The Market Basket Transformer: A New Foundation Model for Retail

Sebastian Gabel[1] and Daniel M. Ringel[2]

[1] Rotterdam School of Management, Erasmus University (gabel@rsm.nl)

[2] UNC Kenan-Flagler Business School (dmr@unc.edu)

November 3, 2024

Declaration of conflicting interest:

The authors declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding statement:

Sebastian Gabel acknowledges funding from the Dutch Research Council (NWO) under a Veni grant and a Google Cloud Platform (GCP) grant for cloud resources.

The authors contributed equally to this research, and they are listed in alphabetical order.

The Market Basket Transformer: A New Foundation Model for Retail

November 3, 2024

Abstract

This paper proposes a *market basket transformer* (MBT), an adaptation of large language model architectures to market basket data. Our transformer learns purchase patterns in unordered product sets, predicting basket composition more than twice as accurately as existing approaches. Three features differentiate the MBT from large language models. First, we incorporate covariates to account for varying purchase probabilities across heterogeneous shopping trips. Second, we improve the pretraining of the transformer by leveraging available data more efficiently. Third, we propose a data curation strategy designed to ensure that the MBT learns meaningful product representations for all products in the retailer's assortment. These extensions ensure that the MBT accurately learns purchase patterns, even for long-tail products typically underrepresented in the training data. We demonstrate the model's adaptability by fine-tuning it to predict coupon redemptions using data from a coupon experiment. Because the MBT learns rich patterns in market basket data, it is a foundation model that can be easily adapted to various retail analytics applications.

Keywords: Transformer models, foundation models, fine-tuning, retailing analytics, market basket analysis, long-tail products

1 Introduction

Retail analytics is a cornerstone of modern marketing, providing retailers and manufacturers with critical insights into consumer behavior, inventory management, and strategic planning. The growing volume and variety of retail data generated from point-of-sale systems and customer interactions create new opportunities for enhancing operational efficiency and customer satisfaction (Wedel and Kannan 2016).

However, retail analytics faces three significant challenges. First, the scale of assortments and the volume of shopping baskets produce high-dimensional data that are difficult to process and analyze effectively (Bell et al. 2014). Second, data sparsity is a pervasive issue: long-tail products are purchased infrequently, making pattern detection and prediction challenging (Anderson 2006, Brynjolfsson et al. 2010). Third, the spectrum of retail analytics tasks is broad and complex. Many challenges, such as personalized marketing and inventory optimization, require specialized analytical approaches and expertise (Grewal et al. 2017).

These challenges parallel those encountered in natural language processing (NLP), where high dimensional and sparse data have historically impeded analysis. Transformer architectures and foundation models have revolutionized NLP by capturing the structure and meaning of language through the self-attention mechanism, leading to breakthroughs in sentiment analysis, machine translation, and conversational AI (Vaswani et al. 2019, Devlin et al. 2019). Transformers excel at modeling contextual relationships in sequential data, demonstrating exceptional performance across diverse applications.

Motivated by these advancements, our research adapts transformer architectures to retail data. By leveraging transformers' ability to understand assortment structures and shopping patterns, we address the inherent challenges of scale, sparsity, and task diversity in retailing. Our model captures complex product relationships and incorporates contextual factors such as store location and time, improving predictive accuracy and enhancing analytical depth. Pretrained MBTs can serve as retail foundation models that can be adapted to specific applications with little effort.

Our contributions are threefold. First, we implement a customized transformer model tailored to market basket data, and compare its performance with state-of-the-art models. Our results show that the MBT predicts missing products in shopping baskets with twice the accuracy of existing approaches. Second, we identify and mitigate a critical limitation in applying transformers to retail data: low data volume and high data sparsity prevent transformers from learning market structure and basket composition, particularly for long-tail products. To ensure that the transformer model performs equally well across all products in the assortment, we incorporate covariates, modify the training approach, and generate additional training examples. These modifications substantially improve the predictive accuracy of high-frequency products. This illustrates how modifying existing machine learning models increases their value for applications in marketing (Liu 2023). Third, we show that our MBT serves as a foundation model for retailing by first pretraining it to understand assortment structure and product relationships, and then fine-tuning it to predict coupon redemptions. The downstream task differs from pretraining objective, and far less data is available. This demonstrates the model's adaptability to specialized retail analytics tasks, such as personalized marketing and promotion optimization.

From a managerial perspective, our model addresses the complexities of retail analytics associated with handling large-scale and sparse data. Furthermore, the MBT is a foundation model that can be adapted to a broad spectrum of analytical tasks. This enables retailers to derive actionable insights with minimal additional training data, facilitating faster implementation and more effective decision-making. Moreover, covariates capture contextual factors that influence shopping behavior, providing valuable insights for retail analytics.

The remainder of this paper is organized as follows. We start by reviewing the related literature (Section 2) and introducing the MBT and its architecture (Section 3). After evaluating the model's performance empirically (Section 4), we present three model extensions that ensure consistent performance across the entire product assortment (Section 5). Finally, we illustrate how our model can be fine-tuned for specific tasks by applying it to coupon redemption

prediction (Section 6). We conclude with a summary of our contributions and suggest future research directions.

2 Literature Review

Our research contributes to two literature streams: applications of transformer models in marketing and methods for market basket analysis. We summarize both literature streams below and highlight our respective contributions.

2.1 Applications of Transformer Models in Marketing

Applications of transformers in marketing broadly fall into three categories. First, researchers use pretrained or fine-tuned transformer models such as BERT (Devlin et al. 2019) and GPT (OpenAI 2023) to *generate embeddings* of texts or images for downstream applications such as text classification, topic modeling, or sentiment analysis. For example, Puranam et al. (2021) examines the effect of a mandated minimum wage increase on consumer perceptions of service quality, leveraging several fine-tuned transformer models to analyze the frequency and sentiment of discussions about service quality attributes. Schöll et al. (2024) uses BERT to examine how politicians adjust their responses to citizen feedback on social media. Wei et al. (2024) encodes responses to open-ended survey questions using a GPT to identify motives for sustainable behaviors.

Second, researchers use generative tools such as GPT and DALL-E to *automate tasks* requiring processing texts or images. For example, Brand et al. (2023) applies large language models to study consumer preferences; Li et al. (2024) investigates the use of large language models as substitutes for human participants in perceptual market mapping; Castelo et al. (2024) explores the creative capabilities of GPT in generating innovative product ideas; Ringel (2023a) investigates the potential of GPT to replace domain experts in classifying complex marketing constructs within microblogs; and Le Mens et al. (2023) investigates the potential of GPT to construct semantic similarity measures for book descriptions and political tweets, specifically focusing on the typicality of texts.

Finally, researchers have started to apply transformers to *marketing-specific datasets* other than text and images. Unlike the applications discussed above, this research cannot rely on pretrained models, but requires training the entire transformer model from raw data. Lu and Kannan (2024) shows that transformers can be used to analyze sequences of customer interactions in multi-channel marketing. Their model learns the relationships between customer touchpoints, and predicts subsequent interactions and conversion probabilities. Bianchi et al. (2020) train a transformer on e-commerce clickstreams. The model uses input sequences to predict two types of events: First, which product is the last product added to a cart? Second, will the session end with an add-to-cart action or not?

Our research builds on this emerging literature by developing a custom transformer designed explicitly for market basket data in brick-and-mortar retailing. We extend this literature by proposing modifications that enable the transformer to handle unordered sets of products and incorporate covariates such as location and time. These adaptations allow the model to contextualize purchase probability predictions to the situational characteristics of the shopping trip. We further show that the standard transformer does not outperform random prediction for long-tail products, which account for most of the assortment. To address this crucial limitation, we improve the masking strategy during model training and develop a new data curation strategy that generates additional training examples. These changes substantially enhance the model's performance across the entire assortment and ensure that the transformer models high-frequency and long-tail products equally well. Finally, we illustrate our model's adaptability by fine-tuning it to predict coupon redemptions in a field experiment, demonstrating that our MBT can serve as a foundation model for retail analytics.

2.2 Market Basket Analysis

Our work also adds to the rich literature on market basket analysis (MBA), which seeks to understand consumer behavior in retail settings. Researchers have proposed a variety of methods for market basket analysis (MBA) that add to our understanding of consumer

behavior in retail settings. Agrawal et al. (1993) laid the groundwork for MBA by introducing an algorithm for mining association rules between products in shopping baskets. Subsequently, researchers have proposed additional metrics that quantify product co-occurrence in shopping baskets (for a summary, see Blattberg et al. 2008). For example, Zhang (2000) proposed an improved association rule metric that normalizes scores against the expected purchase frequency and differentiates between positive and negative associations.

In addition to data mining techniques, researchers have also developed econometric models for analyzing basket data. Manchanda et al. (1999) proposed a multivariate probit model for multicategory purchase decisions, providing insight into both complementarity and co-occurrence in category choice. Russell and Petersen (2000) introduced a multivariate logit model that allows for any type of demand relationship across product categories. Both studies apply their model to four grocery categories. These choice models incorporate cross-category relationships through additional model parameters. Boztuğ and Reutterer (2008) expanded the scope of MBA by combining multicategory choice models with a data-driven approach for basket selection, thereby enriching the understanding of market baskets through customer segmentation. A key benefit of choice models is that their utility framework is grounded in psychological theory, which makes them a suitable tool for policy evaluation.

Researchers have proposed machine learning approaches for MBA to address the limited scalability of econometric methods. Mild and Reutterer (2003) explored collaborative filtering for predicting cross-category purchases based on market basket data. To capture (dynamic) purchase behavior in large-scale retail settings, Jacobs et al. (2016, 2021) developed models based on Latent Dirichlet Allocation (LDA) that leverage customer-level purchase histories to identify purchase motivations. They use these purchase motivations to predict customers' future purchases. Gabel et al. (2019) proposed Product2Vec and P2V-MAP for analyzing market structure using product embeddings. Ruiz et al. (2020) introduced SHOPPER, a sequential probabilistic model of shopping data designed to capture item interactions based on product embeddings and the effect of marketing interventions. Gabel and Timoshenko (2022)

proposed a scalable deep-learning model for multicategory product choice in large assortments, focusing on brand-level analysis and coupon personalization of loyalty card customers. The model captures how a price promotion for a product affects products' purchase probabilities, not only of the promoted product but of all products in large retail assortments. Ringel (2023b) used language models to learn product relationships from clickstream data. They showed that competitive relationships among products can depend on the attributes relevant in a submarket if products compete in multiple parts of a market.

Our research builds on and extends this literature by applying the transformer architecture to MBA. Similar to other machine learning methods, the MBT scales well to large retail assortments and requires few assumptions about market structure. We demonstrate that transformers can learn systematic co-occurrence patterns in shopping baskets, similar to multivariate choice models (Manchanda et al. 1999, Russell and Petersen 2000). Incorporating covariates into the transformer model, addresses the complexity inherent in marketing environments, where multiple factors influence consumer behavior simultaneously (Liu 2023). Our analysis reveals that the MBT learns basket composition more accurately than alternative models. To ensure that the model performs equally well for high-frequency and long-tail products, we modify the training approach and develop a new data curation strategy. With these features, the MBT can serve as a foundation model for retail analytics that can be easily adapted to various tasks with minimal model changes and additional data.

3 Model Input, Model Architecture, and Training Procedure

Adapting transformers to market basket data requires three steps: First, define a neural network architecture that inputs shopping baskets and learns basket composition through scaled dot-product attention (Vaswani et al. 2019); Second, implement a robust training approach that ensures meaningful neural network weights even for rarely observed products; And third, incorporate covariates that characterize shopping trips and capture how purchase probabilities vary across these trips. We detail these steps in the following sections.

3.1 Model Input

We consider a retailer with an assortment that contains J products $j = 1, \ldots, J$. We model purchases at the product level, without any further aggregation to brand or category levels. A shopping basket b of size n_b is a set of products $x_b = \{j_1, \ldots, j_{n_b}\}$ a customer purchases during a single shopping trip. We focus on applications in offline retailing, so shopping baskets are *unordered sets* of products. Retailers track purchases through their checkout systems.

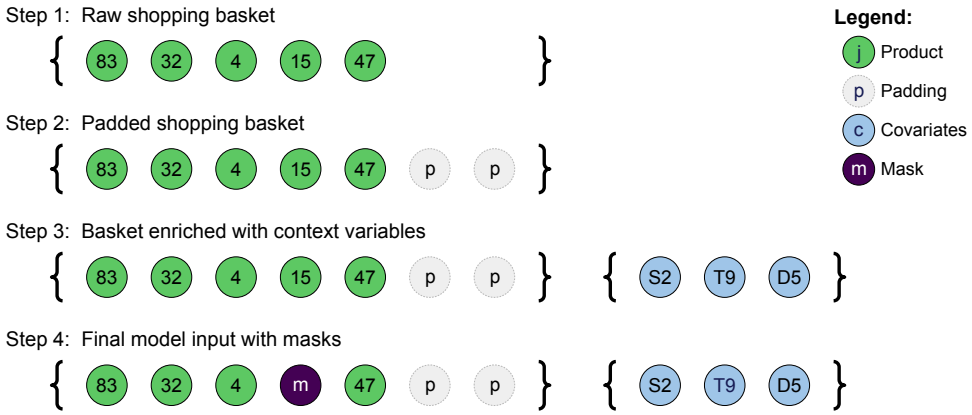
We conduct four data preprocessing steps before we input the market basket data into the transformer (see Figure 1). The starting point is a batch of B shopping baskets (step 1). For simplicity, we depict the case $B = 1$. We do not input product names, descriptions, or one-hot encoded vectors, but a set of product IDs, such as UPCs (universal product code).

Next, we add a pad product p to the product set $S - n_b$ times (step 2). Padding is a common practice in deep learning and facilitates an efficient neural network implementation with tensor multiplication. The pad product is ignored when calculating the loss, so it does not affect the neural network weights. Each padded basket is a set $x_b = \{j_1^b, \ldots, j_S^b\}$ with $|x_b| = S$.

We further input C covariates (step 3). Product purchase probabilities and basket composition likely vary with shopping trip characteristics. For example, customers visiting different stores might have heterogeneous preferences and purchase different products. Customers might also buy some products more frequently in the morning (e.g., breakfast rolls) and others in the evening (e.g., refrigerated beer). Covariates allow the transformer to account for varying purchase frequencies and learn more detailed purchase patterns. The patterns captured by the covariates can also provide insights to marketers, as we will illustrate in Section 5.2. For each basket, the covariates form a set of IDs $x_c = \{x_1^c, \ldots, x_C^c\}$ with $|x_c| = C$.

In our empirical application, we use the store where the shopping trip takes place (`S`), the time of the shopping trip (`T`), and the day of the week (`D`) as covariates. Similar to products, each covariate is encoded as an ID (e.g., `S2` representing one of the retailer's stores) without providing any additional information about the similarity of covariate levels. The transformer

Figure 1: Input Data Preparation



Notes: The transformer inputs shopping baskets as unordered product sets (green). We add pad products p to achieve a fixed set size (light grey). We add shopping trip characteristics such as store IDs (`S2`) as covariates. We replace some products with a mask product m (dark purple).

learns the meaning of covariates by observing the content of shopping baskets. It can be extended to process additional covariates, such as customer characteristics, basket size, basket value (revenue), promotion indicators, or weather data.

Finally, we randomly mask product IDs in baskets during training (step 4). In Figure 1, we replaced Product 15 with the mask product m. Masking is part of the self-supervised transformer training. The learning task is to predict the product replaced with the mask product, based on the surrounding products and the covariates. In every batch of baskets, we dynamically determine which IDs are masked, mirroring the masking proposed by Liu (2019) for NLP transformers. Dynamic masking allows us to learn more from the limited basket data since different products will be masked across the training in each training epoch. In the base transformer model, we use a masking probability of $p_{mask} = 15\%$.[1]

The transformer forward pass (see Section 3.2) outputs a probability distribution

$$p_j(x_b) \coloneqq p(m = j \,|\, x_b,\, x_c) = \{p_1^b, \ldots, p_J^b\} \tag{1}$$

over all products in the assortments. The perfect prediction is a probability of 1 for the true

[1]Many transformer implementations in NLP first determine whether a word should be masked, and then replace only 80% of the words with a mask word, replace 10% of the words with another (randomly selected) word, and keep the original word for the remaining 10% (Devlin et al. 2019). This approach is believed to improve the model training, so we incorporate it in our transformer implementation.

(masked) product and a probability of 0 for all other products, so we define the model (log) loss as

$$L = \sum_j \log p_j(x_b). \tag{2}$$

This learning task forces the model to predict missing products in incomplete baskets, which ensures that the basket transformer accurately captures patterns in basket data. Understanding purchase patterns and basket composition is the basis for many retailing analytics tasks, which is an important requirement for a foundation model.

A challenge for training transformers in retailing is that most shopping baskets are small. If baskets contain four products, a masking probability of 15% would mask one product in every second basket, while the remaining baskets do not contribute to the training. This amplifies the problem of data sparsity. We therefore propose and evaluate two additional transformer modifications in Section 5. We modify the masking procedure by forcing a minimum number of n_{mask} masked products in each shopping basket, and we propose approaches for generating additional training examples from the available training data. We show in the empirical application that these modifications substantially improve model performance, particularly for long-tail products, for which data is limited (see Section 5.3).

3.2 Model Architecture

Next, we describe the model architecture that calculates the probability distribution in Equation 1 (see Figure 2). We input the padded product set x_b and the basket-specific covariates x_c. The transformer uses a product embedding E_j to turn each product ID in the product set into a vector of length D. These vectors represent products in a latent attribute space that captures information such as product categories, brands, prices, etc. (Gabel et al. 2019). Similarly, covariates, such as store and time of day, are represented by D-dimensional covariate embeddings E_c.[2] We then combine product vectors $v_{1..S} = E_j(x_b)$ and covariate vectors

[2]If covariates are continuous variables, they should first be discretized and then represented in the transformer by additional embeddings.

$v_{1..C} = E_c(x_c)$, resulting in a tensor of size $B \times (S + C) \times D$ for each batch. B is the batch size, S is the length of the (padded) product set, C is the number of covariates, and D is the embedding size.

To capture relationships between products and to model products' context-dependent[3] purchase probabilities, we feed the product and covariate vectors into a stack of N ($\geqslant 1$) transformer modules. Each module first calculates the scaled dot-product attention (Vaswani et al. 2019):

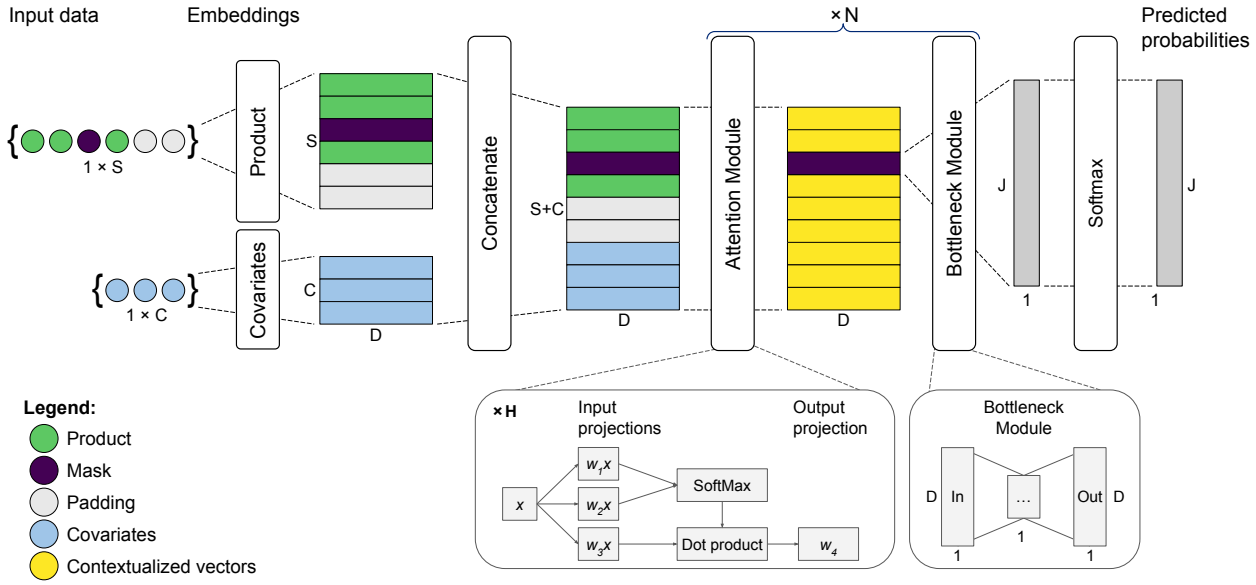
$$\text{Attention}(x) = \text{softmax}\left(\frac{(w_1\, x)(w_2\, x)^T}{\sqrt{D}}\right) w_3\, x \tag{3}$$

where $x = [v_{1..S},\, v_{1..C}]$ is the output of the embedding layer. The attention scores $(w_1\, x)(w_2\, x)^T$ model the relationships among products and covariates. Within each attention module, we use H parallel attention heads. In each head h, the input x is projected using head-specific weight matrices w_1^h, w_2^h, and w_3^h to create distinct representations for the attention calculation. Each head independently performs self-attention (see Equation 3), allowing it to focus on different aspects of basket composition and the effects of covariates. The head outputs are concatenated and linearly projected back to the embedding space by a matrix w_4 to produce a final attention output with the same dimensionality as the attention input. The contextualization by the attention module captures that products' purchase frequencies can depend on other products in the basket and the characteristics of the shopping trip.

Following the attention module, each transformer module includes a bottleneck module to process contextualized vectors further. The bottleneck only operates on the embedding dimension and does not change product relationships; it enables the transformer to model non-linear patterns in the data. We stack N transformer modules, each consisting of an attention module and a bottleneck module, and we use residual connections and layer normalization as proposed by (Vaswani et al. 2019).

[3]We use "contextualization" as a technical term to refer to the effect of the attention layer on the input vectors. We do not use "context" to describe the information captured by covariates. Instead, we refer to the "shopping context" as shopping trip characteristics or covariates.

Figure 2: Model Architecture



The last layer of the MBT is a linear projection to expand the D-dimensional vector for the masked product to the size of the assortment J. The resulting logits vector l_j is scaled to probabilities by a softmax transformation:

$$p_j = \frac{\exp\{l_j\}}{\sum_k \exp\{l_k\}}. \tag{4}$$

These probabilities capture the likelihood that product j occurs in a basket characterized by its products and covariates.

3.3 Training Procedure

We learn the model parameters

$$\theta = \left[E_j,\, E_c,\, \left(w_{1..3}^{Att,1..H},\, w_4,\, w^{FF},\, b^{FF}\right)_{1..N}\right] \tag{5}$$

through batch gradient descent (Hertz 2018). In the backward pass, we calculate the cross-entropy loss from the predicted probabilities

$$L = \sum_j \log\left[p_j(x_b)\right] = \sum_j \log\left[\frac{\exp\{l_j\}}{\sum_k \exp\{l_k\}}\right], \tag{6}$$

and calculate the gradients of the loss function with regard to the model parameters. The loss function compares the predicted probability $p_j(x_b)$ with the true distribution, a one-hot vector

indicating the index of the product in the retailer's assortment that was replaced by the mask product. Based on the gradients, the model updates the parameters to minimize the loss. We compute gradients through backpropagation and use the ADAM adaptive moment algorithm estimation to optimize the model parameters (Kingma 2014). Training the model in batches allows distributed computing and does not require having all training data in memory, so model calibration remains feasible for large data sets.

3.4 Discussion

The proposed MBT model differs from classic NLP transformers in three ways. First, it operates on unordered sets of products rather than ordered word sequences. In traditional NLP models, positional encodings are crucial for capturing the sequential nature of language. For example, grammar rules, the logical structure of the content, and language styles dictate the order of words in sentences and paragraphs. The positional encoding captures such patterns. In most brick-and-mortar settings, such a sequential order does not exist. Instead, our model relies on product embeddings to capture the underlying market structure and product relationships. This allows the basket transformer to learn product substitution and complementarity from purchase co-occurrence data (similar to Gabel et al. 2019), thereby providing a robust foundation for the basket transformer. One can think of market structure as the grammar of market basket data.

Second, the MBT incorporates covariates, which capture differences across shopping trips. The transformer leverages covariates to account for the impact of external factors on product purchase patterns, which enables it to adjust predictions based on the specific characteristics of each shopping trip. It is possible to integrate additional covariates, such as promotional activities, pricing strategies, and seasonal trends, which could further refine its predictive accuracy and broaden its applicability in retail analytics.

Third, in contrast to most existing approaches for market basket analysis, such as P2V, LDA, and SHOPPER, the MBT can serve as a foundation model that can easily be adapted to new

tasks. Fine-tuning pretrained transformers is a common method for adapting the model to a new task not covered in its initial training. This process can involve adding a new output layer tailored to a specific task—such as binary or multilabel classification—and updating the model's weights using labeled data. For classification tasks, the model learns how to assign one or more labels to each input from new and much smaller training data, adjusting its general understanding of purchase patterns to more specialized applications. To prevent overfitting and excessive changes to the pretrained model during fine-tuning, a lower learning rate and smaller batch sizes are typically used during fine-tuning. Examples of fine-tuning tasks include predicting coupon redemptions for specific products (see Section 6), segmenting customers based on the content of a shopping basket, or predicting the likelihood of customer churn. These tasks benefit from the transformer's ability to capture nuanced relationships among products and additional attributes of market basket data.

4 Application to Market Basket Data

In this section, we present results for applying the simplest version of the MBT to data from a German grocery retailer. We discuss the model implementation, evaluate its performance, and identify a limitation of the basic transformer implementation that curtails its applicability to long-tail products. The performance evaluation includes a benchmark with alternative approaches for modeling basket composition.

4.1 Data Set and Descriptive Statistics

We obtain market basket data from 147 stores of a national German grocery retailer. All data were collected between January 2016 and December 2016. In total, the data set contains 38 million shopping baskets. Each shopping basket is a transaction record that details the products purchased together during a single shopping trip.

We use 75,000 randomly sampled baskets as a test set for evaluating the model's predictive performance. We randomly sample 20 million baskets from the remaining baskets for our

training set. We limit the size of the training set, so all baselines can process it.[4] We use the remaining baskets to construct additional test and training sets (see Section 5) for more detailed analyses of predictive performance on long-tail products.

Table 1 provides summary statistics for the training data. The assortment consists of 30,383 products from 143 product categories. The average basket size is 9.24, with an interquartile range of 5 ($Q1 = 6$, $Q3 = 11$). Most products are purchased infrequently. The average product purchase rate, defined as the fraction of shopping baskets containing a focal product, is 0.046% with an interquartile range of 0.035% ($Q1 = 0.005\%$, $Q3 = 0.040\%$). The maximum product purchase rate is 8.3%. The statistics reported in Table 1 are typical of the German grocery retail industry at large.

Table 1: Market Basket Data (Training Data)

Variable	**Value**
Number of baskets	20,000,000
Average basket size	9.24
Basket size quartiles Q1 and Q3	[6, 11]
Basket size interquartile range	5
Number of products	30,383
Number of brands	2,950
Number of categories	143
Average purchase rate	0.046%
Purchase rate quartiles Q1 and Q3	[0.005%, 0.040%]
Purchase rate interquartile range	0.035%

4.2 Model Training

Table 2 summarizes the default MBT hyperparameters we use in this section. Given the limited volume of training data, we choose a low embedding dimensionality ($D = 128$) and a single transformer layer ($N = 1$) to reduce the risk of overfitting. To compensate for the

[4]The implementation of SHOPPER on GitHub does not support processing all products and shopping baskets. To use the entire data set of 38 million baskets, we would have to drop a third of the product assortment, which would prevent SHOPPER from predicting purchase probabilities for these products. We verified that the substantive results for the other models do not change when using all 38 million baskets.

simplicity of the model, we use multiple attention heads ($H = 8$) to capture diverse patterns in product co-occurrence and the influence of covariates.

In Section 5, we evaluate several modifications to the transformer and the training procedure that substantially improve its performance. These modifications include adding covariates to the model, enforcing a minimum number of masked products, and generating additional training examples. The affected hyperparameters are marked with an asterisk in Table 2.

Table 2: Default Model and Trainer Hyperparameters

	Variable	**Value**
Model	Dimensionality of the product embedding	128
	Number of covariates *	0
	Product set size (excluding covariates)	48
	Number of attention modules	1
	Number of attention heads	8
	Number of bottleneck modules	1
Trainer	Default masking probability *	15%
	Default minimum number of masked products *	0
	Addtional training examples *	No
	Number of epochs	10
	Learning rate	Adjusted in optimizer

Note: The parameters marked with an asterisk are the focus of Section 5, where we include covariates, enforce a minimum number of masked products, and generate additional training examples.

4.3 Basket Completion Modeling Task

A good foundation model captures structure and patterns in market basket data. This includes accurately modeling (1) which products co-occur in shopping baskets and (2) how shopping trip characteristics affect basket composition. To evaluate how well the transformer learns these purchase patterns, we use a task that we call *basket composition modeling* (BCM).

Consider an example basket that contains the products:

{`milk, bread, toothpaste, eggs, laundry detergent`}

Products are represented by integer IDs (e.g., UPC); we use product names in this section to simplify the exposition. In each basket, we remove one product and task the model to predict

the missing product, based on the remaining products in the basket. For example, if `bread` is removed from the example basket, the input might appear as:

{`milk, _____, toothpaste, eggs, laundry detergent`}

Although the products in the basket are unordered, the model learns product embeddings that capture market structure, revealing common product associations. In this example, the model might predict that the masked product is `bread` by recognizing associations within the remaining items in the basket. The model outputs a probability distribution over the entire product assortment for the missing product.

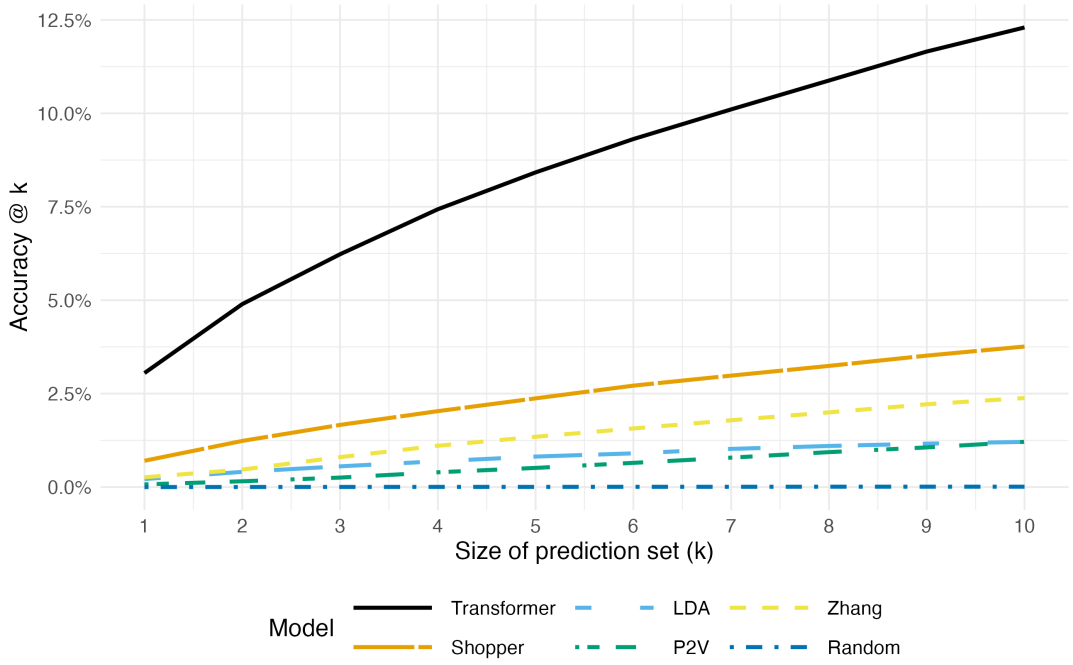
We implement this task on the test set. To evaluate model performance, we use accuracy@k: we take the k products with the highest predicted probabilities and evaluate if the true product is among them. This produces a binary outcome that is one if the true focal product is among the top k predictions and 0 if not. We average this binary outcome across all test baskets to obtain the final accuracy score for a given value of k. Accuracy@k assesses the model's ability to correctly predict the missing product within the k top-ranked predictions. Importantly, we predict missing *products* (N = 30,383), not aggregations such as brands or categories that would simplify the prediction task.

4.4 Model comparison

We compute accuracy@k for the most basic basket transformer (no covariates, standard masking strategy, no additional training examples), and compare its performance with five alternative models: SHOPPER (Ruiz et al. 2020), P2V (Gabel et al. 2019), LDA (Blei et al. 2003), the Zhang metric for market basket analysis (Zhang 2000), and a random prediction baseline. Web Appendix A provides detailed descriptions of the model implementations.

Figure 3 presents the results for values of k ranging from 1 to 10. The transformer's accuracy for $k = 1$ is 3.8%, which is 1,000 times larger than the accuracy of the random baseline ($1/30,383 = 0.003\%$). Among the benchmark models, SHOPPER performs the best, followed by the Zhang metric, with P2V and LDA showing similar but slightly lower performance than

Figure 3: Accuracy in Basket Composition Modeling Task



the Zhang metric. As expected, the accuracy of the random baseline increases linearly with k, reflecting the expanding size of the prediction set. All models significantly outperform the random baseline. The transformer achieves an accuracy@k more than double that of SHOPPER, the best alternative model. Remarkably, despite the very large number of products that could occur in each test basket ($N = 30,383$), one of the transformer's top-10 predictions is correct in 12.5% of the test baskets. Note that the two best models, the transformer model and SHOPPER, were specifically designed to capture basket composition.

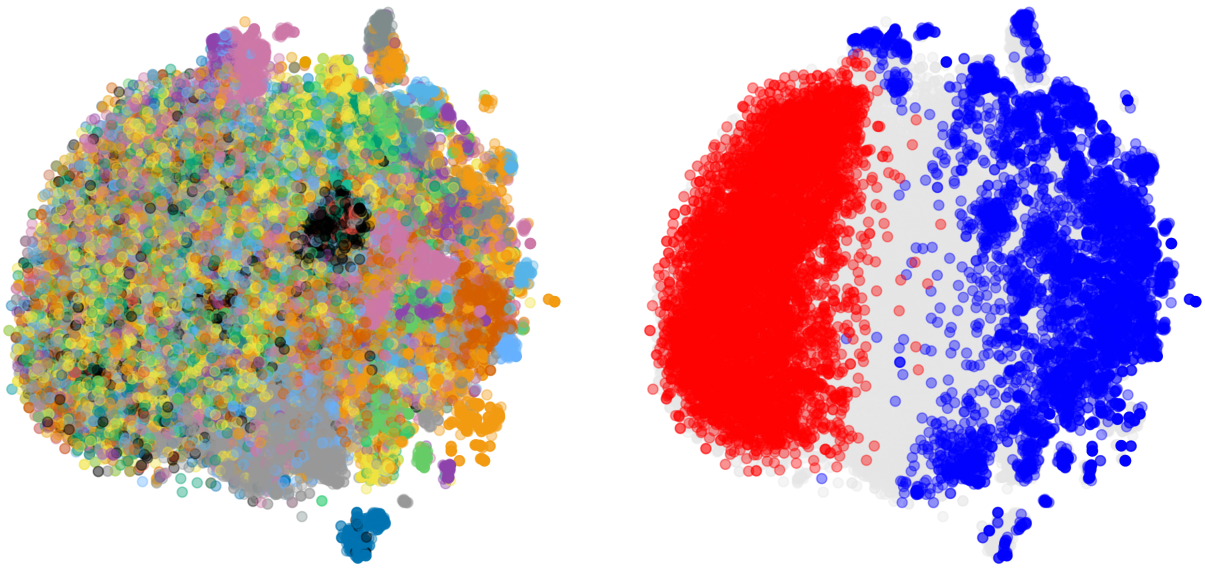
To better understand the transformer's predictive performance, we analyze its product embedding and attention weights in more detail. Figure 4 shows a product map based on the transformer's product embedding, with each bubble representing a specific product. We create a two-dimensional visualization of these high-dimensional embeddings by reducing the dimensionality with t-SNE (Van der Maaten and Hinton 2008).[5] The resulting product map provides a visually interpretable representation of the product embeddings, offering insights into the structure learned by the model.

[5]We use a random initialization for the products' map location and a perplexity of 30.

Figure 4: Product Map

(a) Category Color Overlay

(b) Frequency Color Overlay



Notes: t-SNE representation of the products embeddings. The two panels show the same product map; we use different color overlays. In panel (a), the bubble color indicates product categories. In panel (b), the bubble color indicates products' purchase frequency. Blue denotes products in the top purchase frequency quintile, and red products in the bottom purchase frequency quintile.

The colors in panel (a) represent different product categories. Interestingly, the product map tells two stories. On the right half of the map, products form clusters, each predominantly containing products from the same category. The clustering suggests that the embeddings capture underlying product relationships and similarities. Moreover, clusters often form superclusters, reflecting higher-level relationships between different product categories. In contrast, the left half contains thousands of products, with no apparent clustering. Neighboring products often belong to different categories, resembling the result of "emptying a bag of Skittles on a table." To reveal which products are affected by the lack of structure, we recolored both maps using products' purchase frequency in panel (b). Products in the lowest frequency quintile are colored red, and products in the highest frequency quintile are colored blue. We find that the unclustered products on the left side of the map have a low purchase frequency. In contrast to high-frequency products, the transformer did not learn meaningful product vectors.

Figure 5: Transformer Attention Weights

Note: Darker tiles indicate stronger attention between products.

We would expect that noisy product vectors also limit the transformer's ability to capture basket composition in the attention layer. Recall that the attention weights determine how strongly product vectors are correlated. Attention weights are directly derived from the basket's product vectors x, and the attention weights w_1 and w_2:

$$w_{Att} = \text{softmax}\left(\frac{(w_1 x)(w_2 x)^T}{\sqrt{D}}\right). \tag{7}$$

Attention weights for products that are related to each other (e.g., purchase complements) should be large, whereas attention weights for unrelated products should be close to zero. Figure 5 visualizes the learned attention weight matrix as a heatmap for an example basket, averaged across the eight attention heads. Each weight depicts how the product vector of the row product modulates the vector of the column product. Intuitively, we expect a strong association between tortilla chips and salsa (complements), and between eggs and milk (similar purchase cycle).

Overall, attention heads display sparse activation patterns, with most attention weights being small and only a few being activated. We find that the strongest association is between tortilla chips and eggs. We do not observe an association between tortilla chips and salsa. Both findings are counterintuitive. Overall, the attention weights seem to be larger for more frequently purchased products (eggs and milk). We find similar results for other shopping baskets, indicating that the transformer struggles with long-tail products.

5 Long-Tail Products

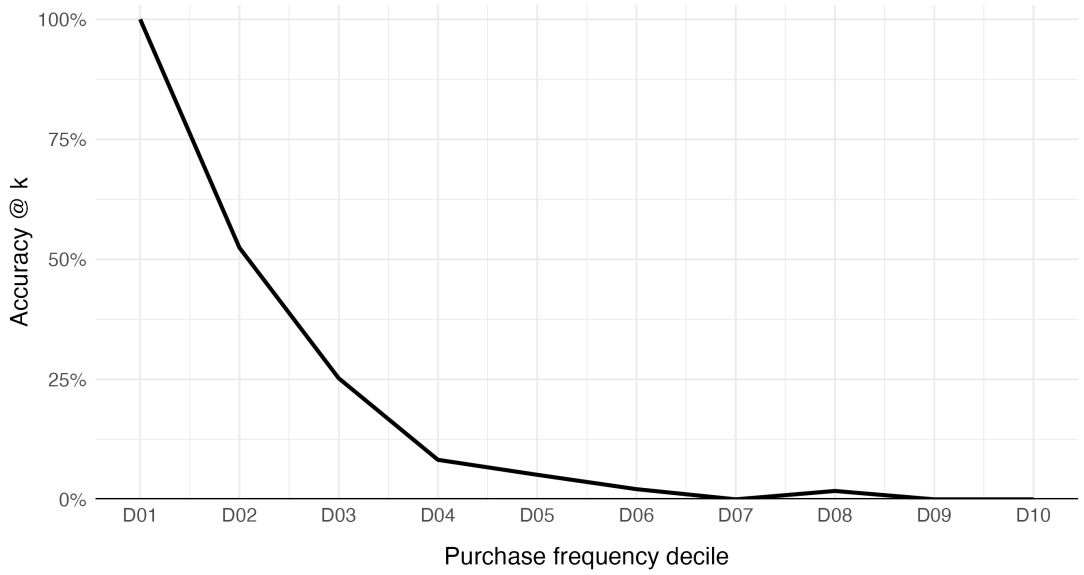
A good foundation model must perform equally well for all products in the assortment. Therefore, we examine the transformer's performance as a function of purchase frequency in more detail. We then assess whether our three transformer modifications—covariates, minimum masking, and synthetic training examples—can address the long-tail problem identified in the previous section.

5.1 Performance of the Basic Transformer Model

To analyze long-tail products, we proceed as follows: First, we construct a new test set from the 18 million shopping baskets, which were not used in our previous analysis. For each product in the original training data, we sample 100 baskets containing the focal product, remove the focal product, and task the transformer to predict the removed product. This produces a test set comprising approximately three million baskets. We then bin products into deciles based on their purchase frequency, ranging from `D01` for the most frequently purchased products to `D10` for the least frequently purchased products. Each decile contains approximately 3,000 products. After predicting the missing products, we calculate the prediction accuracy@k=1 for each frequency bin. Unlike the initial data set, which emphasizes frequently occurring products, the new test set assigns equal weight to all products.

We present the results in Figure 6. The x-axis denotes products' purchase frequency deciles, and the y-axis shows the prediction accuracy@k=1, indicating how often the top prediction is correct. To facilitate the comparison across deciles, we normalize the accuracy to the value

Figure 6: Predictive Performance as a Function of Products' Purchase Frequency



Note: To facilitate a comparison of predictive performance across bins (and analyses), we normalize the accuracy to the value in `D01`.

in decile `D01`. The results reveal the highest predictive performance for products in `D01`, with accuracy declining exponentially as purchase frequencies decrease. The differences between bins `D01` to `D04` are statistically significant ($p < 0.05$). Notably, for half of the assortment, the model's accuracy is indistinguishable from random chance, with no significant improvement over a baseline accuracy of zero.[6]

The results show that the transformer's predictive accuracy is strongly tied to product purchase frequency. While the model performs well for frequently purchased products, its accuracy declines sharply for long-tail products. We posit that the performance drop stems from insufficient training data for long-tail products, limiting the model's ability to learn purchase patterns.[7] In the following sections, we propose and evaluate several modifications to the transformer training and architecture to address this issue. Our objective is to improve the accuracy for long-tail products while maintaining high accuracy for high-frequency products.

[6] We provide a version of the plot in Figure 6 with confidence intervals in Web Appendix B.

[7] An alternative explanation is that the performance decrease is caused by long-tail products co-occurring more frequently with other long-tail products. In Web Appendix C, we show that this is not the case.

5.2 Performance of the Transformer with Covariates

Two reasons motivate the inclusion of covariates: First, a good foundation model for retail analytics must capture as much information about the shopping trip as possible. Incorporating covariates enables the transformer to model purchase frequency heterogeneity as a function of shopping trip characteristics. Second, covariates might mitigate the low performance for long-tail products by contributing additional information about the shopping trip.

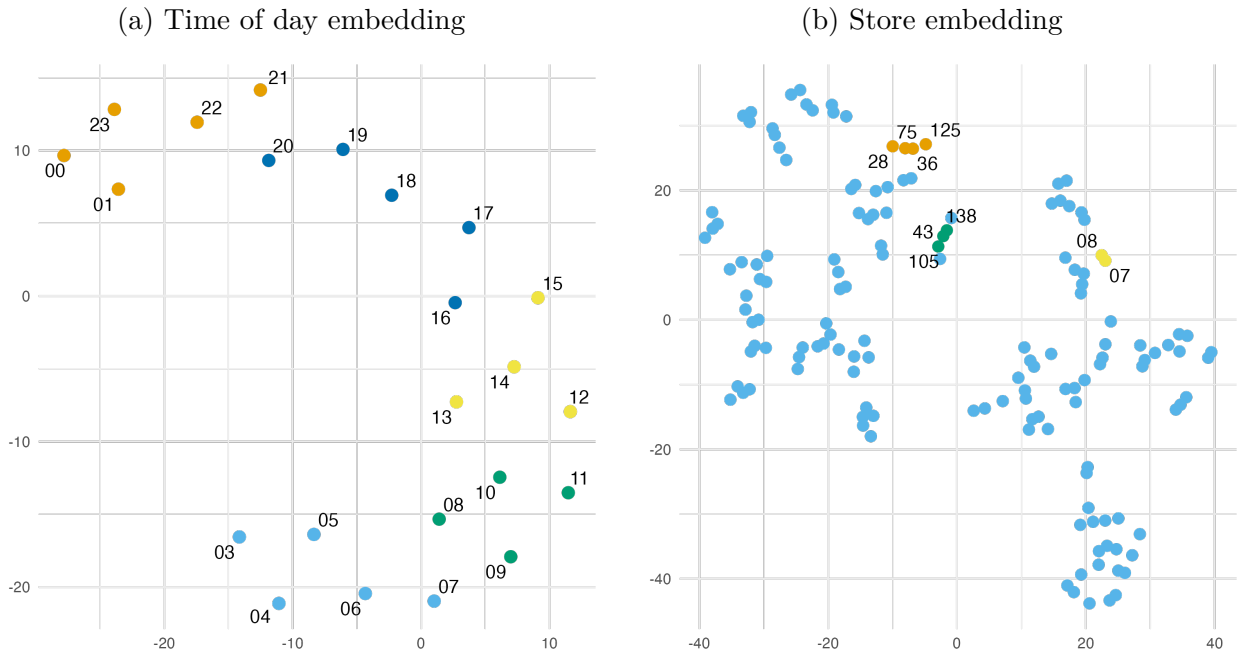
We first analyze the time and store embeddings to assess whether the transformer effectively captures covariate information. The t-SNE visualizations[8] of these embeddings, shown in Figure 7, reveal meaningful patterns. The left panel depicts the *time of day* embedding. Adjacent hours are positioned close to one another, reflecting their similarity. The cluster in the top left represents night hours (orange), while evening hours appear at the top (dark blue), afternoon hours on the right (yellow), morning hours are at the bottom right (green), and early morning hours at the bottom left (light blue). This visualization supports the model's face validity and suggests that shopping trips closer in time tend to have similar product purchase frequencies and basket composition.

In the right panel of Figure 7, we visualize the *store* embedding. We find that similar stores form clusters. For instance, stores in shopping malls in Berlin's tourist areas are clustered together (orange). Similarly, we observe several clusters that contain neighboring stores. For example, the stores in the green and yellow clusters are each located within 1.5 kilometers of each other. The store embeddings learned by the transformer indicate that it effectively captures location information and store similarity.

Notably, we do not provide any information about time or store similarity to the transformer model; we only input abstract IDs for every store and time of day (e.g., `S2` and `T9`), which the model then embeds in latent attribute spaces. The transformer independently learns the similarity of stores and different times during the day during the self-supervised training, just by observing basket content and covariates.

[8] We use a random initialization for the products' map location and a perplexity of 5.

Figure 7: Time of Day and Store Embeddings Learned by the Transformer

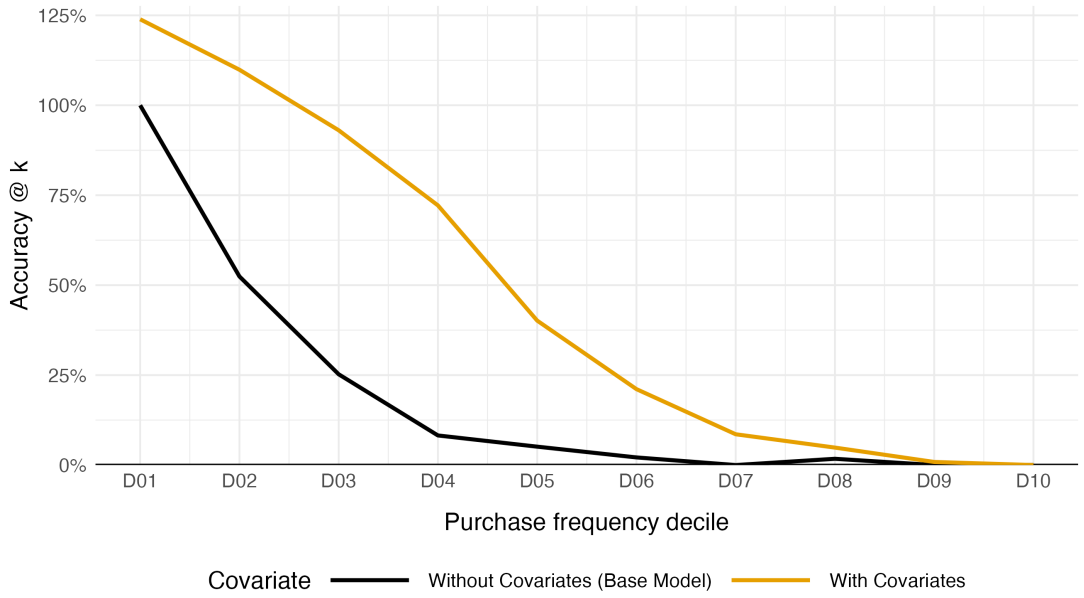


Notes: Each bubble in panel (a) represents a one-hour window in 24-hour format. The colors indicate different parts of the day: night hours in orange, evening hours in dark blue, afternoon hours in yellow, morning hours in green, and early morning hours in light blue. Each bubble in panel (b) is a store. We highlight three interesting store sets: stores in Berlin's tourist areas in orange, and two clusters that contain stores within 1.5 kilometers of each other in green and yellow.

Next, we evaluate how covariates affect the transformer predictions for the basket composition task. Figure 8 shows the prediction accuracy as a function of products' purchase frequency. The black line represents the accuracy of the base transformer in Figure 6, and the orange line the performance of the model with covariates. In addition to generating insights into shopping behavior, covariates substantially improve the model's predictive accuracy, with the largest relative improvement in deciles `D03` and `D04`. However, the long-tail problem persists; for products in deciles `D08` to `D10`, the model's predictive accuracy remains nearly random.

To illustrate that covariates capture meaningful variation in purchase rates, we study the predicted probabilities at the product level. Below, we discuss the results for a beer SKU (a bottle of Berliner Kindl). For beer, we would expect substantial differences in purchase probabilities across stores and time. In this analysis, we sample 250,000 shopping baskets and predict the probability that the focal product would occur in the baskets using the transformer

Figure 8: Effect of Covariates



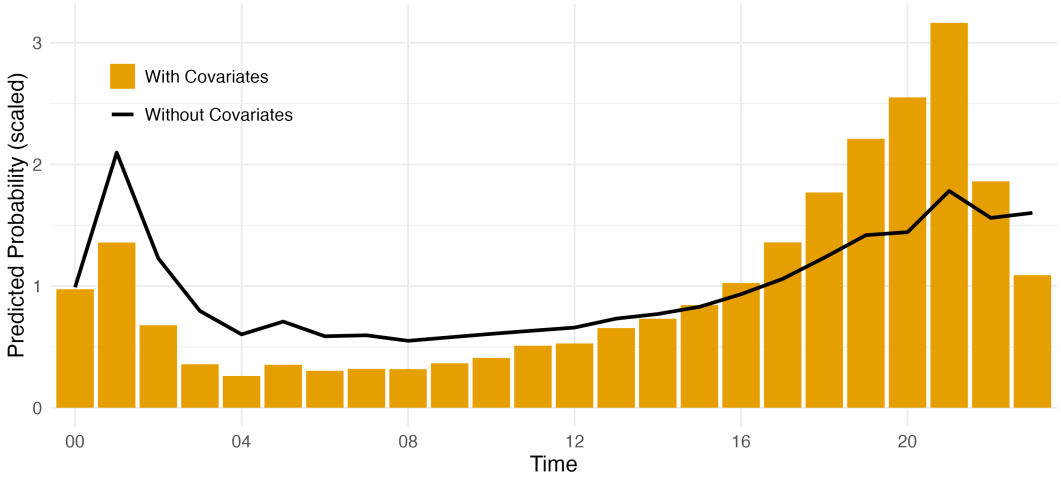
Note: To facilitate a comparison of predictive performance across analyses, we normalize the accuracy to the maximum value of the base model in Figure 6.

model with and without covariates. Figure 9 depicts the resulting probabilities, averaged for every level of the time and store covariates. We normalize the distributions by the average probability. For the store-level probabilities, we sort the probabilities by the model's output without covariates to facilitate the comparison. We find that including covariates increases the variance in the predicted probabilities (orange bars) compared to the model without covariates (black lines).

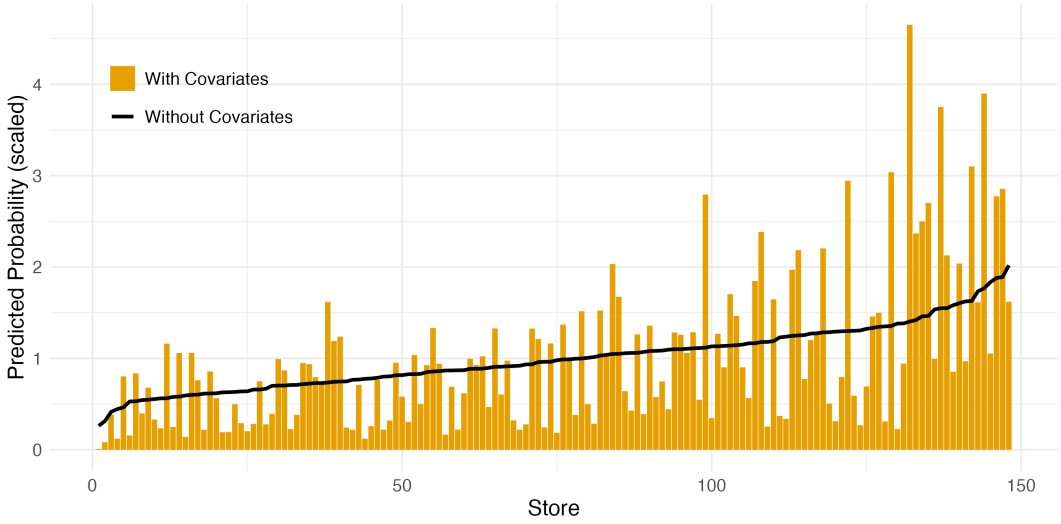
To assess whether the transformer with covariates captures the variance in purchase rates more accurately, we analyze its predictions at each covariate level. Specifically, we calculate the fraction of shopping baskets that contain the focal product for each time of day and store. We then correlate this measure with the predicted probabilities for both models. A higher correlation indicates that the predicted probabilities more accurately capture the observed variations in purchase rates. For the time covariate, both models produce probabilities that are highly correlated with the purchase rates ($\rho_{time}^{Cov} = 0.895$ with $p < 0.001$ and $\rho_{time}^{NoCov} = 0.886$ with $p < 0.001$). The transformer learns time-varying purchase probabilities, even

Figure 9: Average Predicted Probabilities for Time of Day and Stores

(a) Probabilities for time of day



(b) Probabilities for stores



Notes: We sample 250,000 shopping baskets and predict the average probability for a national beer brand with a transformer model with covariates (orange bars) and without covariates (black line). For stores, we sort the probabilities by the output of the model without covariates.

without time covariates. Baskets at different times contain different products. By conditioning purchase probabilities on the other products in the shopping basket, the transformer can use the basket content in the attention layer to adjust the purchase probability for the focal product over time.

For store-level predictions, using covariates increases the correlation substantially: the correlation increases by 87%, from $\rho_{store}^{NoCov} = 0.380$ ($p < 0.001$) to $\rho_{store}^{Cov} = 0.710$ ($p < 0.001$).

The content of shopping baskets does not sufficiently modulate the predicted probabilities to capture the empirically observed variations in purchase frequencies.

Our analysis demonstrates the value of covariates in the transformer model. The covariate embeddings capture information about time and location; this facilitates descriptive analyses (see Figures 7) or model adaptation in other analytics tasks (see Section 6). However, covariates do not entirely mitigate the low model performance for long-tail products.

5.3 Performance of the Transformer with Minimum Masking

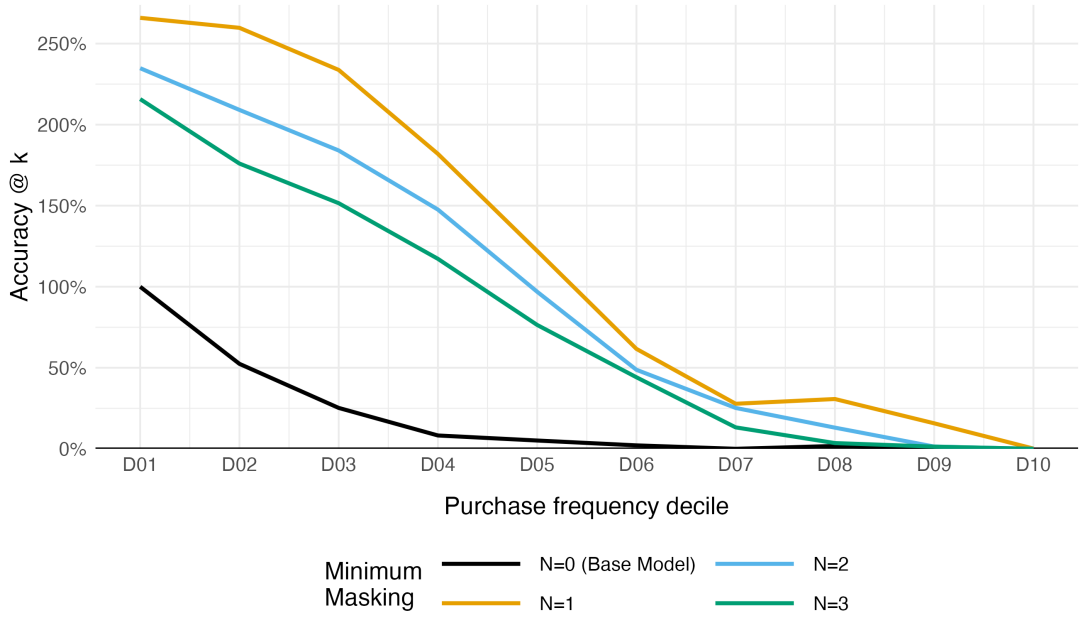
A challenge of the default masking strategy is that smaller baskets—which are common in retail datasets (see Table 1)—may not contain any masked products. Recall the example mentioned in Section 3.1: with the default masking probability of 15%, only half of the baskets containing four products will include a masked product. The other half does not contribute to the training, leading to inefficient use of available data (*undermasking*). Simply increasing the masking probability results in masking too many products, which reduces predictive accuracy as the model struggles with limited context (*overmasking*).[9]

We propose to mask at least $n_{mask} = 1, 2, \ldots$ products in every basket. When the default strategy masks too few products, additional products are randomly masked to meet the n_{mask} threshold. Our strategy mitigates undermasking and overmasking. By ensuring n_{mask} masked products in every basket, we increase the number of baskets contributing to the training. Consequently, all products—including long-tail products—are masked more frequently, which should improve model performance.

Figure 10 shows that all values of n_{mask} more than double the model's predictive performance, with the best result observed for $n_{mask} = 1$. Our minimum masking strategy significantly improves the predictive performance beyond the base model, not only for frequent products, but also for long-tail products. The relative improvement is largest in deciles `D03` to `D05`. Although minimum masking improves predictions, the performance for long-tail products

[9]Web Appendix D shows that tuning the masking probability does not solve the problem of low-accuracy predictions for long-tail products.

Figure 10: Effect of Minimum Masking Strategy



Note: To facilitate a comparison of predictive performance across analyses, we normalize the accuracy to the maximum value of the base model in Figure 6.

(D07 to D10) remains more than an order of magnitude lower than the performance for high-frequency products.

5.4 Training Data Curation

Next, we evaluate whether adding more training data can further improve transformer performance. We consider four data curation strategies to extend the initial training data set:

1. Randomly re-sampling baskets from the training set (*Random Sampling*)
2. Increasing the number of epochs, which leverages that we dynamically mask different products in every epoch (*18 Epochs*)
3. Collecting additional baskets not previously used during training (*New Data*)
4. For each long-tail product, randomly sample baskets containing the focal product from the original training set until each product occurs in at least 1,000 baskets (*Selective Data Sampling*)[10]

[10]Strategy 4 is related to the idea of weighted sampling in language modeling, where the best training data is not always consistent with the distribution of words in human language (Xu et al. 2024).

Strategy 4 (*Selective Data Sampling*) requires adding 15 million shopping baskets (+75%), resulting in a total of 35 million shopping baskets. Some products occur in less than 50 shopping baskets in the original training data. Our upsampling strategy increases the number of baskets by more than 20 times. We implement strategies 1 to 3, such that they also increased the available training data by 75%. For strategy 1, we randomly sample 15 million shopping baskets from the original training data. For strategy 2, we increase the number of training epochs from 10 to 18, such that the (same) baskets are used more often in the (longer) training. For strategy 3, we add 15 million new shopping baskets. We evaluate all data curation strategies with minimum masking $n_{mask} = 1$ and a masking probability of $p_{mask} = 15\%$.

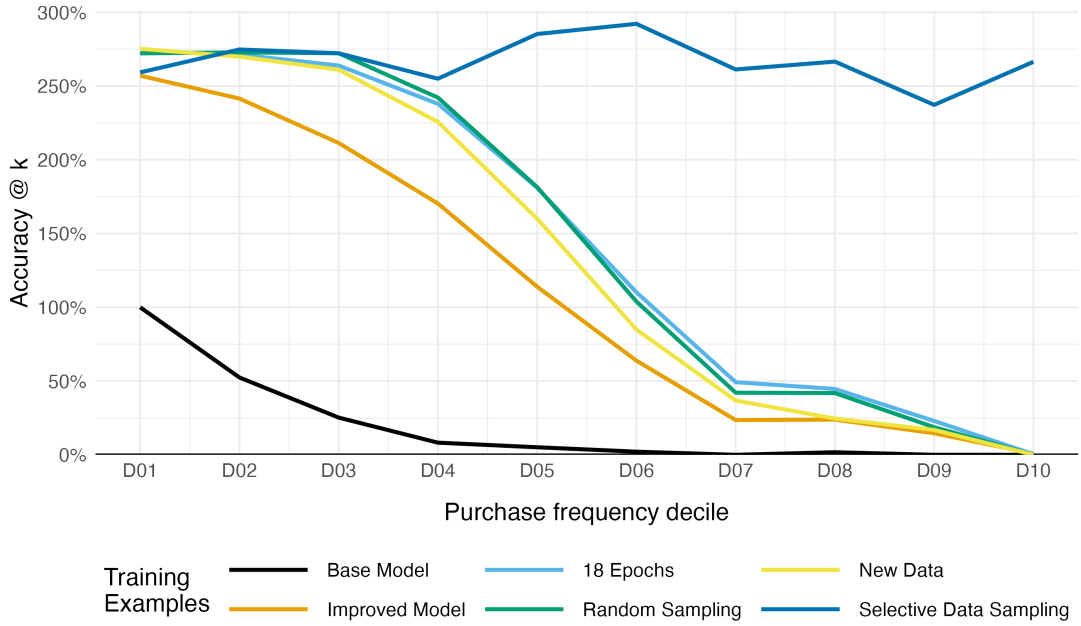
We expect improvements for long-tail products across all data curation strategies because they are now included more frequently in the training data. Selectively upsampling baskets (strategy 4) changes products' purchase frequencies; the resulting concept shift could lower model performance (Simester et al. 2020). However, given that our goal is predicting basket composition and not purchase frequencies, the concept shift may not be problematic.

Figure 11 shows the predictive performance of the base model, alongside the results for the models using the new data curation strategies. We find that strategies 1 to 3 substantially outperform the base model, and generate a small but significant improvement over a model with the same hyperparameters, but the original training data ("Improved Model"). The performance of strategies 1 to 3 does not differ significantly[11]. We observe the largest improvements from these three strategies for medium-frequency products. Consistent with our earlier findings, additional data provides minimal benefit for high-frequency products due to their frequent occurrence (i.e., diminishing returns), while the rarity of low-frequency products limits the benefit of increased data.

In contrast, strategy 4 (*Selective Data Sampling*) creates a substantial and statistically significant improvement even for products in deciles `D05` to `D10`. Notably, this strategy meets

[11] We provide Figure 11 with confidence intervals in Web Appendix B.

Figure 11: Effect of Training Data Curation Strategies



Notes: To facilitate a comparison of predictive performance across analyses, we normalize the accuracy to the maximum value of the base model in Figure 6. Appendix B contains this figure with CIs.

our objectives for an improved transformer model: The bias against long-tail products is mitigated, with significantly higher accuracy in the low purchase frequency deciles, while the performance for high-frequency products remains consistently high.

5.5 Source of Performance Improvement

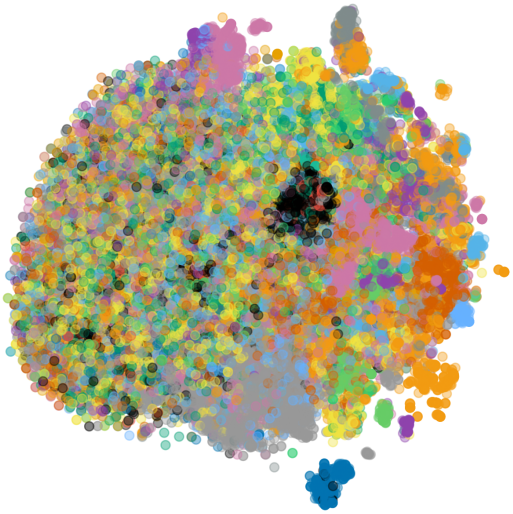
Next, we investigate the source of the performance improvement by examining two models in detail: the basic MBT implementation discussed in Section 5.1 (hereafter, `BASE` transformer) and our best-performing MBT implementation with selective data sampling for long-tail products (hereafter, `BEST` transformer).

First, we plot the product maps for the `BASE` and `BEST` transformers (see Figure 12). The maps for the `BASE` transformer are identical to those in Figure 4. For the `BEST` transformer in panel (b), products from the same category form distinct clusters. As discussed in Section 4, the product map for the `BASE` transformer lacks such a level of structure.

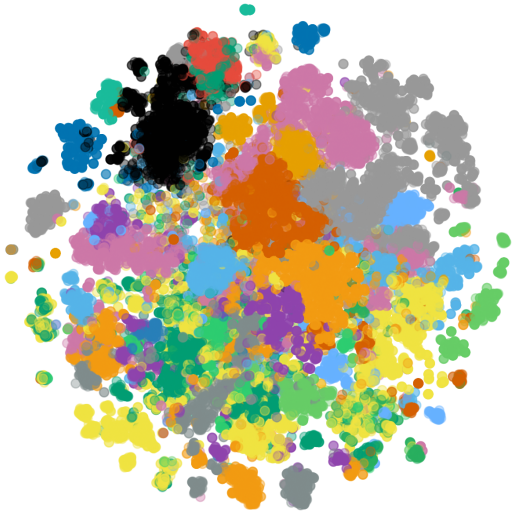
Again, we recolor both maps using the products' purchase frequency and depict the maps in

Figure 12: Product Map

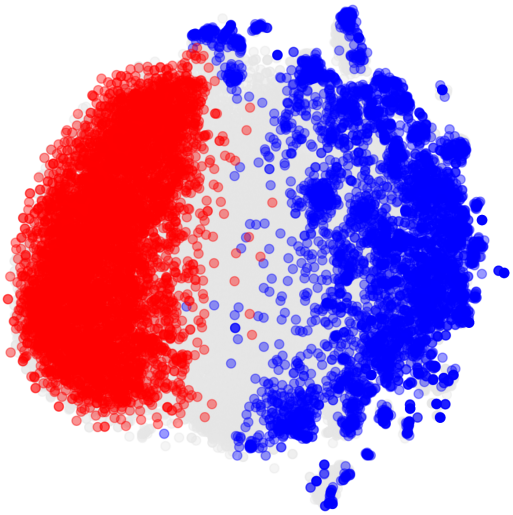
(a) BASE model with category overlay



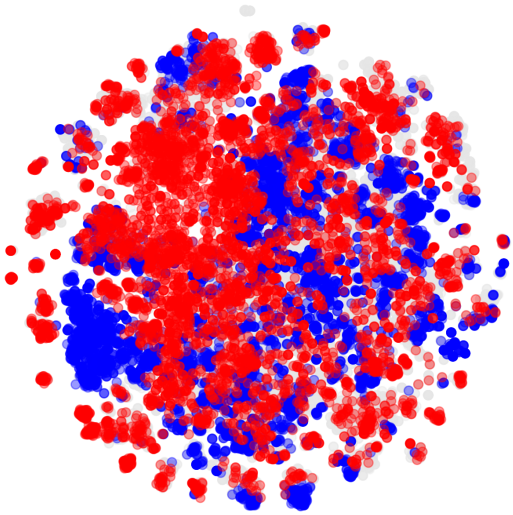
(b) BEST model with category overlay



(c) BASE model with purchase frequency overlay



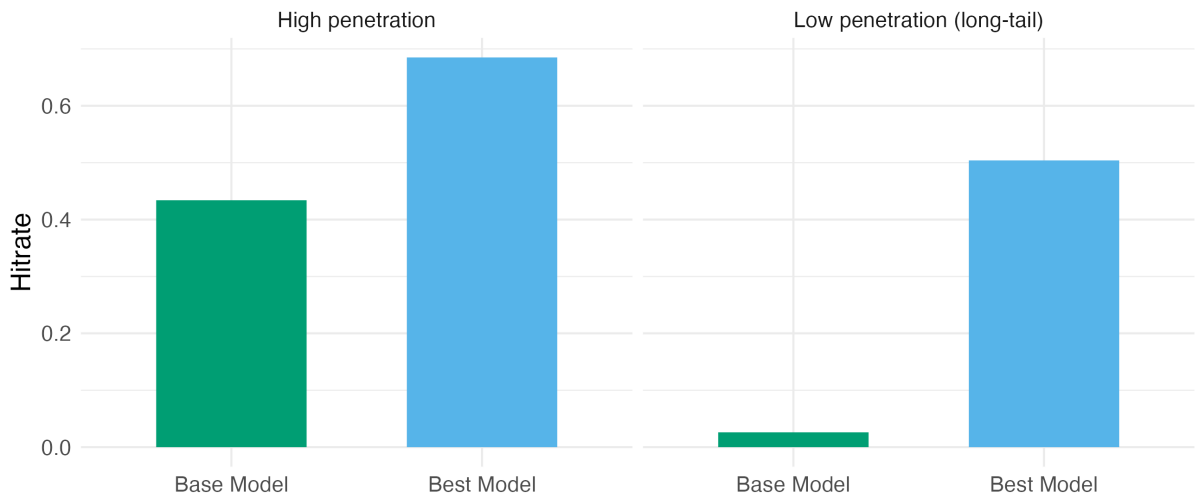
(d) BEST model with purchase frequency overlay



Notes: t-SNE representation of the product embeddings. The two panels show the same product map. We use different color overlays: In panel (a), the bubble color indicates the product category. In panel (b), the bubble color indicates product purchase frequency. Blue bubbles denote products in the top purchase frequency quintile, and red bubbles denote products in the bottom purchase frequency quintile.

panels (c) and (d). Products in the lowest purchase frequency quintile are colored red, and products in the highest purchase frequency quintile are colored blue. In contrast to the BASE transformer map, on which purchase frequency drives cluster formation, products on the BEST transformer map are organized by categories. Instead of forming a big cluster that resembles

Figure 13: Fraction of Same-Category Nearest Neighbors ($k = 10$)



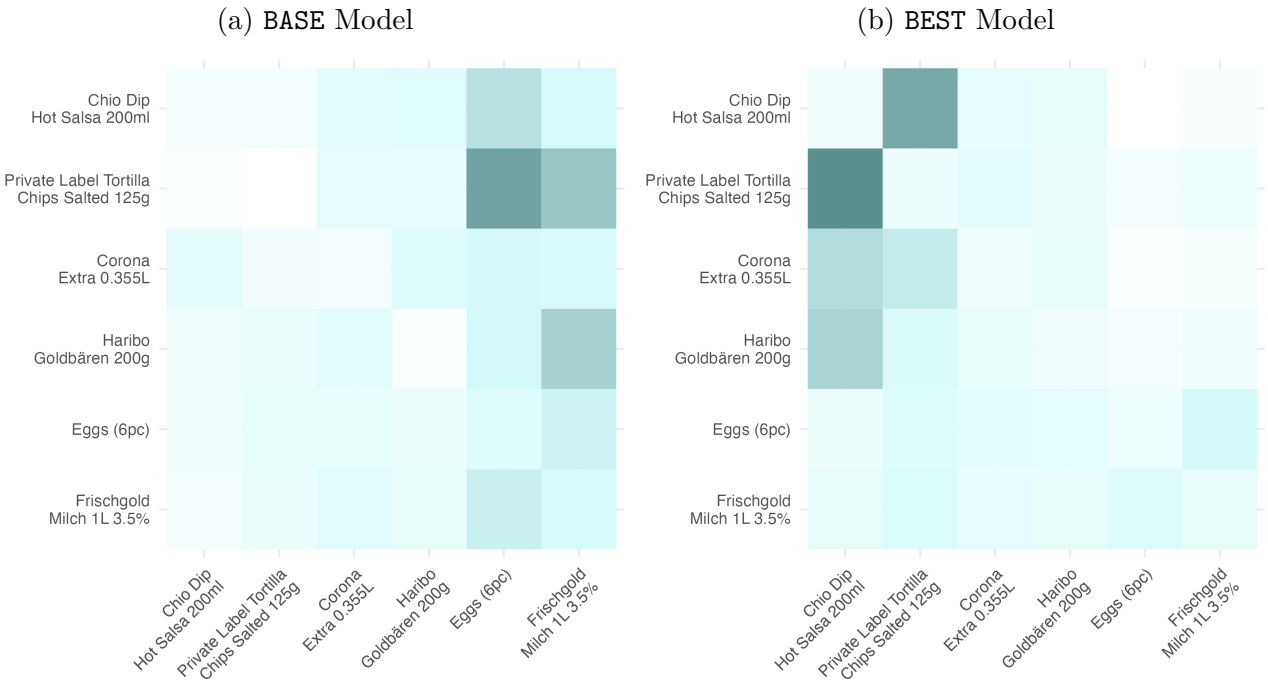
a "bag of Skittles," low-frequency products are now distributed in clusters across the entire product map.

To evaluate whether the low-frequency products are located in the *correct* product cluster, we conduct an additional analysis that evaluates the similarity of neighbors on both product maps. Products situated directly next to each other should be more similar and often associated with the same product category. To evaluate this for both product maps, we sample 100 low-frequency products (from decile `D01`) and 100 high-frequency products (from decile `D09`), and calculate the fraction of the products' neighbors belonging to the same category as the focal product (hit rate). Similar to Ringel and Skiera (2016), we evaluate the ten nearest neighbors. For both models, we then compute this hit rate measure separately for low-frequency and high-frequency products.

The results in Figure 13 show that both models achieve high hit rates for high-frequency products, although the hit rate for the `BEST` transformer is almost 50% higher than for the `BASE` transformer. While the hit rate for the `BASE` model collapses, the `BEST` model maintains a high hit rate even for long-tail products. Web Appendix E shows that this result is robust to the number of nearest neighbors.

Finally, we re-evaluate the attention weights for the shopping basket studied in Section 4.4.

Figure 14: Comparison of Transformer Attention Weights



Notes: Each tile in the heatmap is a single attention weight. The product vectors of row products affect the product vectors of column products. Color scales are identical in the two heatmaps.

Figure 14 visualizes the attention weights for the `BASE` and the `BEST` transformers in heatmaps. The overall sparsity in attention weight is similar for the two heatmaps (the color scales are identical), and we find very few large activated attention weights. Most importantly, the associations identified by the `BEST` transformer seem much more reasonable. Tortilla chips affect salsa and vice versa. Similarly, Corona beer and Haribo gummy bears are both related to hot salsa and tortilla chips. Notably, beer is more related to salted snacks than candy. Attention weights are no longer focused on high-frequency products; instead, they capture face-valid product associations.

In sum, these findings provide empirical support that the proposed modifications lead to substantial improvements.

6 Retail Analytics Application: Predicting Coupon Redemptions

Next, we demonstrate how the pretrained MBT can serve as a foundation model for other retail analytics tasks by adapting it to predict in-store coupon redemptions.

6.1 Foundation Models and Fine-Tuning

Foundation models are large, pretrained machine learning models that can be fine-tuned for tasks different from those they were originally trained on (Bommasani et al. 2022). During pretraining, foundation models learn general patterns and structures from extensive datasets. Fine-tuning adapts the pretrained model to a new downstream task using smaller, task-specific data. This approach uses the general knowledge captured during pretraining, making foundation models particularly effective when task-specific data is limited.

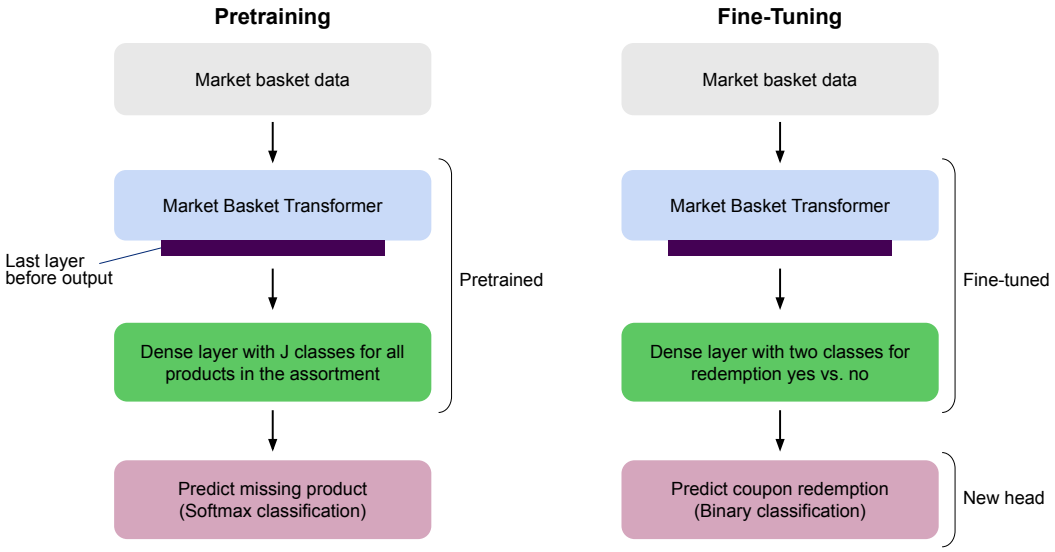
In natural language processing (NLP), foundation models like BERT (Devlin et al. 2019) and GPT (Radford et al. 2019) are pretrained on large text corpora and fine-tuned for sentiment analysis, machine translation, or question-answering. The fine-tuned models inherit the rich representations learned during pretraining, which enhances their performance on specific tasks.

We adopt a similar approach in retail analytics. We *pretrain* the MBT on over 35 million shopping baskets to learn general patterns of product co-occurrence and the influence of covariates such as time and location. During pretraining (left panel of Figure 15), the transformer learns embeddings that capture relationships among products and covariates.

After pretraining, we *fine-tune* the transformer to predict in-store coupon redemptions—a binary classification problem. The fine-tuning step uses a much smaller data set from a coupon field experiment. As illustrated in the right panel of Figure 15, we modify the model by replacing the output layer used for masked product prediction with a new output layer suitable for binary classification. Specifically, we add a linear layer that maps the transformer's output to a single logit, which is then passed through a sigmoid function to estimate the probability of coupon redemption. The model is trained using binary cross-entropy loss, and the parameters are updated using stochastic gradient descent. This application demonstrates how the MBT can serve as a foundation model in retail.

Two considerations should be kept in mind when fine-tuning the MBT: First, fine-tuning is most effective when the products involved in the new task are available in the pretraining

Figure 15: Transformer Pretraining and Fine-Tuning



data. This is the case in our setting. If the products are not present during pretraining, the model lacks embeddings, and learning good representations from scratch during fine-tuning with limited data becomes challenging. Therefore, the effectiveness of fine-tuning depends on the overlap between the products in the pretraining and fine-tuning datasets.

Second, our analysis focuses on a single retailer. In NLP, foundation models can often be applied across different contexts due to the universality of language. In retailing, the applicability of a pretrained model to different retailers depends on factors such as the overlap of product assortments and the use of standardized product identifiers (e.g., UPC). While our model demonstrates potential within one retailer, we leave extending it to a cross-retailer foundation model for future research.

6.2 Predicting In-Store Coupon Redemptions

To evaluate the fine-tuned MBT, we apply it to predict coupon redemptions using data from a coupon experiment at the same grocery retailer. The dataset contains purchase histories of over 250,000 customers who are members of the retailer's loyalty program.[12] The retailer engages with loyalty program members through self-service kiosks located inside stores, where

[12] Coupons are only available for customers participating in the retailer's loyalty program, the customer population differs from the population that generated the market basket data used in the previous sections.

customers can print coupons. Approximately 40% of shopping trips by loyalty program members involve interactions with the kiosks.[13] Coupons are valid on the same day, and are automatically redeemed when customers scan their loyalty card at checkout. In the coupon experiment, which was conducted in 2016, the retailer randomly assigned 965,405 coupons to customers to test the effectiveness of different coupon strategies.

Our objective is to predict whether a customer redeems a randomly assigned coupon. The prediction task is framed as a binary classification problem: given the customer's shopping basket, covariates (e.g., store location, time), and the coupon product, we predict whether the coupon will be redeemed. Accurately predicting whether customers redeem coupons is useful for developing targeted marketing policies. Note that we do not take a stand on the direction of causality in this predictive model: A customer decides to redeem the coupon because the promoted product fits their preferences and the basket content (basket content → coupon redemption). Alternatively, a customer decides to redeem the coupon for the promoted product and then buys products that complement the promoted product (coupon redemption → basket content). Instead, our goals are to (1) demonstrate that model adaptation is possible and (2) benchmark the fine-tuned transformer model with baseline methods trained only on the coupon experiment data.

For fine-tuning, we adapt the pretrained transformer by adding a linear layer for binary classification, as described in Section 6.1. The model inputs include the products in the customer's shopping basket, the covariates, and the promoted product. We introduce a special token `<COUPON>` to separate the coupon product from the other products in the basket. The transformer processes this input and outputs a probability of coupon redemption.

Our goal is to evaluate whether a fine-tuned model can learn the systematic impact of basket content and store location and time on coupon redemptions, regardless of differences between product and their characteristics. We, therefore, balance the data set such that positive examples (coupon redemption) and negative examples (no coupon redemption) are equally

[13] Our data do not track when customers receive the coupon during their shopping trip.

likely. This results in 41,280 observations, of which 50% are redeemed. We split the data into a training set (80%) and a test set (20%). The model is fine-tuned on the training set, and its performance is evaluated on the test set using four metrics: precision, recall, F1 score, and area under the receiver operating characteristic curve (AUC).

We then compare the fine-tuned transformer's performance to two baselines: First, we implement a naïve model that randomly assigns classes (*Random Predictor*). Since the dataset is balanced, the random predictor yields scores of 0.5 for all metrics. Second, we use a Random Forest Classifier. The input features include the basket covariates (same as for the MBT), a basket vector (the average of the product embeddings in the shopping basket), and the product embedding of the coupon product. The product embeddings are obtained from a Product2Vec model (Gabel et al. 2019). Details of the Random Forest Classifier and its hyperparameter tuning are available in Appendix F.

Table 3 presents the evaluation results on the test set. The fine-tuned MBT achieves the highest scores across all metrics. The substantial improvement over the Random Forest Classifier indicates that the transformer effectively leverages its rich representations learned during pretraining to capture complex patterns related to coupon redemption.

These results showcase the MBT's ability to generalize from pretraining to a new task with little additional data. By incorporating information about the customer's shopping basket, covariates, and coupon products, the model can predict coupon redemption more accurately than benchmark models. The successful fine-tuning to coupon redemption prediction showcases the potential of the MBT as a foundation model for retail analytics.

Table 3: Evaluation of Fine-Tuned Transformer and Baselines on Test Set

Model	**Precision**	**Recall**	**F1 Score**	**AUC**
Fine-Tuned MBT	0.786	0.914	0.846	0.905
Random Forest Classifier	0.711	0.773	0.741	0.807
Random Predictor	0.500	0.500	0.500	0.500

7 Conclusion

This paper introduces the market basket transformer (MBT), an adaptation of the transformer architecture to market basket data. While existing transformer implementations work well in NLP and computer vision, the structure and sparsity of market basket data prevent them from being directly applied to market basket data, even after extensive hyperparameter tuning.

We proposed and evaluated several modifications to overcome this problem. First, we introduced covariates such as store location and time to the model. These covariates provide additional information that the MBT uses to model purchase probability heterogeneity across shopping trips. Second, we developed a better training strategy that uses available baskets more efficiently. Third, we created a data curation strategy that re-samples baskets with infrequently purchased products to improve model performance for long-tail products.

Our results show that our MBT more than doubles the predictive accuracy of benchmark models in basket completion modeling. It effectively captures complex purchase patterns and product relationships. By addressing data sparsity, particularly for long-tail products, our transformer achieves consistent predictive performance across all products in large retail assortments.

We also demonstrated our model's potential as a foundation model by fine-tuning it to predict coupon redemptions. The fine-tuned model outperformed benchmark models with minimal additional training data, showcasing the value of adapting the MBT to specialized retail analytics tasks. This flexibility highlights a key advantage of foundation models: they learn generalizable patterns during pretraining, which can be leveraged to efficiently adapt the model to new tasks. In retail analytics, this enables retailers to deploy a single, robust model across a variety of applications, such as targeting and promotion optimization, with minimal task-specific data.

Our modifications ensure that the MBT performs equally well for frequently purchased and long-tail products. This is important for foundation models in retailing. Consider, for ex-

ample, applying the MBT in a recommender system. If the transformer performs poorly for new products, small brands, or niche products, these products would likely be systematically disadvantaged by being recommended less.

We note several promising opportunities for future research. First, it would be interesting to implement the MBT at retailers with varying assortment sizes, including those with substantial non-food items, such as Walmart. Furthermore, e-commerce environments often feature larger assortments and smaller shopping baskets, which may affect transformer performance and the effectiveness of our modifications for long-tail products. Future research could also integrate additional covariates into the transformer architecture. Adding marketing variables and other contextual factors may improve model performance further, generate new insights, and expand transformer applications in retail analytics. In addition, future research could evaluate whether the MBT can serve as a cross-retailer foundation model, and adapt it to additional downstream tasks such as product recommendation, customer segmentation, or inventory management.

Finally, future research could extend the application of transformer models to other settings characterized by data scarcity and long-tail distributions. For example, transformers could model consumer clickstreams during product searches to improve product recommendations and search result rankings. In social media marketing, analyzing sequences of user interactions could enhance understanding of content dissemination and consumer engagement patterns, thereby improving campaign effectiveness. Similar approaches could be applied to investment portfolios by modeling sequences of financial decisions, aiding in risk management and personalized investment advice. By tackling challenges related to data sparsity and capturing complex patterns in marketing-specific data, pretrained transformer models have the potential to advance multiple areas within marketing analytics.

References

Agrawal R, Imieliński T, Swami A (1993) Mining association rules between sets of items in large databases. *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, 207–216.

Anderson C (2006) *The long tail: why the future of business is selling less of more* (Hyperion, New York).

Bell DR, Gallino S, Moreno A (2014) How to win in an omnichannel world. *MIT Sloan Management Review*.

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

Blattberg RC, Kim BD, Neslin SA (2008) Market basket analysis. In *Database Marketing: Analyzing and Managing Customers*, pp. 339–351. Springer New York, New York.

Blei DM, Ng AY, Jordan MI (2003) Latent dirichlet allocation. *Journal of Machine Learning Research* 3(Jan):993–1022.

Bommasani R, Hudson DA, Adeli E, Altman R, ... (2022) On the opportunities and risks of foundation models. *arXiv:2108.07258*.

Boztuğ Y, Reutterer T (2008) A combined approach for segment-specific market basket analysis. *European Journal of Operational Research* 187(1):294–312.

Brand J, Israeli A, Ngwe D (2023) Using gpt for market research. *Harvard Business School Marketing Unit Working Paper* 23(062).

Brynjolfsson E, Hu Y, Smith MD (2010) Research commentary—long tails vs. superstars: The effect of information technology on product variety and sales concentration patterns. *Information Systems Research* 21(4):736–747.

Castelo N, Katona Z, Li P, Sarvary M (2024) How AI outperforms humans at creative idea generation. *SSRN:4751779*.

Devlin J, Chang MW, Lee K, Toutanova K (2019) BERT: Pre-training of deep bidirectional transformers for language understanding. *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.

Gabel S, Timoshenko A (2022) Product choice with large assortments: A scalable deep-learning model. *Management Science* 68(3):1808–1827.

Grewal D, Roggeveen AL, Nordfalt J (2017) The future of retailing. *Journal of Retailing* 93(1):1–6.

Hertz JA (2018) *Introduction to the theory of neural computation* (CRC Press).

Hoffman M, Bach F, Blei D (2010) Online learning for latent dirichlet allocation. *advances in neural information processing systems* 23.

Imbens GW, Rubin DB (2015) *Causal Inference in Statistics, Social, and Biomedical Sciences* (Cambridge University Press).

Imbens GW, Wooldridge JM (2009) Recent developments in the econometrics of program evaluation. *Journal of Economic Literature* 47(1):5–86.

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

Jacobs B, Fok D, Donkers B (2021) Understanding large-scale dynamic purchase behavior. *Marketing Science* 40(5):844–870.

Kingma DP (2014) Adam: A method for stochastic optimization. *arXiv:1412.6980*.

Le Mens G, Kovács B, Hannan MT, Pros G (2023) Uncovering the semantics of concepts using GPT-4. *Proceedings of the National Academy of Sciences* 120(49).

Li P, Castelo N, Katona Z, Sarvary M (2024) Frontiers: Determining the validity of large language models for automated perceptual analysis. *Marketing Science* 43(2):254–266.

Liu X (2023) Deep learning in marketing: a review and research agenda. *Artificial Intelligence in Marketing* 239–271.

Liu Y (2019) RoBERTa: A robustly optimized BERT pretraining approach. *arXiv:1907.11692*.

Lu Z, Kannan P (2024) Measuring the synergy across customer touchpoints using transformers. *SSRN:4684617*.

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

Mild A, Reutterer T (2003) An improved collaborative filtering approach for predicting cross-category purchases based on binary market basket data. *Journal of Retailing and Consumer Services* 10(3):123–133.

OpenAI (2023) Chatgpt (mar 14 version) [large Language Model]. https://chat.openai.com/chat. Technical report, OpenAI.

Puranam D, Kadiyali V, Narayan V (2021) The impact of increase in minimum wages on consumer perceptions of service: A transformer model of online restaurant reviews. *Marketing Science* 40(5):985–1004.

Radford A, Wu J, Child R, Luan D, Amodei D, Sutskever I, et al. (2019) Language models are unsupervised multitask learners. *OpenAI blog* 1(8):9.

Ringel DM (2023a) Creating synthetic experts with generative artificial intelligence. *SSRN:4542949*.

Ringel DM (2023b) Multimarket membership mapping. *Journal of Marketing Research* 60(2):237–262.

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

Ruiz FJ, Athey S, Blei DM (2020) Shopper. *The Annals of Applied Statistics* 14(1):1–27.

Russell GJ, Petersen A (2000) Analysis of cross category dependence in market basket selection. *Journal of Retailing* 76(3):367–392.

Schöll N, Gallego A, Le Mens G (2024) How politicians learn from citizens' feedback: The case of gender on Twitter. *American Journal of Political Science* 68(2):557–574.

Simester D, Timoshenko A, Zoumpoulis SI (2020) Targeting prospective customers: Robustness of machine-learning methods to typical data challenges. *Management Science* 66(6):2495–2522.

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 AN, Kaiser L, Polosukhin I (2019) Attention is all you need. *Advances in Neural Information Processing Systems*.

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

Wei S, Gabel S, Krefeld-Schwalb A (2024) Global evidence on overlapping motives for sustainable behaviors. *OSF:SYKU6*.

Xu C, Sun Q, Zheng K, Geng X, Zhao P, Feng J, Tao C, Lin Q, Jiang D (2024) Wizardlm: Empowering large pre-trained language models to follow complex instructions. *The Twelfth International Conference on Learning Representations*.

Zhang T (2000) Association rules. *Proceedings of 4th Pacific-Asia Conference*, 245–256 (PADKK).

The Market Basket Transformer: A New Foundation Model for Retail

Web Appendix

A Implementation of Baseline Models

To evaluate the performance of our transformer-based model in the Basket Composition Modeling (BCM) task, we implement five baseline models: the Zhang market basket analysis score (Zhang 2000), Product2Vec (Gabel et al. 2019), Latent Dirichlet Allocation (Blei et al. 2003), SHOPPER (Ruiz et al. 2020), and random predictions.

A.1 Zhang Metric

The Zhang metric (Zhang 2000) is a statistical measure to assess the association between products in a shopping basket:

$$m_{ij}^{Zhang} = \frac{P(i \cap j) - P(i)P(j)}{\max(P(i \cap j), P(i)P(j))}, \tag{8}$$

where $P(i)$ and $P(j)$ are the probabilities of products i and j occurring in a shopping basket, and $P(i \cap j)$ is the joint probability that both products are in a shopping basket. It helps determine whether the presence of one product in a basket is positively or negatively correlated with the presence of another product. We use a custom implementation of the Zhang metric.

During training, we calculate the Zhang metric z_{ij} for all product pairs (i, j) using the training data. During inference, we calculate the average Zhang metric between the products in the basket and all products in the assortment (i.e., all potential products that could be the masked item) for each basket b:

$$m_j^{Zhang} = \frac{1}{n_b} \sum_{i \in b} m_{ij}^b \tag{9}$$

We then select the product with the highest average Zhang metric:

$$\hat{j}^{Zhang} = \arg\max_j m_j^{Zhang} \tag{10}$$

A.2 Product2Vec (P2V)

Product2Vec (Gabel et al. 2019) embeds products in latent attribute spaces by learning the co-occurrence of products in shopping baskets. The neural network calculates the probability that pairs of products occur in baskets in the training data as a function of the product vectors. We implement Product2Vec using the Python library `gensim`. Table A1 summarizes the P2V hyperparameters.

Table A1: Hyperparameters for the Product2Vec implementation in `gensim`

Parameter	**Variable**	**Value**
Embedding size	`vector_size`	128
Learning rate	`alpha`	0.025
Window size	`window`	1,000
Minimum observations per product	`min_count`	0
Threshold for downsampling frequent products	`sample`	0.001
Use skip-gram model	`sg`	1
Use hierarchical softmax	`hs`	0
Number of negative samples	`negative`	5
Exponent in sampling weight calculation	`ns_exponent`	0.75
Number of epochs	`epochs`	50

Product2Vec generates two product embeddings V and W that capture product attributes and relationships based on their co-occurrence in shopping baskets. During inference, we use the first embedding to derive a basket vector for each shopping basket in the test set as the average over the product vectors v_i for all products i in basket b

$$v_b^{P2V} = \frac{1}{n_b} \sum_{i \in b} v_i. \tag{11}$$

We then calculate the co-occurrence metric (Gabel et al. 2019) between the basket vector and the vector of each product in the assortment

$$m_j^{P2V} = \cos(w_j,\, v_b^{P2V}), \tag{12}$$

and select the product with the highest co-occurrence metric as the predicted missing product

$$\hat{\jmath}^{P2V} = \arg\max_j m_j^{P2V} \tag{13}$$

A.3 Latent Dirichlet Allocation (LDA)

LDA (Blei et al. 2003, Hoffman et al. 2010) is a generative probabilistic model used for uncovering latent topics in texts. LDA has been applied to market basket data by Jacobs et al. (2016). We implement LDA using the Python library `gensim`. Table A2 summarizes the LDA hyperparameters.

Table A2: Hyperparameters for the LDA implementation in `gensim`

Parameter	**Variable**	**Value**
Embedding size	`num_topics`	128
Number of baskets in each batch	`chunksize`	2000
Number of epochs	`iterations`	50
Number of iterations over corpus	`passes`	50
Batch learning	`update_every`	1
A-priori belief on document-topic distribution	`alpha`	symmetric
A-priori belief on topic-word distribution	`eta`	symmetric
% of lambda that is forgotten in new document	`decay`	0.5
Step slow down for first iterations	`offset`	0.5

During training, we learn topic vectors for all products. Each product is assigned a probability distribution over latent topics based on the other products it tends to co-occur within the same baskets b in the training data set. During inference, we first compute the basket's topic distribution t_b by averaging the topic distributions of the products in the basket for all baskets in the test set

$$t_b^{LDA} = \frac{1}{n_b} \sum_{i \in b} v_i. \tag{14}$$

We then calculate the similarity metric between this basket topic distribution and the topic distributions of all products in the assortment

$$m_j^{LDA} = \cos(v_j, t_b^{LDA}) \tag{15}$$

and select the product with the highest metric as the predicted missing product

$$\hat{j}^{LDA} = \arg\max_j m_j^{LDA} \tag{16}$$

A.4 SHOPPER

SHOPPER (Ruiz et al. 2020) is a probabilistic choice model that integrates consumer preferences and product interactions. During training, SHOPPER uses customers' baskets in the training set to learn product embeddings. Each product is embedded into a vector space; distances between products reflect their likelihood of co-occurring in baskets. Because we use market basket data without customer IDs, we do not learn customer embeddings. We train SHOPPER using the authors' code with the hyperparameters depicted in Table A3.

Table A3: Hyperparameters for SHOPPER implementation

Parameter	**Variable**	**Value**
Embedding size	`K`	128
Intercept for products	`itemIntercept`	1
Data includes customer ID	`userVec`	0
Data includes price	`price`	0
Normalize price	`normPrice`	0
Data includes dates	`days`	1
Number of negative samples	`negsamples`	25
Learning rate	`eta`	0.01
Maximum number of iterations	`max-iterations`	1000
Batch size	`batchsize`	1000

Note: We thank the authors for supporting us during our implementation of the SHOPPER model.

During inference, we average the embeddings of the products in the basket b to obtain a basket vector (Equation 1 in Ruiz et al. 2020)

$$v_b^{SHOPPER} = \frac{1}{i-1} \sum_{j=1}^{i-1} \alpha_{y_j}. \tag{17}$$

We then compute the co-occurrence metric between this basket vector and the vector of each product in the assortment:

$$m_j^{SHOPPER} = \rho_j \cdot v_b^{SHOPPER}, \tag{18}$$

and select the product with the highest score as the predicted missing product

$$\hat{j}^{SHOPPER} = \arg\max_j m_j^{SHOPPER} \tag{19}$$

The weight matrices α and ρ are learned by SHOPPER.

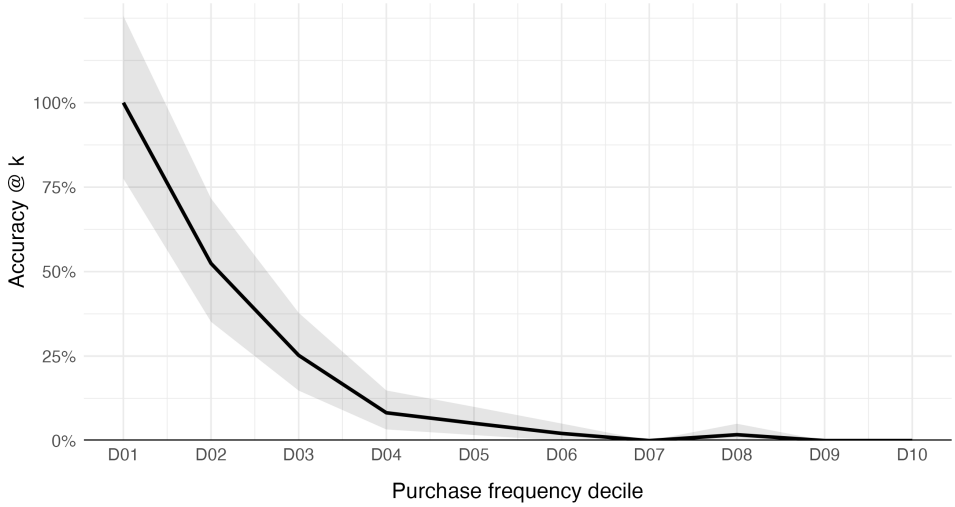
A.5 Random Predictions

The random baseline serves as a naïve control model. For each test basket, we randomly select a product from the entire assortment as the predicted missing product, without taking into account the other products in the basket or any product relationships.

B Basket Composition Modeling Plots with Confidence Intervals

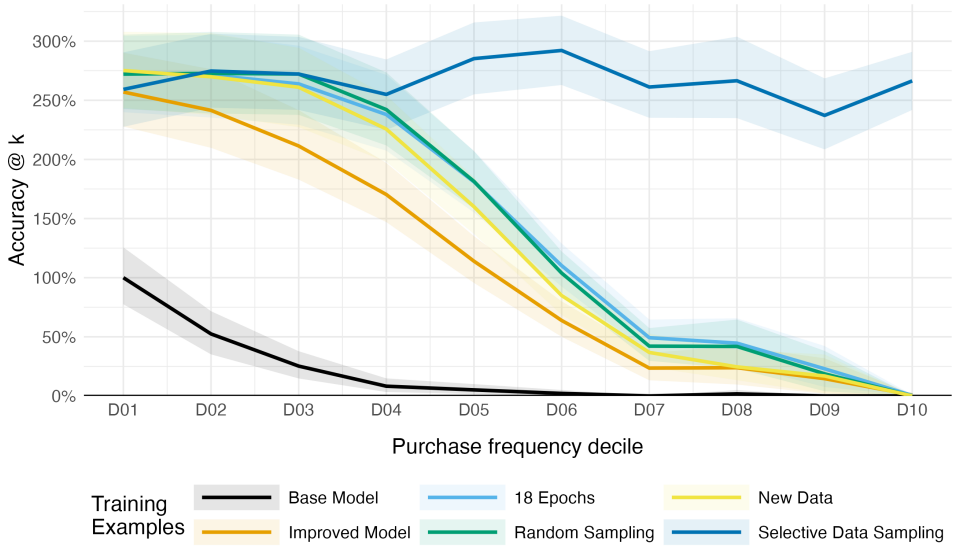
Figures A1 and A2 depict versions of Figures 6 and 11 with 95% confidence intervals. We calculate CIs using a non-parametric bootstrap, resampling products within each bin with replacement. Figure A1 confirms that differences between the first four bins are statistically significant ($p < 0.05$). Figure A2 confirms that the data curation strategies 1 to 3 do not produce models with significantly different predictive performance.

Figure A1: Predictive Performance as a Function of Product Purchase Frequency



Notes: We normalize the accuracy to the maximum value of the base model in Figure 6.

Figure A2: Effect of Training Data Curation Strategies



Notes: We normalize the accuracy to the maximum value of the base model in Figure 6.

C Do long-tail products co-occur more frequently with other long-tail products?

In this section, we explore a possible explanation for the difference in the predictive performance between long-tail and frequently purchased products. Baskets containing long-tail products may have systematically different compositions compared to those with high-frequency products. For example, long-tail products might occur more frequently with other long-tail products, and the low number of observations for long-tail products might make it more challenging for the transformer model to learn robust patterns.

We investigate this possibility by sampling 50 products from the decile with the highest purchase rates and 50 products from the decile with the lowest purchase rates. For each focal product, we calculate the average product purchase rate in baskets containing that product (excluding the product itself) and obtain the mean and variance of these purchase frequencies across baskets. This analysis is conducted separately for high-frequency products ($\overline{X}_h$ and S_h) and long-tail products ($\overline{X}_l$ and S_l). Following Imbens and Wooldridge (2009), we calculate the standardized mean difference (SMD) as a scale-free measure of the difference in distributions:

$$\Delta_X = \frac{\overline{X}_h - \overline{X}_l}{\sqrt{S_h^2 + S_l^2}}.$$

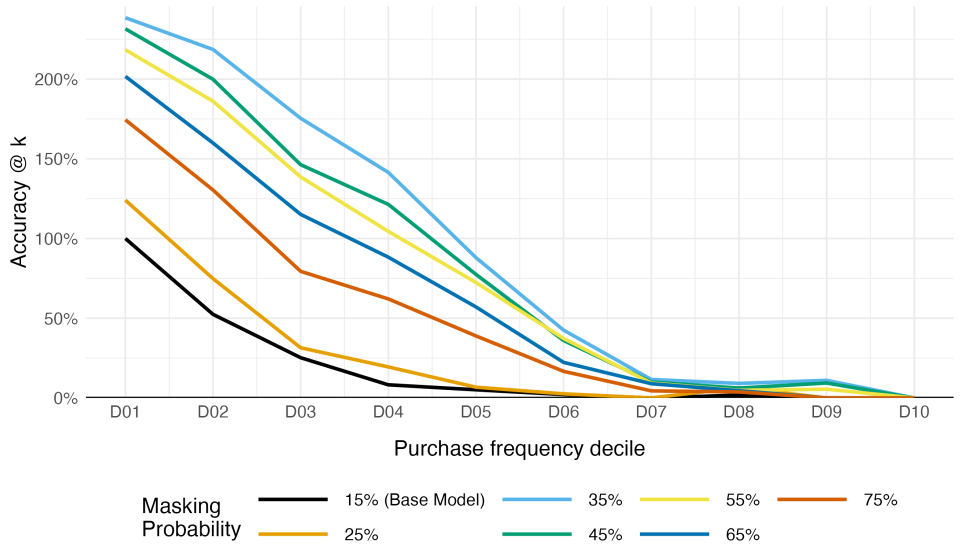
According toImbens and Rubin (2015), SMD values below 0.25 indicate no meaningful differences between groups. Using a non-parametric bootstrap, we estimate the standard error for the difference in average product purchase frequency.

We find a normalized difference of $\Delta_X = 0.199$ ($SE = 0.110$), with frequency values of $\overline{X}_h = 0.0030$ and $\overline{X}_l = 0.0025$. Our finding suggests that while long-tail products tend to co-occur with other long-tail products, the overall distributional differences between high-frequency and long-tail products are minor.

D Transformer Performance at Different Masking Probabilities

In Section 5.3, we evaluate how a fixed number of masked products per basket affects the performance of the MBT. Here, we confirm our earlier proposition that low and high masking probabilities for the default masking approach lead to undermasking and overmasking, thereby reducing the transformer's predictive performance. To assess the impact of varying the masking probability on transformer performance, we systematically vary the masking probability from 15% to 75% in 10% increments, while holding all other model hyperparameters constant.

Figure A3: Effect of Masking Probability



Notes: To facilitate a comparison of predictive performance across deciles, we normalize the accuracy to the maximum value in Figure 6 in the main text.

Figure A3 shows the results of our experiments. For all masking probabilities, the accuracy decreases consistently as we move from high-frequency products (`D01`) to low-frequency products (`D10`), with all curves running parallel. Increasing the masking probability from 15% to 25% and then to 35% improves model performance overall, but further increases beyond 35% lead to a decline in accuracy, consistent with our expectation that both undermasking and overmasking can deteriorate model performance. Specifically, increasing the masking probability from 15% to 35% creates substantial improvements for products in `D04` to `D06` and moderate improvements in `D01` to `D03`. However, none of the tested masking probabilities significantly increase accuracy for products in `D07` to `D10`, which collectively account for 40% of the retailer's assortment. We explain this finding by the low rate with which long-tail products occur in shopping baskets: High-frequency products are observed so frequently that the marginal benefit of additional data is minimal. Conversely, low-frequency products are so rare

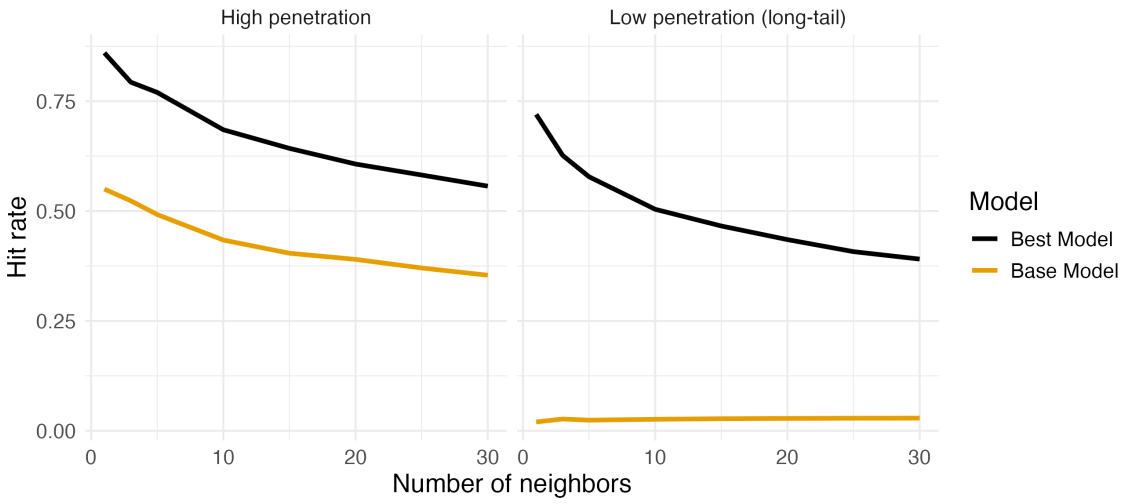
that the impact on performance remains negligible even with increased data. However, for medium-frequency products, each additional basket substantially improves model accuracy.

Our findings indicate that although increasing the masking probability improves overall model performance, it does not effectively address the challenges posed by low-frequency products. Consequently, additional modifications are necessary to overcome the inherent difficulties associated with long-tail products in market basket analysis.

E Robustness of Nearest Neighbors Analysis

In Section 5.5, we show that our transformer modifications increase the quality of product embeddings for long-tail products. Here, we show that the results presented in Figure 13 are robust to the number of nearest neighbors.

Figure A4: Fraction of Same-Category Nearest Neighbors



We consider the same 200 products we focused on in the main text (100 low-frequency products and 100 high-frequency products) and calculate what fraction of products' neighbors on the product maps for the `BASE` and the `BEST` transformer belong to the same category as the focal product. We calculate this hit rate separately for low-frequency and high-frequency products. Instead of using $k = 10$ nearest neighbors, we vary the number of neighbors of the focal product considered in the analysis from 1 to 30. Although the hit rates in Figure A4 decrease with an increasing number of neighbors, the conclusion in Section 5.5 remains valid: Both models achieve high hit rates for high-frequency products. The `BEST` model maintains the hit rate even for long-tail products, whereas the hit rate for the `BASE` model breaks down entirely.

F Random Forest Hyperparameter Tuning

We tune the following the hyperparameters of the Random Forest model to achieve better performance: The number of trees, the maximum tree depth, the minimum samples required for split, and the minimum number of samples per leaf node. We use 5-fold cross-validation on the training data to select the hyperparameters (using the F1 score). Table A4 shows the hyperparameter ranges considered in the hyperparameter optimization. We chose the hyperparameters that produced the best average performance across five folds.

Table A4: Random Forest Hyperparameters

Parameter	**Name**	**Grid**	**Chosen**
Number of trees	`n_estimators`	[25, 250]	149
Maximum tree depth	`max_depth`	[3, 10]	9
Minimum samples required for split	`min_samples_split`	[2, 10]	8
Min samples per leaf node	`min_samples_leaf`	[1, 5]	4