Vector Embedding Solution for Recommendation System

Vidya P V
College of Engineering Trivandrum
d-tve23jul018@cet.ac.in

Ajeesh Ramanujan
College of Engineering Trivandrum
ajeesh@cet.ac.in

Abstract

We propose a vector embedding approach for recommendation systems aimed at identifying product affinities and suggesting complementary items. By capturing relationships between products, the model delivers highly relevant recommendations based on the context. A neural network is trained on purchase data to generate word embeddings, represented as a weight matrix. The resulting model predicts complementary products with top-20 and top-50 precision scores of 0.59251 and 0.29556, respectively. These embeddings effectively identify products likely to be co-purchased, enhancing the relevance and accuracy of the recommendations.

1 Introduction

Words can be represented as real-valued vectors in lower dimensional space, that encodes its meaning based on the context. The words that appear closer in the vector space are considered to have similar meaning. Word2Vec [8] and Glove [5] are two most important word embeddings that captures the semantic similarity of words.

The idea of word embeddings can be applied to commercial products as well, where the products are encoded in vector space. This aids the product recommendation system to identify and predict related products based on the purchase trends by customers. Product embeddings, which captures the affinities between products, is very useful when applied to to information retrieval and recommender systems, as it enable customers to find and discover highly relevant products based on context, as the embeddings capture the semantic relation between the products, by learning their co-purchase pattern. Given a list of products bought together in a single session, the products would best complement the products being considered, can be easily deduced when embedding is applied.

2 Related Works

[2] uses the bidirectional transformer architecture, BERT, to create contextual embeddings for products, called *Prod2BERT*, by capturing the online shopping behaviour of the customer. It uses the technique of masked session modelling, where each shopping session is considered as a "sentence" and the products which belong to each session as words of that sentence. The *Prod2BERT* model, thus developed can be used to predict related products and also provides context-specific embeddings for products.

The idea of dual embedding is implemented in [3] for representing products which can be created from co-purchased data. This helps in recommending complementary products that can used in recommendations systems. The recommended model uses both input and output weight matrices for computing the similarity, which results in identifying the related words. The model also uses synthetic samples to augment the model so that it can handle the sparsity of purchase data.

Alternatives to a given product can be identified by extracting relationship between products based on their frequency of occurrence and co-purchase history. [1] applies Word2Vec and FastText models to obtain embeddings to help product recommendation. [7] uses Word2Vec model to identify similarity between items purchased in a session and analyse the number of common recommendations of selected items, to obtain the products that can be purchased together.

A model to recommend products to the customer on an e-commerce platform, based on the customer profile is implemented in [6]. It uses the purchase history and product reviews, along with the co-purchase patterns to create profiles for users and products. The interaction between products and users are identified using word embedding techniques.

[1] and [7] uses context-independent traditional embedding techniques such as Word2Vec and FastText and [1] uses the tedious process of analysing product reviews and creating user and product profiles. [3] uses the idea of dual embedding, but with lower precision values. Our model proposes multiple embedding representations when compared to the models discussed above, providing flexibility, with greater precision values when compared to an existing model using the same dataset. The model was able to achieve comparable results with minimal dataset, smaller embedding dimensions and

within the computational limitations.

3 Methodology

3.1 Dataset

The dataset used to implement this approach is Instacart dataset [4], a relational set of files describing customers' orders over time. Instacart is an American company that operates a grocery delivery and pick-up service in the United States and Canada. The dataset is anonymized and contains a sample of over 3 million grocery orders from more than 200,000 Instacart users. For each user, the dataset includes between 4 and 100 of their orders, with the sequence of products purchased in each order.

There are 6 correlated csv files which can be easily merged using the primary keys - *aisles.csv, departments.csv, order_products__prior.csv, orders.csv, products.csv and sample_submission.csv*. Among these, *orders.csv*, *products.csv* and *order_products__prior.csv* are used to perform the market-basket analysis in this work. *order_products__prior.csv* specify which products were purchased in each order. It contains previous order contents for all customers. *orders.csv* contains the details of each order. *products.csv* contains the details of products. The dataset contains 3421083 orders and 49688 products.

3.2 Methodology

From the dataset 1500 order details are used to create the word embeddings. It constitutes 155 orders and 1148 unique products. Each product is represented as a one-hot vector of dimension 1148, which is the size of vocabulary. Instead of deciding a fixed size context window to identify the contextual affinity between two products, a product within an order is assumed to be related to all the other products in the same order. These word pairs are used to create the training data. There are 19124 product pairs that constitutes the training data. In the training data, the first product in each pair is added to the input matrix **X** and the second product in the pair is added to the target **Y**. This training data is used to build and train the word embedding network.

The input to the network is a sparse vector containing one-hot encoded vectors corresponding to the products, applying the weight matrix of the first layer of the feed forward network to these vectors transforms these one-hot encoded vectors to dense vectors in lower dimension. These dense vectors when multiplied with the weight matrix of second layer gives another set of dense vectors. Hence each row of the weight matrix belonging to the first layer encode meaningful semantic information of the products from the training data, making it a product embedding. Similarly, each column of the weight matrix belonging to the second layer can be considered as another embedding. The last layer of the network is a softmax layer, where the output of the second layer is passed as input and is converted into probability vectors whose elements sum up to one. This final output can be considered as recommendations, i.e. the products that are likely to be bought along with the given product.

The weights are initialized randomly by drawing samples from a uniform distribution, over the half-open interval [-1, 1). Each product will be represented as vectors in ten-dimensional space, i.e., the model creates a ten-dimensional embedding. Batch gradient descent algorithm is used for back-propagation and iterated for 50 epochs.

The cross entropy error is defined as follows:

$$H(Y, Z) = -\sum_{x \in X} Y(x) log Z(x)$$

where each $Y(x)$ is the one-hot encoded target vector corresponding to the sample x and $Z(x)$ is the predicted probability corresponding to the sample x. The closer the value of the prediction is to 1, the smaller the cross entropy, and vice versa. The error is calculated while training and the weights are updated accordingly. At times, due to the random initialization of weights for both input and output weight matrices, the gradient value might explode, resulting in stopping the further gradient updates. To prevent the gradient from exploding, gradient clipping can applied on both weights, with a threshold value of 70. After completing the iterations, the weight matrices thus obtained can be considered as the embeddings corresponding to the products.

Given the input matrix X, with dimension 19124×1148, the one-hot encoded representations of products, the output of the first layer can be calculated as:

$$A_1 = X \cdot W_1$$

where W_1 is the weight matrix in the first layer, with dimension 1148×10. The resultant matrix A_1 has a dimension of 19124×10. The output from the first layer is given as input to the second layer, and using the weight matrix W_2 of this layer, with dimension 10×1148, the output is calculated as:

$$A_2 = A_1 \cdot W_2$$

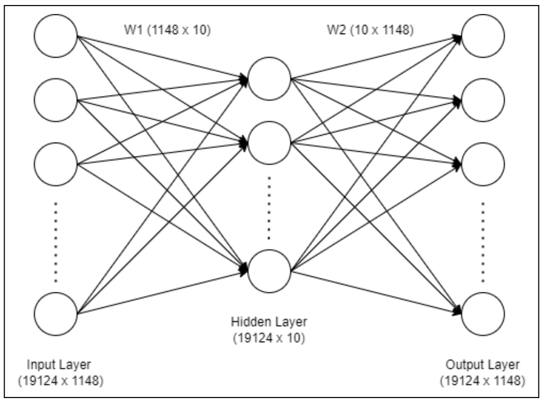
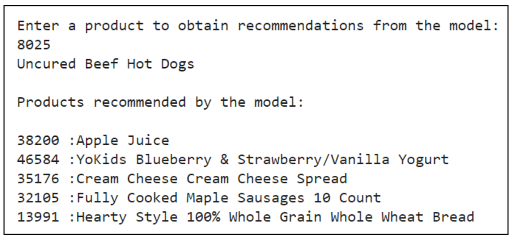


Figure 1: Network Diagram



```
Enter a product to obtain recommendations from the model:
8025
Uncured Beef Hot Dogs

Products recommended by the model:

38200 :Apple Juice
46584 :YoKids Blueberry & Strawberry/Vanilla Yogurt
35176 :Cream Cheese Cream Cheese Spread
32105 :Fully Cooked Maple Sausages 10 Count
13991 :Hearty Style 100% Whole Grain Whole Wheat Bread
```

Figure 2: Model Recommendation

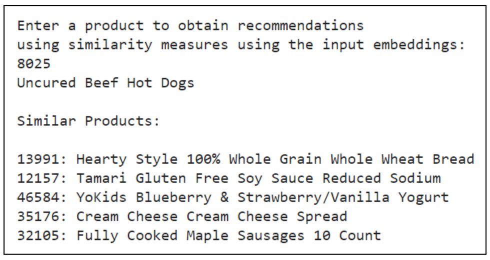
The softmax function is applied to each row of the output of the second layer to obtain the prediction probability corresponding to each row of input matrix **X**.

$$Z = Softmax(A_2)$$

The diagram of the network is shown in Figure 1.

3.3 Results

The model created recommends complementary products for a given product. For all the unique products in the purchase list, average precision is calculated by predicting top 5, top 10, top 20 and top 50 products. The values are listed in Table 1. The results obtained are compared to the precision values at top 20 and top 50 for the dual embedding method implemented in [3] and shows that the model implemented in this paper outperforms the dual embedding method.

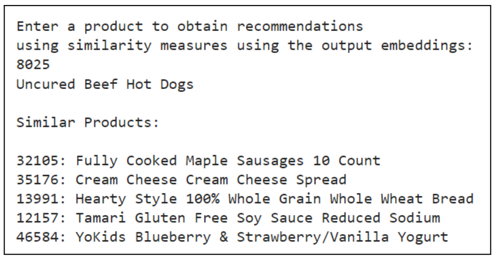


```
Enter a product to obtain recommendations
using similarity measures using the input embeddings:
8025
Uncured Beef Hot Dogs

Similar Products:

13991: Hearty Style 100% Whole Grain Whole Wheat Bread
12157: Tamari Gluten Free Soy Sauce Reduced Sodium
46584: YoKids Blueberry & Strawberry/Vanilla Yogurt
35176: Cream Cheese Cream Cheese Spread
32105: Fully Cooked Maple Sausages 10 Count
```

Figure 3: Similarity using input embedding



```
Enter a product to obtain recommendations
using similarity measures using the output embeddings:
8025
Uncured Beef Hot Dogs

Similar Products:

32105: Fully Cooked Maple Sausages 10 Count
35176: Cream Cheese Cream Cheese Spread
13991: Hearty Style 100% Whole Grain Whole Wheat Bread
12157: Tamari Gluten Free Soy Sauce Reduced Sodium
46584: YoKids Blueberry & Strawberry/Vanilla Yogurt
```

Figure 4: Similarity using output embedding

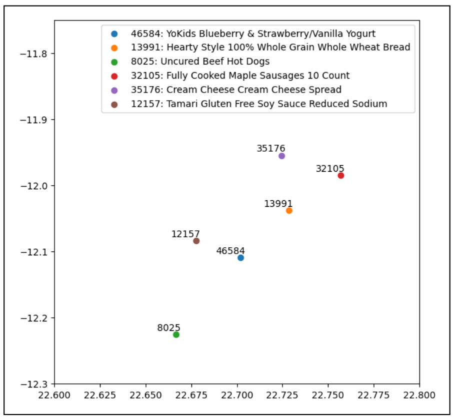


Figure 5: Sample Representation of Input Embedding

Cosine similarity is used to find out the products that are co-purchased more often with a product. Both the input weight embedding and output weight embedding acts as the vector representations of the products to calculate the similarity. Comparing the top 5 predictions from the feed forward neural network and the products found closer in the vector space using similarity measures, it was found the results overlapped. This indicates the vector embeddings created for the products learn the co-purchase pattern in the order data. Figure 2, 3 and 4 shows sample recommendation by the model, similarity using the input and output embedding respectively. The products are listed along with their product ids given in the dataset.

The embeddings from both weight matrices are used to visualize the vectors that are similar, by plotting these vectors in a vector space. T-distributed stochastic neighbor embedding (t-SNE) is used to visualize these high-dimensional vectors. Figure 5 and 6 shows the sample representations of input and output embeddings. Complementary products are identified using cosine similarity. It is observed that the products identified as similar to the given products are same when both embeddings are used.

Another set of embedding is created by multiplying the embeddings obtained from the weight

Table 1: Precision@K Values of the Model

Model	Precision@5	Precision@10	Precision@20	Precision@50
Without Gradient Clipping	0.83624	0.78798	0.60675	0.29758
With Gradient Clipping	0.77073	0.74085	0.59251	0.29556
Dual Embedding Model [3]			0.0437	0.0322

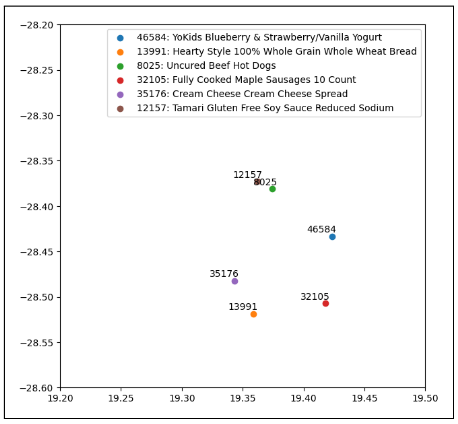
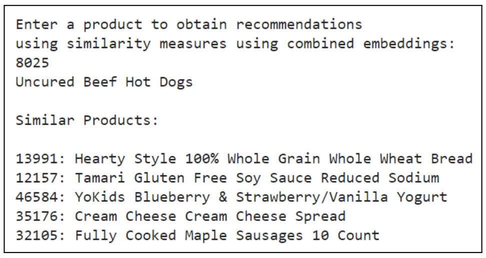


Figure 6: Sample Representation of Output Embedding



```
Enter a product to obtain recommendations
using similarity measures using combined embeddings:
8025
Uncured Beef Hot Dogs

Similar Products:

13991: Hearty Style 100% Whole Grain Whole Wheat Bread
12157: Tamari Gluten Free Soy Sauce Reduced Sodium
46584: YoKids Blueberry & Strawberry/Vanilla Yogurt
35176: Cream Cheese Cream Cheese Spread
32105: Fully Cooked Maple Sausages 10 Count
```

Figure 7: Similarity using combined embedding

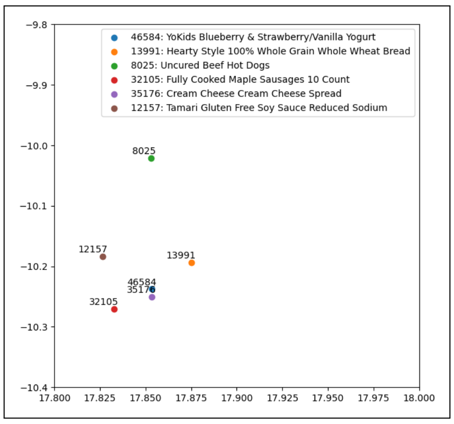


Figure 8: Sample Representation of Combined Embedding

matrices of first and second layers of the network. The resultant embedding equally captures the contextual similarity as the other two embeddings. The results of similarity check using this embedding overlapped with the results obtained previously. Figure 7 shows the similarity using combined embedding and 8 shows the sample representation of combined embedding.

4 Conclusion

This paper addressed the challenge of representing the commercial products as vectors in a low dimensional space, with the primary objective of improving the recommendations in an e-commerce platform. The feed forward network designed gives a top 20 and top 50 precision of 0.59251 and 0.29556 when compared to the existing dual embedding model [3] with precision values 0.0437 and 0.0322 respectively. The reason behind smaller value of precision for dual embedding may be due to the size of the training data and also because of the synthetic data generated using augmentation. The findings of this paper provide a practical solution for identifying the complementary products which aids the recommendation system. The product embeddings contributes to the understanding of product semantics, relationships, and user interactions by converting the information about products from each purchase session into a vector space. This enables more effective search, recommendation, analysis, and enrichment of product-related information.

References

[1] Ahmet Tugrul Bayrak. 2022. A session-based recommendation approach with word embeddings. *2022 International Conference on INnovations in Intelligent SysTems and Applications (INISTA)*.

[2] Jacopo Tagliabue Federico Bianchi, Bingqing Yu. 2021. Bert goes shopping: Comparing distributional models for product representations. *Proceedings of the 4th Workshop on e-Commerce and NLP (ECNLP 4), Association for Computational Linguistics*, pages 1–12.

[3] Nishan Subedi Mohammad Hajiaghayi Giorgi Kvernadze, Putu Ayu G. Sudyanti. 2022. Two is better than one: Dual embeddings for complementary product recommendations. *2022 IEEE International Conference on Knowledge Graph (ICKG)*.

[4] Instacart. 2021. Instacart online grocery basket analysis dataset.

[5] Christopher D. Manning Jeffrey Pennington, Richard Socher. 2014. Glove: Global vectors for word representation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

[6] Abdel-Fattah Hegazy Omar Attia, Ahmed Dahroug. 2023. A proposed model for enhancing product recommendation based on word embedding. *2023 International Conference on Computer and Applications (ICCA)*.

[7] Hassana Abdullahi Ramazan Esmeli, Mohamed Bader-El-Den. 2020. Using word2vec recommendation for improved purchase prediction. *2020 International Joint Conference on Neural Networks (IJCNN)*.

[8] Greg Corrado Jeffrey Dean Tomas Mikolov, Kai Chen. 2013. Efficient estimation of word representations in vector space. *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*.

Vector Embedding Solution for Recommendation System

Vidya P V, Ajeesh Ramanujan

Responses to the Questions by Reviewer 1

1. Could you elaborate on why you chose to use only 1,500 orders out of the 3.4M available in the Instacart dataset? How might this smaller sample size have influenced your results?

   Due to computational constraints, including limited processing power and memory, we were only able to use a subset of the dataset for training and model development. However, despite the smaller size, the model's performance on the subset was consistent when compared to another model that used the same dataset, and it was able to recognize the complementary products. We were able to identify the clusters using visualizations. While a larger dataset would provide more comprehensive insights, the results from this subset seem reliable given the constraints, and future work with more computational resources could expand the model's evaluation

2. Could you explain the process you followed to select the key model parameters, such as 10-dimensional embeddings, 50 training epochs, and the gradient clipping threshold of 70?

   The size of the embedding was selected at random without exhaustive tuning, the model still performed well, likely due to the fact that the embedding was able to capture the product information and it was able to recognize the complementary products. We were able to identify the clusters using visualizations, suggesting that the choice was reasonable. The number of epochs was initially set to 50 based on typical training practices for similar models and tasks. The choice of 50 epochs also balanced training time and resource constraints, allowing for good performance without overfitting. We will experiment with different epoch counts to further optimize the training. The model was trained with different gradients clipping values in range 50 to 150. When gradient clipping value was set to values larger than 70, the gradients were not be clipped sufficiently and still caused instability in training, such as the weights becoming NaN, making training ineffective. With gradient clipping values smaller than 70, the performance of the model was very low due to loss of information as gradients are overly scaled down, resulting in underfitting. We will conduct more targeted hyperparameter optimization to ensure the best possible settings.

3. How does your model address new products that were not present in the training data?

   The cold-start problem will be considered as a research problem, and will be addressed in future works

4. Could you provide an analysis of the failure cases encountered during your study?

   A very few failure cases were found, with No Sugar Added Variety Fruit Bars being predicted along with

Classic Scent Liquid Dish Soap as an example. The intuition behind this failure cases of embedding being that since the products in the same order are considered as related, there may be rare samples in the dataset where unrelated items are purchased in the same order. This can happen when the order size is too large. Such failure cases can be considered as a research problem which will be addressed in future works.

5. Would you mind sharing a comparison of your model's performance with other commonly used baseline recommendation approaches?
   The model currently uses Instacart dataset, and limited work is available using the same. We will perform the comparison of the model's performance with commonly used approaches, considering different datasets while extending this work, in future, so as to improve the generalization of the model.

Responses to the Questions by Reviewer 2

1. In the related works section, please mention the research gaps and how your proposed system differ from the systems found in the literature
   The research gaps and the advantages of the proposed system over the models reviewed in literature has been mentioned in the paper in page 1, last paragraph.

2. Please provide citation for the Instacart dataset
   Citation added for Instacart dataset in the paper in page 2, paragraph 1 and added as 4th reference in the Reference section

3. How representative was the 1500 order details to create word embeddings?
   For the purpose of model development, we selected a portion of the dataset consisting of the first 1500 rows, without considering potential biases. Moving forward, we plan to evaluate the model on a broader, more random sample of the data to ensure the model generalizes well across the full dataset.

4. How the weight matrix of the first layer of the feed forward network defined?
   Random initialization was performed for the weight matrix of the first layer of the feed forward network. The number of rows of the weight matrix corresponds to the unique products under consideration and columns correspond to the size of the embedding (10 in our case). After completing the model training, each row of the weight matrix will give the word embedding corresponding to each product

5. How you identified the threshold value for gradient clipping?
   The model was trained with different gradients clipping values in range 50 to 150. When gradient clipping value was set to values larger than 70, the gradients were not be clipped sufficiently and still caused instability in training, such as the weights becoming NaN, making training ineffective. With gradient clipping values smaller than 70, the performance of the model was very low due to loss of information as gradients are overly scaled down, resulting in underfitting