated until no (cluster of) products with purchase frequency lower than 500 are left for the considered aisle.

This product aggregation procedure results in 9,407 products or product clusters.

B P2V maps for selected departments

Figures 11 and 12 show the P2V maps of the products in the *dry goods pasta* and *pantry* departments, respectively. Different color represent different aisles. In line with the full maps, some similar clusters are found by both BaskERT and P2V. In general, however, clusters are less clearly separated and less homogeneous compared to the BaskERT map.

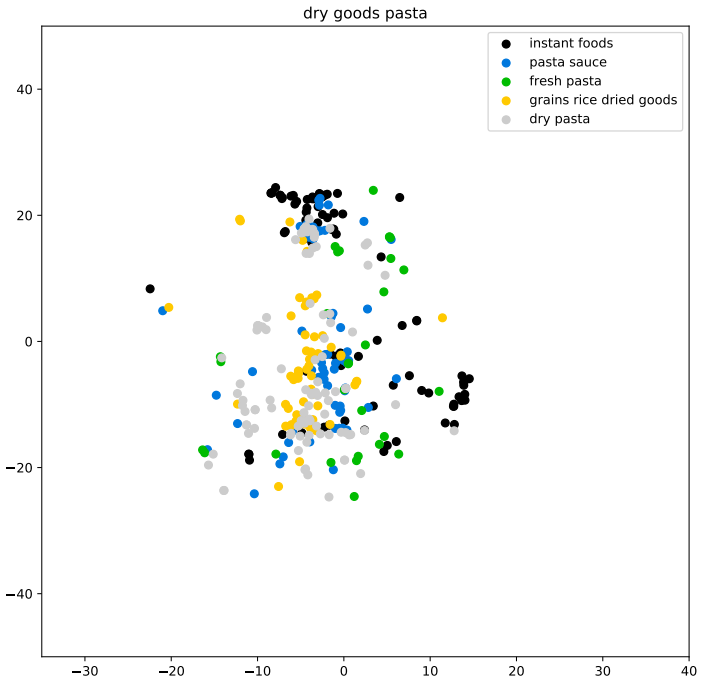

Figure 11: t-SNE map of products from the *dry goods pasta* department, with embeddings learned by P2V. Colors represent the aisles the product are assigned to by Instacart.

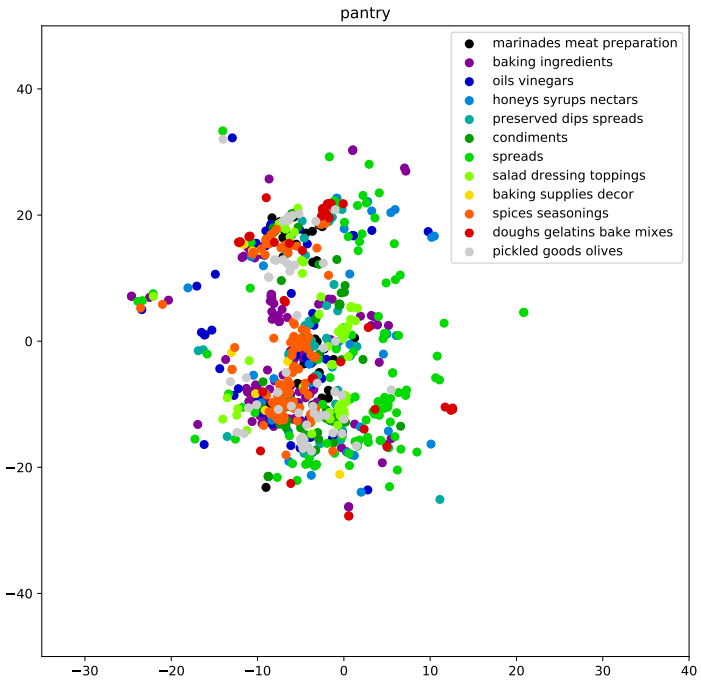

Figure 12: t-SNE map of products from the *pantry* department, with embeddings learned by P2V. Colors represent the aisles the product are assigned to by Instacart.