Profit-Oriented High Utility Sequential Recommendation in Big Dataset with MapReduce*

Esther Umoh[1], C.I. Ezeife[2][0000−0002−9424−9223], and Ritu Chaturvedi[3][0000−0003−0233−674X]

[1] School of Computer Science, University of Windsor, Windsor, Canada
umoh1@uwindsor.ca

[2] School of Computer Science, University of Windsor, Windsor, Canada
cezeife@uwindsor.ca

[3] School of Computer Science, University of Guelph, Guelph, Canada
chaturvr@uoguelph.ca

Abstract. Online transactions platforms such as e-Commerce and social networks generate massive volumes of data, which are complex and with sizes that can reach terabytes (10^{12} bytes) or petabytes (10^{15}), making data mining and recommendation tasks increasingly challenging. Existing systems for high utility sequential pattern mining (HUSPM) extract valuable (e.g., profitable) e-commerce sequential products to recommend to buyers considering both frequency of items and utility (e.g., profit, importance) of itemsets. Existing HUSPM systems include those named as HUSREC21, HUSP21, and HUSP-SP23, mine high utility sequential patterns using a single machine for these HUSPM tasks, resulting in inefficient processing of big sized datasets, longer execution times and high memory consumption.

This paper proposes a system called Big High Utility Sequential Pattern Recommendation System (BIGHUSREC), that extends the HUSREC21 system to mine high-utility sequential patterns from big sized datasets through a "Top-K" approach integrated with the MapReduce framework. The focus is on extracting the Top-K most valuable (profitable) and yet relevant to the user patterns, with the aim of improving recommendation accuracy, while minimizing execution time. The proposed system uses MapReduce to partition data into smaller parts (Mapping), and analyzing each part in parallel to identify profitable patterns before aggregating the results (Reducing) for a comprehensive output. By combining purchase and clickstream data, the proposed BIGHUSREC effectively improves recommendation accuracy.

Keywords: High Utility Sequential Pattern Mining · Data Mining · Big Data · MapReduce · Sequential Database · E-commerce Recommendation Systems.

* This research was supported by the Natural Science and Engineering Research Council (NSERC) of Canada under an Operating grant (OGP-0194134) and a University of Windsor grant.

1 Introduction

Big data are large, complex datasets that are difficult to analyze, process, or visualize using traditional methods [13]. Big datasets define such data as website clicks, customer purchase histories, and interactions on social media. A dataset domain of interest to businesses for big data analysis is customer purchase data, which includes details as what products customers buy, when they buy them, and in what order. For example, Amazon collects this data to predict what customers might want next and recommends additional products they are likely to purchase. Traditional methods like association rule mining (ARM) [9] and sequential pattern mining (SPM) [14] have been used to analyze such data. However, these methods only consider frequency and do not account for profitability or importance to the business. High Utility Sequential Pattern Mining (HUSPM) mines patterns that not only occur frequently but also generate significant profits [19]. For example, while customers may often purchase phone cases, HUSPM might reveal that purchasing a phone with an extended warranty is more profitable. The use of HUSPM to suggest products that maximize both customer satisfaction and profits is commonly done in e-Commerce platforms such as Amazon and Alibaba. Since traditional high utility sequential pattern mining algorithms are designed to run on a single computer, they are limited by the size of datasets they can process efficiently at a time. Real-world datasets often reach terabytes (TB) (10^{12} bytes) or even petabytes (PB)((10^{15} bytes) in size, making them impossible for a single system to manage efficiently. To address this limitation, technologies like Hadoop and Apache Spark [7] were developed. Hadoop includes the Hadoop Distributed File System (HDFS) and MapReduce [6]. HDFS stores data across multiple machines, ensuring reliability through data replication, while MapReduce (MR) allows for distributed and parallel processing of data. MR breaks down large datasets into smaller chunks that are processed across different machines simultaneously. Once processing is complete, the results are combined to form the final output. By combining tools like MapReduce with advanced data mining techniques such as HUSPM, businesses can efficiently analyze large datasets to uncover actionable insights. These insights help improve customer satisfaction, boost sales, and enhance profitability. Methods like classification, clustering, frequent pattern mining, and high utility itemset mining (HUIM) are used by recommendation systems to make sense of customer purchase data and provide meaningful product suggestions [9].

Service providers may want to use recommender systems technology to: (i) Generate increased sales; (ii) Promote long-term loyalty of buyers; (iii) improve users' experience; (iv) Discover novel and diverse items that may not be in the frequently purchased list but which are liked by users; (v) Understand users' preferences and popular interests and tailor services to cater for such interests [10]. An effective RS must balance the needs of both the provider and the user, and high utility sequential pattern mining focuses on this aspect.

1.1 Some Important Concepts

1. Recommender System Techniques [1] include the three main broad techniques of collaborative filtering (CF), content based (CB) filtering and a hybrid approach that can model user interactions using various mining and machine learning algorithms. Recommendation algorithms represent users and items interactions in a user item ratings matrix and use learning algorithms (e.g., classification, clustering, association rule and sequential pattern mining) in a data-driven manner on ratings matrix to define user-item dependencies model which are used to make predictions for new or targeted users. Generally, a collaborative filtering algorithm accepts a user item rating matrix as input and predicts missing ratings by going through the steps of: (i) Computing the similarity of $user_i$ and $user_j$ for all users by computing the cosine similarity between pairs of users. (ii) Secondly, selecting the top K most similar neighbors to a target user whose rating is being predicted as the users with the highest cosine similarity values on the item column being predicted. (iii) Finally, the target user's predicted rating is computed using a weighted or mean centered average of the top K neighbors of the target user from step 2. The algorithm for the content based filtering is similar but while the rows of the input rating matrix are users, the columns of this matrix are features or properties of items or products being rated [16].
2. Data Mining Techniques include the process of finding useful patterns, models, and information in large sets of data [9], using association and sequential pattern mining, clustering and classification techniques. Association rule mining techniques find meaningful relationships or associations between items in large databases. The Apriori algorithm [14] is a widely used method for mining frequent itemsets and discovering association rules in transactional databases. An association rule expression is of the form X $\rightarrow$ Y, which indicates that if the set of items in X (the rule antecedent) are purchased, then, the items in Y (the rule consequent) are purchased next. The strength of an association rule is evaluated using two key metrics, support and confidence. Support of an association rule is defined as the count or percentage of records (tuples) in the database that contains all the set of items in the rule. The confidence measures the percentage of all the antecedent records that also contain all the set of items in the rule. Support is defined as $|antecedent- > consequent|/|database|$ while confidence = $|antecedent- > consequent/|antecedent|$. When mining for frequent patterns from a given database of purchase records, the association rule mining algorithms will first find all frequent combinations of items or patterns that meet a specified minimum support. Sequential pattern mining tasks compute frequent sequential patterns from a given sequential database that would meet a given minimum support. Sequential pattern mining techniques follow the same general reasoning and methodology as the Apriori algorithm [9] except that the input database is a sequential database and the technique for calculating candidate sequences will perform a join of the large (i+1)-sequence with itself using a different technique that is suitable

for sequential data called the gsp-join when using the similar basic sequential pattern mining algorithm called the GSP algorithm [14]. The sequential pattern rule computed from these sequential mining algorithms is in the form of $X- > Y$ where both the antecedent (X) and consequent (Y) are sequential items.

3. High Utility Itemset Mining Techniques would identify item sets that can yield substantial profits. The difference between high utility problem and frequent pattern mining is that with the high utility pattern mining, each of the input database item (e.g., purchased product) has a utility value also attached to it to measure for internal utility, such as the quantity of such product that was purchased, and for external utility value, to measure a qualitative value of the item or product such as price or profit of the product. In this context, the term "itemset utility" refers to the perceived interest, significance, or profitability of an item to users. High utility itemset mining focuses on discovering itemset whose utility meets or exceeds a user-specified minimum utility threshold u. The utility of an itemset is calculated based on both internal utility and external utility [19]. Utility can be defined as "a measure of how 'useful' (i.e., profitable) an itemset is." It is a quantitative expression of user preference. The total utility U (X, T) of an itemset X in transaction T is the sum of the product of internal utility and external utility for each item in the itemset. It is calculated as:

$$U(X,T) = \sum_{i \in X} q(i,T) * p(i)$$

for internal utility(i, T) = q(i, T) for quantity of i in transaction T, and external utility(i) = p(i) for price of item i.

4. High Utility Sequential Pattern Mining (HUSPM) is similar to high utility frequent pattern mining except that the database in this case is sequential database rather than itemset database. High Utility Sequential Pattern Mining (HUSPM) provides a weight to each item based on its relative importance. HUSPM considers non-binary purchase quantities in a sequential order. A sequence is classified as a High Utility Sequential Pattern (HUSP) if its utility exceeds a user-defined minimum utility threshold (count) [20]. The work in [2] proposed an algorithm named HUSP-Miner for HUSPM. The task of High Utility Sequential Pattern Mining is to extract frequent sequential patterns from a sequential database given the internal (quantity) and external utility (profit) measures which give the importance of each item in the sequence. Formally, a Sequential Database D is defined as follows. Let there be the set I of all items I = $\{i_1, i_2, \ldots, i_m\}$. A quantitative transaction database D is a set of sequences, denoted as D = $\{s_1, s_2, \ldots, s_n\}$ where each transaction, s_q is a set of items (i.e., $s_q \subset I$), and has a unique identifier q called its SID (Sequence Identifier). Every item $i \in I$ is associated with a positive number p(i), which is called its external utility (e.g., profit of item). Every item i appearing in a transaction s_q has a positive number q(i, s_q) called its internal utility, which represents quantity of i in sequence s_q.

5. Big Data Features. Big datasets are generated at an unprecedented scale and speed, encompassing a wide range of formats and requiring advanced technologies to extract meaningful insights [5]. The characteristics of big Data are (i) Volume: for huge amount of data measured in gigabytes, terabytes (TB), and over. For example, social media and e-commerce platforms like Facebook and Amazon generate massive amounts of data from customer purchases, click history, posts, comments, and messages.
   Variety: Data come in different forms. It can be structured (like data in tables), semi-structured (like emails or logs), or unstructured (like text, videos, photos, and social media posts).
   Velocity: This is about the speed at which data is generated and processed. Some data, like online transactions or real-time weather data, need to be processed quickly for fast decision-making.
   Veracity: The trust worthiness or quality (reliability) of data is referred to as veracity. Value: Value is the advantage (insight) that the data provide to the organization or business from analyzing data.
6. Map Reduce Architecture. As the volume of data continues to grow rapidly, traditional database management systems like ORACLE, Microsoft SQL, MYSQL struggle to efficiently handle and process big data. These systems rely on a centralized server to store and process data, and this becomes a major limitation when dealing with massive datasets [5]. Handling enormous volumes of data, such as petabytes, often creates bottlenecks in centralized systems, making scalability difficult and rendering traditional databases inadequate for such tasks. To address this challenge and provide a more efficient solution, Google introduced the MapReduce algorithm [6], a programming framework designed for parallel processing of large datasets across distributed systems. It works in three main stages: the map phase, where tasks are broken down and distributed; the sorting and shuffling phase; and the reduce phase, where results are aggregated and combined into a final output. This framework allows large tasks to be divided into smaller subtasks that are processed simultaneously on multiple systems, with the results merged to form a complete dataset for analysis. The MapReduce algorithm sets the foundation for many modern algorithms designed to handle large-scale data mining through parallel processing on distributed machines, effectively [20]. In the MapReduce framework, two primary tasks, Map and Reduce, are divided into smaller subtasks. The Map task processes large datasets by breaking them into smaller, manageable subsets of data, breaking each item into tuples known as key-value pairs so that multiple computers can process them at the same time. This division is based on different criteria to make sure that the data remains meaningful and is processed efficiently. Some of the criteria for partitioning data are based on continuous ranges of transaction IDs (TIDs), transaction or total utility values, where high utility transactions can be kept apart from low utility transactions with (key, value) like (itemset, utility) pairs to be assigned to different nodes. A simple example is: Given as sequential database, SDB, with 4 sequences with ids, SID 1, 2, 3, 4; and the 4 sequences with their quantities purchased (inter-

nal utility) corresponding to these sequence ids are: < (apple 2, bread 1), (cake 1), (apple1) >; < (apple 1), (bread 1, cake 2) >; < (cake 1), (apple 1, bread 1), (doughnut 3) >; < (bread 2), (cake 1), (apple 2, bread 1)>. Also given is the candidate 1-itemset with their external utilities (profit) as: (apple:3; bread:2; cake:4; dougnut:1); a minimum utility threshold. The task is to compute the high utility sequential patterns from this SDB using the MapReduce method. The three key stages of the MapReduce process, are shown in Figure 1 as the Map stage, the Shuffle stage, and the Reduce stage.
i. Input: The input data is divided into smaller chunks, with each chunk processed by a separate machine (or node) in the cluster.
ii. Map Stage: The divided data chunks are processed in parallel across the distributed system. Using the example SDB above, two fragments of SDB consisting of fragment 1 for SID 1 and 2, fragment 2 for SID 3 and 4 are obtained based on SID. Each node computes the total utility of its sequences' subsequences as the sum of the product of quantity and profit of each of its items as given in the total utility formula. For subsequences $s_1 \ldots s_n$ in SID 1 (e.g., for s_1 in SID 1, there is apple with total utility of 3 x 3 = 9, bread with 1 x 2 = 2, cake with 1 x 4 = 4) and for s_2 in SID 1, there is (apple, bread) with total utility of (2 x 3) + (1 x 2) = 8, etc. As done in even Apriori algorithm, the total utilities of the big datasets are shared and computed by different nodes faster at this stage. The second stage of this Map function involves presenting the results of the nodes as key-value pairs and in this example, it is (itemset, total utility) pairs. For example in fragment 1, there are (apple: 9) and (apple: 3) for the two SIDs 1 and 2, and in fragment 2, there are (apple: 3) and (apple:6) for SIDs 3 and 4. These itemsets with total utilities are next sorted during the Shuffle and Sort phase to prepare the data for the Reduce phase in order to identify high utility sequential patterns.
iii. Shuffle & Sort Stage: This stage reorganizes the mapped data (e.g., the key-value pairs in different nodes), grouping similar keys (e.g., itemset like apple) together to get the total utilities for the key. With our example SDB, there are grouped utility results for itemsets like for apple: (9, 3, 3, 6); bread: (2, 2, 2, 4); cake: (4, 8, 4, 4); doughnut:(3); (apple,bread): (8, 5, 10); (apple, cake): (10); (bread, cake): (10).
iv. Reduce Stage: The grouped data (e.g., total utilities for each itemset or key) is aggregated and combined (summed up) to generate the results (e.g., total utility for apple, bread, etc.) and compared against the minimum utility threshold of say 20 in this case and to identify patterns (itemsets) with total utility (TU) greater than or equal to 20 as mined high utility patterns (HUP). The HUPs with their TUs are: (apple:21; cake: 20; (apple, bread): 23) .
v. Output: The results are written to a storage system for further use.

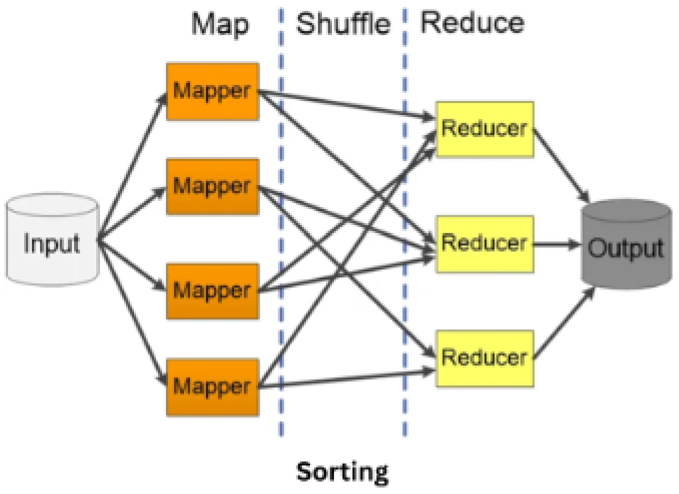

Fig. 1: Map Reduce Architecture

1.2 Paper Contributions and Outline

This paper makes the following contributions to the problem of high utility sequential pattern mining on big datasets.

1. Handling scalability of big Size data with MapReduce architecture.
2. Optimizing HUSPM execution time using HUS-SPAN miner on large datasets.
3. Enhancing recommendations using high utility patterns focused on sellers.

This paper proposes the BIGHUSRec system, which first creates big volume sized high utility purchase sequential database (HUPSDB) of purchases and clicks, finds high utility sequential patterns of purchases and clicks in this database using the HUS-Span [17] algorithm in a MapReduce framework to mine High Utility Sequential Patterns (HUSPs) from the HUPSDB. The mined patterns are aggregated and filtered to produce high utility sequential rules (HUSR). These rules are then used to update the initial item rating matrix M, resulting in an enriched matrix M_1. Using a similar method to that in HSPRec19 [4]. Then, the proposed algorithm also uses sequential pattern rules to find the next possible purchases and consequential bonds between clicks and purchases. These mined rules and bonds between clicks and purchases information are used to estimate ratings (values between 0 and 1) for unrated items in the M_1 user item matrix, showing possibility of the item being purchased by the user as learned from the rules. The result of the update to M_1 is the matrix M_2 that now has reduced sparsity problem and will increase recommendation accuracy when used as input to the collaborative filtering for selecting the top K items to recommend to a user.

2 Related Work

Some of the key techniques that impact or contribute to techniques for addressing the challenges of high utility sequential recommendation in big data environment have been introduced in section 1. These techniques include collaborative filtering, traditional sequential pattern mining algorithms like the GSP [14], high utility sequential pattern mining algorithms like the HUSPM algorithm in [2], and techniques for addressing big size data challenges using MapReduce. Relevant recent work in these categories are further summarized in this section.
1. PrefixSpan (Prefix-projected sequential pattern mining) algorithm by [12] finds the sequential patterns without candidate generation by constructing a projected database, consisting of a collection of the original database's sub-patterns with prefix suffixes. In order to mine the patterns that have the support threshold in the projected database, the PrefixSpan first finds the frequent patterns of size 1. From there, it builds its projected database. Every element of the pattern identified in the projected database is concatenated with the pattern of length 1 to create patterns of length 2, a process that continues until the projected database is empty.
2. Early high utility sequential pattern mining algorithms include Utility Span algorithm (US) by [2] and USpan Algorithm [19], and HUS-SPAN algorithm by [17]. Sequential pattern mining generally treats all items as having the same importance/utility and assumes that an item appears at most once in the database, which does not reflect several real-life scenarios, and thus the useful information of sequences with high utilities (high profits) are lost. High utility sequential pattern considers external utility (e.g., unit profits) and internal utility (e.g., quantity) of items such that it can provide users with patterns having high utility values. Utility Span Algorithm (US) by [2] mines high-utility sequential pattern mining with a pattern growth approach that allows it to cut down on time spent on multiple scans of huge databases for support counts. It scans the database a maximum of three times. The problem solved by this US algorithm is to find the high utility sequential patterns based on their minimum sequence utility from a sequence database which has internal utility, sequence utility and profits gained per item in a given sequential database. The algorithm first calculates the high utility sequential patterns of each item based on their minimum sequence utility from a sequence database which has internal utility, sequence utility and profits gained per item using the utility computation formula. Next the US algorithm scans the SDB once to detect length-1 frequent sequences. Subsequently, it generates projected databases by considering length-1 frequent sequences as prefixes with a second database scan. Then, using a pattern growth approach, it divides the search spaces (projected databases) recursively and applies the same technique into them. The US algorithm only generates the high-frequent sequences without generating many intermediate candidates. A third database scan is needed to discover the high-utility sequential patterns from the high-frequent sequences. The second algorithm in this category is the USpan Algorithm [19], which finds sequences having a maximum utility. USpan algorithm satisfies Downward Closure Property. USpan is composed of a

lexicographic quantitative-sequence tree (LQS-tree) which is the search space, and two concatenation mechanisms of I-Concatenation and S-Concatenation it uses to generate newly concatenated utility-based sequences. It has two pruning strategies. Based on the LQS-tree structure, USpan [19] adopts the sequence-weighted utilization (SWU) measure and the sequence weighted downward closure (SWDC) properties to prune unpromising sequences and to improve the mining performance. The third algorithm in this category is an efficient algorithm for high utility sequential pattern mining (HUS-SPAN) by [17]. It first constructs Utility Matrix with each q-sequence in the input database converted into a utility matrix. This matrix forms the foundation for estimating sequence utility efficiently during pruning. In its second step, it forms a Lexicographic Q-Sequence (LQS) Tree rooted at (). The HUS-Span algorithm organizes candidate patterns using a Lexicographic Q-Sequence Tree (LQS-tree) where each node represents a sequence (e.g., (b), (c)) and stores its corresponding utility. The root node is the empty sequence (). From the root, all promising 1-sequences are generated as direct children, other steps are followed to complete the generation of high utility sequential patterns.
3. E-Commerce Recommendation Systems. Two related work in E-commerce recommendation arena are HPCRec18 by [18] and the HSPRec19 system by [4]. HPCRec18 normalizes the historical purchase frequency matrix to improve rating quality and mines the session-based consequential bonds between clicks and purchases to generate potential ratings to improve the rating quantity. The primary goal of the proposed sequential recommendation system HSPRec19 is to improve a user-item rating matrix by mining past E-commerce data for common sequential trends. The authors in [4] developed a daily purchase sequence database of customer database based on consequential bond between the click and the purchase database. They also proposed two algorithms, such as HSPRec19 (Historical sequential recommendation) and SHOD (Sequential Historical Periodic Database) System.
4. Recent High Utility Sequential Pattern Mining Algorithms include MapReduce_HUSP21 by [11] which contributes a novel three-tier MapReduce model for scalable mining of high-utility sequential patterns (HUSPs) in large-scale datasets. Two key data structures are introduced, the sidset and Utility-Linked List, to improve computational efficiency by optimizing memory usage and speeding up utility calculations. In the first phase (Identification), MapReduce identifies promising items by calculating their utility based on the formula u(ir, X) = q(ir, X) * pr(ir) where q(ir, X) is the quantity and pr(ir) is the profit of an item ir in a transaction. The second phase (Local Mining) applies the HUS-Span algorithm across partitions of the dataset, discovering local high-utility patterns within each partition. The local utility of a sequence in a partition is calculated using

$$uL(t, D_i) = \sum_{s_j \in D_i} U(t, S_j)$$

where u(t, S_j) is the utility of a sequence in each transaction. The sidset and Utility-Linked List structures are used to speed up the process by reducing repeated utility calculations. In the third phase (Integration), local patterns are aggregated, and the global utility of a sequence

$$uG(t, D) = \sum_{D_i \in D)} Ul(t, D_i)$$

is computed across all partitions. Only sequences retained are those whose global utility satisfies $uG(t, D) \geq \delta\ x\ u(D)$ where δ is the user-defined minimum utility threshold.
The second system in this category is High Utility Sequential Pattern Recommendation (HUSRec21) [16] system, which improves e-commerce recommendations by integrating high utility sequential pattern mining to maximize seller profitability and recommendation accuracy. The system first converts historical purchase data into a high utility sequential purchase database (HUSPDB) using both internal utilities (quantity of items bought) and external utilities (price or profit per item). It then applies the USpan algorithm to mine high utility sequential patterns from the purchase data and the PrefixSpan algorithm to mine frequent patterns from clickstream data. These patterns are combined to enrich the user-item matrix, reducing sparsity and improving recommendations.
The third system in this category is HUSP-SP23, Faster Utility Mining on Sequence Data [20], which introduces an efficient methodology for High-Utility Sequential Pattern Mining (HUSPM) through a combination of innovative data structures and advanced pruning techniques namely (Early Pruning (EP) and Irrelevant Item Pruning (IIP)). The core of the system is the SeqPro structure, a compact representation of sequence data that stores both the sequence and its utility information in an array format, significantly reducing memory usage and computational time. This structure is designed to efficiently handle the growing number of candidate patterns by utilizing the extension-list, which tracks the utility and extension positions of sequences. A key contribution of the authors [20] is the introduction of Tighter Reduced Sequence Utility (TRSU), a more refined upper bound compared to traditional methods like Sequence Weighted Utility (SWU) and Reduced Sequence Utility (RSU).

3 THE Proposed Big High Utility Sequential Recommender System (BIGHUSREC)

The proposed BIGHUSREC system, as in the unpublished thesis [15], builds upon existing systems HUSREC21 [16] and HUSP-SP23 [20], both of which have limitations when applied to real-world, large-scale e-commerce scenarios. The HUSRec21 system extended an earlier HSPRec19, improved recommendations by moving from frequent sequential pattern mining to high-utility sequential pattern mining, recognizing not just how often items appear together, but also their business value. However, HUSRec21 was primarily tested on small datasets, limiting its scalability. As e-commerce platforms now handle millions of transactions daily, this system struggles to process such volumes effectively

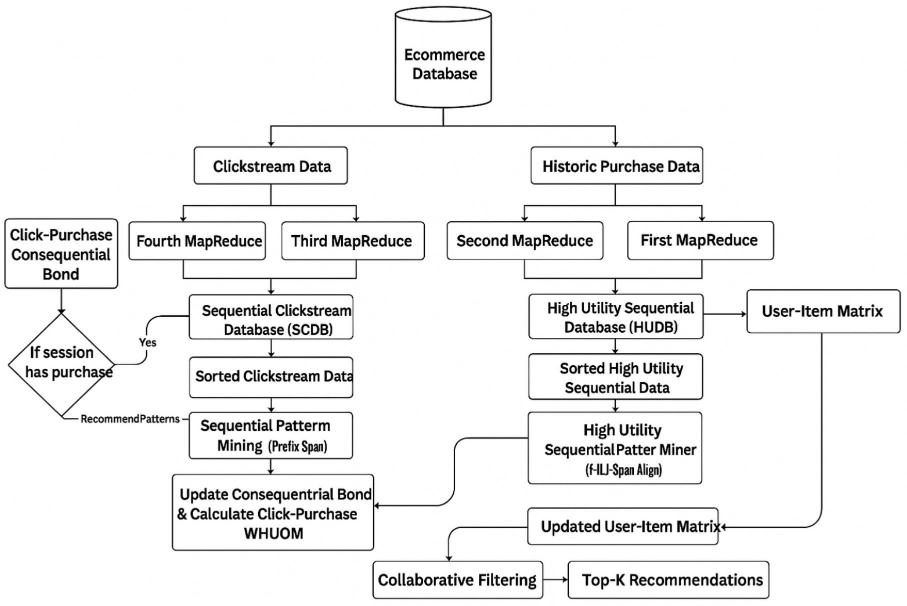
Ecommerce Database
Clickstream Data
Historic Purchase Data
Click-Purchase Consequential Bond
Fourth MapReduce
Third MapReduce
Second MapReduce
First MapReduce
Sequential Clickstream Database (SCDB)
High Utility Sequential Database (HUDB)
User-Item Matrix
If session has purchase
Yes
Sorted Clickstream Data
Sorted High Utility Sequential Data
RecommendPatterns
Sequential Patterm Mining (Prefix Span)
High Utility SequentialPatter Miner (f-ILJ-Span Align)
Update Consequentrial Bond & Calculate Click-Purchase WHUOM
Updated User-Item Matrix
Collaborative Filtering
Top-K Recommendations

Fig. 2: BIGHUSREC Architecture

and to uncover deeper, more useful patterns in larger datasets. Moreover, HUS-Rec21 and HUSP-S23 suffer from inefficiencies in the mining process. Both systems tend to generate many low-utility candidate patterns, which increases processing time and reduces the quality of recommendations. These unpromising candidates consume memory and computational resources without contributing meaningfully to the recommendation output. The proposed BIGHUSREC addresses these limitations by incorporating a scalable MapReduce-based architecture [11] and replacing older mining algorithms like USpan with the more efficient HUS-Span algorithm. This enables the system to handle much larger datasets, eliminate unnecessary candidates early, and deliver more accurate and timely recommendations, making it better suited for the current scale and demands of e-commerce recommendation systems. The system architecture of the proposed BIGHUSREC is shown in Figure 2. BIGHUSREC consists of 5 major steps which will be explained in the next section. The main algorithm for BIGHUSREC is shown as Algorithm 1.

3.1 Details of Algorithm Steps

1. BigDataGen: The proposed BIGHUSREC algorithm starts by calling the BigDataGen algorithm. This module creates a synthetic dataset of approximately 100GB in size using a custom python script that leverages the Faker library for data generation and employs CSV partitioning to ensure scalability. This script produced over 500 million records of user session data, modeled after the

Algorithm 1: BIGHUSREC to mine High Utility Sequential patterns from Big data.

```
Input: Desired Data size (S), Number of files (N), Number of Rows per file
    (R), click-to-purchase ratio (a) Historical click database, Reference
    dataset (D(ref)), AWS S3 bucket (B), S3 folder path (F), Historical
    Cleaned clickstream and purchase datasets (D(click, clean),
    D(purchase, clean) uploaded to S3, User query matrix (M)
Data: High Utility Purchase Sequential Database (HUPSDB), Sequential
    Clickstream Database (SCDB), High Utility Sequential Rules (HUSR),
    Sequential Pattern Rules (SPR), Updated matrices (M1, M2),
    Weighted High Utility Occupancy Matrix (WHUOM), User-item rating
    scores (r(ui))
Output: Normalized matrix (M2), Top K item recommendations per user.
1 BIGHUSREC calls the BigDataGen() module as summarized in section
    3.1, to generate large scale synthetic purchase and clickstream datasets;
2 BIGHUSREC calls MRHHUSDB( ) to generate High Utility Purchase
    Sequential Database (HUPSDB), Sequential Clickstream Database (SCDB),
    and initial item frequency matrix (M) (summarized in subsection 3.1);
3 BIGHUSREC calls BigDataMiner() to Mine High Utility Sequential
    Rules (summarized in section 3.1);
4 BIGHUSREC calls IntegrateClickPurchasePatterns() to correlate click
    and purchase behavior or mine Sequential Pattern Rules (SPR) and generate
    item rating matrix $M_2$, and WHUOM;
5 BIGHUSREC calls RecommenderEngine() to generate user-item scores,
    normalize the enriched matrix, $M_2$, and return the Top-K item
    recommendations per user;
```

YOOCHOOSE dataset [3]. Each record captures sequences of product clicks and purchases with varying lengths and patterns. The session data includes key attributes such as session identifiers, timestamps, item identifiers, click and purchase events, product prices, quantities purchased, and randomized session durations. After the data generation, the raw dataset is pre-processed for data quality, consistency, and completeness. This involves standard data cleaning, transformation, integration, and reduction steps. Following pre-processing, the cleaned click stream and purchase datasets are uploaded to Amazon S3 using the Amazon Web Service (AWS) Command Line Interface (CLI), allowing for centralized, scalable storage within the BIGHUSREC framework. The output of this step is a cleaned dataset, which is then passed to the MRHUSDB algorithm summarized next.

2. Data Conversion and Matrix Construction, MRHHUSDB: The next step in the BIGHUSREC algorithm is to prepare the cleaned data set for mining and recommendation tasks by extracting high-utility sequences, creating click-based session data, and setting up the initial user-item matrix, M_1, which will be further improved in later stages. This module MRHHUSDB handles these tasks with scripts that use MapReduce to work with big sized datasets. The high util-

ity purchase sequential database (HUPSDB) is created using this MRHUSDB algorithm.
3. BigDataMiner module is for mining high utility sequential rules using MapReduce and HUS-Span frameworks. The HUPSDB purchase sequential database is fed into the high utility sequential pattern miner (HUSPM) module, and this module analyzes the sequence of purchases to generate high-utility sequential rules that capture high profit patterns across sessions. It goes through the Map Phase to mine local patterns using the HUS-Span sequential pattern mining algorithm, where each Map task receives one or more sequences from the sequence database, HUPSDB and applies the HUS-Span algorithm locally to mine high utility sequential patterns. Within each mapper, the sub steps in the HUS-Span miner are done. Secondly, it performs the shuffle and sort phase. The algorithm groups all identical subsequences emitted by the mappers based on key-value pairs. Thirdly, comes the reduce phase where pruning and final rule selection is done, and only patterns with utility greater than or equal to the minimum utility threshold are retained.
4. IntegrateClickPurchasePatterns module focuses on linking user intent shown through click stream data with actual purchase behavior, as captured in the transaction sequences from earlier steps. This is used to incorporate sessions (cold start cases) that do not include purchased data, but clicked data may generate purchases next, as done in some of the earlier systems [4] as a mechanism to reduce the sparsity and cold start problem often seen in recommendation systems. To achieve this, the High Utility Sequential Rules (HUSR), Click-Purchase Similarity (CPS) metrics, and Sequential Pattern Rules (SPR) [4] mined from the clickstream data using the PrefixSpan algorithm, are all used. These elements are then used to enrich the Weighted High Utility Occupancy Matrix (WHUOM) and update the user-item matrix by adding inferred user preferences based on implicit behavior. WHUOM is a weighted high utility rating matrix, defined using the utility occupancy (uo) [8] of an itemset. The uo of an itemset, e.g., X in a transaction, Tq denoted as uo(X, Tq) is the ratio of the utility of X in that transaction divided by the total utility of that transaction. vii. Click purchase similarity CPS (X (click sequence), Y (purchase sequence)) uses the frequency and positioning of goods inside the user's click and purchase sequences to compute the similarity between click and purchase sequences for each session [4] as a numeric value between 0 and 1. CPS(click sequence, purchase sequence) is computed and used as the weight or probability that user u will purchase the entire click sequence. The Weighted High Utility Occupancy Matrix (WHUOM) module takes weight assigned by the CPS module as input and generates weighted frequent purchase items for each user where the weight of the patterns are the CPS(user's click sequence, SDB sequences) with weight for each SDB sequence as the computed CPS. Weighted High Utility Occupancy Matrix (WHUOM) now uses the weight assigned to the SDB sequences by the CPS module as input and generates frequent items with weight high utility occupancy matrix using these weights to cover for cold start items and highly sparse items with unknown ratings.

5. RecommenderEngine module is called finally by the proposed BIGHUSREC algorithm to use the enriched user-item matrix M_2 constructed earlier with actual purchases, inferred preferences from clickstream behavior, and high utility sequential rules to generate Top-K item recommendations for each user/session, especially those with sparse or no purchase data. This step uses user-based collaborative filtering over modified rating matrix M_2, augmented with implicit utility-weighted ratings derived from WHUOM to generate recommendations.

4 Experimental and Performance Analysis

The dataset utilized came from YOOCHOOSE GmbH, provided for ACM RecSys 2015 [3]. This dataset captures interactions from a European online retailer, consisting of two files documenting click events and purchase transactions. These events are spread across 9,512,786 unique sessions, over 52,739 distinct products classified into 339 categories. Due to the significant data volume required for the experiment, it was necessary to generate approximately 100GB of synthetic data, modeled after the YOOCHOOSE dataset, while maintaining the structural and statistical characteristics of the original dataset. The synthetic data replicated key features, including session-based click stream records and purchase events, while adhering to realistic distributions such as session lengths, item popularity, category proportions, and temporal patterns. The purchase data is structured as follows: (SessionId, Timestamp, ItemID, Price, Quantity). The performance of user-based collaborative filtering approach using a historical dataset as discussed above, considering metrics such as Mean Absolute Error (MAE), Precision, Recall, F1 Score and execution time are recorded. Higher mean absolute errors indicate less efficiency in accurate rating prediction, while lower mean absolute errors signify greater efficiency in predicting accurate ratings. The proposed BIGHUSREC is evaluated for performance in comparison with existing high utility sequential mining systems, such as the HUSREC21 [16], and HUSP-SP23 [20]. These existing systems show poorer performance in precision and execution time as the dataset size increase as shown in Table 1. The proposed BIGHUSREC

Table 1: Recommendation Accuracy Comparison Between Recommendation Systems

System	Precision	Recall	F1	MAE score
HUSRec21	0.742	0.650	0.692	0.384
HUSP-SP23	0.781	0.675	0.723	0.369
BIGHUSREC	0.856	0.750	0.799	0.324

demonstrates significantly improved scalability, achieving lower execution times compared to HUSP-SP23 [20] across all data sizes, and completing the 100 GB dataset in 24,400 seconds; a substantial reduction relative to HUSP-SP23. These results in Table 2 highlight the effectiveness of BIGHUSREC in handling large-scale high-utility sequential pattern mining tasks with better efficiency. For time

Table 2: BIGHUSREC Versus HUSREc and HUSP-SP23 systems across data sizes

data size	HUSREC	HUSP-SP23	BIGHUSREC
1	300	170	115
10	300	5,000	1280
30	300	10,200	3950
60	300	36,000	12000
100	300	86,000	24400

complexity analysis of the proposed BIGHUSREC, the most time consuming step is done with mining frequent sequential patterns of big sized N number of sequences in the SDB. This N sequences is split into N/d for c nodes in the MapReduce process. The time complexity of BIGHUSREC is O(MNL/c) where M is the average length of the sequences, N is the number of sequences, and L is the average length of frequent sequential patterns.

5 Conclusions and Future Work

This paper presents the Big High Utility Sequential Recommendation (BIGHUSREC) system for high utility sequential pattern mining of large scale datasets. Unlike traditional systems that suffer from decreased efficiency with increasing data volumes, proposed BIGHUSREC sustains stable precision and delivers efficient execution times. The system achieves this by combining the distributed MapReduce framework with the HUS-SPAN algorithm, enabling effective partitioning, processing, and extraction of high utility sequential patterns. Experimental results demonstrate that proposed BIGHUSREC outperforms existing systems, offering improved precision and faster execution.
Future work may consider i. exploring extending high utility sequential mining datasets with other V's of big data apart from big volume; ii. Integrating high utility mining of big data with advanced machine learning models (deep learning, re-inforcement learning, use of large language models) to uncover more complex patterns; iii.Explore using other distributed and parallel frameworks e.g Apache-eSpark to discover interesting HUSPs on large-scale databases in distributed environments.

References

1. Aggarwal, C.C.: Recommender systems, vol. 1. Springer (2016)

2. Ahmed, C.F., Tanbeer, S.K., Jeong, B.S.: A novel approach for mining high-utility sequential patterns in sequence databases. ETRI journal 32(5), 676–686 (2010).
3. Ben-Shimon, D., Tsikinovsky, A., Friedmann, M., Shapira, B., Rokach, L., Hoerle, J.: Recsys challenge 2015 and the yoochoose dataset. In: Proceedings of the 9th ACM Conference on Recommender Systems. pp. 357–358 (2015)
4. Bhatta, R., Ezeife, C.I., Butt, M.N.: Mining sequential patterns of historical purchases for e-commerce recommendation. In: Big Data Analytics and Knowledge Discovery: 21st International Conference, DaWaK 2019, Linz, Austria, August 26–29, 2019, Proceedings 21. pp. 57–72. Springer (2019)
5. Chen, M., Mao, S., Liu, Y.: Big data: A survey. Mobile networks and applications 19, 171–209 (2014).
6. Dean, J., Ghemawat, S.: Mapreduce: simplified data processing on large clusters. Communications of the ACM 51(1), 107–113 (2008).
7. Elmasri, R., Navathe, S.: Fundamentals of Database Systems, vol. 7. Pearson (2016).
8. Gan, W., Lin, J. C. W., Fournier-Viger, P., Chao, H. C., & Yu, P. S. 2021. A Survey of Utility-Oriented Pattern Mining. IEEE Transaction on Knowledge and Data Engineering. Vol. 33, No. 4, pp. 1306-1327.
9. Han, J., Pei, J., Tong, H.: Data mining: concepts and techniques. Morgan kaufmann (2022).
10. He, X., Liu, Q., Jung, S.: The impact of recommendation system on user satisfaction: A moderated mediation approach. Journal of Theoretical and Applied Electronic Commerce Research 19(1), 448–466 (2024.)
11. Lin, J.C.W., Djenouri, Y., Srivastava, G., Li, Y., Yu, P.S.: Scalable mining of highutility sequential patterns with three-tier mapreduce model. ACM Transactions on Knowledge Discovery from Data (TKDD) 16(3), 1–26 (2021).
12. Pei, J., Han, J., Mortazavi-Asl, B., Pinto, H., Chen, Q., Dayal, U., Hsu, M.C.: Mining sequential patterns by pattern-growth: The prefixspan approach. IEEE Transactions on Knowledge and Data Engineering 16(11), 1424—-1440 (2004).
13. Sagiroglu, S., Sinanc, D.: Big data: A review. In: 2013 international conference on collaboration technologies and systems (CTS). pp. 42–47. IEEE (2013).
14. Srikant, R., Agrawal, R.: Mining sequential patterns: Generalizations and performance improvements. In: International conference on extending database technology. pp. 1–17. Springer (1996).
15. Umoh, E.: High Utility Sequential Pattern Based Recommendation Systems of Big Data Using Map Reduce. Master's thesis, University of Windsor (Canada) (2025).
16. Virk, K.: Improving E-Commerce Recommendations using High Utility Sequential Patterns of Historical Purchase and Click Stream Data. Master's thesis, University of Windsor (Canada) (2021).
17. Wang, J.Z., Yang, Z.H., Huang, J.L.: An efficient algorithm for high utility sequential pattern mining. In: Frontier and innovation in future computing and communications. pp. 49–56. Springer (2014).
18. Xiao, Y., Ezeife, C.I.: E-commerce product recommendation using historical purchases and clickstream data. In: International Conference on Big Data Analytics and Knowledge Discovery. pp. 70–82. Springer (2018).
19. Yin, J., Zheng, Z., Cao, L.: Uspan: an efficient algorithm for mining high utility sequential patterns. In: Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 660–668 (2012).
20. Zhang, C., Du, Z., Zu, Y.: An efficient algorithm for extracting high-utility hierarchical sequential patterns. Wireless Communications and Mobile Computing 2020(1), 8816228 (2020).