Zepto

Recommendation Case Study

Arhat Amitabh

(21EE38023 / 21EE30037)

Department of Electrical Engineering,

IIT Kharagpur

Link to Notebook

Overview

This case study focuses on building a recommendation engine for a market basket dataset using Graph Neural Networks (GNNs). The dataset contains transactions where each row corresponds to a list of items bought in one session. The goal is to create a recommendation system that can suggest relevant products based on the items in the basket.

Exploratory Data Analysis

1. **Number of Unique Products**
   The dataset contains a limited number of unique products (430), suggesting a concise product catalogue.

   Code:

```
# Calculate the number of unique products
unique_products = set(item for transaction in transactions for item in transaction)
num_unique_products = len(unique_products)
print(f'Number of unique products: {num_unique_products}')
```

```
Number of unique products: 430
```

---

2. **Transaction Statistics**
   Number of Transactions: 165335 transactions.
   Average Number of Items per Transaction: 2.2 items.
   Maximum Number of Items in a Transaction: 9 items.
   Minimum Number of Items in a Transaction: 2 items.

   Code:

```
# Calculate the number of transactions
num_transactions = len(transactions)
print(f'Number of transactions: {num_transactions}')
```

```
Number of transactions: 165335
```

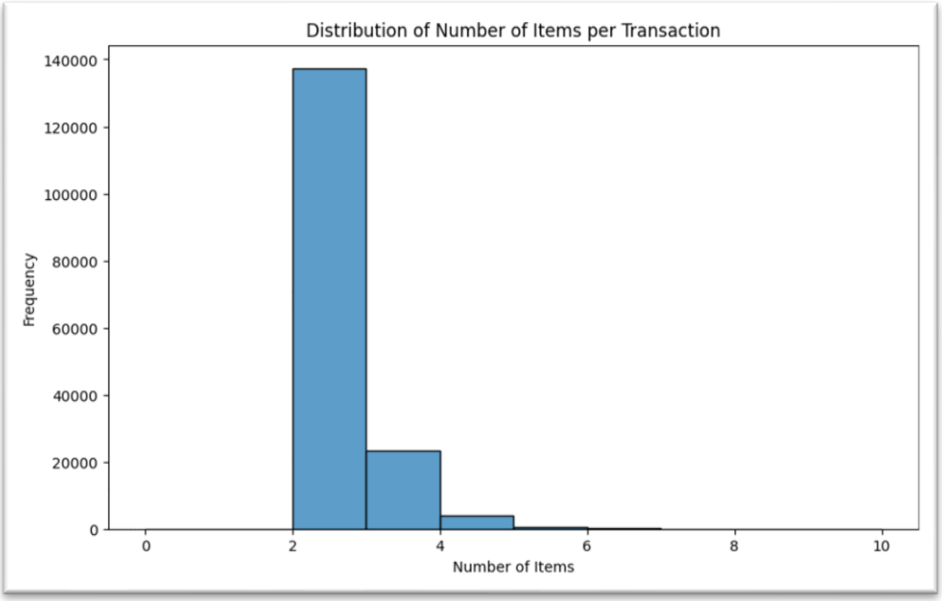
```
# Distribution of the number of items per transaction
num_items_per_transaction = [len(transaction) for transaction in transactions]
print(f'Average number of items per transaction: {np.mean(num_items_per_transaction):.2f}')
print(f'Maximum number of items in a transaction: {np.max(num_items_per_transaction)}')
print(f'Minimum number of items in a transaction: {np.min(num_items_per_transaction)}')
```

```
Average number of items per transaction: 2.20
Maximum number of items in a transaction: 9
Minimum number of items in a transaction: 2
```

---

3. Items per Transaction Distribution

Plot:

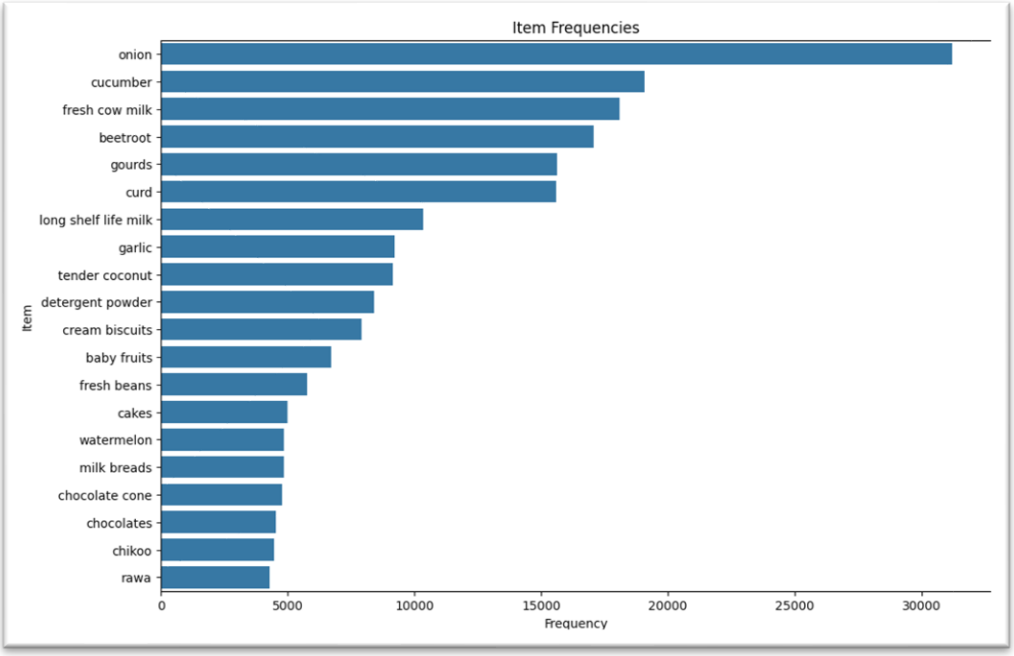


---

4. Item Frequency Analysis

Most Frequent Items: Onion appears in the majority of transactions, indicating its high demand.

Plot:



Recommendation Engine ML Architecture

1. Pre-Processing

- Text Cleaning: Replaced ‘&’ characters with commas for a uniform separator.
- Preparing Dataset: Split strings of each row into a list of words corresponding to individual items
- Combining Sessions: Multiple rows may correspond to same session id, so combining the lists corresponding to each unique session ids.
- Converting product names to unique ids by simply assigning integers using dictionary. Since these integers shall be nodes for the graphs, there is no need for one-hot encoding
- Transforming transactions into relations between product ids instead of product names.

2. Graph Construction

- Create an edge list representing item co-occurrences in transactions.
- Construct a co-occurrence matrix to be used as input for the GNN.

3. Graph Neural Network

- Utilize a GCN (Graph Convolutional Network) model.
- Nodes represent products, and edges represent co-occurrences weighted by frequency.
- The GCN learns item embeddings based on graph structure.

4. Model Training

- Train the GCN using the co-occurrence matrix and item features.
- Optimize using Cross Entropy Loss and Adam optimizer.
- Training model over the whole dataset as testing dataset not required for the given problem statement.

5. **Recommendation Generation**
    - Normalize item embeddings to ensure better similarity search.
    - Calculate similarity scores between items by taking dot product.
    - Generate top-k recommendations excluding already purchased items

---

Results and Takeaways

1. **Model Performance**
    - The GCN model effectively captures item co-occurrences and generates relevant product recommendations.
    - Normalizing embeddings and excluding purchased items improve recommendation diversity and relevance.
    - Further, for qualitative analysis, Apriori Algorithm was used to extract most common corelations and were cross checked by feeding the same to the model.

2. **Insight from Recommendation**
    - The system suggests items frequently purchased together, reinforcing observed patterns from the data.
    - A few examples are given below:

```
Recommendations for incense sticks: ['agarbatti', 'chandan', 'dhoop', 'matchbox', 'sambrani']
```

```
Recommendations for cucumber: ['beetroot', 'fresh beans', 'watermelon', 'garlic', 'blueberry']
```

```
Recommendations for curd: ['fresh cow milk', 'garlic', 'gourds', 'watermelon', 'fruit juices']
```

Which make sense both qualitatively and mathematically using Apriori. The results of Apriori is present in the Notebook.

3. Assumptions and Proposed Solution

- **Assumption:** Product relationships are static and based solely on historical co-occurrences.
  Solution: GCN captures these relationships to generate relevant recommendations. The model can be further improved by adding a categorical variable/feature which categorises items into categories like vegetables, fruits, dairy etc.
- **Assumption:** Points in the dataset which correspond to the same item but with different name in different languages are treated as separate items.
  For example: Incense Sticks (English) and Agarbatti (Hindi)
  Solution: The Recommender system can have an added level during text preprocessing where words of different languages such as Hindi can be translated into a single language such as English for uniformity.
  This issue can be seen in the recommender output for the same:

```
Recommendations for incense sticks: ['agarbatti', 'chandan', 'dhoop', 'matchbox', 'sambrani']
```

Scope for Further Improvement

Given additional data fields, the recommendation engine can be enhanced:

1. User/Real-Time Information

- Incorporate user demographics, purchase history, and preferences for personalized recommendations.
- Use user-product interaction graphs to capture individual user behaviour.
- Account for temporal patterns in transactions to recommend seasonal or trending products.

2. Product Features

- Leverage product attributes (e.g., category, price) to improve recommendation relevance.
- Adapt the model to handle larger datasets with thousands of products and transactions.

3. **Hybrid Models**
    - Combine GNNs with other recommendation techniques (e.g., collaborative filtering, content-based) for hybrid recommendations.
    - Experiment with different GNN architectures (e.g., GraphSAGE, GAT) for improved performance, as here only one architecture is used given the limited time.

Conclusion

This case study demonstrates the application of GNNs to build an effective recommendation engine for a market basket dataset. The proposed architecture, grounded in insights from EDA, highlights the potential of GNNs in capturing complex item relationships. Future improvements leveraging additional data fields and advanced techniques can further enhance recommendation quality and scalability.

The link to the Notebook for the above is attached at the beginning of this document. Thank You!

Fin.