Available online at http://ijdea.srbiau.ac.ir

Int. J. Data Envelopment Analysis (ISSN 2345-458X)

Vol. 11, No. 4, Year 2023 Article ID IJDEA-00422, Pages 65-78
Research Article

International Journal of Data Envelopment Analysis

Science and Research Branch (IAU)

Dynamic Web Personalization Using a Hybrid Recommender System with Sequential Pattern Detection and Link Sequence Similarity

Vahid Saffari*[1], Karamollah Bagherifard[1], Hamid Parvin[2], Samad Nejatian[3], Vahideh Rezaei[4]

[1] Department of Computer Engineering, Islamic Azad University, Yasooj, Iran.
[2] Department of Computer Engineering, Islamic Azad University, Nourabad Mamasani, Iran.
[3] Department of Electronic Engineering, Islamic Azad University, Yasooj, Iran.
[4] Department of Mathematics, Yasooj Branch, Islamic Azad University, Yasooj, Iran.

Received 20 May 2023, Accepted 14 July 2023

Abstract

In this paper, we propose a dynamic hybrid web recommender system aimed at personalizing websites through sequential pattern discovery and link sequence similarity detection. The system is evaluated on two standard datasets, Zanbil and NASA, containing extensive web server logs. After preprocessing the logs by removing irrelevant data and segmenting user interactions into sessions, we perform user clustering using the PAM algorithm with three similarity metrics: Levenshtein Distance, Longest Common Subsequence (LCS), and Needleman-Wunsch (NW). The optimal number of clusters is determined through evaluation of Precision, Recall, and F-measure, with the best results found at 350 clusters for Zanbil and 500 for NASA.

User profiles are generated using FP-Growth and SPADE, which help in identifying frequent navigation patterns. The model is then evaluated, yielding optimal Precision of 0.91 and recall of 0.83 for SPADE combined with LCS. Results show that this combination produces the best performance, effectively capturing user behavior and providing superior personalized recommendations.

The study demonstrates that this hybrid approach enhances the personalization of web, delivering more relevant suggestions to users based on their previous interactions.

Keywords: Web Recommender System, Website Personalization, Sequential Pattern Discovery, Link Sequence Similarity, Hybrid Model.

* Corresponding author: Email: Vahid.Saffari@iau.ac.ir

1. Introduction

Web recommender systems have become one of the key tools in enhancing user experience and increasing user engagement with websites. These systems, by analyzing user behavior and recommending relevant content, not only provide a personalized experience but also help websites improve their efficiency in attracting and retaining users. With the growing amount of data available on websites and the diversity of user behaviors, the need for efficient and accurate recommender systems has become more crucial than ever [1].

These systems are essential not only for improving user experience but also for increasing profitability and customer satisfaction in e-commerce environments [2].

Website personalization through recommender systems is based on analyzing user behavior and identifying similarities among them. Users, through their interaction with websites, leave behind vast amounts of direct or indirect data. This data can include clicks, page visit times, and content interactions, which are stored in server logs. Web logs, as rich sources of user information, are highly valuable [3].

These data can be used to better understand user behavior and identify frequent patterns [4].

One method of analyzing log data is identifying users' "navigation sequences" based on their clicks. These sequences represent the paths users take over time on a website. Identifying these navigation sequences and analyzing the similarities between them is one of the key tools in developing recommender systems [5].

By segmenting users into sessions, typically defined by time intervals such as 30 minutes since the last activity, user behavior can be divided into analyzable units.

This method helps us extract meaningful patterns from each session separately [6].

Identifying sequential patterns and discovering similarities among user sequences is a significant challenge in web data analysis. These patterns may include common paths that users follow on a website. Using various algorithms, these sequences are analyzed to identify similarities and repeated patterns [7].

One of the major challenges in this field is selecting the appropriate algorithm for sequence analysis and optimizing the computational processes for large datasets [8].

Several algorithms have been proposed for identifying similarities and discovering sequential patterns, each with its own strengths and weaknesses. Algorithms such as Levenshtein, LCS, and Needleman-Wunsch are commonly used for calculating similarities between sequences [8]. While these algorithms offer high accuracy in identifying similarities, selecting the best algorithm for each dataset and optimizing it for large datasets remains a fundamental challenge. Additionally, frequent pattern mining algorithms such as FP-Growth and SPADE are used to identify sequential patterns in user data [7].

In this paper, we aim to dynamically examine all possible combinations of similarity detection and sequential pattern discovery algorithms. The main goal is to generate various models by combining these algorithms and then evaluate the accuracy of each model using evaluation metrics such as Precision, Recall, and F-measure. Ultimately, by analyzing the results, we will propose a dynamic hybrid model for web recommender systems that can contribute to more accurate and efficient website personalization.

2. Related Work

One of the prominent studies in this field was conducted by Dacrema and colleagues. In this research, the authors explored the challenges of reproducibility in recommender system research. They found that many papers in this domain have irreproducible results, which may be due to a lack of sufficient data, lack of access to source codes, or flaws in evaluation methods. The study recommends that researchers adopt more transparent and well-documented approaches in their work to achieve better results in the development of recommender systems [9].

Another significant study was conducted by Sun a colleague, in which the BERT4Rec model was introduced. BERT4Rec uses transformer techniques to generate recommendations. By employing bidirectional encoding and learning temporal sequences, BERT4Rec significantly improved the accuracy of recommendations compared to traditional and advanced models. This model offers a better understanding of user interactions with the system, providing more accurate recommendations [10].

Another important study was carried out by Zhou and colleagues, where the S3-Rec model was introduced. This model uses a self-supervised learning approach with mutual information maximization. By leveraging mutual information between user interactions, the model achieved significant improvements in recommendation accuracy. This approach has led to a better understanding of users' needs and preferences [11].

In a separate study, Ma and colleagues introduced memory-augmented graph neural networks. These models, using graph structures and augmented memories, helped improve recommendation accuracy. By combining graphical information with augmented memories, they outperformed previous models and showed significant improvements in recommendation precision [12].

Another study by Fan and colleagues introduced temporal collaborative graph networks, which enhanced recommender systems by continuously considering time. This model demonstrated that considering time continuously can lead to significant improvements in recommendation accuracy, as user interactions with systems evolve over time, and the model is better able to capture these changes [13].

Gao and colleagues, in their study, introduced a state-preserving RNN framework that retains state information over time to improve recommendation accuracy. This model, by preserving state information, outperformed traditional RNNs and showed that state retention can significantly enhance recommendation precision [14].

Gong and colleagues introduced a contrastive self-supervised learning model, which uses robust reinforcement to improve recommendation accuracy. This model showed that using robust reinforcements can lead to significant improvements in recommendation accuracy and enhance the performance of recommender systems [15].

Li and colleagues introduced temporal self-attention networks, which consider the time intervals between user interactions and use this information to improve recommendation accuracy. The model demonstrated that considering time intervals can lead to significant improvements in recommendation precision [16].

In a study by Wang and colleagues, implicit feedback refinement methods were introduced to filter out incorrect and unreliable feedback, thereby improving recommendation accuracy. The study showed that refining implicit feedback can lead to significant improvements in accuracy and reduce issues associated with incorrect implicit feedback [17].

Xie and colleagues introduced hierarchical attention networks that consider the hierarchical structures of user interactions and use this information to improve recommendation accuracy. This model showed that leveraging hierarchical attention structures can lead to significant improvements in recommendation precision [18].

Research in the area of sequential recommendations is expanding. One study offers media and product suggestions to customers based on their past behavior and interests. Recent machine learning algorithms for sequential recommendation rely on deep learning and transformers. Given the competitive and sudden performance of simple nearest-neighbor algorithms for session-based recommendations, the author examined nearest-neighbor techniques for the challenges of sequential recommendation. In two out of four datasets, nearest-neighbor methods outperformed the transformer-based BERT4Rec algorithm. Deep learning also outperformed simpler methods on larger datasets, supporting the idea that neural methods improve with data growth [19].

3. Proposed Method

In this section, we present a dynamic hybrid web recommender system aimed at personalizing websites by leveraging sequential pattern detection and link sequence similarity analysis. The proposed method involves multiple stages, starting from data preprocessing, session generation, and user identification to clustering users based on their browsing patterns using three similarity metrics: Levenshtein Distance, Longest Common Subsequence (LCS), and Needleman-Wunsch (NW). User profiles are generated through two powerful sequential pattern mining algorithms: FP-Growth and SPADE, which help in identifying frequent patterns in user behavior. The system utilizes the K-Nearest Neighbors (KNN) algorithm to recommend personalized content based on the closest user profiles. The process flow is detailed in Figure 1, which visually depicts the major steps of the proposed method.

Our flowchart has been illustrated in Figure 1:

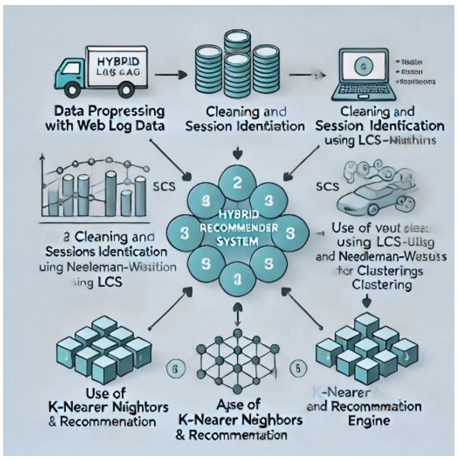
HYBRID LOG CAG
Data Propressing with Web Log Data
Cleaning and Session Identiation
Cleaning and Session Identfication using LCS-Niathins
SCS
2 Cleaning and Sessions Identication uing Neeleman-Wastion sing LCS
2
3
3
3
3
HYBRID RECOMMENDER SYSTEM
S
S
1
SCS
Use of vout using LCS-Ulisg and Needleman-Wasuss for Clusterings Clastering
Use of K-Nearer Niightors & Recomenation
6
Ajse of K-Nearer Neghbors & Recommentation
5
K-Nearer and Recommmation Engine

Figure 1: Proposed Method Flowchart

3.1 Datasets

In this research, we utilized two well-known and standard datasets: Zanbil and NASA [20,21]. These datasets are widely used in web recommendation systems to evaluate user behavior and navigation patterns based on web server logs.

Zanbil Dataset: This dataset contains web server logs from an e-commerce website and includes user navigation data and clickstreams, providing valuable insights into customer behavior on the platform.

NASA Dataset: This dataset consists of web server logs from NASA's Kennedy

Space Center web server, capturing user activities such as page visits.

The details of each dataset, including the number of records, are summarized in Table 1. These datasets serve as the foundation for user session generation and subsequent processing in this study.

Table 1: Number of records in Zanbil and NASA datasets.

Dataset	Number of Records
Zanbil	4,477,843
NASA	1,048,574

3.2 Data Preprocessing and User Identification

Web mining is the process of extracting valuable insights from web data, and data preprocessing is a critical initial step in this process. Web server log files, typically stored in the Common Log Format (CLF), contain essential information such as client IP addresses, timestamps, request details, HTTP methods, and response status codes. However, raw web logs often include unnecessary or noisy data, making preprocessing a fundamental step to ensure data quality.

In this study, we refined the raw log files from both Zanbil and NASA datasets to extract meaningful user behaviors. The preprocessing phase involved several key tasks:

1. URL Normalization: We standardized URL formats, removing redundant parameters such as session IDs and irrelevant query strings.

2. Exclusion of Irrelevant Content: Non-essential requests such as images, stylesheets, JavaScript files, and multimedia content (e.g., videos) were excluded to focus on primary interactions, such as user clicks on web pages.

3. Error Handling: Requests resulting in HTTP error codes (such as 404 or 500) were filtered out to eliminate incomplete or failed interactions.

4. User Identification: Users were identified based on their IP addresses and user-agent strings to differentiate between unique visitors and bots.

5. Session Identification: We employed time-based heuristics, where a session was defined as a sequence of user activities with no more than 30 minutes of inactivity. This approach allowed us to group related user actions into meaningful sessions for further analysis.

Through these steps, we aggregated and refined the raw data, which was essential for generating user sessions and preparing the datasets for subsequent phases of pattern recognition and recommendation. Data preprocessing plays a vital role in enhancing the quality and reliability of web data, laying the groundwork for accurate analysis and insights.

After completing the preprocessing phase, Table 2 provides the statistics of each dataset, including the number of records, unique users, sessions, and unique URLs.

Table 2: Post-processed statistics for Zanbil and NASA datasets.

Data Set	***Number of Records***	***Sessions Count***	***Users Count***
Zanbil	4,477,843	252,518	76,744
NASA	1,048,574	98,372	46,351

3.3 Dataset Splitting

After the preprocessing phase, the datasets were divided into three subsets: training, validation, and testing. This splitting is a crucial step in machine learning workflows to ensure that models can generalize well to unseen data. In our study, the datasets were split as follows:

- **Training Set**: 60% of the data was allocated to the training set. This subset was used to train the clustering and recommendation models.
- **Validation Set**: 20% of the data was reserved for validation purposes. The primary role of this set was to fine-tune the model and determine the **optimal number of clusters (k)** for the clustering algorithm. We applied validation techniques to evaluate various values of k and select the one that yields the best results in terms of clustering performance.
- **Testing Set**: The remaining 20% of the data was used as the testing set, which served to evaluate the final model's performance on unseen data and measure its ability to provide accurate web recommendations.

The validation set plays a key role in identifying the optimal number of clusters (k) in our clustering algorithm (PAM). By experimenting with different k-values, we ensured that the clustering structure best represents the underlying user sessions, which is essential for generating meaningful user profiles and improving the recommendation system's accuracy.

This systematic division of the data ensures that the model is not only trained effectively but also tested for its ability to generalize and make accurate recommendations based on user behavior patterns.

Here is an updated version of the text with the mathematical explanation for PAM clustering added:

3.4 Clustering Step

In this step, we use the PAM (Partitioning Around Medoids) algorithm to cluster users based on the similarity of their web navigation sequences. The clustering process is crucial because we aim to group users with similar browsing patterns, which will later be used for building recommendation profiles.

The clustering phase can be divided into three main parts:

3.4.1. Determining the Optimal Number of Clusters:

To identify the optimal number of clusters, we begin by testing different cluster counts ranging from 5 to $\sqrt{n}$, where n is the number of samples. For each dataset and for each possible number of clusters, we calculate evaluation metrics such as Precision, Recall, and F-measure. The process continues iteratively until further increases in the number of clusters do not significantly reduce the error. As shown in Figures 2 and 3, by analyzing the F-measure, we observe that for Dataset 1, the optimal cluster count is 500, and for Dataset 2, the optimal count is 350. The optimal values of k for Datasets 1 and 2 are also shown in Tables 3 and 4.

Table 3: Determining the Optimal Number of Clusters for Dataset1

Number of Clusters (K)	Evolution Metrics		
	Precision	***Recall***	***F-Measure***
10	0.72	0.65	0.68
40	0.78	0.70	0.74
80	0.82	0.76	0.79
130	0.86	0.81	0.83
200	0.88	0.84	0.86
300	0.89	0.85	0.87
500	**0.90**	**0.86**	**0.88**
600	0.88	0.84	0.86

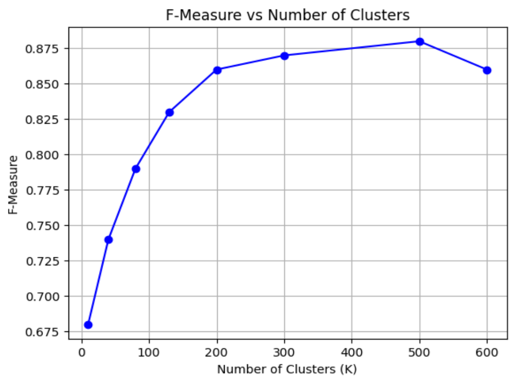

Figure 2: F-Measure Comparison for the Zanbil

Table 4: Determining the Optimal Number of Clusters for Dataset2

Number of Clusters (K)	Evolution Metrics		
	Precision	***Recall***	***F-Measure***
10	0.68	0.60	0.64
50	0.74	0.69	0.71
100	0.79	0.74	0.76
200	0.85	0.80	0.82
350	**0.91**	**0.88**	**0.89**
500	0.89	0.86	0.87
700	0.87	0.83	0.85

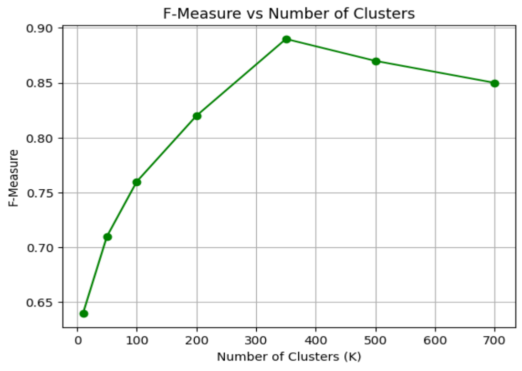

Figure 3: F-Measure Comparison for the NASA

3.4.2. Similarity Metrics:

To measure the similarity between users based on their click sequences, we employ three different sequence similarity algorithms: Levenshtein Distance, Longest Common Subsequence (LCS), and Needleman-Wunsch (NW).

- Levenshtein Distance: Measures the minimum number of single-character edits (insertions, deletions, or substitutions) required to change one sequence into another.

$$d_{lev}(s_1, s_2) = \min(\text{edits to transform } s_1 \text{ into } s_2)$$

- Longest Common Subsequence (LCS): Finds the longest subsequence present in both sequences without changing the order of elements.

$$LCS(s_1, s_2) = \max(\text{length of common subsequence between } s_1 \text{ and } s_2)$$

- Needleman-Wunsch (NW): A dynamic programming algorithm used to align two sequences optimally by maximizing the similarity score.

$$d_{nw}(s1, s2) = max(alignment\ score\ between\ s1\ and\ s2)$$

Each similarity measure will be applied separately during clustering to evaluate the best fit for the dataset.

3.4.3. Clustering with PAM:

Once the similarity measures are calculated, we apply the PAM algorithm to cluster users. PAM is particularly suitable for handling sequence data because it identifies representative sequences (medoids) that minimize the total dissimilarity within clusters.

1. Initial Medoid Selection: Given a dataset X= $\{x_1, x_2 \ldots, x_n\}$, PAM initially selects k random points from the dataset to serve as medoids:

$$M = \{m_1, m_2, \ldots, m_k\},\ m_i \in X$$

These medoids represent the initial cluster centers.

2. Assignment of Points to Medoids: Each point x_i in the dataset is assigned to the nearest medoid based on the selected similarity metric:

$$C_j = \{x_i \mid d(x_i, m_j) = \min_{1 \leq j \leq k} d(x_i, m_j)\}$$

where d (x_i, m_j) is the distance between point x_i and medoid m_j.

3. Medoid Update: PAM iteratively replaces medoids with other points in the cluster to minimize the total dissimilarity:

$$T = \sum_{j=1}^{k} \sum_{x_i \in C_j} d(x_i, m_j)$$

The goal is to find the medoids that minimize the total distance T, which represents the sum of the dissimilarities within each cluster.

4. Stopping Criterion: The algorithm terminates when changing the medoids no longer reduces the total dissimilarity.

5. Final Clustering:
   Using the optimal number of clusters determined from the validation dataset, we apply PAM clustering with each of the three-similarity metrics (Levenshtein, LCS, and NW) separately. This allows us to evaluate the clustering results and identify the most effective sequence similarity measure for grouping users with similar browsing behaviors.

3.5. User Profile Generation

User Profile refers to a structured representation of a user's preferences, behavior, and interests, typically constructed from their interaction history. In the context of web recommendation systems, a user profile is created based on the user's activity, such as the sequence of URLs they visit, the time spent on pages, and the actions they take. These profiles are vital for personalizing content and improving the accuracy of recommendation algorithms. The key to an effective recommendation system is extracting frequent patterns from user sessions and using these patterns to build robust profiles for future predictions.

To identify frequent patterns, we will employ two well-known sequential pattern mining algorithms: FP-Growth and SPADE. Each algorithm uncovers frequent patterns in a dataset, which can be used to form user profiles.

FP-Growth (Frequent Pattern Growth) is an efficient algorithm for mining frequent itemset without candidate generation. The algorithm works in two steps:

1) Construction of the FP-Tree: This is a compact structure that retains the itemset information, where each node represents an item and its frequency.
2) Mining frequent itemset from the FP-Tree: The tree is recursively divided into conditional FP-trees, and frequent patterns are extracted.
3) The mathematical representation of FP-Growth is as follows:
4) Let D represent the dataset of transactions and $I = \{i_1, i_2, \dots, i_i\}$ be the set of items.
5) The support of an itemset X is defined as the number of transactions in which X appears:

$$\text{Support}(X) = \frac{\text{Frequency of } X \text{ in } D}{|D|}$$

6) The FP-tree is built by sorting items in descending order of their frequency and inserting them into the tree.
7) Frequent patterns are mined by

recursively generating conditional trees for subsets of items and extracting patterns that meet a minimum support threshold.

SPADE (Sequential Pattern Discovery using Equivalent Class) is designed to mine sequential patterns efficiently by exploring frequent sequences in a depth-first search manner. It uses lattice structures to generate sequence patterns and applies constraints to reduce the search space.

The mathematical description of SPADE is as follows:

1. Let D be the sequence database, where each sequence is a list of itemset ordered by time.
2. A sequence S = $(a_1, a_2, \ldots, a_k)$ is frequent if the number of sequences in which SSS appears is greater than or equal to a predefined minimum support threshold.
3. SPADE uses vertical database representation to store sequences, meaning each item is associated with a list of all transactions in which it appears.
4. The support of a sequence is calculated as:

$$\text{Support}(S) = \frac{\text{Number of sequences containing } S}{|D|}$$

5. The algorithm explores frequent sequences by intersecting lists of items and recursively combining them to form longer sequences.

Both algorithms, FP-Growth and SPADE, will be implemented to generate user profiles. These profiles will then be used to make personalized recommendations based on user behavior. By using two different pattern-mining techniques, we aim to compare their efficiency and accuracy in building comprehensive user profiles. The performance of these methods will be evaluated to determine the optimal approach for sequence pattern discovery.

3.7. Model Evaluation

For the evaluation of the recommendation model, we rely on the test dataset and three primary metrics: Precision, Recall, and the F-measure. These metrics are widely used to assess the performance of recommendation systems, providing insight into the accuracy and relevance of the recommendations made by the system.

1. Precision is defined as the ratio of correctly recommended links (true positives) to the total number of recommended links (true positives + false positives):

$$Precission = \frac{TP}{TP + FP} \times 100$$

Precision indicates how many of the recommended links are relevant to the user.

2. **Recall** is the ratio of correctly recommended links to the total number of relevant links in the test dataset (true positives + false negatives):

$$Recall = \frac{TP}{TP + FN} \times 100$$

Recall measures how many of the relevant links were successfully recommended by the system.

3. **F-Measure** is the harmonic mean of Precision and Recall, providing a balance between the two:

$$F\text{-measure} = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

This metric helps evaluate the trade-off between Precision and Recall, providing a single score that summarizes the model's effectiveness.

To select the most appropriate profile and recommend links to users, we utilize the **K-Nearest Neighbors (KNN)** algorithm. KNN is a non-parametric method used to find the closest profiles based on the similarity between users' behavior sequences. It works as follows:

1. **Profile Matching**: For each user in the test set, we compute the similarity between their session sequences and the profiles generated during the training phase using a similarity metric.
2. **KNN-based Recommendation**: The K most similar user profiles are identified using KNN. The system then aggregates the recommendations from these profiles to suggest the most relevant links for the target user.
3. **Prediction Process**: The KNN model recommends the next possible link for the user based on the most frequent patterns found in the K nearest profiles.

The evaluation metrics will be calculated for each recommendation model configuration to determine the most effective combination of pattern mining algorithms, similarity metrics, and clustering methods. In Table 5, 6 the Precision, Recall, and F-measure values for each model configuration will be presented, helping us identify the most accurate model for dynamic web recommendation.

3.8. Link Recommendation and Website Personalization

In this phase, the system leverages the user profiles selected by the K-Nearest Neighbors (KNN) algorithm from the previous step. Each user profile contains a sequence of previously visited links, forming a behavioral pattern. Based on this sequence, we can dynamically predict the next likely link the user might choose, providing personalized content recommendations in real-time.

3.9. Link Prediction Process

1. Profile-based Link Sequence: Once a user selects a link, the system uses the profile's link sequence to suggest the next possible link. This approach ensures that the recommendations are tailored to the user's browsing behavior, providing personalized and relevant options.

2. Dynamic Recommendations: As the user interacts with the website by clicking on various links, the system continually updates the recommendations. Each new interaction refines the sequence of suggestions based on the patterns discovered in the user's profile, improving personalization.

3. Website Personalization: By continuously predicting the next link the user might visit; the system dynamically personalizes the browsing experience. This method allows the website to adapt to the user's behavior, presenting tailored content and enhancing user engagement.

In the next section, we will evaluate the system's performance in making accurate link predictions by reviewing the Precision, Recall, and F-measure metrics. These results will demonstrate the effectiveness of the dynamic web recommendation model in personalizing the user experience.

4. Results Analysis

The first step in the analysis is to visualize the results of the clustering process performed using the PAM algorithm across both datasets: Zanbil and NASA. Clustering was done using the three similarity measures discussed earlier—Levenshtein Distance, Longest Common Subsequence (LCS), and Needleman-Wunsch (NW). The figures (Figures 4-7) below demonstrate the clustering distribution for each dataset based on these similarity metrics.

Next, we evaluate the performance of the recommendation models generated based on different configurations for each dataset. The evaluation is conducted based on three metrics: Precision, Recall, and F-measure. The results are presented in separate tables for the Zanbil and NASA datasets. Each table summarizes the combination of clustering algorithms, sequential pattern mining algorithms (FP-Growth and SPADE), and similarity measures (Levenshtein, LCS, NW).

Table 5: Model Evaluation Results for Zanbil Dataset

Sequential Pattern Mining Algorithm	Similarity Metric	Precision	Recall	F-Measure
FP-Growth	Levenshtein	0.82	0.75	0.78
	LCS	0.85	0.77	0.81
	NW	0.78	0.71	0.74
SPADE	Levenshtein	0.88	0.81	0.84
	LCS	0.91	0.83	0.87
	NW	0.83	0.77	0.80

Table 6: Model Evaluation Results for NASA Dataset

Sequential Pattern Mining Algorithm	Similarity Metric	Precision	Recall	F-Measure
FP-Growth	Levenshtein	0.76	0.70	0.73
	LCS	0.80	0.74	0.77
	NW	0.72	0.68	0.70
SPADE	Levenshtein	0.84	0.78	0.81
	LCS	0.86	0.80	0.83
	NW	0.79	0.73	0.76

In addition to the tabular results, we provide a visual comparison of the Precision and Recall values for each dataset across the three-similarity metrics (Levenshtein, LCS, and NW) and for both sequential pattern mining algorithms (FP-Growth and SPADE). The x-axis in each chart represents the three-similarity metrics, while the y-axis shows the Precision or Recall scores. Each similarity metric will have two values corresponding to the results of FP-Growth and SPADE, respectively.

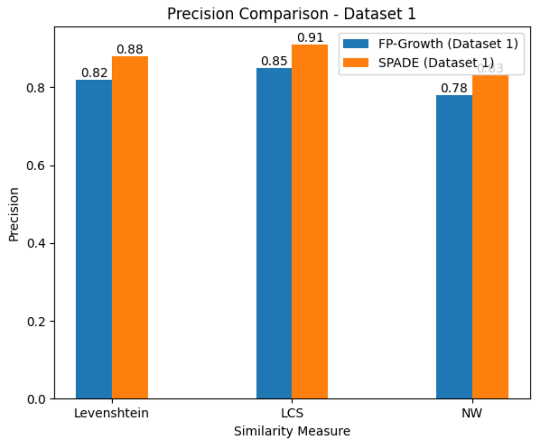

Figure 4: Precision Comparison for the Zanbil

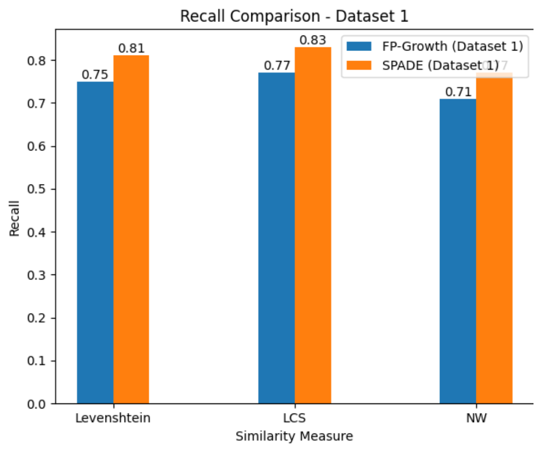

Figure 5: Precision Comparison for the NASA Dataset

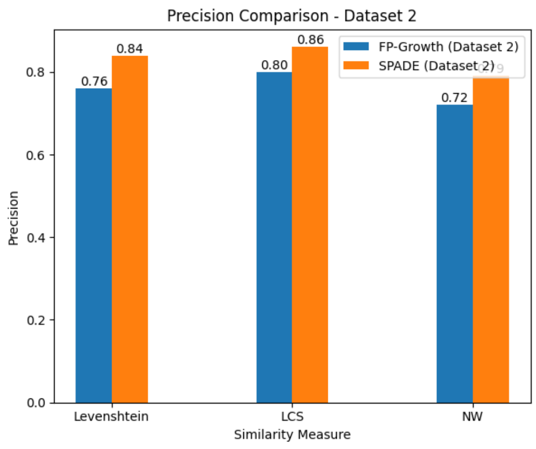

Figure 6: Recall Comparison for the Zanbil Dataset

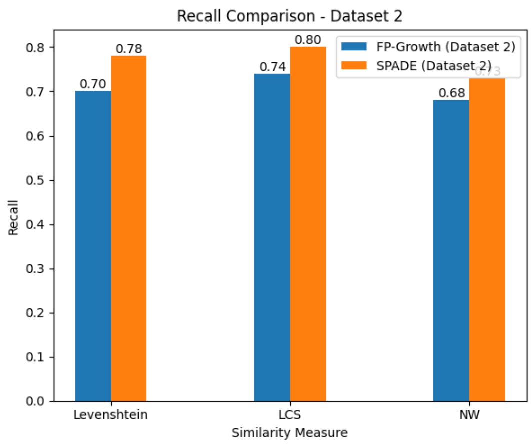

Figure 7: Recall Comparison for the NASA

These visual comparisons provide a clearer understanding of how each similarity metric and pattern mining algorithm performs, helping us determine the most effective configuration for both datasets.

5. Conclusion

In this research, we designed a dynamic hybrid web recommender system aimed at personalizing websites for users by leveraging sequential pattern mining and link sequence similarity detection. By clustering users based on the similarity of their browsing patterns and using various similarity metrics (Levenshtein, LCS, and NW), we were able to build and evaluate personalized recommendation models. The system was tested on two distinct datasets—Zanbil and NASA—to verify its effectiveness. Our results show that the SPADE algorithm combined with the LCS similarity metric consistently outperformed other configurations, providing higher precision and recall scores, making it the most suitable approach for both datasets.

For future work, we suggest enhancing the user profiling process by incorporating Recurrent Neural Networks (RNNs), which are well-suited for modeling sequential data. By using RNNs, the system could potentially better capture the complex, long-term dependencies in users' browsing behaviors, leading to more accurate predictions and an even more effective recommendation engine. Additionally, experimenting with more advanced deep learning models such as LSTM and GRU could further improve recommendation accuracy in dynamic environments.

References

[1] Aggarwal, C. C. (2016). *Recommender Systems: The Textbook.* Springer.

[2] Ricci, F. R. (2015). *Recommender Systems Handbook.* Springer.

[3] Tan, P. N. (2018). *Introduction to Data Mining. Pearson Education.* Pearson.

[4] Cooley, R. M. (1999). Data Preparation for Mining World Wide Web Browsing Patterns. *Knowledge and Information Systems 1*, 5–32.

[5] Liu, B. L. (2018). Sequence-Aware Recommender Systems: An Overview. *ACM Computing Surveys (CSUR), 54(4),* 1-36.

[6] Mobasher, B. D. (2002). Discovery and Evaluation of Aggregate Usage Profiles for Web Personalization. *Data Mining and Knowledge Discovery 6*, 61-82.

[7] Zaki, M. (2020). An efficient algorithm for mining frequent sequences. *Machine Learning, 42,* 31–60.

[8] Navarro, G. (2021). A Guided Tour to Approximate String Matching. *ACM Computing Surveys.*, 33.

[9] Dacrema, M. F. (2021). A Troubling Analysis of Reproducibility and Progress in Recommender Systems Research. *ACM Transactions on Information SystemsVolume 39*, 1–49.

[10] Sun, F. L. (2019). BERT4Rec: Sequential Recommendation with Bidirectional Encoder Representations from Transformer. *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, (pp. 1441–1450).

[11] Zhou, K. W. (2020). S3-Rec: Self-supervised learning for sequential recommendation with mutual information maximization. *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, (pp. 1893–1902).

[12] Ma, R. S. (2020). Memory augmented graph neural networks for sequential recommendation. *Proceedings of the AAAI Conference on Artificial Intelligence 34(04)*, (pp. 5045-5052).

[13] Fan, Y. M. (2021). Continuous-time sequential recommendation with temporal graph collaborative networks. *The 30th ACM International Conference on Information and Knowledge Management*.

[14] Gao, C. G. (2020). A state-preserving RNN framework for sequential recommendation. *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*.

[15] Gong, S. C. (2021). Contrastive self-supervised sequential recommendation with robust augmentation. *Proceedings of the 30th ACM International Conference on Information and Knowledge Management (CIKM).*

[16] Li, S. W. (2020). Time interval aware self-attention for sequential recommendation. *Proceedings of the 13th International Conference on Web Search and Data Mining (WSDM).*

[17] Wang, S. Y. (2021). Denoising implicit feedback for recommendation. *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*.

[18] Xie, Y. Z. (2020). Sequential recommender system based on hierarchical attention network. *Proceedings of the 13th ACM International Conference on Web Search and Data Mining (WSDM).*

[19] S. Latifi, D. J. (2022). Sequential recommendation: A study on transformers , nearest neighbors and sampled metrics. *Inf. Sci.(Ny)., vol. 609*, 660–678.

[20] Zaker, F. (2019). *Harvard Dataverse, V1*. Retrieved from Online Shopping Store - Web Server Logs: https://doi.org/10.7910/DVN/3QBYB5.

[21] N.S. (2018). Retrieved from NASA: https://www.kaggle.com/datasets/souhagaa/nasa-access-log-dataset.