Erasmus University Rotterdam

Erasmus School of Economics

Master Thesis – Data Science and Marketing Analytics

Knowledge Injection into a Transformer-Based Recommender System: An Online Grocery Shopping Dataset

Max van de Kamp (60112)

Supervisor:

Luuk van Maasakkers

Second Assessor:

Dr. Michiel van Crombrugge

February 6, 2023

Abstract

In the age of online shopping, recommender systems are an essential tool for businesses to understand and anticipate the purchasing patterns of their customers. These systems help customers navigate through vast product offerings and improve their shopping experience, ultimately leading to customer loyalty and satisfaction. In this thesis, we present a novel type of recommender system called KI-BaskERT, which is specifically designed for session-based recommendations for basket completion. KI-BaskERT combines the state-of-the-art concept of knowledge injection with a transformer network-based recommender system. We first review the literature on recommender systems and describe in detail how KI-BaskERT is constructed, including the key components of the transformer network, specifically the BERT architecture and its attention mechanism and masked modelling training procedure. The results of this research indicate a significant improvement in performance of our KIBaskERT model when compared to the model that does not utilize knowledge injection.

Contents

1 Thank Word

I would like to express my sincerest gratitude to my supervisor, Luuk van Maasakkers, for his invaluable guidance and support during the process of building and training the KI-BaskERT recommender system. His expertise as a neural network specialist from Erasmus University Rotterdam, as well as his patience and helpfulness, made this project a success. I have immense respect for him and without his help, this thesis would not have been possible. Thank you, Luuk.

2 Introduction

The digital revolution in the last decades has altered the rules of communication, personal relationships, businesses, and the way people occupy their time. Linked to this revolution is electronic commerce, better known as e-commerce, which can be seen as the whole of commercial activities carried out via the internet (Knight & Mann, 2017). E-commerce has become an integral part of the global business environment nowadays. The increase in the global availability of electronic equipment and the consequent technological development in services offered via the internet have led to a trend towards more e-commerce (Jamsheer, K., 2019).

With internet access and usage growing rapidly worldwide, the number of digital consumers continues to increase every year. To get a sense of this enormous growth, e-commerce sales in 2021 in the retail sector globally reached about 4.65 trillion euros. This figure is expected to rise by 50% over the following four years, to approximately 7.02 trillion euros in 2025 (Pasquali, M., 2022).

The main advantages of e-commerce over ordinary commerce are the easy comparison of products and services of competitors, the 24-hour availability and convenience, that it is a global market, comfortability, personalisation and at least the usually lower price as pure online providers have lower costs for buildings and employees compared to brick and mortars (Jamsheer, K., 2019).

The advantages, opportunities and tremendous growth of this sector have created one of the largest transformative industries on the planet, leading to a huge competitive market in which new players are constantly entering and trying to capture a share of the market and its growth. This competition is reflected in the growth of the number of goods and services offered online, the fact that more and more offers and discounts are provided, the facilitation of payment procedures, as well as the facilitation of the searching process for goods and services for customers according to their preferences. In this work, research will be conducted on the latter topic, the facilitation of the search process for customers.

2.1 Problem Description

At the end of the last century, it was realised that companies must change from a world of mass production where the focus is on selling standardised products in homogeneous markets to one of mass customisation and variety. At the very least, companies should be able to offer products that meet the diverse needs of their diverse consumer base. In this sense, e-commerce had not necessarily enabled companies to produce more products, but it has enabled them to serve consumers with a wider range of products (Schafer et al., 2001).

Due to this explosive increase in the number of products and services offered online, it became increasingly difficult for the customer to find the products that are relevant to him/her. Whereas physical shops can only display part of their product range, online shops do not face this limitation. This is both an advantage and a disadvantage because the question that arises is which products to show to which specific customer. On the other hand, consumers do not have the capacity or the time to browse through all the thousands of products of an e-commerce website to search for the desired product. It was realised that e-commerce websites must be adapted and personalised to the customer. In the late 90's, Jeff Bezos, Amazon's still current CEO stated: "If I have three million customers on the web, I should have three million stores on the web".

Mainly due to the increasing capabilities of the Internet technology and the growing e-commerce market, the demand for recommender systems that could recommend the most suitable products to specific customers became more necessary than ever. In addition to just their functionality, recommender systems have many advantages for e-commerce companies. Three major ways in which recommender systems enhance an e-commerce system is by assisting buyers who have no prior experience of online purchasing, by cross-selling products to regular customers, and by creating greater customer loyalty.

In the literature, recommender systems are defined as "information filtering systems that address the problem of information overload by filtering critical information fragments out of a large amount of dynamically generated information based on consumers' preferences, interests, and past behaviour" (Isinkaye et al., 2015). An alternative definition for a recommender system given by Apáthy, S. (2022) is "An information filtering system that assists the

consumer in a particular decision-making situation by narrowing down the range of possible choices and prioritising related elements in the specific context". Prioritisation can be based on the explicit or implicit preferences expressed by the consumer as well as on the previous behaviour of consumers with similar preferences. Explicit preferences are preferences that are obtained via direct participation of the user. These preferences can for example be gathered by implementing a like/dislike button in a movie recommender system. Implicit preferences on the other hand are preferences where no specific user participation is needed. Here the system automatically tracks the users' preferences by observing the performed interactions with the system such as where a user has clicked or how long they have been visiting a page for example. In conclusion, both definitions mention the process of narrowing the number of displayed products based on explicit or implicit behaviour and context.

As inferred, recommender systems are not a recent invention. The recommender system was first introduced in the 90's - in a very rudimentary form - as a novelty, but it has become indispensable in today's e-commerce environment. However, faster, more intelligent, and more efficient recommender systems are constantly being developed by researchers to anticipate customer preferences and purchase context even better as well as obtaining a competitive advantage for companies.

These more sophisticated recommender systems result from the explosive growth of artificial intelligence and machine learning capabilities and applications over the course of the last two decades. This work will focus on neural networks, a form of deep machine learning algorithms that are already well established in the machine learning field, but of which constantly improved, and more intelligent modifications are being developed (Jannach, D. et al., 2021). More specifically, the focus is on Transformer-based neural networks that were introduced in 2017 by a team from Google Brain in the ground-breaking paper *Attention is all you need* by Vaswani, A. et al. (2017). Building transformer networks presents significant challenges, including the substantial data requirements and extensive training time even with high-performance hardware, without guaranteeing accurate predictions. Nonetheless, earlier studies (Sun et al., 2019; Bianchi et al., 2020) have demonstrated the potential of transformer networks for creating effective recommender systems. This work aims to address the issue of accuracy and endeavors to enhance the performance of these complex networks.

2.2 Research Objective

The objective is to construct an improved type of recommender system in an e-commerce setting named KI-BaskERT based. The name of this model is divided into several parts, namely the first two letters stand for novelty this work attempts to bring, namely Knowledge Injection. Followed by the word BaskERT which refers to the BERT architecture derived from the transformer model introduced by Google. And finally, Bask stands for the application to the completion of baskets of products. Although there exist similar applications in the marketing domain related to the transformer networks, the novelty brought by this work will be the addition of the concept of Knowledge Injection. In the methodology section the concept of Knowledge Injection will be discussed in more detail. Expressed briefly, the aim is to inject additional knowledge about the products during the training of the model in order to make the model more intelligent in recognising structures in the data. It is expected that with the additional knowledge about the products in the baskets, the KI-BaskERT model will outperform its predecessors. This brings us to the following research question with sub-questions:

- How can theory and insights from related and prior transformer network-based recommender systems combined with the state-of-the-art concept of Knowledge Injection be deployed to develop a novel type of recommender system in an online shopping environment?
  - To what extent can we correctly predict which products a customer will buy or would like to buy and then recommend them based on products from the current basket when using knowledge injection in transformer networks?
  - Will Knowledge Injection in transformer networks provide increased accuracy in predicting the next product in the basket compared to previous models without Knowledge Injection?

This study will be carried out by first training and using a BaskERT model without Knowledge Injection to compare it with the results of the KI-BaskERT model both applied to the Instacart Shopping dataset. The proposed model will be, to our knowledge, the

first BERT-based model that uses Knowledge Injection in a recommender system related context. This fills a gap in the literature as these transformer networks are mainly studied in the Natural Language Processing (NLP) environment but have few applications in the marketing field. Knowledge Injection already shows promising applications in the Natural Language Processing field, making this a valuable study.

Following this introduction is the literature review section in which the most important concepts in the existing literature regarding recommender systems, neural networks, transformer networks, BERT and knowledge injection will be reviewed. The subsequent section is the data section discussing the Instacart Shopping dataset and its characteristics. Thereafter, in the methodology section, the complete KI-BaskERT model will be elaborated upon in order to interpret the results in the succeeding section. Finally, the conclusion and potential areas for improvement will be presented.

3 Literature Review

The invention of the traditional recommender systems dates to 1992, back when e-mail was becoming popular. People were flooded with emails and could not distinguish between relevant and irrelevant emails. This gave rise to a novel filtering system called Taperstry invented by the Xerox Palo Alto Research Center that worked based on a technique called collaborative filtering (Goldberg, Nichols, Oki, & Terry, 1992). This technique was also applied to build the first recommender systems and was soon followed by the second most important filtering technique named content-based filtering. These two techniques are still used in many applications today and dominated the landscape for many years. Collaborative filtering and content-based filtering recommender systems both base their recommendations on historical interactions and user/item characteristics. Content-based recommendations are primarily based on the specific item and profile characteristics of the user, whereas collaborative filtering systems look for the preferences of a similar audience (Sciforce, 2022).

Matrix factorisation has been the most widely adopted method for constructing recommender systems among all collaborative filtering techniques. This approach involves projecting customers and products into the same vector space and maximizing their inner product (Koren, Y. & Bell, R., 2011; Salakhutdinov, R. & Mnih, A., 2007). However, despite its years of success, researchers gradually realised that with the increasing complexity of the problem setting, such as the surge in the number of variables and product sequence length, these techniques led to reduced accuracy in many cases. A significant limitation was observed in their performance with session-based data, where the model relies on information from a brief period, such as a customer's short visit to a website. (Hidasi, B. et al., 2015).

Later with the emergence of deep machine learning algorithms, stimulated by the broad success in other application areas including speech and image recognition (Russakovsky et al., 2014; Hinton et al., 2012), neural networks were also applied to build recommender systems. Many different types of neural networks such as feedforward neural networks, convolutional neural networks, recurrent neural networks, and combinations such as recurrent convolutional networks were proposed and tested. As recently as 2015, recurrent neural networks were used to develop recommender systems as a solution for session-based data and to consider

the sequence of clients' purchases, as previous techniques were unable to do so (Hidasi, B. et al., 2015; Donkers, T. et al., 2017).

The standard paradigm of recurrent neural networks was to encode a client's historical interactions in a vector using a left-to-right sequential model. Products were presented in a sequence and the model used the information about the products at that step to predict the next product in the sequence. Despite being prevalent and efficient, left-to-right unidirectional models have been criticized for their inadequacy in learning and modeling optimal representations for clients' behavioral sequences. The main drawback of such recurrent neural networks is that each item in the sequence can only encode the information of previous items and cannot grasp the full historical sequence of products (Lipton, Z. et al., 2015; Sun, F. et al., 2019).

From that moment on, numerous attempts have been made to increase the capabilities of recurrent neural networks and encoder-decoder architectures (Jozefowicz R. et al 2016; Wu, Y. et al, 2016). These networks confronted severe limitations when working with long sequences, as their ability to retain information from the first inputs was lost when new inputs were added to the sequence.

The real breakthrough came in 2017 through a work by Vaswani, A. et al. in the paper *Attention is all you need* (2017). The authors proposed a simple but powerful network architecture for automatic multilingual translation, named the transformer network. Like most competing neural sequence translation models (Bahdanau, D. et al., 2014; Cho, K. et al., 2014), the transformer network utilises an encoder-decoder architecture. However, the major innovation brought by their work is the attention mechanism. The attention mechanism enables the encoder to process the input text sequence by seeking out important parts and generating a word embedding for each word depending on its relevance to other words in the sentence. To clarify, a word embedding is the numerical representation of a word determined in a vector that the model employs to identify a word. After the encoder has processed the sequence, the decoder leverages the output of the encoder, i.e. the embeddings, and converts those embeddings back into a text output in a different language. The attention mechanism enabled the encoder to focus, i.e. draw attention, to certain parts of the sequence to understand context while ignoring irrelevant parts of the input sequence. The transformer

architecture made training the network significantly faster due to parallelisation and showed increased accuracy in translation tasks compared to previously discussed neural network architectures.

It was quickly realised by researchers that this type of attention mechanism could have many promising applications beyond text translation. It would not take long until Devlin, J. et al. published another important work named *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding* (2018) in which the authors proposed a new architecture derived from the previously described transformer networks for Natural Language Processing (NLP). As its name suggests, the aim of this model was not to translate, but to understand and represent language in the most accurate way by using the attention mechanism to comprehend the context of the input sequences.

The decoder side of the original transformer network architecture was discarded, and BERT was constructed using a multi-layer encoder architecture which is no different from stacking multiple transformer encoders. The purpose of the encoders is to process the input sequence and encode, compress, and summarise the information with the help of the attention mechanism to generate improved embeddings that result in more accurate text representations. The novelty brought by this work was the use of a "Masked Language Model" (MLM) training objective also referred to as the Cloze task (Taylor, W.,1953), making the model bidirectional. This technique involves randomly masking words from the input sentences during the training phase of the language model and having the model predict these masked words. By masking just a few words and allowing the model to look at the rest of the sentence, the authors obtained bidirectionality. This bidirectionality was a big improvement since previous transformer-based architectures such as OpenAI's GPT did not allow this, which constrained the choice of architectures at the training phase.

In 2019, researchers from Alibaba Group implemented the BERT architecture for an in-session recommender system in their work *BERT4Rec: Sequential Recommendation with Bidirectional Encoder Representations from Transformer* (Sun, F. et al., 2019). The authors argue that the BERT architecture is more suitable than unidirectional models as they can only process the sequence of products from left to right. Such left-to-right encoding limits the model to comprehend the full sequence of products in the basket, i.e. the context they appear

in, as each product can only encode the information of the previous products in the sequence. Sun, F. et al (2019) mention two important disadvantages of such left-to-right architecture: "unidirectional architectures limit the power of hidden representation in users' purchase sequences, and they often assume a strictly ordered purchase sequence, which is often not the case in practice". An additional disadvantage of unidirectional architectures is that they impede parallelisation since one must "wait" for the entire sequence to be processed one by one. To overcome these shortcomings, the authors designed a bidirectional recommender system for modelling customer behavioural sequences that can exploit the contexts of both the left and right parts of the sequence of products to base its predictions on.

A similar study by Bianchi, F.et al. (2020) compared the accuracy of product embeddings produced by BERT and the embeddings of the *prod2vec* method through comprehensive experiments on different commercial datasets (Grbovic, M. et al., 2015). Product embeddings, like word embeddings, are the numerical representation of a product in a vector of a predetermined size. The authors found that the product embeddings produced by BERT showed significantly higher accuracy and improved contextual comprehension. Compared to the study on BERT4Rec that uses academic datasets, in this research datasets of existing anonymous companies are used. However, a critical note is made by the autors that for accurate product embeddings generated by the BERT architecture, there is a need for a significantly large number of training data observations, making such an implementation only valuable for large companies with a lot of data.

The original BERT model from Devlin, J. et al. (2018) was designed for Natural Language Processing (NLP) tasks and is therefore trained on large-scale open-domain textual corpora including the full corpus of Wikipedia to acquire general language representations. After the pre-training phase comes the fine-tuning phase where the language model is adapted for downstream tasks, e.g. question answering, classification, named-entity recognition. This can lead to a discrepancy between the general knowledge that the model possesses and the domain-specific downstream tasks for which it is tailored. For instance, BERT, which is pre-trained on Wikipedia, cannot fully live up to its value when it comes to tasks in the medical world, such as medical records analysis.

To overcome this, Liu, W. et al. (2020) propose K-BERT to enable language representa-

tion with Knowledge Graphs for BERT, thereby achieving the possibility of common-sense or domain knowledge in a Natural Language Processing setting. This research is very much in line with the research proposed in this work as the authors inject knowledge into the model using Knowledge Graphs. This is done by adding domain-specific words to each word in a sentence, by using triples of words, to create sentence trees that make the embeddings of the words more meaningful for that specific domain.

In their research, domain knowledge from three Chinese Knowledge Graphs (CN-DBpedia, HowNet and MedicalKG) was injected into sentences as they perform various NLP tasks, while using the pre-trained parameters from the original BERT model. In this way, the researchers try to mimic the way human experts make inferences regarding relevant knowledge. For instance, a doctor reading a medical text will recognise and understand certain medical terms that an average reader will not. By knowing those particular words, the doctor can make inferences to understand the text. In the setting of BERT, the pre-trained BERT is the average reader and Knowledge Graphs can help BERT to make inferences to words in order to possess domain knowledge.

Previous resarch has proven that incorporating additional information of products in models like recurrent neural networks and convolutional neural networks can increase the performance of recommender systems. In 2016, Hidasi, B. et al. proposed a parallel recurrent neural network that consists of multiple recurrent neural networks, one for every representation/aspect of the product to be predicted including one for the product ID, the product image and one for the textual information. Afterwards, the hidden states of each network are aggregated to obtain the scores for the most likely products to be recommended.

On the other hand, Tuan, T. X., and Phuong, T. M.(2017) present a different approach. In their work they propose an 3D convolutional neural network to predict the next item to be added to an online shopping cart by allowing to combine different product information, e.g. product ID, product name and other relevant metadata. All the additional information is incorporated by using one character-level encoded matrix where the aggregate information is contained in a concatenated matrix. This concatenated matrix is then fed to the neural network.

One can say that different ways of adding additional information are still being explored.

However, to our knowledge, this has not yet been applied in a recommender system setting to the new BERT architecture. The novelty brought by this work is the concept of Knowledge Injection for a BERT architecture recommender system. The extra knowledge/information that is injected, is the aisle or department information to which the products belong to allow the network to better understand the context of the baskets. To give a practical example, imagine a customer wants to bake a cake and therefore needs ingredients such as flour, sugar, eggs, chocolate baking chips etc. Flour and sugar are two products from the "baking ingredients" aisle. Including that both of those products belong to the same aisle as supplementary information to the recommender system, the system will better understand how the products flour and sugar relate to each other. We expect the model to become better at understanding the relationships between products by including the aisle or department information and therefore making better recommendations based on which products are in the in-session based basket of the customer.

Important to mention here is that whereas the literature usually speaks of sequential recommendation, i.e. taking into account the order and time in which a customer purchases a certain product, this work does not take into account the order in which the products are placed in the basket. In an e-commerce setting like Sun, F. et al. (2019), it is argued that it is important that when one has bought an iPhone for example, that one offers a charger afterwards. In this work however, the in-session based recommender system that is built does not attempt to capture such patterns. The focus here is more on the context of the goods purchased together and to better predict the next product based on that context. The order in which the products are added to a basket is therefore irrelevant in this application. In addition, this recommender system is for session-based basket recommendation and therefore does not take into account a stretched time period in which different products are bought, unlike the iPhone example.

4 Data

This section discusses all the important aspects of the publicly available Instacart Online Grocery Shopping dataset (Instacart Market Basket Analysis | Kaggle, 2017) that will be used to train and test the recommender system.

Instacart is a American company established in 2012 that specializes in offering a pick-up service for groceries across the United States and Canada. Its services are available through both a website and mobile application. Customers can place an order for groceries through the platform, and a personal shopper is then assigned to collect the items from participating retailers and deliver them directly to the customer. The dataset comprises information on which products are purchased in a single purchase transaction, these purchased products are also called a basket of products. The term basket will come up a lot in this work. A basket consists of the selection of products a customer has picked via the Instacart app during a session. As a result, Instacart has an enormous amount of data on what products are in the baskets of its customers. The dataset was published in 2017 in an anonymised way to be used for a competition to improve Instacart's recommender system.

As widely recognized in the literature, training neural networks requires a large amount of data to produce dependable outcomes. In line with this, the dataset used in this study is suitable, as it contains over 3.2 million baskets collected from Instacart's customers. However, prior to feeding the KI-BaskERT model, the dataset needs to be cleaned and prepared. To this end, products that were purchased infrequently were eliminated from the dataset. Specifically, all products that were purchased less than 300 times throughout the dataset ($38,251$ products) were removed, resulting in a product range of $11,426$ products. This step is critical, as the model's ability to recognize structures in the dataset could be hampered if many products in the dataset are purchased infrequently. Furthermore, this could lead to the identification of insignificant relationships between products, ultimately resulting in an inaccurate model.

In addition to reducing the product assortment, the dataset only includes baskets that contain two or more products but less than or equal to 50 products. Baskets that contain a single product are not useful for training the recommender system as no information is

available for the system to base its predictions on. Additionally, baskets with more than 50 products are considered oversized, as they inundate the model with excessive information, reducing the coherence of products and creating inconsistencies and computational power difficulties during the model's training process. In addition, we do not observe many baskets in the dataset with more than 50 products. Following the data cleaning process, the dataset consists of a total of $3,014,658$ baskets. In Figure 1, a histogram of the basket sizes is presented, revealing that the majority of baskets contain less than 15 products, with an average of 10 products per basket and a standard deviation of 7.2 products.

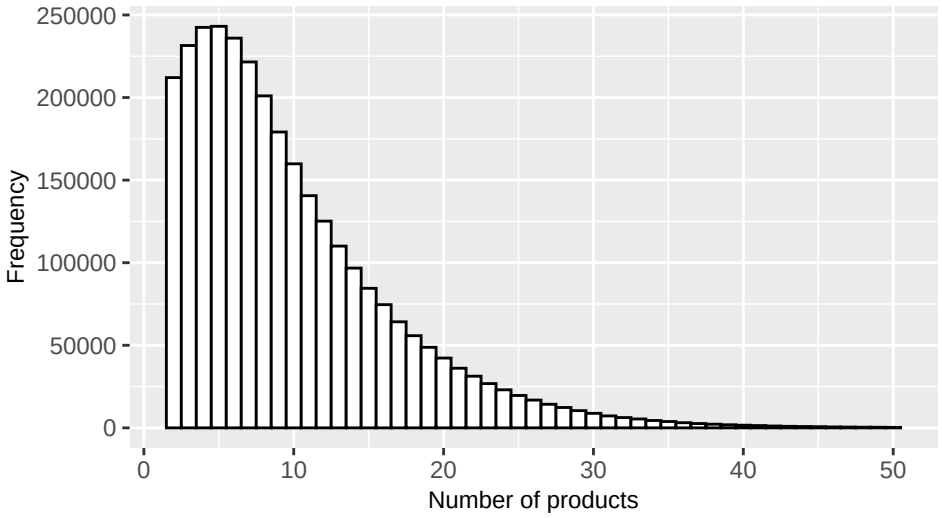

Figure 1: Distribution of Basket Sizes among Customers

Figure 2 depicts a significant graph that showcases the most frequently purchased products. The graph displays the top 30 products sold along with their respective quantities, as a proportion of the total number of baskets. Notably, the most popular products sold are bananas, followed by organic bananas, which amount to $852,015$ purchases out of $3,014,658$ baskets. Thus, on average, 28.26% of the baskets contain bananas. It is worth mentioning that fresh fruits and vegetables dominate the top 30 most purchased products.

This examination of the most commonly purchased products is essential as it serves as a benchmark for our trained model, both with and without knowledge injection. It enables us to evaluate the model's predictive value by comparing the accuracy of the model against the benchmark model, which suggests the top 100 best-selling products for each product

recommended. The benchmark model will be elaborated on later in this work.

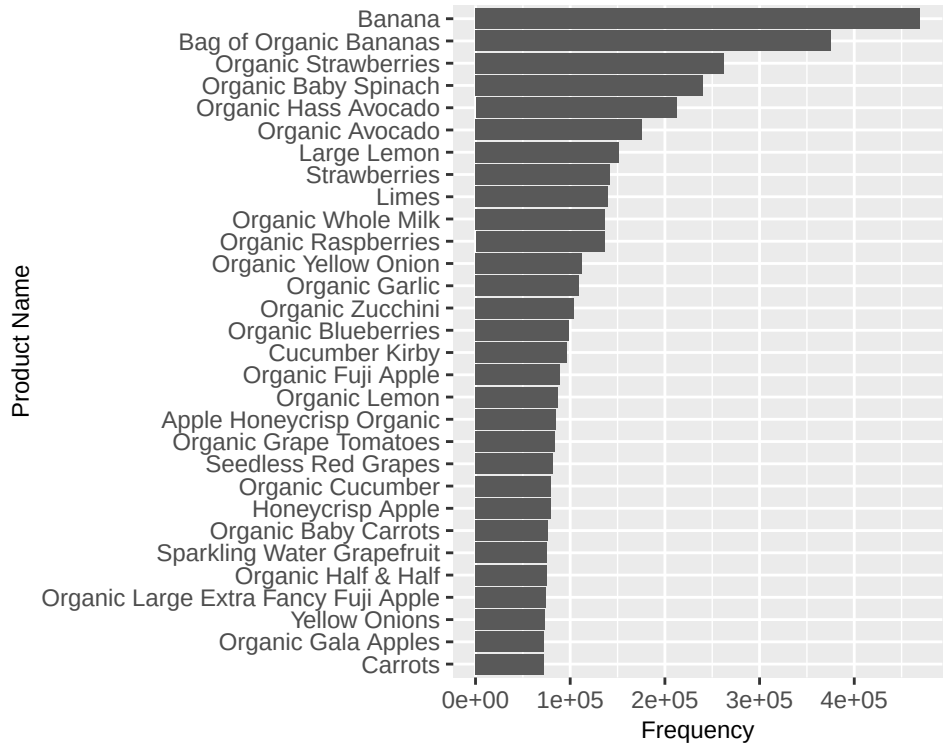

Figure 2: 30 Most Frequently Purchased Products

The final step of the data preparation phase is to divide the dataset into a training set, test set and validation set to evaluate the model efficiency. The training dataset is used, as the name suggests, for training the model. It is used to fit the internal model parameters. After this, the fitted model is used to make predictions on the observations in the validation dataset to obtain a first unbiased evaluation on observations unseen by the model. Different values for the hyperparameters are also examined on the validation set to obtain the final model. Finally, the test dataset is a dataset used to conduct an unbiased evaluation of the final model. Due to the large number of total observations, an $80-10-10$ split was opted for, with 80% of the total observations, i.e. $2,411,727$ baskets, composing the training dataset, and the validation set and test set each consisting of 10% ($301,465$ baskets) of the total number of observations. Thereby, all baskets were distributed in a randomised manner across the training, validation, and test set. The literature frequently mentions a method for enhancing the robustness of results, known as cross-validation, where

the data is iteratively divided into train, validation, and test sets through different splits, and the performance measures are averaged over these splits. Unfortunately, implementing this approach was not feasible within the constraints of time and computational resources of this work. In Appendix 1, a representation of the data after data preparation can be found. In order to incorporate additional knowledge into the model, we will utilise two datasets in addition to the basket datasets described above. These datasets provide information regarding the products' corresponding aisle and department. During the training phase, this information will be incorporated into the model by injecting it to the corresponding products in the basket. The aisle dataset includes a total of 134 different aisles, such as prepared meals, soap, coffee, and pasta sauces. The department dataset represents a higher level of subdivision, with only 20 different departments such as beverages, bakery, frozen, and breakfast. A depiction of the dataset that shows to which aisle and department a product belongs can be found in Appendix 2.

5 Methodology

Previous research has shown that several neural network architectures, such as recurrent and convolutional neural networks, have been successfully applied in the context of recommender systems. However, these unidirectional architectures suffer from limitations, including the inability to perform parallel computation and the restriction of encoding historical purchasing interactions from left to right only. To address these deficiencies, this work proposes the use of a bidirectional sequence model based on the BERT architecture. This architecture allows the model to simultaneously view and process the entire sequence of products in a basket, both left and right, to generate more contextualised product embeddings that result in improved predictions. Sun, F. et al. (2019) and Bianchi, F. et al. (2020) previously showed that recommendation systems applications based on the BERT architecture demonstrate improvements over predecessors and are therefore suitable methods for modelling clients' interests accurately.

The KI-BaskERT model that we introduce in this work is largely built upon the deep bidirectional self-attention model BERT from the work from Devlin, J. et al. (2018) but taken one step further with the novelty of Knowledge Injection. The inclusion of knowledge injection is an attempt to improve the just-mentioned BERT-based recommender systems. In the further course of the methodology section, the specific training procedure will be explained as a starting point as it can be seen as a prediction task itself. Then, the inner working of the BERT architecture including the ubiquitous method of self-attention (Vaswani A. et al., 2017) will be demonstrated that helped improve the performance of neural machine translation applications enormously. Afterwards, the explanation of the diverse methods for integrating knowledge into the KI-BaskERT model will be examined. Lastly, the measures for calculating our model performance will be elaborated upon.

5.1 Training Procedure

To effectively learn word characteristics and relations, the authors of BERT proposed a training objective named Masked Language Modelling (Devlin, J. et al., 2018) inspired by the Cloze task (Taylor, W.,1953). The Cloze task is often applied in primary schools to test

children their ability to understand context and vocabulary. Children are given a portion of language where certain words are removed, and they are asked to complete the missing words. Devlin, J. et al. (2018) replicated this task during the training phase of their language model by simply masking some percentage of the tokens from the input sequence at random and let the model predict those masked tokens, hence the name Masked Language Modelling. The meaning of a token depends on the application, e.g. in natural language processing a sequence of tokens may represent the words in a sentence, as opposed to in a recommendation system where a sequence of tokens represents a basket of products. The authors would thus feed the model with sentences where a percentage of the words are left out and let the model predict these words. In our recommender system setting this means randomly masking a product from a basket and letting the model predict that masked product. The training objective can therefore intuitively be seen as a product prediction task itself. Since the baskets of products from our dataset contain at most 50 products, and are therefore not excessively large sequences, to train the KI-BaskERT model one random product from each basket/sequence will be masked during training.

In practice we start with the complete training data containing $2,411,727$ baskets in total. A random selection of 128 baskets from the full training dataset is sampled, referred to as a batch. This is because the training procedure is an iterative process in which small pieces of data are processed by the model. The literature mentions taking a batch as a fraction of the training dataset between 64 and 512 observations (Keskar, N. S. et al., 2016). An excessive batch size may lead to a weaker ability of the model to generalise, which is why 128 was decided as the batch size in this work. In the subsequent step, the model randomly replaces one product from each basket from the batch with the mask token. This masked batch is fed to the KI-BaskERT model that computes probabilities per masked product per basket based on the information about the composition and context of the basket. These probabilities per masked product are determined in a vector in which each product of the total product assortment is assigned a probability defining how likely it is that a product is the masked product of that basket.

Subsequently, the performance of the model is calculated on the predictions of the batch. To measure the performance, or rather the loss of the predictions accompanied with the cur-

rent internal parameter values, Cross-Entropy Loss (De Boer, P. T. et al., 2005) is used in this work. Cross-entropy loss is widely used as a loss measure when optimising neural networks that involve binary target values. It quantifies the error between the predicted values, i.e. the outcome probabilities of the masked products, with the actual values and expresses this in a single real number. The smaller the loss, the better our model performs. The computation of Cross Entropy Loss is formulated by: $L_{CE} = -\frac{1}{N} \sum_{i=1}^{N} \log(Prob(maskedproduct_i))$ (Equation 1), where $Prob(maskedproduct_i)$ represents the probability the model outputs for the masked product of basket i. Thereafter, the logarithm is taken to project the probability on a number between minus infinity and zero. Taking the logarithm is advantageous since poor predictions, characterised by probabilities close to zero, result in a large negative value after taking the logarithm, thus resulting in a high positive loss due to the minus sign in the formula. Conversely, good predictions made by the model, i.e. probabilities close to one, provide small negative values after logarithm and thus result in a low loss. The total loss is calculated by averaging the loss values for all masked products of one batch. Since we have a batch of 128 randomly selected baskets of which one product each is masked, N is 128 in this application.

After the model has processed a batch, the resulting predictions are compared with the true masked products by calculating the cross-entropy loss that examines "how far" the predictions are from the real values. From this error, an update algorithm is used to adjust the learnable parameters of the model to conceivably improve the model before the next batch is processed. What these learnable parameters entail will be discussed later. This explains why we feed only small, masked batches of training data to the KI-BaskERT model, as it is an iterative optimisation process that updates the learnable parameters after each batch.

Upon completion of processing a batch, the internal model parameter values are updated by calculating the gradients of the loss function with respect to the model parameters. The update algorithm then attempts to find the optimal parameter values by examining local minima of the loss function with respect to the model parameters. The gradients of the loss function are used to guide the update algorithm in the direction of decreasing loss. In the literature, examining these local minima is referred to as descending along the gradient for finding the optimal values for a parameter, hence the name Gradient Descent

(Ruder, S. 2016). Stated otherwise, gradient descent aims to discover a set of internal model parameter values that excel in accordance with our performance measure, cross-entropy loss. More specifically, to build our KI-BaskERT model, we use the Adam optimiser algorithm, a special gradient descent approach that is said in the literature to provide faster convergence of model parameters. The exact internal workings of the Adam optimiser algorithm are not part of this work, for which we refer to the literature (Kingma, D. P., & Ba, J., 2014).

When all observations from the full training dataset, drawn at random, have been processed once by the model, this is referred to as an epoch. This means that each observation, i.e. basket from the training dataset, has had an opportunity to exert its influence on the internal model parameters. The complete training procedure consists of multiple epochs until model convergence and all learnable parameters are optimised. By model convergence we mean that the loss and accuracy of the model on the training dataset do not improve significantly anymore by updating the model parameters. Considering the computational resources available for this work, we take the cut-off of ten epochs, i.e. if over the most recent ten epochs the loss and accuracy do not improve significantly anymore, the model has converged and with a maximum of 70 epochs.

On figure 3 the training procedure is visualised for an example of one basket. In practice, this procedure is performed on the whole batch at once. The way in which the model arrives at these probabilities will be clarified later when we unravel the KI-BaskERT model.

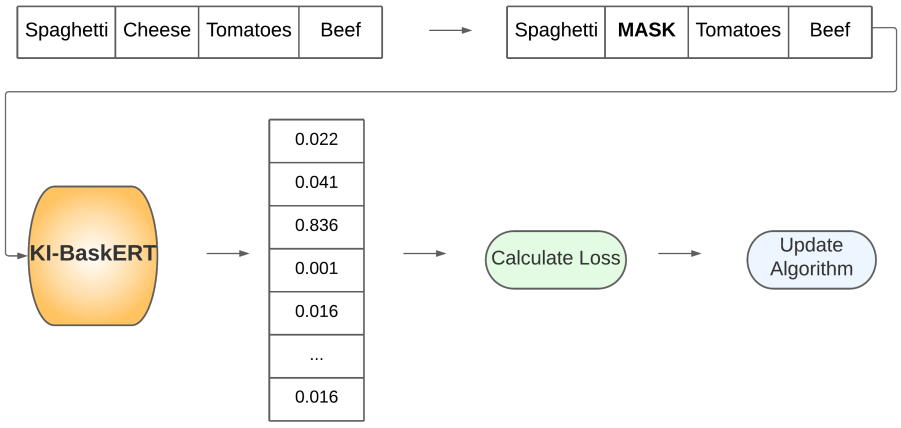

Figure 3: Training Procedure for one basket

5.2 KI-BaskERT Model Architecture

The KI-BaskERT model architecture is predominantly derived from the BERT architecture (Devlin, J. et al., 2018). BERT, in turn, is a derivative of the Transformer network. Inspired by the success of the BERT architecture in understanding and representing text, this architecture is adopted in this work to model sequential product recommendation. The following section will provide a more in-depth explanation of the product embeddings, a crucial component of the model for understanding products. This will be preceded by a depiction of the encoder and its various components including the self-attention mechanism and the feedforward neural network.

5.2.1 Product Embedding

As briefly touched upon in the literature review, product embeddings are the numerical representation of products in a vector. The model uses these vectors to identify and comprehend products. Product embeddings are one of the learnable parameters of the KI-BaskERT model, i.e. they are updated after a batch is processed. Based on these product embeddings, the model understands the composition of the baskets and leverages this information to make predictions about the masked products.

At the end of the training process, we expect the model to have learned the similarities and differences between products. These similarities and differences are reflected in the product embeddings. By "similar", we mean that certain values in the product embedding vectors are the same or at least in the same direction (positive or negative). This can be interpreted as a high-dimensional vector space, in which the embedding vectors represent the products and similar products are close together in this high-dimensional vector space. To illustrate this concept, consider the following example with four products: gouda cheese, parmesan cheese, spaghetti, and toilet paper. It is expected that the product embeddings of parmesan cheese and gouda cheese are very similar because they are both types of cheese. In addition, the product embeddings of parmesan cheese and spaghetti are also likely to be related because they are often used together in pasta recipes. On the other hand, toilet paper is not related to the other products and is therefore expected to have a completely

different product embedding.

Determining the length of the embedding vector is an important architectural choice. A trade-off must be made between the size of the embedding vector itself, the size of the product assortment, the duration of the training phase and model performance. An embedding vector that is too small may prevent the model from adequately encompassing the products, resulting in low model performance. Conversely, larger product embeddings do provide improved performance, but are accompanied by longer computation time during the training phase. Additionally, larger product embeddings only provide enhanced model performance to a certain extent, as embedding vectors that are too large can furnish the model with an excess of information, hindering model convergence. A further disadvantage of excessively large embeddings is the risk of overfitting, which is when a model performs well on the training data but poorly on new data. This is because large word embeddings can contain a vast amount of information and can be highly detailed, which can make it easier for a model to memorise the training data rather than generalise to new data. To take all these considerations into account while remaining in line with the work of Devlin et al. (2018), we use the same ratio of embedding size to assortment size in this work. The implementation of BERT utilised a vocabulary of $30,000$ distinct words associated with a word embedding vector of size 768, hence a product embedding vector of size 294 is used in this work for a product range of $11,426$ products.

When the model is invoked for the first time, the product embeddings are Xavier uniform initialised. This means that the initial values of the product embedding vectors are set randomly to values drawn from a Xavier uniform distribution in such a way that the variance across all parameters is the same. These values are thus assigned in a random manner after which they are updated after each batch processing. The following step the model is invoked, the updated product embeddings are selected and further updated until model convergence.

5.2.2 Encoder

The encoders mainly define the architecture of the KI-BaskERT model. The purpose of the encoders is to encode and process the information from the baskets by constantly producing improved product embeddings that yield more accurate predictions. As can be observed from figure 4, each encoder consists of two sublayers, namely a multi-head self-attention mechanism and secondly a fully connected feedforward network which will be both outlined in the subsequent sections.

Around every sublayer, a residual connection is employed to allow a smooth gradient flow through the model. Residual connections are separate paths for the data to reach latter parts of network by skipping some computations. In our model this means that the outcome resulted from the previous sublayer is added back to the next sublayer outcome. In the literature, it is claimed that the model converges faster with the use of a residual connection (He, K. et al., 2016). Furthermore, Figure 4 demonstrates that before the masked baskets are fed to the encoder, they are first converted to their corresponding product embeddings. Lastly, the "x N" on Figure 4 refers to stacking multiple encoders on top of each other, named layers, where the output of the prior encoder is the input of the subsequent encoder.

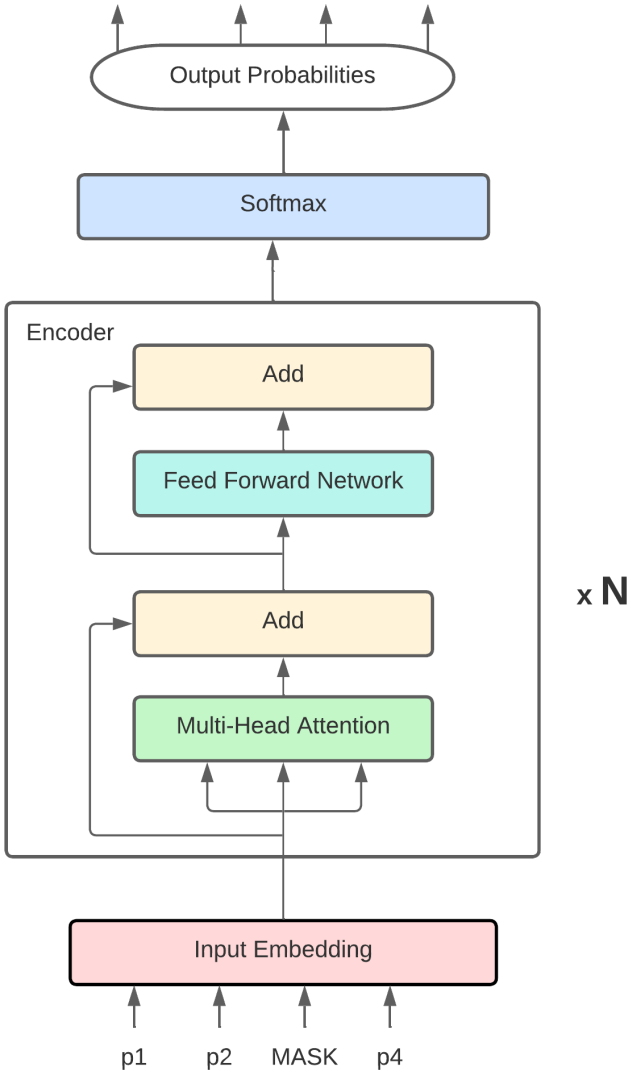

Figure 4: KI-BaskERT Architecture

5.2.2.1 Attention Mechanism The biggest novelty introduced in the Transformer network is the addition of the concept attention. For a moment, consider the definition of the word attention in psychology. Attention stands for the ability to actively process specific information in the environment while ignoring other details. A neural network is considered an attempt to mimic the human brain actions in a facilitated way. Similarly, the attention mechanism is an attempt to implement the same action of selective concentration on relevant parts of the basket while ignoring others. More specifically, the paper by Devlin, J. et al (2018) refers to the concept of self-attention which comes down to allowing a token to look at itself and the relation with other tokens in the sequence to enrich the embedding of the token in question. Briefly applying this to our recommender system setting, self-attention allows a product to look at or pay attention to the product itself and how it relates to other products contained in the basket. The objective is to better understand the context in which products are bought together and consequently enhance product embeddings with this information to yield better model performance.

Remember the pasta recipe example, where spaghetti and Parmesan cheese are likely to be frequently purchased together. If these products frequently appear together in the baskets in the training data, the attention mechanism will will attach importance to this relationship. Moreover, it will focus on this relationship while disregarding less important product associations in the basket, such as the co-occurrence of pasta and toilet paper, which are not typically related. The objective is to concentrate on the smaller yet meaningful segments of the product sequence and ignore other parts. In practice, the model obtains this form of attention by computing similarity scores between products. The advantage of the BERT architecture demonstrated here is the bidirectionality that allows the model to immediately capture significant relationships along both sides of the basket, which was not possible with unidirectional architectures. In addition, it does not matter how far these products are apart from each other in the basket, where recurrent neural network architectures did encounter difficulties handling long sequences.

Attention is computed using an attention function that calculates for all tokens in a sequence how relevant the token itself is and, the relationships with other tokens are in the form of a similarity score. Devlin, J. et al (2018) define the attention function as:

> Mapping a query and a set of key-value pairs to an output, where the query, keys, values, and output are all vectors. The output is computed as a weighted sum of the values, where the weight assigned to each value is computed by a compatibility function of the query with the corresponding key.

This definition refers to queries, keys and values that are matrices obtained by multiplying the product embedding by three different learnable parameter matrices. These learnable parameter matrices, like the product embeddings, are parameters of the model that are initialised at the beginning of the training phase to be updated during the training phase. At first glance, the definition of attention and the meaning of the query, key and value vectors seem opaque. An analogy can be made with the functioning of a search system such as YouTube, for example. This search engine will match the search term entered, the query, against the set of keys, which in this example are the names of the videos in the database. To finally recommend a list, the values, of videos that best match the request. In our recommender system setting this means matching a product name (the query) against all the products in the product assortment (the keys) to suggest a list of best matching products (the values). In practice, these are matrices filled with values on which mathematical operations are performed. Therefore, let us see how this process works mathematically in the context of a recommender system.

In the following example illustrated on Figure 5 the attention is computed for the second product p_2 in a single basket containing n products (maximum 50) where the sequence of products is fed simultaneously to the attention mechanism. As can be observed on Figure 5, the first step considers the product embeddings of the different products in the basket $(p_1, p_2, \ldots, p_n)$. In what follows, all steps must be carried out on a product-by-product basis. In the second step, the product embedding vectors are multiplied by their learnable product-specific weights vectors W^q, W^k and W^v to obtain the queries, keys, and values vectors for each product.

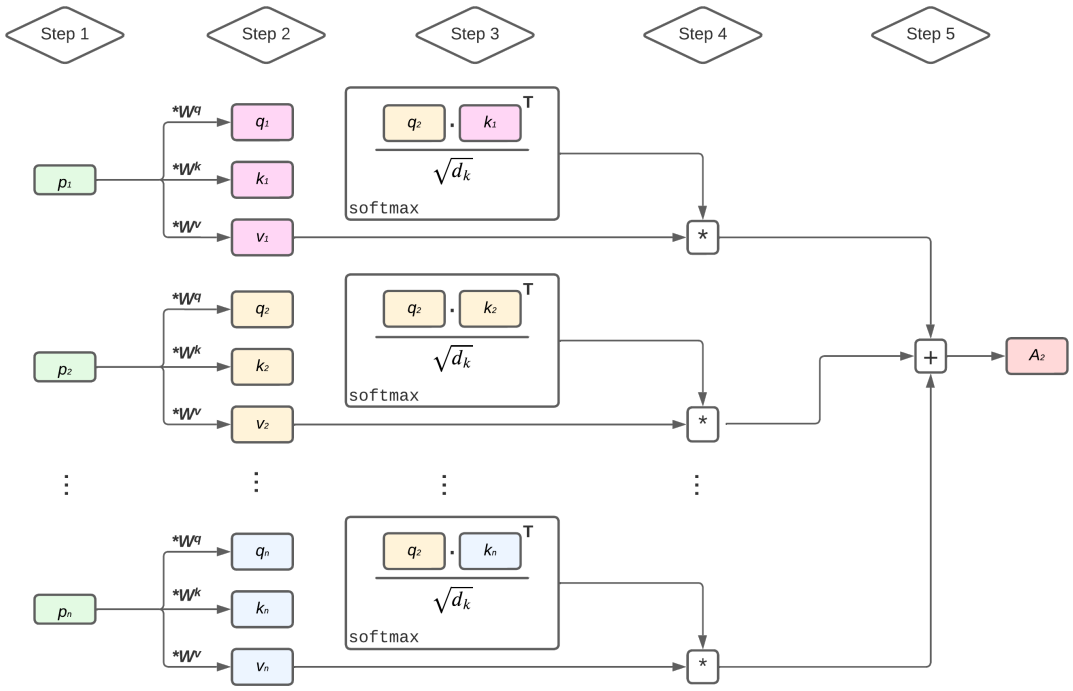

Figure 5: Calculation of Attention for second product in a basket of n products

The third step is very important as the dot product is performed between the queries vector (q_2) of the product of which we are calculating the attention for, here the second product p_2, and the transposed keys vectors of the all the corresponding products in the basket. This step is also called multiplicative attention which computes a form of similarity or compatibility measure. Note that by this multiplication with the transposed keys vector, the resulting similarity score is a real number. This real number is subsequently divided by the root of the dimension of the keys vector to prevent the values from becoming too large to ensure more stable gradients. All the outcomes resulting from step three are passed through a SoftMax function. The SoftMax function, represented by $\sigma(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \quad for\ i = 1, 2, \ldots, K$ (Equation 2) is a normalisation function that transforms a sequence z of K real values into a probability distribution with the property that all probabilities in that sequence sum up to one. The SoftMax function hereby forces low or negative values to convert to very small probabilities and high values to higher probabilities. This SoftMax outcome per product

indicates a score that expresses how much attention is paid at each product from the basket and the product itself. Intuitively, a higher score indicates greater focus on the product itself or other products in the basket.

Subsequently, in step four, these weighted SoftMax scores are multiplied by the value vectors of the corresponding products. The intuition is to keep the value vectors of the product or products we want to focus on intact and diminish irrelevant product value vectors. After multiplication with the SoftMax score, more attention is given to high-scoring products and conversely less attention to low-scoring products. Ultimately, the outcome of the attention layer for the second product is obtained in step five by summing all the resulting value vectors from step four. This amounts to a weighted average of all the value vectors based the normalised SoftMax scores. This vector is added back to the original product embedding of the second product of the specific basket by means of the residual connection to eventually be fed to the feed-forward network, the second sublayer in the encoder block.

Retake the example basket consisting of the products pasta, parmesan cheese, and toilet paper. We demonstrate the computation of attention for the product pasta. Remember that pasta and parmesan cheese have some form of relationship as they are both ingredients of a pasta recipe. Therefore, it is expected that pasta and parmesan cheese often appear together in baskets as opposed to with toilet paper. In step three when computing the similarity scores, the largest dot product value will be that of the pasta query with the pasta key vector itself. Subsequently, it is expected that the dot product between the pasta query and parmesan cheese key vector will provide a much higher value compared to the dot product between pasta with toilet paper due to their relationship. Therefore, after the SoftMax function, this low dot product value will be converted to a very low similarity score. As a result, the values of the toilet paper value vector will almost be vanished after multiplication with the low similarity score. This is important as we do not want to give attention to this product. We eventually obtain the attention vector for the product pasta as the weighted sum of the value vectors of all three products in the basket. The resulting attention vector contains information about pasta itself but also about its relationship with the product cheese, while omitting the association with toilet paper for that basket. The attention mechanism thus implements the action of selective concentration on a few relevant

aspects of products while ignoring others by mathematical operations.

In practice, however, attention is not calculated for one product at a time with product-level vectors but for all products in the basket together, using matrix operations to make this process faster and computationally more efficient. In the original BERT paper this is referred to as Scaled Dot-Product Attention (Devlin, J. et al., 2018). The first step in calculating attention using matrix operations is depicted in Figure 6. Here, the product matrix P with the rows representing the products in the basket and the number of columns being the size of the product embedding, is multiplied by three learnable weight matrices W^Q, W^K and W^V to produce the queries, keys, and values matrices of a basket.

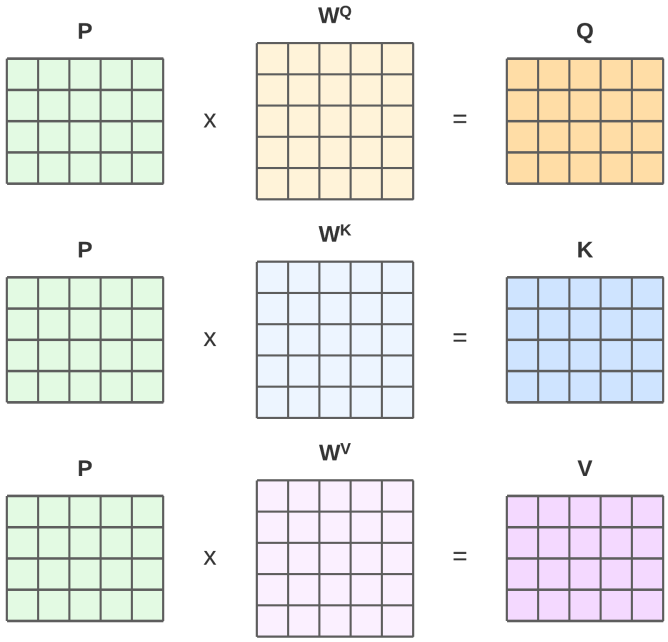

Figure 6: Calculation of Queries, Keys and Values matrices

The dimensions of the weight matrices are equal to the size of the product embedding for both the number of rows and columns. In the next step, shown on Figure 7, the queries matrix is multiplied by the transposed keys matrix to then be divided by the square root of the dimension of the keys matrix d_k. Thereafter, the entire matrix passes through the SoftMax function with respect to the rows to become all the similarity scores for a basket. This result is multiplied by the values matrix of the basket before arriving at the attention matrix of that specific basket. Each row from the attention matrix A can be seen as the

weighted sum of the values vectors of all the other products weighted by the similarity score determined by the SoftMax function. Consequently, information about the context of the products in the basket and their relationships is stored in the attention matrix. The attention matrix is added to the output of the residual connection to be fed to the feed forward network in the subsequent sublayer as shown in Figure 4. Important to notice, during the implementation of the KI-BaskERT model, a third dimension comes into play, namely a dimension for the batch. This enables us to efficiently calculate attention for all products in the basket, and for all baskets in our batch at the same time.

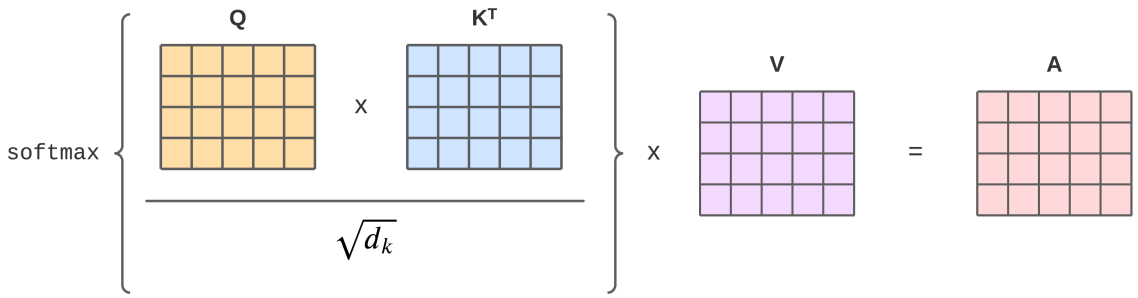

Figure 7: Scaled Dot-Product Attention

Mathematically the computation of the attention for one basket using matrix multiplication is described as: $Attention\,(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V$ (Equation 3) where Q, K and V are the queries, keys, and values matrices of a specific basket, displayed in Figure 7.

5.2.2.2 Multi-Headed Attention The computation of attention is taken one step further to what is called Multi-Headed Attention. This is a very clever and advantageous extension of attention. It comes down to running the attention mechanism part as shown in Figure 4 in parallel in the so-called heads, hence the multiple arrows on the figure leading to the multi-headed attention mechanism. In other words, for each head, separate queries, keys, and values weight matrices are initialised, which in turn provide different queries, keys, and values matrices. This set of different queries, keys and values matrices is also referred to as creating different "representation subspaces". Attention with multiple heads allows the model to simultaneously attend to information from various representation subspaces at different positions. When there is only a single attention head, this is obstructed by averag-

ing. This has the great advantage for our recommender system that it allows the attention mechanism to pay attention to different relationships or aspects of the sequence of products in the basket.

Consider another example with a basket containing the products milk, bread, butter, and eggs, and the goal of the recommender system is to make recommendations based on these items. The recommender system in this example features a multi-headed attention mechanism with two heads, which means that it can consider two different aspects of the basket at the same time. The first head of the multi-headed attention mechanism will be focusing on the relationships between milk and butter, and eggs and butter. These products are often used together in cooking and baking recipes, so the model is considering the possibility that the client might be interested in baking recipes that include these ingredients. The second head of the multi-headed attention mechanism is focusing on the relationships between milk and bread, and eggs and bread. These products are often consumed together as a breakfast meals or snacks, so the model is considering the possibility that the client might be interested in breakfast recipes. By considering both relationships and aspects of the input basket simultaneously, the model can make more accurate and personalised recommendations based on the specific context of the items in the shopping basket. In this example, the model might recommend products related to a recipe for scrambled eggs and toast, or a recipe for a milk-based smoothie to go with the bread.

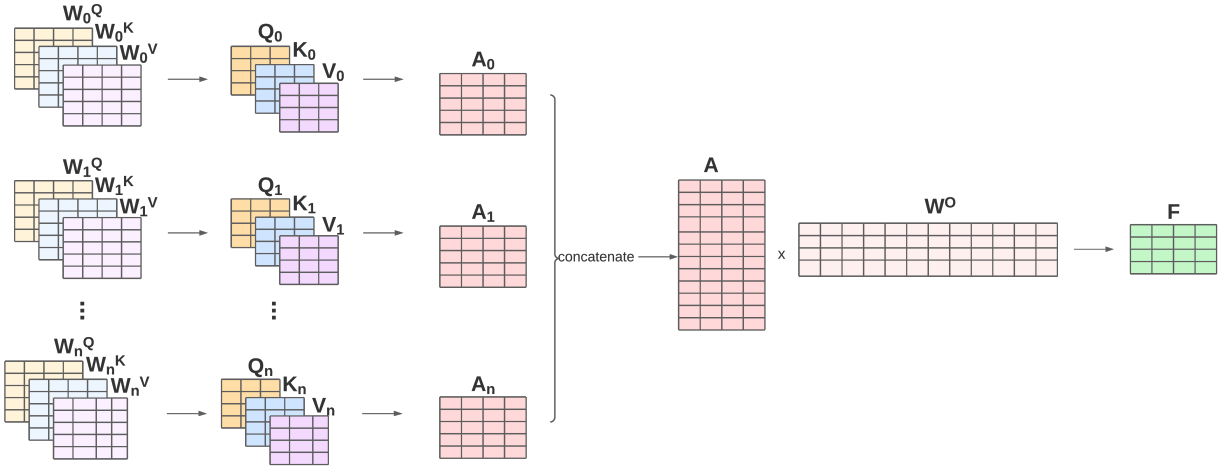

Figure 8: Multi-Headed Attention

However, initialising separate queries, keys, and values weight matrices creates a complication, namely the feedforward network expects only a single matrix to process. If we work with e.g. six heads, we will have six output matrices at the end of the computation of the multiheaded attention. For this reason, the number of output matrices, depending on how many heads there are, are concatenated and multiplied by another additional weight matrix W^O. This weight matrix is another learnable parameter of the model and therefore likewise updated during the training phase. Because we get a large, concatenated matrix due to our different layers, multiplying by W^O ensures that the final output matrix of the multi-headed attention mechanism achieves the desired dimensions for the feedforward neural network. The resulting matrix F (shown in Figure 8) is essentially the product embedding matrix for all the baskets in the batch but now added with attention, or in other words weighted by the similarity scores determined by the attention mechanism. F is ultimately added to the residual connection output to be sent through the feed forward network.

5.2.2.3 Feedforward Neural Network The feedforward neural network is the second important sublayer (see Figure 4) of the encoder block. The output of the multi-headed attention mechanism, i.e. the product embeddings of the baskets combined with the attention matrix, is used as input to the feedforward network to compute contextualised and knowledge-enriched product embeddings. To do this, the feedforward network first unpacks and expands the information from the product embeddings by incorporating non-linearity into the model, and then repackages it to its original size. With the use of the update algorithm, the feedforward neural network learns to map the input data to the desired output by adjusting the weights and biases of the connections between the nodes. The feedforward neural network is an essential component of the architecture because it enables the model to learn complex relationships between the baskets with a masked product and the masked product itself, and to make intelligent decisions based on those relationships. The resulting contextualised product embeddings are then used to produce scores about the masked products, which are eventually converted into the final outcome probabilities.

The feedforward neural network is a type of neural network that is the earliest and simplest form of a deep learning architecture. It is called "feedforward" because the data

flows only in one direction through the network, from the input nodes through any hidden layers (which may consist of multiple layers) and finally to the output nodes. In contrast to other types of neural networks, such as recurrent neural networks, feedforward neural networks do not have feedback connections that allow data to flow in both directions. Each layer of a feedforward neural network consists of nodes, and each node in a given layer receives input from every node of the previous layer and sends output to every node in the subsequent layer. This is known as being "fully connected".

As for the size of the feedforward neural network for the KI-BaskERT model, there is opted for a single hidden layer as this is sufficient for the majority of problems trying to solve with a feedforward neural network. Regarding the size of the hidden layer, i.e. the number of nodes in the hidden layer, the same proportion as in the original BERT paper is used. Namely, the size of the hidden layer is set at four times the size of the embedding. A visual representation of the network can be found on Figure 9.

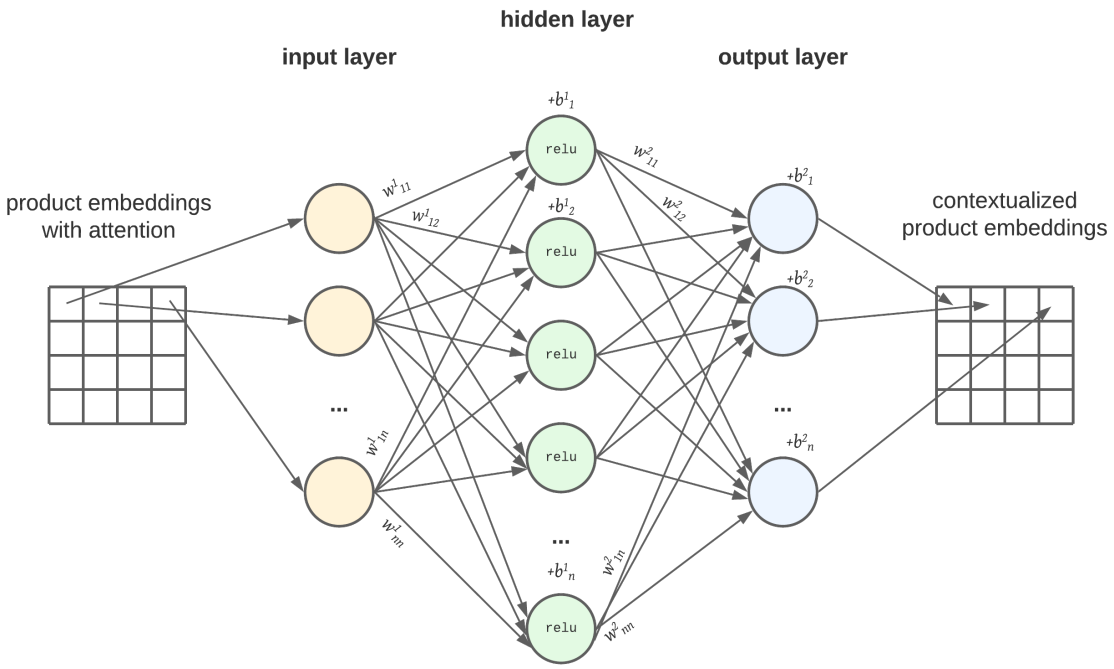

Figure 9: Feedforward Neural Network

The mathematical representation of the implemented feedforward neural network in this

work is described as $FFN(x) = \max(0,\ xW_1 + b_1)W_2 + b_2$ (Equation 4). In this equation, x is the input of the network and consists of the output of the multi-headed attention mechanism (see matrix F from figure 8) added with the output of the residual connection, i.e. the baskets represented by their corresponding product embeddings. W_1, W_2, b_1 and b_2 are the learnable parameters of the feedforward network. The weights (W_1, W_2) regulate the signal or strength of the connection between two layers in the network. In other words, the weights determine how much influence the input (of that layer) has on that layer. More specifically, they convey the importance of an input on the output. The biases (b_1, b_2), which are constants, are an additional value to be added to the input of a layer after multiplication with the weights. They can shift the input of the activation function or the final output. The max-function in the formula is the Rectified Linear Unit (ReLU) activation function of the hidden layer. It is a function that will output the inputs of the hidden layer directly if the output is positive and will output zero otherwise. The main purpose of the activation function is to introduce position-wise non-linearity into the output of a node, and it decides whether a node will be activated or not. Notice that there are as many distinctive W_1, W_2, b_1 and b_2 matrices as there are layers, i.e. encoders, in our model architecture.

To prevent overfitting, a computationally cheap and remarkably effective regularisation technique named dropout (Srivastava, N. et al., 2014) is applied to our feedforward network. Overfitting means that the model learns the statistical noise of the training data and therefore performs poor on the unseen validation dataset. With dropout, randomly selected nodes from the hidden layer are disabled making the training phase noisy on purpose, forcing the remaining nodes in the hidden layer to probabilistically take on greater or lesser responsibility in calculating the output. The reasoning behind this is that by dropping nodes randomly, one can turn a single neural network into many different network architectures that function as an ensemble of networks, thereby reducing overfitting. Different dropout rates are validated in the results section to determine the optimal dropout rate.

5.2.3 Knowledge Injection

As discussed in the literature review, similar applications of the BERT architecture with its multi-headed attention mechanism have already been investigated in a recommender system

setting and showed promising results. A successful implementation is the BERT4Rec model proposed by Sun, F. et al. (2019). However, this model solely relies on product identifiers and co-occurrences of products in the baskets and ignores other possible sources of information. Here we propose four different approaches to incorporate additional information by including the aisle or department information of the products into the model. We expect the model to become smarter in recognising structures of the baskets. Retake the cake recipe example where a client would like to bake a cake and therefore needs products including flour, sugar, eggs, chocolate baking chips etc. Adding the aisle information to these products, we expect the model to recognise that flour and sugar are related to each other as they belong to the same aisle. We therefore expect that adding this additional information will provide more accurate predictions on masked products.

Four different approaches of Knowledge Injection will be elaborated and tested to ultimately be compared with each other. The first approach is to reserve a percentage of the embedding for aisle information. This is done by training a supplementary embedding, namely the aisle embedding, in addition to the product embedding. These are merged into the so-called total embedding just before going through the KI-BaskERT model. Different percentages of allocation of aisle information from the total embedding are tested ranging from 20% to 40%. If the total embedding for one product consists of a vector of length 294, the first positions in this vector are filled with values from our known product embedding and the last e.g. 20% of the positions in this vector are reserved and filled with data from the corresponding aisle embedding values. The aisle information is pasted behind the product embedding, so to speak. Since many products belong to a particular aisle, all products belonging to the same aisle will show the same values for their last percentage of positions of their total embedding. Since products become intuitively more similar in the high dimensional vector space by allocating positions of the total embedding to the aisle information, we hope that the model can now understand relationships between products more effectively.

The second approach is to train an aisle embedding alongside the product embedding that is of the same length as the product embedding itself. The total embedding is then obtained by adding the aisle embedding per product to the corresponding product embedding. As a result, we are not bound by the number of positions of the vector representing the total

embedding that we must allocate to product information and aisle information. Note that in previous approach we had to reserve a percentage of the positions of the vector for aisle information by "pasting" it behind the product information. Now both product embedding and aisle embedding are of size 294 and are added together before proceeding the attention mechanism. This allows them both to possess and add the same amount of information to the attention mechanism. If for each product the same values are subtracted or added to its associated product embedding, determined by the values of the accompanying aisle embedding, we expect that the model has received additional information and can therefore make better predictions. In the literature, it is believed that summing the two embeddings provides more desirable results since we are not bound by dimension differences (ENCCS, 2022). On top of that, one could also think of the concatenation of the previous approach as an addition of the two embedding matrices where they contain zeros for the places where they exclude each other.

The third approach is a variant of the first approach, now adding information is taken one step further by adding the department information of the product on top of the aisle information. There are even fewer departments than there are aisles so the total embedding matrix will show a kind of cascading pattern of information. Namely, the first positions in the product embedding vector are allocated to the product information values, followed by a proportion of the positions allocated to the aisle information and department information. Again, different proportions of allocation to aisle and department information will be tested. We expect that this will further improve the model's ability to recognise product subdivisions. We make the analogy with a physical store, take the product Blueberry Muffins as an example. If were go to a physical store and would search for it, we first need to go the bakery department, then to the breakfast bakery department to find the muffins there. We want to implement this same way of reasoning in the model via knowledge injection by first including the department information and thereafter the aisle information.

A latter evident approach is to add the product, aisle, and department embeddings of the same size for the corresponding products in the basket. We depart from the same reasoning of adding embeddings as in the second approach. Now, three different types of information are added together with the expectation of even better model performance.

5.2.4 Cross-Entropy Loss & Accuracy

To evaluate the performance of our models, we report the cross-entropy loss and accuracy. The cross-entropy loss is the average loss over all predictions made in one epoch and is defined in Equation 1 in the methodology section. Accuracy is defined in this work as the ratio of the number of times the model correctly recommends the masked product in an epoch to the total number of predictions made in that epoch. The models are allowed to recommend the top hundred most probable products for each basket. If the masked product is in this top hundred for that specific basket, the prediction is considered correct. If the masked product is not in the list, the prediction is considered incorrect. We chose to allow the model to recommend the top hundred products because it is unlikely that the model will be able to predict the exact masked product from a product assortment of $11,426$ items with a high probability. By allowing the model to recommend the top hundred products, we give the model the opportunity to suggest its top 1% most probable products (0.875% in reality). Additionally, it is more realistic for a recommendation system on a website or app to suggest multiple products based on the composition of the basket instead of just one.

6 Results

In this section, we assess the ability of our Transformer-based recommender systems to predict the masked product for a given basket. We also investigate whether the different configurations of Knowledge Injection improve model performance by providing additional information to the model.

Initially, various architectural choices must be made prior to arriving at the final models. Consequently, we first validate various hyperparameters on the BaskERT model to subsequently implement them in our KI-BaskERT models. Thereafter, we present the results of six different models. In addition to the results from the four configurations of the KI-BaskERT model, we also report the results from two additional models: a BaskERT model without the use of Knowledge Injection, and a benchmark model. The BaskERT model serves as a point of comparison to assess whether the models with Knowledge Injection outperform the same model without Knowledge Injection. Secondly, the benchmark model always recommends the top hundred most purchased products for each masked product and does not rely on a neural network or any learning process. Note, as a result this model does not produce any loss values. We use the results of the benchmark model to determine whether our BaskERT and KI-BaskERT models have explanatory value and learn meaningful structures in the data.

6.1 Determination of the hyperparameters

Before we can assess our KI-BaskERT models on the test dataset, decisions must be made about various hyperparameters, including the number of layers in the architecture, the number of heads in the multi-headed attention mechanism, the dropout rate, and the proportion of Knowledge Injection (for the first and third approach). We report the results of different model configurations of the BaskERT model without knowledge injection and use the optimal hyperparameter values to evaluate our KI-BaskERT models. In what follows, the size of the embedding will always be 294, as previously determined in the methodology section.

Table 1
Dropout

Model	Dropout Rate	Training Cross-Entropy Loss	Training Accuracy (Top 100)	Validation Cross-Entropy Loss	Validation Accuracy (Top 100)
BaskERT	0.00	6.983	0.413	7.024	0.408
BaskERT	0.10	7.090	0.398	7.115	0.393
BaskERT	0.25	7.199	0.379	7.205	0.378
BaskERT	0.50	7.401	0.346	7.402	0.346

6.1.1 Dropout

The BaskERT model with a product embedding of size 294 and 2 layers was tested with different levels of dropout to determine the optimal rate. The rates tested were: $0.00, 0.10, 0.25$, and 0.50, representing the proportion of random nodes in the hidden layer that are disabled when the feedforward network is invoked. The results in Table 1 show that as the dropout rate increases, loss increases and accuracy decreases. This suggests that higher dropout rates cause the feedforward network to underfit the data, leading to a decrease in model performance. Therefore, a dropout rate of 0.00 was chosen as there were no indications of overfitting, which suggests that the model generalises well.

6.1.2 Layers and Heads

To determine the optimal number of layers and heads in the multi-headed attention of the BaskERT model, we incrementally increased the number of layers and heads from 1 to 3. The results in Table 2 show that increasing the number of layers and heads improves model performance. However, this improvement comes at the cost of an increase in the average epoch duration. The average time per epoch increases significantly when going from one layer and head to two. Specifically, the average epoch time increases by 15.82%. Adding another layer and head results in another increase of 14.98% in the average epoch time, but the improvement in model performance is lower in the latter case. We conclude a diminishing marginal increase in model performance as the number of layers and heads increase. Considering the limited improvement in performance and the substantial increase

Table 2
Layers and heads

Number of layers and heads	Average time per epoch (in seconds)	Training Cross-Entropy Loss	Training Accuracy (Top 100)	Validation Cross-Entropy Loss	Validation Accuracy (Top 100)
1	196	7.035	0.405	7.052	0.403
2	227	6.983	0.413	7.024	0.408
3	251	6.956	0.418	6.995	0.413

in training time, the final model chosen has 2 layers and heads, which provides sufficient flexibility for the recommendation task.

6.1.3 Proportion of Knowledge Injection

The final decision to be made concerns the proportion of Knowledge Injection for the KI-BaskERT models that allocate a part of their total embedding to aisle and/or department information. These models concatenate the aisle and/or department embedding behind the product embedding to become the total embedding. For the KI-BaskERT model that concatenates the aisle embedding to the product embedding (first approach), different proportions of Knowledge Injection are evaluated in Table 3 to determine their impact on the model's performance. To clarify, a proportion of 20% implies that the last 59 places of the total embedding vector are reserved for values of aisle embedding. For propotions 30% and 40%, these are 88 and 118 places respectively.

According to Table 3, the most favorable result is obtained when 30% of the total embedding is allocated to aisle information. Furthermore, it can be observed that both insufficient and excessive allocation of aisle information can negatively affect the model's performance, although the differences in performance are minor. Therefore, in the subsequent comparison of the models, we choose to allocate 30% of the total embedding to Knowledge Injection for the models that concatenates aisle and/or department. For the KI-BaskERT model that concatenates both the aisle and department information, this 30% is further divided with half being assigned to aisle information and the other half to department information.

Table 3
Proportion of Knowledge Injection for KI-BaskERT that concatenates product and aisle embedding

Proportion of Knowledge Injection	Training Cross-Entropy Loss	Training Accuracy (Top 100)	Validation Cross-Entropy Loss	Validation Accuracy (Top 100)
0.2	7.062	0.402	7.063	0.402
0.3	6.954	0.418	6.990	0.414
0.4	6.977	0.417	7.014	0.411

6.2 Comparison of the models

After determining the hyperparameters, we can evaluate the performance of our six models by comparing their cross-entropy loss and accuracy on the test dataset. As previously mentioned, all of our models (except the benchmark model) have a two-layer architecture and utilise a multi-headed attention mechanism with two heads, a feedforward neural network with 1176 hidden nodes, a product embedding of size 294, and a dropout rate of 0.00. Lastly, it was decided to allocate 30% of the total embedding to aisle and/or department information for the KI-BaskERT models that assign a portion of their total embedding to aisle and/or department information (first and third approach of Knowledge Injection).

The outcomes of this comparison are presented in Table 4, where the performance on the test dataset is reported. It is crucial to evaluate the models on the test dataset, as this data has not been utilised during the training and validation phase and provides the most reliable indication of the models' ability to generalise to previously unseen data. Table 4 reveals a large, significant difference in accuracy between the benchmark model and the other models ($p < 0.05$). See Appendix 3 for a more detailed description of the performed T-tests. This indicates that our Transformer-based recommender systems have effectively learned valuable structures from the data and, as a result, have provided improved product recommendations. Our results thus demonstrate that the use of a Transformer-based recommender system leads to a 68.91% increase in model accuracy compared to the benchmark model.

The second and most important finding related to our research question is that we find a highly significant difference in terms of performance between the BaskERT model and the KI-BaskERT models ($p < 0.05$) except for the KI-BaskERT model that adds the product,

aisle and department embedding (last approach). Again, we refer to Appendix 3 for the T-test statistics. Almost all our attempts to incorporate additional knowledge have significantly improved the performance of the recommender system. The KI-BaskERT model that add the product and aisle embeddings to create a total embedding demonstrated the lowest cross-entropy loss and highest accuracy on the test dataset. This is consistent with previous research which suggests that adding embedding vectors is more effective than concatenating them (ENCCS, 2022). More specifically, we see an increase in accuracy of 4.01% of the KI-BaskERT model that adds the product and aisle embedding versus the BaskERT model.

Furthermore, our findings reveal that the incorporation of department information in the latter approaches resulted in a slight decrease in performance compared to the KI-BaskERT models that only inject aisle information. Adding the product, aisle and department embeddings even leads to an insignificant difference in performance ($p > 0.05$) compared to the BaskERT model. This is likely due to the excess of information being added, as the model performance decreased. Our hypothesis that incorporating department information in addition to aisle information would result in a finer subdivision of information and therefore improved model performance has not been confirmed. The models that incorporate or add solely aisle information resulted in the highest accuracy compared to the other models, indicating that in this application the recommender system does not benefit from the additional source of department information.

Table 4
Comparison of the models

Model	Test Cross-Entropy Loss	Test Accuracy (Top 100)	Accuracy Standard Deviation
Benchmark	/	0.243	0.038
BaskERT	7.01	0.409	0.043
KI-BaskERT: concatenating product and aisle embedding	6.99	0.415	0.043
KI-BaskERT: adding product and aisle embedding	6.926	0.426	0.044
KI-BaskERT: concatenating product, aisle and department embedding	7.009	0.412	0.045
KI-BaskERT: adding product, aisle and department embedding	7.016	0.411	0.043

7 Conclusion

In today's digital age, the interaction between humans and machines has become increasingly prevalent and complex. As technology continues to advance, machines will play an increasingly important role in shaping human online behaviour and decision-making. In the business context, recommender systems are becoming increasingly important as they help guide clients through the vast array of products and services offered online. These systems are expected to becoming even more crucial in the future as the number of products and services offered online continues to grow and therefore the need for personalised recommendations becomes more urgent. For this reason, it is essential that improved recommender systems are constantly being developed, ensuring to respond appropriately to this increasingly complex task. In this thesis, the aim is to contribute to this effort by conducting a study on the bidirectional KI-BaskERT model, a recommender system that leverages the encoders of the transformer network and incorporates the state-of-the-art concept of Knowledge Injection with the expectation to improve the accuracy of product recommendations in a session-based recommendation setting. After providing an introduction and a literature review on the history and development of recommender systems, we extensively described the

methodology behind the KI-BaskERT model and our different approaches to inject knowledge. In the following conclusion, we will summarise the main findings and contributions of this work, and discuss the implications of our results and potential avenues for future research.

The first important finding is that our transformer-based recommender system, BaskERT, demonstrates a substantial and statistically significant improvement over the benchmark model. Despite the benchmark model's simplicity and low likelihood of practical application, this highly significant ($p < 0.05$) increase of 68.91% in accuracy illustrates the capability of transformer-based recommender systems in learning valuable structures regarding basket composition, thereby allowing for more informed and calculated product recommendations. This finding aligns with the results of previous studies, such as Sun et al. (2019) and Bianchi et al. (2020), which also concluded that the development of a transformer-based recommender system leads to an increase in accurate predictions and customer satisfaction.

Although the accuracy of the BaskERT model may seem relatively low at first glance, it is important to note that the recommendations made by these transformer-based recommender systems are still valuable for customers. Although the masked product only appears among the top hundred predicted products 41% of the time in the test dataset, this does not necessarily indicate that the model is not providing beneficial product recommendations for clients. As illustrated in Appendix 4, even when the model's prediction is deemed inaccurate, the top hundred recommendations still comprise products that are closely related to the masked product or alternative products that could easily be selected by the customer. An additional argument for the value of the product recommendations provided by the BaskERT model is that the reported accuracy does not account for the likelihood of customers purchasing multiple products from the recommended top hundred product list. This added value is not captured by our simple accuracy measure. Despite the relatively low accuracy, these models proved that they are still able to learn valuable structures in the data and make valuable recommendations that are useful for clients and companies.

One potential explanation for the relatively low accuracy of the model is the sheer size of the product range. The model must recommend a mere fraction of 100 products from a total of $11,426$, based solely on a limited number of products in a given basket. Furthermore,

there are specific products that are purchased only 300 times across the entire datasets, resulting in a scarcity of co-occurrences for the model to learn during the training phase. As a result, the model may struggle to provide well-informed recommendations about these products as they are not purchased very often.

Another factor that may have contributed to the relatively low accuracy of the model is the large number of baskets containing a small number of products. For these baskets, the recommender system may have had insufficient information, making it more challenging to produce accurate predictions. This is especially true if the masked product is unrelated to the other products in the basket, as it can lead the model to form an incorrect understanding of the context of the basket and therefore makes totally different recommendations. This fact is also substantiated by Figure 1, which demonstrates that 25% of all baskets consist of only five products. If one product of these is already masked, the recommendation system has only four products to base its prediction on with a total product assortment of $11,426$ products to choose from.

The second important finding addresses the final sub-question of our research question, which assesses whether knowledge injection leads to increased accuracy in predicting the next product in the basket compared to the BaskERT model. The majority of our approaches to inject additional knowledge into the BaskERT model were successful in achieving a significant improvement in model performance ($p < 0.05$). Only the latter approach to inject knowledge by adding the product, aisle and department embedding did not significantly outperform the BaskERT model ($p > 0.05$). The KI-BaskERT model that adds the product and aisle embedding shows the highest accuracy and shows a 4.01% increase in accuracy compared to the BaskERT model. This is a substantial increase, especially if one deploys this recommender system on a large scale. Given the large number of customers being served by the company Instacart, this increase in performance is extremely valuable, as it has the potential to result in a corresponding increase in customer satisfaction and sales of the company.

We argue that the improvement in the model performance of the KI-BaskERT model compared to the BaskERT model can be attributed to the enhanced representation of products in the high-dimensional vector space of the product embeddings through the injection of

aisle information. The embeddings of the products resulting from training the KI-BaskERT model become richer and more diverse, which can capture more complex relationships between products. This is especially true for less frequently purchased products. When the co-occurrence data is sparse, the aisle information provides additional information beyond the co-occurrence data, which can help the model understand these less frequently purchased products better and make more informed predictions about these products.

It is also worth noting that adding additional sources of information, in this case department information, may not always lead to improved model performance. Our results show that models that incorporate department information perform worse than the KI-BaskERT models that only incorporate aisle information. This suggests that not all sources of information contribute equally to significantly improved product recommendations. This finding highlights the importance of selecting the most appropriate source of information to inject into the recommender system. Further research into knowledge injection in transformer-based recommender systems could explore the potential of other sources of information, such as product ratings, prices, nutritional value, or ingredients, for example.

As another potential direction for future research, an extension of this study could involve exploring the impact of product assortment size on the model's performance. The current study utilised a product assortment of 11,426 products and found that baskets were primarily composed of fresh vegetables and fruit. As a result, it is possible that the model struggles to accurately predict infrequently purchased products. To address this issue, one option could be to gather additional basket data that includes a greater number of baskets containing these less frequently bought products in order to achieve a more balanced distribution of products in the basket data.

In conclusion, our study has achieved its objective of constructing and implementing a transformer-based recommender system, drawing on the theories and insights of preceding works in the field. By injecting aisle information into the BaskERT model, we have significantly improved the performance of the KI-BaskERT model, with a substantial increase of 4.01% in accuracy as compared to the BaskERT model. This demonstrates the potential injecting knowledge in recommender systems, but also knowledge injection in general in neural network-based models. With the continuous advancement of technology and the discovery of

more sophisticated recommender systems and methods of injecting knowledge, we anticipate that they will continue to improve, becoming even faster, more effective, and applicable to an increasing range of use cases. These developments hold immense promise for the future and will undoubtedly benefit the field of recommendation systems in many exciting ways.

8 References

Apáthy, S. (2022). *History of recommender systems: overview of information filtering solutions.* Onespire Ltd. - SAP and Management Consulting. https://www.onespire.net/news/history-of-recommender-systems/

Bahdanau, D., Cho, K., & Bengio, Y. (2014). *Neural machine translation by jointly learning to align and translate.* arXiv preprint arXiv:1409.0473.

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

De Boer, P. T., Kroese, D. P., Mannor, S., & Rubinstein, R. Y. (2005). *A tutorial on the cross-entropy method.* Annals of operations research, 134(1), 19-67.

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

Donkers, T., Loepp, B., & Ziegler, J. (2017). *Sequential user-based recurrent neural network recommendations.* In Proceedings of the eleventh ACM conference on recommender systems (pp. 152-160).

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). *Learning phrase representations using RNN encoder-decoder for statistical machine translation.* arXiv preprint arXiv:1406.1078.

Goldberg, D., Nichols, D., Oki, B. M., & Terry, D. (1992). *Using collaborative filtering to weave an information tapestry.* Communications of the ACM, 35(12), 61-70.

Grbovic, M., Radosavljevic, V., Djuric, N., Bhamidipati, N., Savla, J., Bhagwan, V.,

& Sharp, D. (2015). *E-commerce in your inbox: Product recommendations at scale.* In Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining (pp. 1809-1818).

He, K., Zhang, X., Ren, S., & Sun, J. (2016). *Deep residual learning for image recognition.* In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).

Hidasi, B., Karatzoglou, A., Baltrunas, L., & Tikk, D. (2015). *Session-based recommendations with recurrent neural networks.* arXiv preprint arXiv:1511.06939.

Hidasi, B., Quadrana, M., Karatzoglou, A., & Tikk, D. (2016). *Parallel recurrent neural network architectures for feature-rich session-based recommendations.* In Proceedings of the 10th ACM conference on recommender systems (pp. 241-248).

Hinton, Geoffrey, Deng, Li, Yu, Dong, Dahl, George E, Mohamed, Abdel-rahman, Jaitly, Navdeep, Senior, Andrew, Vanhoucke, Vincent, Nguyen, Patrick, Sainath, Tara N, et al. *Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups.* Signal Processing Magazine, IEEE, 29(6):82–97, 2012.

Instacart Market Basket Analysis | Kaggle. (2017, May). Retrieved May 2, 2022, from https://www.kaggle.com/c/instacart-market-basket-analysis

Isinkaye, F., Folajimi, Y., & Ojokoh, B. (2015). *Recommendation systems: Principles, methods and evaluation.* Egyptian Informatics Journal, 16(3), 261–273. https://doi.org/10.1016/j.eij.2015.06.005

Jamsheer, K., (2019, August 27). *Impact of e-Commerce On Society: Advantages and Disadvantages.* Woocommerce Product Addons. https://acowebs.com/impact-ecommerce-society/

Jannach, D., Pu, P., Ricci, F., & Zanker, M. (2021). *Recommender systems: Past, present, future.* Ai Magazine, 42(3), 3-6.

Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M., & Tang, P. T. P. (2016). *On large-batch training for deep learning: Generalization gap and sharp minima.* arXiv preprint arXiv:1609.04836.

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

Knight, S., & Mann, C. *Electronic Commerce.* Oxford Research Encyclopedia of International Studies. Retrieved 3 May. 2022, from https://oxfordre.com/internationalstudies/view/10.1093/acrefore/9780190846626.001.0001/acrefore-9780190846626-e-85.

Lipton, Z. C., Berkowitz, J., & Elkan, C. (2015). *A critical review of recurrent neural networks for sequence learning.* arXiv preprint arXiv:1506.00019.

Liu, W., Zhou, P., Zhao, Z., Wang, Z., Ju, Q., Deng, H., & Wang, P. (2020). *K-bert: Enabling language representation with knowledge graph.* In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 34, No. 03, pp. 2901-2908).

OECD (1999-01-01), *Economic and Social Impact of E- commerce: Preliminary Findings and Research Agenda*, OECD Digital Economy Papers, No. 40, OECD Publishing, Paris. http://dx.doi.org/10.1787/236588526334

Pasquali, M. (2022, November 17). *E-commerce worldwide - statistics & facts.* Statista. https://www.statista.com/topics/871/online-shopping/

Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu.

Exploring the limits of language modeling. arXiv preprint arXiv:1602.02410, 2016.

Ruder, S. (2016). *An overview of gradient descent optimization algorithms.* arXiv preprint arXiv:1609.04747.

Ruslan Salakhutdinov and Andriy Mnih. 2007. *Probabilistic Matrix Factorization.* In Proceedings of NIPS. Curran Associates Inc., USA, 1257–1264.

Russakovsky, Olga, Deng, Jia, Su, Hao, Krause, Jonathan, Satheesh, Sanjeev, Ma, Sean, Huang, Zhiheng, Karpathy, Andrej, Khosla, Aditya, Bernstein, Michael S., Berg, Alexander C., and Li, Fei-Fei. *Imagenet large scale visual recognition challenge.* CoRR, abs/1409.0575, 2014. URL http://arxiv.org/abs/1409.0575.

Schafer, J. B., Konstan, J. A., & Riedl, J. (2001). *E-Commerce Recommendation Applications. Data Mining and Knowledge Discovery*, 5(1/2), 115–153. https://doi.org/10.1023/a:1009804230409

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

EuroCC National Competence Centre Sweden [ENCCS]. (2022). *Summing or concatenating embeddings? — Graph Neural Networks and Transformers documentation.* https://enccs.github.io/gnn_transformers/notebooks/session_1/1b_vector_sums_vs_concatenation/

Sun, F., Liu, J., Wu, J., Pei, C., Lin, X., Ou, W., Jiang, P.: *BERT4Rec: sequential recommendation with bidirectional encoder representations from transformer.* In: Proceedings of the 28th ACM International Conference on Information and Knowledge Management - CIKM 2019. ACM Press (2019). https://doi.org/10. 1145/3357384.3357895

Sciforce. (2022, February 2). *Deep Learning Based Recommender Systems* - Sciforce. Medium. Retrieved August 17, 2022, from https://medium.com/sciforce/deep-learning-based-recommender-systems-b61a5ddd5456

Taylor, W. L. (1953). "Cloze procedure": *A new tool for measuring readability.* Journalism quarterly, 30(4), 415-433.

Tuan, T. X., & Phuong, T. M. (2017, August). *3D convolutional networks for session-based recommendation with content features.* In Proceedings of the eleventh ACM conference on recommender systems (pp. 138-146).

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). *Attention is all you need.* Advances in neural information processing systems, 30.

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., ... & Dean, J. (2016). *Google's neural machine translation system: Bridging the gap between human and machine translation.* arXiv preprint arXiv:1609.08144.

Yehuda Koren and Robert Bell. 2011. *Advances in Collaborative Filtering.* In Recommender Systems Handbook. Springer US, Boston, MA, 145–186.

9 Appendix

9.1 Appendix 1

10799	9205	6365	2299	4911	5198	8036	9315	4055	5752	7490	9534	-1	-1	-1	-1	-1	-1
7015	3516	10073	8744	8615	7923	6169	8779	6760	-1	-1	-1	-1	-1	-1	-1	-1	-1
9275	2064	10647	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
6509	5635	3402	3913	8765	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
8220	7601	10057	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
3870	9084	9979	10614	6700	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
3259	5130	4051	1399	9735	2795	8545	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
10061	4141	7293	2145	46	3964	4888	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
3035	226	8203	1112	9381	2667	8756	9979	9158	487	3065	2764	1036	2300	-1	-1	-1	-1
4105	4725	1335	8562	6111	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
3038	10888	6526	7751	9171	5499	9162	6263	9858	1118	9113	3035	6764	5884	-1	-1	-1	-1
6212	9775	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
4105	4311	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
10260	6545	4925	1093	6121	2654	6651	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
3035	8234	7694	3618	6107	3040	9689	5696	4025	1181	9011	7122	10936	3402	2292	9976	6380	-1
10108	3986	3431	7041	42	8348	6915	9809	11155	10550	6451	1973	1261	2054	5943	9697	-1	-1
5833	9417	11014	3198	5028	10013	7287	3901	10594	11319	-1	-1	-1	-1	-1	-1	-1	-1
10991	4266	6866	2536	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
99	6036	5038	5561	1036	1273	5410	2957	4105	7206	7231	291	-1	-1	-1	-1	-1	-1
6410	3776	3637	8564	11426	5697	2631	10882	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
2857	10592	4889	875	3035	8308	8473	9704	11284	6785	-1	-1	-1	-1	-1	-1	-1	-1
7582	7422	548	4782	5697	3642	10458	4929	11024	6369	-1	-1	-1	-1	-1	-1	-1	-1
337	3012	5338	3035	1149	3494	379	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
9114	4782	3911	1835	5283	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
10753	3870	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
10757	9295	10583	3340	8397	10178	5907	3441	6380	2462	1957	1107	1919	256	7400	10175	1081	6760
4117	9362	3035	9150	6206	10170	2323	1077	1717	9029	6686	6594	4853	4105	1874	5268	8653	10516
4542	3870	3035	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1

Figure 10: Representation of basket data (each row represents a basket from which a product is masked)

9.2 Appendix 2

original_product_id	product_name	aisle_id	aisle_name	department_id	department
1	Chocolate Sandwich Cookies	60	cookies cakes	19	snacks
4	Smart Ones Classic Favorites Mini Rigatoni With Vodka Cream Sauce	37	frozen meals	1	frozen
10	Sparkling Orange Juice & Prickly Pear Beverage	114	water seltzer sparkling water	7	beverages
23	Organic Turkey Burgers	48	packaged poultry	12	meat seafood
25	Salted Caramel Lean Protein & Fiber Bar	2	energy granola bars	19	snacks
26	Fancy Feast Trout Feast Flaked Wet Cat Food	40	cat food care	8	pets
28	Wheat Chex Cereal	120	cereal	14	breakfast
32	Nacho Cheese White Bean Chips	106	chips pretzels	19	snacks
34	Peanut Butter Cereal	120	cereal	14	breakfast
35	Italian Herb Porcini Mushrooms Chicken Sausage	105	hot dogs bacon sausage	12	meat seafood
37	Noodle Soup Mix With Chicken Broth	68	soup broth bouillon	15	canned goods
44	Sparkling Raspberry Seltzer	114	water seltzer sparkling water	7	beverages
45	European Cucumber	82	fresh vegetables	4	produce
47	Onion Flavor Organic Roasted Seaweed Snack	65	asian foods	6	international
49	Vegetarian Grain Meat Sausages Italian - 4 CT	13	tofu meat alternatives	20	deli
54	24/7 Performance Cat Litter	40	cat food care	8	pets
63	Banana & Sweet Potato Organic Teething Wafers	91	baby food formula	18	babies
79	Wild Albacore Tuna No Salt Added	94	canned meat seafood	15	canned goods
83	100% Whole Wheat Pita Bread	127	tortillas flat bread	3	bakery
87	Classics Earl Grey Tea	93	tea	7	beverages

Figure 11: Representation of data used for Knowledge Injection

9.3 Appendix 3: T-Tests

The Welch t-test was performed to compare the accuracy of the different models. The null hypothesis was that there was no difference in accuracy between the reference model and the compared model in Table 5. The alternative hypothesis was that compared model had a higher accuracy than the reference model. The t-statistic was calculated using the formula $t = \frac{\bar{x}_1 - \bar{x}_2}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}}$. Where $\bar{x}_1$,s_1^2 and $\bar{x}_2$,s_1^2 are the accuracies and the standard deviations of the reference model and compared model respectively. These values can all be retrieved from Table 4. The number of observations (n) is equal to the number of baskets in the test dataset $(301,465)$.

Table 5

Results of Welch's t-test Comparing Accuracies. Note: (*) indicate the level of statistical significance, where $p < 0.01$ (***), $p < 0.05$ (**), and $p < 0.10$ (*)

Reference Model	Compared Model	T-value	Degrees of Freedom	p-value	Significance
Benchmark	BaskERT	140.857	4618.893	0.000	***
Benchmark	KI-BaskERT (concatenating product and aisle embedding)	144.984	4615.120	0.000	***
Benchmark	KI-BaskERT (adding product and aisle embedding)	153.557	4601.324	0.000	***
Benchmark	KI-BaskERT (concatenating product, aisle and department embedding)	140.401	4569.118	0.000	***
Benchmark	KI-BaskERT (adding concatenating product, aisle and department embedding)	143.683	4640.529	0.000	***
BaskERT	KI-BaskERT (concatenating product and aisle embedding)	4.091	4709.958	0.000	***
BaskERT	KI-BaskERT (adding product and aisle embedding)	12.924	4709.128	0.000	***
BaskERT	KI-BaskERT (concatenating product, aisle and department embedding)	2.303	4703.891	0.011	**
BaskERT	KI-BaskERT (adding concatenating product, aisle and department embedding)	1.264	4708.405	0.103	

9.4 Appendix 4: Demonstration of the KI-BaskERT recommender system

The functioning of the KI-BaskERT (addition of product and aisle embedding) is shown here. Although the masked product was not among the top hundred recommended products, we demonstrate here that the recommender system still provides added value, as we see that many highly related products are recommended that could equally lead to purchase. Note that the full top hundred recommended products is not shown here.

Original basket:

- Whipped Cream Cheese
- 100% Whole Wheat Bread
- Organic Whole Wheat Fusilli
- Cheese Puffs Original
- Organic Reduced Fat Milk
- *MASK*
- Cheddar Bunnies Snack Crackers
- White Corn
- Organic Yokids Lemonade/Blueberry Variety Pack Yogurt Squeezers Tubes

Masked product:

- Organic Mixed Berry Yogurt & Fruit Snack

Is masked product among recommendations:

- False

The recommendations (ranked):

- Bananas
- Organic Strawberry Grassfed Whole Milk Yogurt
- Organic Cashew Nondairy Vanilla Yogurt
- Grassfed Whole Milk Strawberry Yogurt

- Organic Mango Yogurt
- Organic Cashew Nondairy Blueberry Yogurt
- Half & Half
- Organic Nondairy Lemon Cashew Yogurt
- 100% Raw Coconut Water
- Bag of Organic Bananas
- Grassfed Whole Milk Blueberry Yogurt
- Organic Whole Milk Washington Black Cherry Yogurt
- Organic Coconut Yogurt
- Coconut Almond Unsweetened Creamer Blend
- Dairy Free Coconut Milk Chocolate Yogurt Alternative
- Dairy Free Coconut Milk Vanilla Yogurt Alternative
- Almond Milk Peach Yogurt
- Organic Strawberries
- Unsweetened Whole Milk Blueberry Greek Yogurt
- Whole Milk Yogurt Organic Indonesian Vanilla Bean
- Organic Banana
- Organic Whole Milk
- Cream Top Smooth & Creamy Maple Yogurt
- Almond Milk Blueberry Yogurt
- Coconut Yogurt
- Organic Lowfat Mango Kefir
- French Vanilla Yogurt Vanilla
- Organic Plain Unsweetened Nondairy Cashew Yogurt