DATA 607 Final Report:

Sequence Sensibility: LSTM for Dynamic Product Recommendations

University of Calgary

Faculty of Science

DATA 607: Statistical and Machine Learning

Instructor: Placida Dassanayake

April 10, 2023

Table of Contents

Introduction

Predictive modeling plays a pivotal role, aiding decision-making processes by forecasting future trends based on historical data. For example, in the context of business operations, predicting a customer's future basket items, can emerge as a powerful tool that benefits both consumers and businesses. In recent years, Long Short-Term Memory (LSTM) networks have been leveraged as a way to retain long-term dependencies while selectively forgetting irrelevant information. For this project, we hope to leverage LSTM to predict a customer's future basket items based on its previous transactions.

The Dataset

The dataset used in our analysis originates from the UCI Machine Learning Repository (Chen, 2015). This dataset comprises transnational records encompassing all transactions that took place between December 1, 2010, and December 9, 2011. The population under scrutiny comprises customers of a UK-based and registered non-store online retail, with a notable proportion being wholesalers.

Exploratory Data Analysis (EDA)

For exploratory data analysis, we have outlined three main guiding questions:

What are the most frequently occurring products based on description?

To address this inquiry, we employed two distinct wordcloud visualizations. The first wordcloud dissects all words into individual components, providing insight into the most prevalent terms across product descriptions. Meanwhile, the second wordcloud considers entire string entries from the dataset, offering a broader perspective on frequently occurring phrases.

Upon analysis of the wordclouds, several prominent products consistently emerge as popular purchases. These include items such as "Metal Sign," "Jumbo Bag," "Light Holder," "Lunch Bag," and "Red Retrospot." The visualizations highlight the recurrent presence of these products within the dataset, indicating their significance in customer transactions.

Furthermore, the second wordcloud reinforces these findings by showcasing similar patterns in product descriptions. Phrases like "World War 2 Gliders," "Jumbo Bag Red Retrospot," and "Paper Craft" prominently feature, further emphasizing the prevalence of certain items in customer baskets.

These observations provide valuable insights into customer preferences and purchasing behaviors. By identifying the most frequently occurring products based on description, businesses can optimize their inventory management strategies, tailor marketing efforts, and enhance the accuracy of predictive models, such as LSTM, aimed at forecasting the next item in a customer's basket.

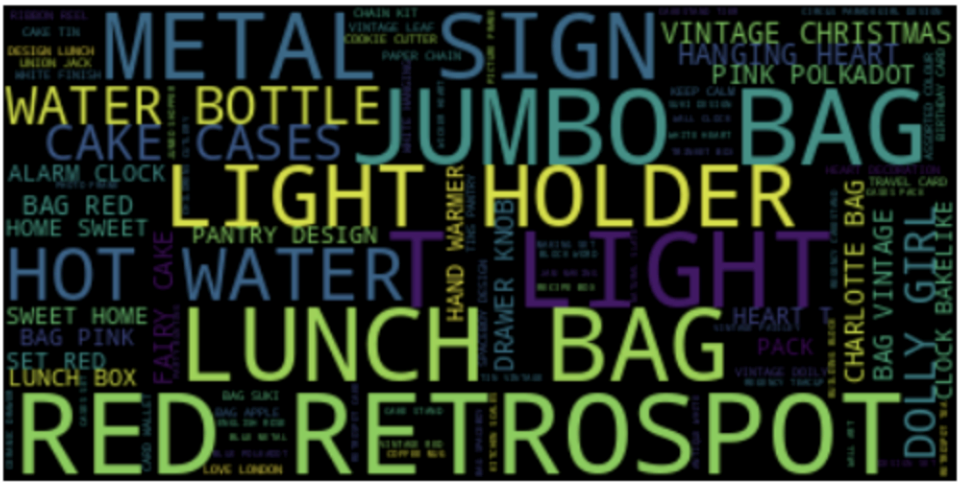


Figure 1. Word cloud by single words

Figure 2. Word cloud by entry

What are the top ten sold inventory items?

To delve into this inquiry, we conducted a thorough analysis to identify the top ten inventory items that experienced the highest sales volume within the dataset. By scrutinizing transaction records, we were able to extract valuable insights into the most sought-after products by customers.

Our analysis culminated in the visualization depicted in Figure 3, showcasing the top ten sold inventory items. Each item is ranked based on its frequency of occurrence across transactions, providing a clear representation of the products that garnered the highest demand.

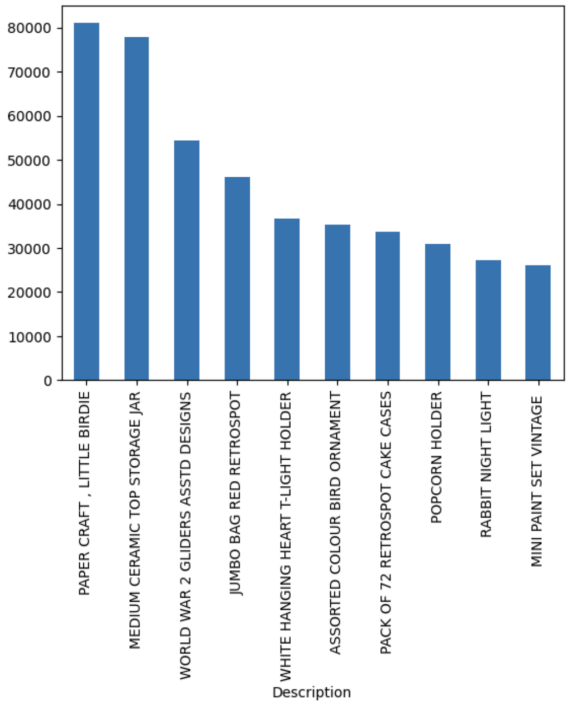


Figure 3. Top ten sold inventory items

How have transaction patterns evolved over time?

Examining our dataset reveals a discernible trend in transaction volumes over time. Notably, there is a gradual uptick in the number of transactions, albeit punctuated by fluctuations rather than following a consistent linear trajectory. These fluctuations suggest cyclical patterns in consumer purchasing behavior, with certain periods exhibiting higher transactional activity than others. While the data exhibits variability, there is an overall positive trend indicating an increase in purchases as time advances.

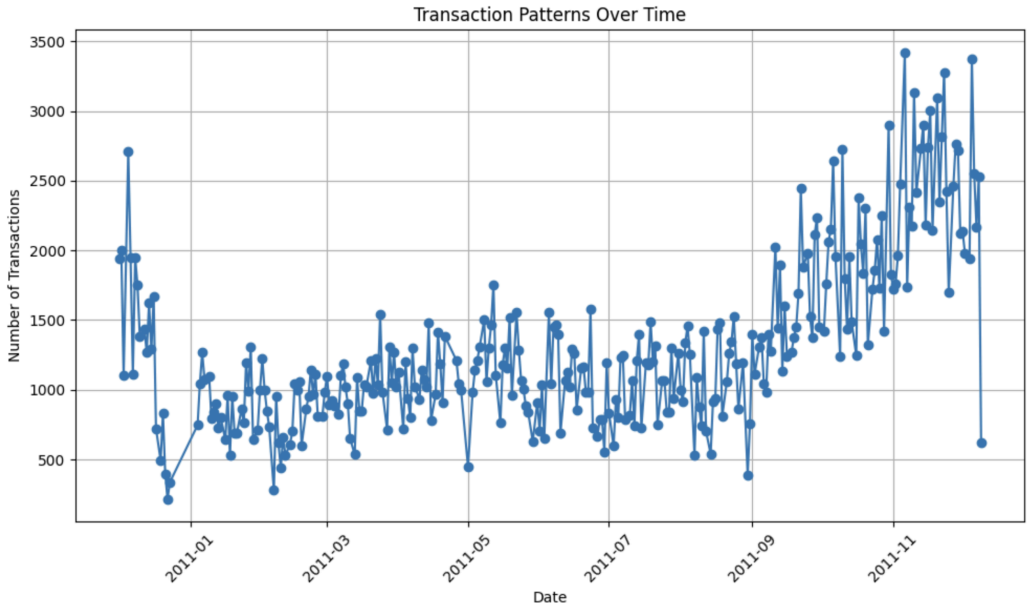


Figure 4. Transaction trends over time

Data Preprocessing and Cleaning

For data cleaning, the initial stages include transforming the dataset to ensure the fit for our LSTM prediction model.

1. **Handling Missing Values:** We identified missing values in the 'CustomerID' and 'Description' columns. Given the role of 'Description' as an input for our model, along with the potential bias from 'CustomerID', we made the decision to drop NA values present for both columns.
2. **Treatment of Duplicated Records**: There are instances when duplicated records exist. We have decided to keep these duplicates as scenarios may arise where items are scanned in a non-sequential order, leading to duplicates. For example, item A might be scanned, followed by items B and C, before item A is scanned again. In such cases, the system may not aggregate the quantities but instead treats each scan as a separate transaction entry.
3. **Handling Negative Quantity:** Records with a quantity less than zero were removed from the dataset, as these entries reveal instances that refer to returns, refunds, or write-offs. Such entries are deemed as irrelevant to our predictive output.
4. **Encoding Descriptions to Numerical**: The 'Description' column was transformed into numerical format using encoding techniques. This conversion method facilitated the representation of each item in a manner compatible with our LSTM model.
5. **Sequence Generation**: Sequences were generated to construct the input-output pairs for model training. For example, each transaction (which is grouped by 'CustomerID', "invoiceNo', and 'InvoiceDate'), the first three items were designated as input features, while the subsequent item was designated as the output label.

6. **Output Label Conversion:** To facilitate binary classification, the output labels (‘y’) were converted into a binary vector format. Here, a value of ‘1’ indicated that the corresponding item was bought, while a value of ‘0’ indicates that the item was not bought.

Our Data Model

LSTM networks represent a specialized form of recurrent neural networks (RNNs) designed to tackle the challenges associated with learning from sequential data. Unlike traditional RNNs, which often struggle to capture long-range dependencies due to the vanishing gradient problem, LSTMs are adept at retaining and utilizing information over extended sequences.

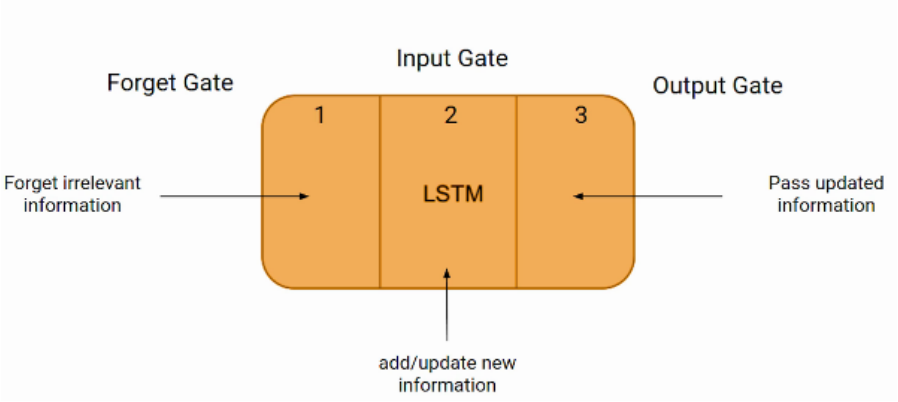


Figure 5. Example LSTM flow

In the context of our project, which aims to predict a customer's future basket items based on their past transactions, LSTM networks offer a powerful framework for sequence processing. By training an LSTM model on historical transaction data, we can capture complex patterns and dependencies in the customer's purchasing behavior over time. The LSTM's ability to retain long-term information allows it to infer potential future items that a customer is likely to purchase, based on their past buying patterns. Through the sequential processing capabilities of LSTMs, our model can adaptively learn from the sequential nature of the data, making it well-suited for tasks involving time series prediction and sequential pattern recognition.

Initial Approach - Label Encoding

For our initial approach, we utilized the one-hot encoded representation for the output variable (Y), which consisted of a vector of size 345,758 (number of transactions) by 3,877 (number of unique items). Each entry in the vector indicates whether a particular item was purchased in a given transaction, with a value of 1 representing a purchase and 0 representing no purchase. This encoding scheme allowed us to handle the multi-class nature of the prediction task, where each item served as a separate class.

We constructed a base LSTM model for this approach, leveraging the sequential processing capabilities of LSTMs to capture temporal dependencies in the customer's purchasing behavior. The model architecture incorporated LSTM layers followed by dropout and dense layers with appropriate activation functions, culminating in a softmax activation function to generate probabilities for each item as shown in Table 1.

First Model Architecture **Table 1.0**

Layer (type)	Output Shape	Param #
lstm (LSTM)	(none, 3, 100)	40800
lstm_1 (LSTM)	(none, 100)	80400
dense (Dense)	(None, 3877)	391577
Total params: 512, 777 Trainable params: 512, 777 Non-trainable params: 0		

Upon evaluating the model's performance, we observed an average validation accuracy of 0.46% and a validation loss of 7.5247 after running 10 epochs. While this initial result provides a benchmark, it also highlights areas for potential improvement.

Second Approach - Label Encoding with a Reduced Number of Items

One limitation we identified in our initial approach is the large number of unique items present in our dataset. This abundance of items may introduce noise and complexity into the model, potentially hindering its ability to generalize effectively. To address this challenge, we devised a strategy to reduce the number of items by filtering out those items that were purchased infrequently.

By setting a threshold of 100 for the minimum number of purchases required to retain an item, we aimed to focus the model's attention on the most relevant and frequently purchased items. This approach helps mitigate the impact of rare or outlier items on the model's performance and streamlines the learning process by prioritizing the most informative features.

To assess the impact of this reduction in item diversity on model performance, we retrained the LSTM model using the revised dataset containing a subset of items that met the minimum purchase threshold. We maintained the same architectural configuration as the initial model to ensure consistency in the experimental setup.

Model Architecture	Table 2.0

Layer (type)	Output Shape	Param #
lstm (LSTM)	(none, 3, 100)	40800
lstm_1 (LSTM)	(none, 100)	80400
dense (Dense)	(None, 3877)	120089
Total params: 241,289 Trainable params: 241,289 Non-trainable params: 0		

From the results, however, we can see that there is no significant improvement, with an average validation accuracy of only 0.575%.

Final Approach - Universal Sentence Encoder

In our third approach, we sought to explore a different method for building our model and preparing our data to improve the predictive performance of our LSTM network. Unlike our previous approaches that utilized one-hot encoded representations of item purchases, this approach leverages the Universal Sentence Encoder (USE) for embedding item descriptions into dense vector representations. Our approach is as follows:

1. ***Utilizing the Universal Sentence Encoder (USE)***
   We employed the Universal Sentence Encoder (Google, 2020), a pre-trained deep learning model developed by Google, to encode item descriptions into dense vector embeddings. These embeddings capture semantic similarities and contextual information within the item descriptions, facilitating more nuanced feature representation than traditional one-hot encoding.
2. ***Description Embeddings Mapping***
   We mapped each item description in our dataset to its corresponding embedding vector using the Universal Sentence Encoder. This mapping enabled us to transform the raw textual descriptions into high-dimensional numerical representations, which capture the semantic meaning and context of each item more effectively.
3. ***Nearest Centroid Encoding***
   To further enhance the feature representation of the item descriptions, we employed a Nearest Centroid classifier to cluster similar embeddings and encode them with representative centroids. This encoding scheme helps consolidate the semantic

information contained within the embeddings, facilitating more efficient learning and prediction.

4. ***Sequential Data Processing***
   We organized our transaction data into sequences, considering each transaction as a sequence of purchased items. Utilizing the grouped transaction data, we constructed input-output pairs by defining a fixed number of steps in and steps out for each sequence.
5. **Sequence Creation with Embeddings**
   For each transaction sequence, we created input sequences consisting of a fixed number of preceding item embeddings (n_steps_in) and corresponding output sequences representing the subsequent item embeddings (n_steps_out). This sequential processing approach allows our model to learn temporal dependencies and patterns in the customer's purchasing behavior.
6. **Transformation of Output Sequences (Y)**
   Notably, the transformation of output sequences (Y) involves converting item descriptions into their corresponding embeddings. This step is crucial for aligning the input and output sequences with their respective dense vector representations, ensuring consistency in feature space and facilitating effective learning during model training.

By adopting this innovative approach, we aim to capitalize on the semantic richness captured by the Universal Sentence Encoder and enhance the predictive capabilities of our LSTM model. The utilization of dense vector embeddings for item descriptions offers a more nuanced representation of the underlying data, potentially leading to improved accuracy and generalization performance in predicting future basket items based on past transactions.

Ultimately, our final model integrates both classification and regression components, departing from our initial approaches which focused solely on classification. This decision was driven by the following considerations:

1. *Regression Aspect:* The LSTM model is adept at directly predicting continuous outputs (vectors) in a high-dimensional space, making it well-suited for regression tasks. Our goal is to generate vectors that closely approximate the true vectors representing recommended products.
2. *Classification Aspect:* We introduce a classification element in the final step by mapping the predicted vectors to discrete product descriptions or categories. Essentially, this involves categorizing each predicted vector into one of several predefined product categories.

Model Architecture

The architecture of the model was constructed using the Keras Sequential API.

Final Model Architecture	Table 3.0

Layer (type)	Output Shape	Param #
lstm (LSTM)	(None, 100)	245200
repeat_vector (RepeatVector)	(None, 1, 100)	0
lstm_1 (LSTM)	(None, 1, 100)	80400
time_distributed (TimeDistributed)	(None, 1, 512)	51712

Total params: 317,312
Trainable params: 377,312
Non-trainable params: 0

Model Components

1. **LSTM Layer:** This layer contains 100 units. Each unit can be thought as a neuron that learns from the data. A higher number of units allows the model to learn more complex patterns but at the cost of increased computational complexity. ReLU (Rectified Linear Unit) is used as the activation function. It introduces non-linearity into the model, allowing it to learn more complex patterns and is popular for its computational efficiency and ability to mitigate the vanishing gradient problem.
2. **RepeatVector Layer:** Following the first LSTM layer, the RepeatVector layer acts as a bridge between the encoding and decoding phases of the model. It repeats the encoded representation of the input sequence n_steps_out times to prepare for the generation of the output sequence. This repetition ensures that the dense information captured about the input sequence is available at each step of the output sequence generation, allowing the model to maintain context as it predicts each item in the sequence of recommendations.
3. **Second LSTM Layer:** Mirroring the first in terms of units (100) and activation function (ReLU), the second LSTM layer operates in a decoding capacity. It takes the repeated encoded vector and processes it through another sequence of LSTM operations, this time configured to return sequences. This ensures that the model outputs a sequence of vectors, each vector intended as a prediction corresponding to a step in the output sequence. This layer further refines the sequence prediction, leveraging the context distributed across the repeated encoded inputs.

4. **TimeDistributed Dense Layer:** The TimeDistributed wrapper around a Dense layer, enables the application of the same Dense operation to each time step of the sequence independently. This means that for each predicted item in the sequence, the Dense layer transforms the LSTM's output vector into a final prediction vector with the correct dimensionality (n_features). This layer is crucial for mapping the high-level representations learned by the LSTM back into the same space as the input embeddings, effectively translating the LSTM's internal language back into actionable product recommendations.

Loss Function

The loss function MSE is highly suitable for this LSTM model due to its direct alignment with the model's regression-based approach to predicting product recommendations. It supports the model's objectives of accurately predicting high-dimensional output vectors, optimizing in a continuous space, and maintaining semantic integrity in the embedding representations. This makes MSE an effective choice for quantifying and minimizing the difference between the model's predictions and the actual product vectors, driving towards more accurate and relevant product recommendations.

Table 4.0 **DESCRIPTION OF PACKAGES USED**

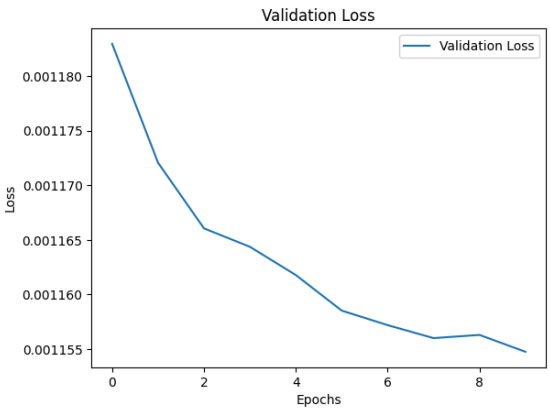
Package	Usage
tensorflow.keras.models (Sequential)	Utilized to construct the model architecture in a sequential manner
tensorflow.keras.layers (LSTM, Dense, Dropout)	These layers are used to define the components of the neural network model

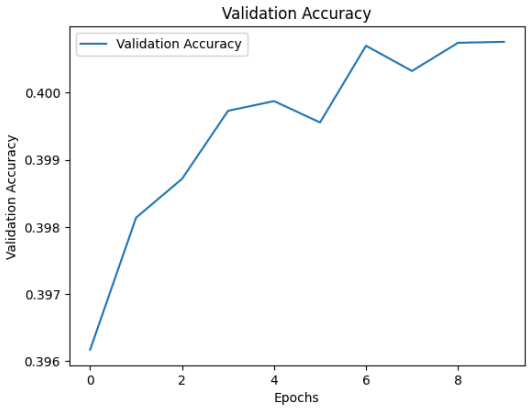
Model Training

To train our LSTM prediction model, we followed a rigorous approach aimed at optimizing its performance and ensuring robustness in predicting future basket items based on past transactional data. The model training process encompassed several key steps, each meticulously designed to facilitate effective learning and generalization.

1. **Data Partitioning**
   We began by partitioning our dataset into distinct subsets for training, validation, and testing. Specifically, we allocated 60% of the data for training, 20% for validation, and reserved the remaining 20% for testing. This partitioning scheme enabled us to train the model on a sufficiently large dataset while also providing separate datasets for validation and testing, essential for assessing the model's performance and generalization capabilities.
2. **Fit Data to the Model**
   With the data partitioned into train, validation, and test sets, we proceeded to fit the training data to the LSTM model. Leveraging the sequential processing capabilities of LSTMs, the model learned to capture temporal dependencies and patterns in customer purchasing behavior from the training data. This process involved iterating over multiple epochs, with each epoch representing a complete pass through the entire training dataset.
3. **Training the Model**
   Despite initially setting the model to undergo training for a total of 100 epochs, we incorporated an early stopping mechanism based on validation loss. After each epoch, the model's performance was rigorously evaluated. If the validation loss did not exhibit significant decrease for a predefined number of epochs (in our case, 10 epochs), the training process was terminated ahead of schedule. This proactive measure was crucial in averting overfitting by halting training when further enhancement in validation loss was improbable, all within the designated training duration of 100 epochs. This integration of early stopping with a predetermined training duration ensured the optimal utilization of computational resources while safeguarding against the risk of overfitting, ultimately enhancing the model's generalization capabilities and predictive accuracy for forecasting future basket items based on historical transactional data.

Model Evaluation





In the evaluation of our LSTM prediction model, we conducted a comprehensive analysis to assess its performance and effectiveness in predicting future basket items based on past transactional data. A key aspect of our evaluation involved monitoring the validation loss and validation accuracy metrics over the course of training epochs, providing valuable insights into the model's learning dynamics and generalization capabilities.

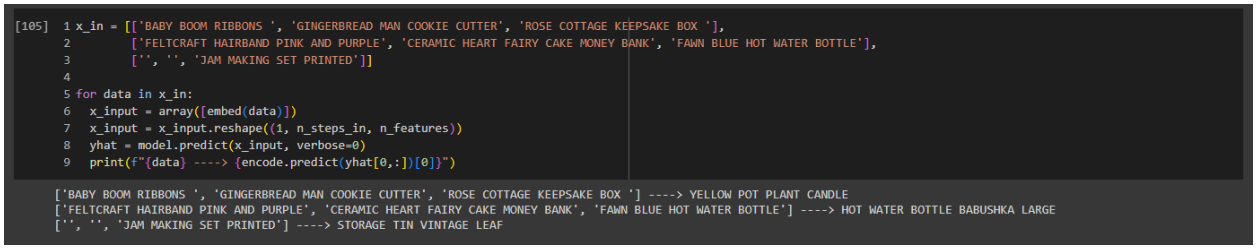
Figure x illustrates the trend of validation loss over the number of epochs. The validation loss metric serves as a measure of the model's predictive accuracy on unseen validation data, with lower values indicating better performance. Our analysis revealed a gradual decrease in validation loss over successive epochs, indicative of the model's ability to effectively minimize prediction errors and improve its predictive accuracy over time. This downward trend underscores the efficacy of our LSTM model in capturing complex patterns and dependencies in customer purchasing behavior, leading to more accurate predictions of future basket items.

Additionally, Figure x showcases the trend of validation accuracy over the number of epochs. Validation accuracy represents the proportion of correctly predicted outcomes relative to the total number of validation samples, providing a measure of the model's overall predictive performance. Our evaluation demonstrated a corresponding increase in validation accuracy as training progressed, indicating the model's capacity to learn from the training data and generalize effectively to unseen validation instances. This upward trajectory in validation accuracy reaffirms the robustness and efficacy of our LSTM model in making accurate predictions of customer purchase behavior.

Model Prediction

Our model receives three item inputs and predicts the next item for the customer’s basket. In our code, we define an array containing descriptions of three products, and our output displays the

predicted next item. Additionally, our model is designed to handle empty inputs and provide appropriate responses, as demonstrated in the screenshot.



```
[105] 1 x_in = [['BABY BOOM RIBBONS ', 'GINGERBREAD MAN COOKIE CUTTER', 'ROSE COTTAGE KEEPSAKE BOX '],
      2         ['FELTCRAFT HAIRBAND PINK AND PURPLE', 'CERAMIC HEART FAIRY CAKE MONEY BANK', 'FAWN BLUE HOT WATER BOTTLE'],
      3         ['', '', 'JAM MAKING SET PRINTED']]
      4
      5 for data in x_in:
      6   x_input = array([embed(data)])
      7   x_input = x_input.reshape((1, n_steps_in, n_features))
      8   yhat = model.predict(x_input, verbose=0)
      9   print(f"{data} ----> {encode.predict(yhat[0,:])[0]}")

['BABY BOOM RIBBONS ', 'GINGERBREAD MAN COOKIE CUTTER', 'ROSE COTTAGE KEEPSAKE BOX '] ----> YELLOW POT PLANT CANDLE
['FELTCRAFT HAIRBAND PINK AND PURPLE', 'CERAMIC HEART FAIRY CAKE MONEY BANK', 'FAWN BLUE HOT WATER BOTTLE'] ----> HOT WATER BOTTLE BABUSHKA LARGE
['', '', 'JAM MAKING SET PRINTED'] ----> STORAGE TIN VINTAGE LEAF
```

Limitations & Challenges

A number of challenges arose as we progressed throughout our project:

Imperfect Data Entries:

The quality of the data poses a significant challenge due to potential variations in how transactions are recorded. For instance, inconsistencies may arise based on how individual cashiers input data into the system. As an illustration, one cashier might record the sale of six apples as six separate transactions, while another cashier might register the sale as a single transaction with a quantity of six. These inconsistencies can introduce noise and discrepancies in the dataset, affecting the accuracy and reliability of the predictive model. Addressing this challenge may require implementing robust data cleaning and preprocessing techniques to standardize and reconcile disparate data entries.

Large Number of Unique Items:

The dataset comprises a vast array of unique items, totaling 3877 distinct products. Handling such a large number of unique items poses several challenges, including increased computational complexity, memory requirements, and potential overfitting issues. With a large number of unique items, the model's ability to generalize patterns and make accurate predictions may be hindered. Moreover, the presence of rare or infrequently occurring items may further exacerbate the challenge of model training and inference. Mitigating this challenge may involve strategies such as feature selection, dimensionality reduction, or exploring alternative modeling approaches tailored to handle high-dimensional data.

Limited Computing Resources and Training Time:

Training deep learning models, such as LSTM networks, demands substantial computational resources and time, particularly when dealing with large datasets and complex architectures. The computational demands escalate further when confronted with the abundance of unique items and the need to process sequential data. Limited access to high-performance computing infrastructure or constrained computational resources may prolong the model training process,

impeding experimentation and iterative model refinement. Consequently, protracted training times may hinder the exploration of alternative architectures, hyperparameter tuning, and optimization strategies. Addressing this challenge may necessitate leveraging distributed computing resources, cloud-based solutions, or optimizing model architectures to improve training efficiency without compromising performance.

Conclusion

In conclusion, our project demonstrates a promising start in utilizing LSTM models to predict sequences of product recommendations, achieving a considerable accuracy of 40%. This success underscores the model's capability in capturing meaningful patterns and relationships within the dataset. It validates our foundational approach of employing deep learning and natural language processing to anticipate consumer preferences.

However, there remains ample opportunity for improvement to further enhance the model's accuracy, thereby refining the relevance and precision of the product recommendations it generates. Moving forward, continued refinement and optimization of our approach will be pivotal in unlocking the full potential of predictive analytics in enhancing the customer experience and driving business success.

References

Chen, Daqing. (2015). Online Retail. UCI Machine Learning Repository. https://doi.org/10.24432/C5BW33.

Google. (2020). Universal Sentence Encoder. Kaggle. https://www.kaggle.com/models/google/universal-sentence-encoder/frameworks/tensorFlow2/variations/multilingual-large/versions/2?tfhub-redirect=true