J Inf Process Syst, Vol.17, No.1, pp.107~123, February 2021
https://doi.org/10.3745/JIPS.04.0209

ISSN 1976-913X (Print)
ISSN 2092-805X (Electronic)

JOURNAL OF INFORMATION PROCESSING SYSTEMS JIPS

# Effective and Efficient Similarity Measures for Purchase Histories Considering Product Taxonomy

Yu-Jeong Yang* and Ki Yong Lee*

**Abstract**

In an online shopping site or offline store, products purchased by each customer over time form the purchase history of the customer. Also, in most retailers, products have a product taxonomy, which represents a hierarchical classification of products. Considering the product taxonomy, the lower the level of the category to which two products both belong, the more similar the two products. However, there has been little work on similarity measures for sequences considering a hierarchical classification of elements. In this paper, we propose new similarity measures for purchase histories considering not only the purchase order of products but also the hierarchical classification of products. Unlike the existing methods, where the similarity between two elements in sequences is only 0 or 1 depending on whether two elements are the same or not, the proposed method can assign any real number between 0 and 1 considering the hierarchical classification of elements. We apply this idea to extend three existing representative similarity measures for sequences. We also propose an efficient computation method for the proposed similarity measures. Through various experiments, we show that the proposed method can measure the similarity between purchase histories very effectively and efficiently.

**Keywords**

Hierarchical Classification, Purchase History, Sequence Similarity, Similarity Measure

## 1. Introduction

In an online shopping site or offline store, products purchased by each customer over time form the purchase history of the customer. Like a purchase history, where products purchased by a customer are listed in the order of purchase, the data in which items are listed in some order is called a sequence. Because purchase histories, which are a kind of sequences, contain the behavioral characteristics and consumption patterns of customers, many companies analyze these purchase histories to develop marketing strategies or increase their sales. In particular, companies can find customers with similar purchase histories for product recommendation or group customers with similar purchase histories to run targeted marketing [1-4].

Meanwhile, in most retailers, products have a product taxonomy, which represents a hierarchical classification of products, as shown in Fig. 1. Given a product taxonomy, the lower the level of the category to which two products both belong, the more similar the two products. For example, in Fig. 1, “Coke” and “Sprite” both belong to the same group “Beverage”, while “Coke” and “Candy” belong to different groups but belong to the same division “Food”. In this case, “Coke” and “Sprite” are more similar than “Coke” and


Manuscript received March 19, 2020; first revision June 22, 2020; accepted July 3, 2020.
**Corresponding Author:** Ki Yong Lee (kiyonglee@sookmyung.ac.kr)
* Dept. of Computer Science, Sookmyung Women’s University, Seoul, Korea (diddbwjd96@sookmyung.ac.kr, kiyonglee@sookmyung.ac.kr)

"Candy" because "Coke" and "Sprite" belong to the same category with a lower level than "Coke" and "Candy". In other words, "Coke" and "Sprite" are closer in the hierarchical classification tree than "Coke" and "Candy".

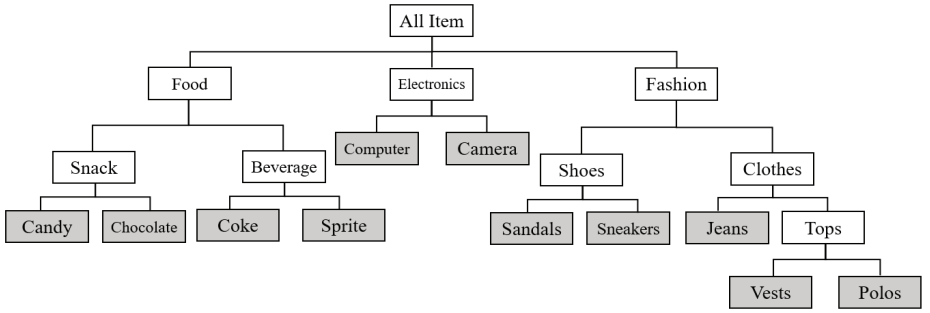


**Fig. 1.** Example of a hierarchical classification of products.

So far, many studies have been conducted on measuring the similarity between sequences. However, most of them do not consider a hierarchical classification of elements in sequences. For example, in Fig. 1, most existing measures do not distinguish the difference between "Coke" and "Sprite" and the difference between "Coke" and "Candy". As a result, they may not be able to accurately distinguish between similar and dissimilar purchase histories. For example, consider three purchase histories $s_1$ = <"Coke", "Sandals">, $s_2$ = <"Sprite", "Sneakers">, and $s_3$ = <"Candy", "Camera">. In this case, although $s_1$, $s_2$, and $s_3$ consist of completely different products, we can see that $s_1$ and $s_2$ are more similar than $s_1$ and $s_3$ because "Coke" and "Sprite" are more similar than "Coke" and "Candy", and "Sandals" and "Sneakers" are more similar than "Sandals" and "Camera" considering the hierarchical classification of products. However, most existing similarity measures cannot distinguish the difference between $s_1$ and $s_2$ and the difference between $s_1$ and $s_3$ because they only consider whether products composing the purchase histories match or not. As a result, most existing similarity measures produce the same similarity value for $s_1$ and $s_2$ and $s_1$ and $s_3$. Therefore, in this paper, we propose new similarity measures for purchase histories considering not only the purchase order of products but also the hierarchical classification of products. Even if two purchase histories consist of completely different products, the proposed method considers the two purchase histories to be similar if their products are close in the hierarchical classification tree. Unlike the existing methods, where the similarity between two elements in sequences is only 0 or 1 depending on whether two elements are the same or not, the proposed method can assign any real number between 0 and 1 considering the hierarchical classification of elements. We apply this idea to extend three existing representative similarity measures for sequences, i.e., the Levenshtein distance [5], the Needleman-Wunsch similarity [6], and the dynamic time warping (DTW) distance [7]. Besides the three similarity measures, the proposed method can be generally applied to other similarity measures where the similarity between two elements is measured only as 0 or 1.

We also propose an efficient computation method for the proposed similarity measures. The proposed computation method uses a segment tree to compute the similarity between two products in a hierarchical classification very efficiently. This allows the proposed measures to be used very efficiently even when the number of purchase histories to be compared is very large. Through various experiments, we show that the proposed measures can measure the similarity between purchase histories very effectively and efficiently, when we are given a hierarchical classification of products. Our contributions are summarized

as follows:

- For the first time, we propose new similarity measures for purchase histories that consider not only the purchase order of products but also the hierarchical classification of products.
- We extend existing representative similarity measures for sequences to make them consider the hierarchical classification of products.
- We also propose an efficient computation method for the proposed similarity measures, which allows the proposed measures to be used for a large number of purchase histories.

The rest of the paper is organized as follows. Section 2 reviews the related studies, and Section 3 describes the proposed measures in detail. Section 4 presents the performance evaluation of the proposed measures. Finally, Section 5 summarizes and concludes the paper.

# 2. Related Work

## 2.1 Sequence Similarity Measures

A sequence is an ordered list of elements. Examples of sequences include DNA sequences, protein sequences, web surfing histories, user activities, and product purchase histories [8–10]. To measure the similarity between sequences, we need to consider the order of elements in sequences [11]. Existing similarity measures for sequences can be largely divided into three categories, according to how the similarity between sequences is calculated:

**Edit-based similarity measures**

These measures compute the similarity between sequences by counting the number of edit operations required to make one sequence the same as the other sequence. The Levenshtein distance [5], also known as the edit distance, is a representative similarity measure for strings, which can be seen as sequences of characters. The Levenshtein distance is defined as the minimum number of edit operations required to transform one string to the other. The edit operations used to change a string include insertions, deletions, and substitutions. Each edit operation is performed on one character. Each time a character is added, deleted, or substituted in a string, 1 is added to the distance. Therefore, these measures do not consider the hierarchical classification of elements in sequences.

**Alignment-based similarity measures**

These measures measure the similarity between sequences as the cost of aligning or moving the sequences to match the same parts in both sequences. These measures are mainly used in bioinformatics to compare protein sequences or nucleic acid sequences [12]. They are also divided into local and global alignment methods depending on the scope of finding similar parts between two sequences. The Needleman-Wunsch similarity [6] is a representative global alignment method. This distance inserts gaps to align two sequences so that the similarity between them is maximized. It then calculates the distance by adding the cost of inserting gaps and the cost of mismatch between elements. However, these methods also do not consider the hierarchical classification of elements in sequences.

**Matching-based similarity measures**

These measures compute the similarity between sequences as the cost of matching each element in a sequence to one or more elements in the other sequence. The DTW distance [7] is a representative

matching-based similarity measure. The DTW distance is originally proposed to measure the similarity between two time series sampled at different rate. Fig. 2 shows an example of computing the similarity between two sequences $<a_1, a_2, \ldots, a_7>$ and $<b_1, b_2, \ldots, b_7>$ using the Euclidean distance and DTW distance, respectively. In the Euclidean distance, each element $a_i$ is matched only to $b_i$, which is the element in the same position in the other sequence. On the contrary, the DTW distance can match $a_i$ to any number of elements at any position in the other sequence. The DTW distance measures the distance between two sequences by computing the minimum cost of matching elements in a sequence to elements in the other sequence. Therefore, it is known that the DTW distance is very effective for measuring the distance between sequences of different lengths [13,14]. However, the DTW distance does not consider the hierarchical classification of elements in sequences when matching them.


(a) (b)

**Fig. 2.** Example of computing the DTW distance: (a) Euclidean matching and (b) DTW matching.

## 2.2 Segment Tree

Unlike the previous measures that do not consider the similarity between elements in sequences, the proposed measures consider the similarity between products in purchase histories considering the hierarchical classification of products. In order to quickly compute the similarity between products, the proposed method uses a segment tree [15], whose example is shown in Fig. 3. In a segment tree, as shown in Fig. 3, each node maintains some information about an interval or segment (e.g., sum, minimum, maximum). Furthermore, the interval of a parent node is the sum of the intervals of its child nodes. For example, in Fig. 3, each node in the segment tree maintains a sum value about an interval, which corresponds to the sum of sum values of its child nodes. The segment tree is known to be efficient in finding the minimum value for any given interval because each node stores information about a sub-interval computed in advance [16,17]. In this paper, we construct a segment tree to store information about the hierarchical classification of products and use it to quickly compute the distance between products.

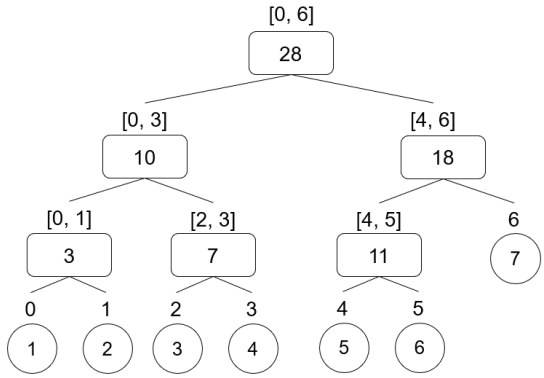


**Fig. 3.** Example of a segment tree.

# 3. Proposed Method

## 3.1 Overview

In this paper, we propose new similarity measures for purchase histories considering the hierarchical classification of products (i.e., a product taxonomy). We also propose an efficient computation method for the proposed similarity measures. A purchase history is a sequence of products listed in order of purchase. Unlike the previous measures for sequences, the proposed measures consider both the order of products and the similarity between products to compute the similarity between purchase histories. For this purpose, we extend the existing representative similarity measures for sequences (i.e., the Levenshtein distance, the Needleman-Wunsch similarity, and the DTW distance) to consider the hierarchical classification of products. In the next subsections, we first present the basic idea of the proposed measures and then describe how to extend the existing measures to consider a product taxonomy.

## 3.2 Basic Idea

As mentioned in Section 2, the previous similarity measures for sequences assign the distance between two products as 0 if they are identical and 1 if they are different. Thus, they do not consider the hierarchical classification of products. On the other hand, by considering the hierarchical classification of products, the proposed measures assign a value between 0 and 1 as the distance between two products. Let *dist*($x$, $y$) be the distance between two products $x$ and $y$. Then, in the previous measures, *dist*($x$, $y$) is defined as follows:

$$dist(x,y) = \begin{cases} 0, & if\ x = y \\ 1, & if\ x \neq y \end{cases} \quad (1)$$

Let $T$ be a tree representing a given hierarchical classification, whose example is shown in Fig. 1. We call $T$ a hierarchical classification tree. Given $T$, we define *dist*($x$, $y$) as follows:

$$dist(x,y) = \frac{shortestPathLen(x,y,T)}{longestPathLen(T)} \quad (2)$$

Here, *shortestPathLen*($x$, $y$, $T$) represents the length of the shortest path between product $x$ and $y$ in $T$, and *longestPathLen*($T$) represents the length of the path between the two leaf nodes that are the farthest within $T$. *dist*($x$, $y$) has a value between 0 and 1. *dist*($x$, $y$) has a smaller value as two products $x$ and $y$ are closer in $T$, i.e., the lower the level of the category to which two products both belong. On the other hand, *dist*($x$, $y$) has a larger value as two products $x$ and $y$ are farther in $T$, i.e., the higher the level of the category to which two products both belong. For example, in Fig. 1, *dist*("Coke", "Sprite") = 2/7 because *shortestPathLen*("Coke", "Sprite", $T$) = 2 and *longestPathLen*($T$) = 7. Also, *dist*("Coke", "Candy") = 4/7 because *shortestPathLen*("Coke", "Candy", $T$) = 4 and *longestPathLen*($T$) = 7. Therefore, we can see that *dist*("Coke", "Sprite") < *dist*("Coke", "Candy") as we expected.

Based on this idea, we can extend the existing similarity measures for sequences that assign only 0 or 1 as the distance between two products to reflect the hierarchical classification of products. In the next section, we describe how to extend the three existing representative similarity measures for sequences, i.e., the Levenshtein distance, the Needleman-Wunsch similarity, and the DTW distance.

## 3.3 Extensions of Existing Similarity Measures

Suppose we are given two purchase histories $s_1 = \langle x_1, x_2, \cdots, x_n \rangle$ and $s_2 = \langle y_1, y_2, \cdots, y_m \rangle$, where $x_i$ and $y_i$ represent the $i$th purchased product in $s_1$ and $s_2$, respectively. Given $s_1$ and $s_2$, the Levenshtein distance, the Needleman-Wunsch similarity, and the DTW distance all create two-dimensional array $M$ of $n \times m$ size as shown in Fig. 4. Initially, all elements $M[i][j]$ are set to the specified values depending on the distance. Then, these distances update each element $M[i][j]$ by using its neighboring three elements $M[i\text{-}1][j\text{-}1]$, $M[i][j\text{-}1]$, and $M[i\text{-}1][j]$. Each distance uses a different formula to obtain $M[i][j]$ from $M[i\text{-}1][j\text{-}1]$, $M[i][j\text{-}1]$, and $M[i\text{-}1][j]$, which will be described in detail below. After obtaining all $M[i][j]$ for $i = 1, 2\ldots, n$ and $j = 1, 2, \ldots, m$, $M[n][m]$ is returned as the distance between $s_1$ and $s_2$. Now we describe how to extend each distance to consider the hierarchical classification of products.

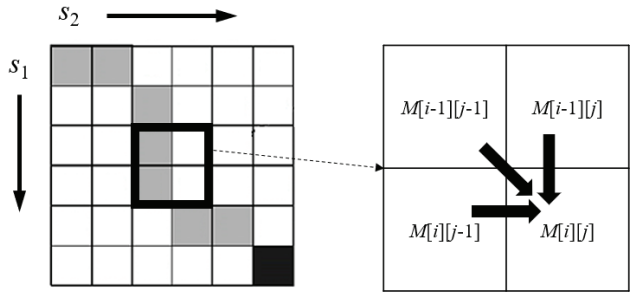


**Fig. 4.** Computation of the distance between two purchase histories $s_1$ and $s_2$.

**Extension of the Levenshtein distance**

In the Levenshtein distance, $M[i][0]$ and $M[0][j]$ are initialized to $M[i][0] = i$ and $M[0][j] = j$, respectively, for $i = 1, 2\ldots, n$ and $j = 1, 2, \ldots, m$. $M[i][j]$ is then updated using the following equation:

$$M[i][j] = \min(M[i][j\text{-}1] + 1, M[i\text{-}1][j] + 1, M[i\text{-}1][j\text{-}1] + 1_{(xi \neq yj)}), \tag{3}$$

where $1_{(xi \neq yj)}$ is the indicator function equal to 0 when $x_i = y_j$ and equal to 1 otherwise. In other words, the Levenshtein distance considers the distance between $x_i$ and $y_j$ to 0 if $x_i = y_j$, and 1 if $x_i \neq y_j$. Thus, we extend Eq. (3) to allow the distance between $x_i$ and $y_j$ to have a value between 0 and 1 as follow:

$$M[i][j] = \min(M[i][j\text{-}1] + 1, M[i\text{-}1][j] + 1, M[i\text{-}1][j\text{-}1] + dist(x_i, y_j)), \tag{4}$$

where $dist(x_i, y_j)$ is defined as Eq. (2), which corresponds to the distance between $x_i$ and $y_j$ in the hierarchical classification tree $T$. In this way, we can extend the Levenshtein distance to consider the hierarchical classification of products.

**Extension of the Needleman-Wunsch similarity**

In the Needleman-Wunsch similarity, $M[i][0]$ and $M[0][j]$ are initialized to $M[i][0] = -i$ and $M[0][j] = -j$, respectively, for $i = 1, 2\ldots, n$ and $j = 1, 2, \ldots, m$. $M[i][j]$ is then updated using the following equation:

$$M[i][j] = \max(M[i][j\text{-}1] - 1, M[i\text{-}1][j] - 1, M[i\text{-}1][j\text{-}1] + s(x_i, y_j)), \tag{5}$$

where $s(x_i, y_j)$ is a match score function that returns 1 if $x_i = y_j$ and 0 otherwise. In other words, the Needleman-Wunsch similarity assigns 1 as the match score between $x_i$ and $y_j$ if $x_i = y_j$, and 0 if $x_i \neq y_j$. Thus, we extend Eq. (5) to allow the match score between $x_i$ and $y_j$ to have a value between 0 and 1 as follow:

$$M[i][j] = \min(M[i][j\text{-}1] + 1, M[i\text{-}1][j] + 1, M[i\text{-}1][j\text{-}1] + (1 - dist(x_i, y_j))), \quad (6)$$

where $dist(x_i, y_j)$ is defined as Eq. (2), which corresponds to the distance between $x_i$ and $y_j$ in the hierarchical classification tree $T$. In this way, we can extend the Needleman-Wunsch similarity to consider the hierarchical classification of products.

**Extension of the DTW distance**

In the DTW distance, all elements $M[i][j]$ are initialized to $\infty$ for $i = 1, 2\ldots, n$ and $j = 1, 2, \ldots, m$. $M[i][j]$ is then updated using the following equation:

$$M[i][j] = d(x_i, y_j) + \min(M[i][j\text{-}1], \mathrm{M}[i\text{-}1][j], M[i\text{-}1][j\text{-}1]), \quad (7)$$

where $d(x_i, y_j) = |x_i - y_j|$ assuming that $x_i$ and $y_j$ are numbers. Although the DTW allows a value other than 0 and 1 as the distance between $x_i$ and $y_j$, the distance is defined only for numbers. Thus, we extend Eq. (7) to allow the distance between two products $x_i$ and $y_j$ as follow:

$$M[i][j] = dist(x_i, y_j) + \min(M[i][j\text{-}1], \mathrm{M}[i\text{-}1][j], M[i\text{-}1][j\text{-}1]), \quad (8)$$

where $dist(x_i, y_j)$ is defined as Eq. (2), which corresponds to the distance between $x_i$ and $y_j$ in the hierarchical classification tree $T$. In this way, we can extend the DTW distance to consider the hierarchical classification of products.

## 3.4 Efficient Computation Method for Similarity Measures

In this subsection, we propose an efficient computation method for the proposed similarity measures presented in Section 3.3. We first describe a naïve method and then present an improved version that eliminates the disadvantage of the naïve method.

### 3.4.1 Naïve method

In order to compute the proposed similarity measures, we need to compute $dist(x_i, y_j)$ repeatedly for all pairs of $x_i$ and $y_j$ in $s_1$ and $s_2$. Therefore, as $s_1$ and $s_2$ become longer, the computation cost increases. When we compute $dist(x_i, y_j)$, since $longestPathLen(T)$ is a fixed value in $dist(x_i, y_j)$, it is very important to efficiently obtain $shortestPathLen(x_i, y_j, T)$ to reduce the computation cost. A simple way to obtain $shortestPathLen(x_i, y_j, T)$ for given $x_i$ and $y_j$ is as follows: We first find the path from the root node of $T$ to $x_i$ and $y_j$, respectively. We then compare the nodes in the two paths one by one to count the number of different nodes. For example, Fig. 5 shows the process of computing *shortestPathLen*("Sandals", "Vests", $T$) using this naïve method. The two paths in Fig. 5 represent the path from the root node of $T$ to "Sandals" and "Vests", respectively. The naïve method then compares the two paths one by one from the root node. This process continues until the two nodes do not match. Finally, the number of

mismatched nodes in the two paths represents the length of the shortest path of the two products. Therefore, in Fig. 5, *shortestPathLen*("Sandals", "Vests", $T$) = 2 + 3 = 5.

However, the naïve method has the disadvantage that the cost of computing *shortestPathLen*($x_i$, $y_j$, $T$) increases as the size of $T$ increases, because the length of a path from the root node of $T$ to a leaf node increases. In particular, this problem becomes more serious as the lengths of $s_1$ and $s_2$ increase, because we need to compute *dist*($x_i$, $y_j$) repeatedly for all pairs of $x_i$ and $y_j$ in $s_1$ and $s_2$. Therefore, we propose an efficient computation method to minimize this problem.

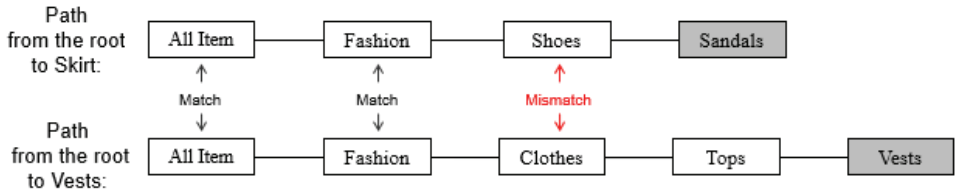


**Fig. 5.** Example of the naïve method to compute *shortestPathLen*($x_i$, $y_j$, $T$).

### 3.4.2 Proposed computation method

To minimize the problem of the naïve method, we propose a very efficient method to compute *shortestPathLen*($x_i$, $y_j$, $T$). The proposed computation method uses a segment tree, which we described in Section 2.2. Before computing *shortestPathLen*($x_i$, $y_j$, $T$), the proposed method traverses the hierarchical classification tree $T$ once to build a segment tree, which is then used for computing *shortestPathLen*($x_i$, $y_j$, $T$).

Fig. 6 shows an example of a segment tree built by the proposed computation method. In order to build a segment tree, we first assign a product number to each leaf node of the hierarchical classification tree $T$ in the order of 0, 1, … from left to right (e.g., Sprite = 0, Coke = 1, Candy = 2, …). In the segment tree built from $T$, each node is associated with an interval $[i, j]$ of product numbers and stores *depth*[*LCA*($i$, $j$, $T$)], which is the depth of the lowest common ancestor (LCA) of product $i$ and product $j$ in $T$. Here, the depth of a node is the number of edges from the root node to the node. The LCA of two products is the node with the lowest level that contains both products. For example, in Fig. 6, the node 'Food' in the segment tree is associated with the interval [0, 2], which represents the range of product numbers for

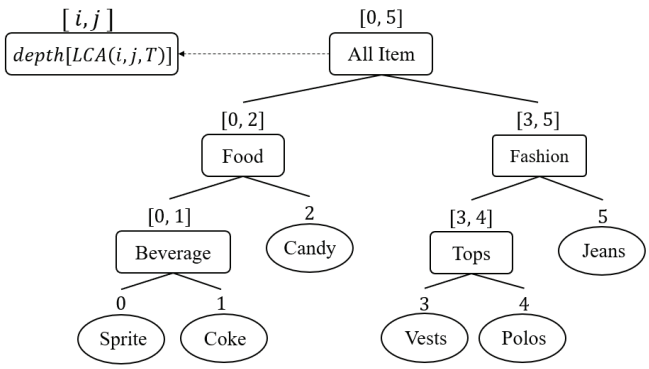


**Fig. 6.** Example of a segment tree built by the proposed computation method.

products belonging to the "Food" category. The node "Food" also stores $depth[LCA(0, 2, T)]$, which is 1 because the depth of the LCA of product 0 (i.e., Sprite) and product 2 (i.e., Candy) is 1 in $T$.

In the naïve method described in Section 3.3.1, the final step in computing $shortestPathLen(x_i, y_j, T)$ is to compare the paths from the root node of $T$ to $x_i$ and to $y_j$ to find the last matching node. In fact, the last matching node corresponds to the LCA of $x_i$ and $y_j$. Therefore, the proposed method uses the segment tree to quickly find the depth of the LCA of $x_i$ and $y_j$ and uses it to compute $shortestPathLen(x_i, y_j, T)$. More specifically, we compute $shortestPathLen(x_i, y_j, T)$ as follows:

$$shortestPathLen(x_i, y_j, T) = depth[x_i] + depth[y_j] - 2 \times depth[LCA(p(x_i), p(y_j), T)], \quad (9)$$

where $p(x_i)$ and $p(y_j)$ are the product number of $x_i$ and $y_j$, respectively. The length of the path from $x_i$ to $LCA(p(x_i), p(y_j), T)$ is the $depth[x_i] - depth[LCA(p(x_i), p(y_j), T)]$, and the length of the path from $LCA(p(x_i), p(y_j), T)$ to $y_j$ is the $depth[y_j] - depth[LCA(p(x_i), p(y_j), T)]$, so the length of the shortest path between two products $x_i$ and $y_j$ can be expressed by Eq. (9).

The proposed method first generates two one-dimensional arrays by traversing the product classification tree $T$. The first array stores the nodes in order of visit and the second array stores the depth of each node. In the hierarchical classification tree $T$, let us assume that the node ID of a parent node is always smaller than those of its child nodes. Then, we can guarantee that the LCA of $x_i$ and $y_j$ in $T$ is the node with the smallest node ID among the nodes in the traversal path from $x_i$ to $y_j$. Therefore, we can build the segment tree using these two arrays, which store the nodes in order of visit and the depth of each node, respectively.

When computing $shortestPathLen(x_i, y_j, T)$ defined in Eq. (9), we can use the segment tree to quickly obtain the value of $depth[LCA(p(x_i), p(y_j), T)]$. Therefore, $shortestPathLen(x_i, y_j, T)$ can be easily obtained without finding the paths from the root node of $T$ to $x_i$ and $y_j$ and then comparing the nodes in the two paths one by one. Consequently, the cost of computing $dist(x_i, y_j)$ is greatly reduced. We will present the performance evaluation of the proposed computation method in Section 4.2.

# 4. Performance Evaluation

In this section, we present the performance evaluation of the proposed measures. First, we evaluated the effectiveness of the proposed similarity measures presented in Section 3.3. We then evaluate the efficiency of the proposed computation method presented in Section 3.4. The proposed method was implemented using Python, and the experiments were conducted on a PC running Windows 10 with Intel Core i7-5829 3.3 GHz CPU and 8 GB memory. To construct a hierarchical classification tree for products, we referred to the actual product classification system of Amazon.com, a representative online shopping site.

## 4.1 Effectiveness Evaluation

First, we evaluated whether the proposed similarity measures, which consider the product taxonomy, is more effective in measuring the similarity between purchase histories, compared to the existing measures. Fig. 7 shows the set of purchase histories used in this experiment. The 9 purchase histories (i.e., $s_1$, $s_2$, …, $s_9$) are divided into three groups Group 1, Group 2, and Group 3, each of which consists

of similar purchase histories. Group 1 is from Amazon customers' real purchase histories, and Groups 2 and 3 are synthetic purchase histories. Each product in Group 1 represents an actual product sold on Amazon, which is shown in Table 1.

Note that purchase histories in each group have products with similar categories in a similar order. We measured the distances between all pairs of $s_1$, $s_2$, …, $s_9$ using the three existing similarity measures (i.e., the Levenshtein distance, Needleman-Wunsch distance, and DTW distance) and the proposed extended versions of them, and compared the results.

$$
\begin{aligned}
&Group\ 1 = \{s_1, s_2, s_3\}\\
&Group\ 2 = \{s_4, s_5, s_6\}\\
&Group\ 3 = \{s_7, s_8, s_9\}\\
&\\
&s_1 =< Candy, Computer, Sprite >\\
&s_2 =< Chocolate, Camera, Coke >\\
&s_3 =< Cookie, Bread, TV, Water, Tea >\\
&\\
&s_4 =< Sneakers, Sprite, Skirt, Cake >\\
&s_5 =< Loafer, Sandals, Sprite, Coke, Vests, Polos, Chocolate >\\
&s_6 =< Sandals, Coke, Shirt, Snack >\\
&\\
&s_7 =< Cheese, Beef, Bracelet, Skirt, Lighting >\\
&s_8 =< Milk, Yogurt, Pork, Watch, Necklace, Camcorders, Lenses >\\
&s_9 =< Butter, Chicken, Earrings, Glasses, Computer, Video >
\end{aligned}
$$

**Fig. 7.** The set of purchase histories used to evaluate the effectiveness of the proposed measures.

**Table 1.** Actual products in Group 1

| Product in Group 1 | Actual product |
|---|---|
| Bread | King Arthur Flour, Pumpkin Bread + Muffin Mix, Gluten Free, 12 oz |
| Camera | Canon EOS 4000D DSLR Camera w/Canon EF-S 18–55 mm F/3.5–5.6 III Zoom Lens + Case + 32 GB SD Card (15pc Bundle) |
| Candy | SKITTLES, STARBURST & LIFE SAVERS Valentine's Day Candy Fun Size Variety Mix 22.7-Ounces 80 Pieces |
| Chocolate | Betty Crocker Hershey's Milk Chocolate Frosting, Gluten Free, 16 oz |
| Coke | Diet Coke Soda Soft Drink, 12 fl oz, 12 Pack |
| Computer | HP 21.5-Inch All-in-One Computer, AMD A4-9125, 4 GB RAM, 1 TB Hard Drive, Windows 10 (22-c0010, White) |
| Cookie | Pepperidge Farm, Classic Cookie Favorites, 13.25 oz |
| Sprite | Sprite Lemon Lime Soda Soft Drinks, 12 fl oz, 12 Pack |
| Tea | Ahmad Tea Twelve Teas Variety Gift Box, 60 Foil Enveloped Teabags |
| TV | TCL 32S325 32 Inch 720p Roku Smart LED TV (2019) |
| Water | Nestle Pure Life Purified Water, 8 fl oz. Plastic Bottles (24 Pack) |

Fig. 8 shows the distances between $s_1$, $s_2$, …, $s_9$ measured by the Levenshtein distance and the proposed extended version. The original Levenshtein distance computes the similarity between two sequences based on how many products match. Therefore, even if two sequences have similar products in the same order, it considers these two sequences to be completely different. On the other hand, the proposed method measures the similarity between two purchase histories more accurately by considering whether they are composed of products with similar categories. For example, the original Levenshtein distance measures the distance between $s_1$ and $s_2$ as 3 because they consist of completely different products.

However, “Candy” and “Chocolate”, “Compute” and “Camera”, and “Sprite” and “Coke” are similar to each other, respectively, because they belong to the same low-level category. If we use the proposed measure, the distance between $s_1$ and $s_2$ is measured as 0.75, which is more accurate. Note that, in Fig. 8(b), the purchase histories belonging to the same group have smaller distances (i.e., those in the dotted boxes) compared to other purchase histories.

Fig. 9 shows the similarities between $s_1$, $s_2$, …, $s_9$ measured by the Needleman-Wunsch similarity and the proposed extended version. Like the Levenshtein distance, the Needleman-Wunsch similarity computes the similarity between two sequences based on how many elements are identical. Therefore, also in the case of the Needleman-Wunsch distance, if two sequences consist of totally different products, they are considered completely different sequences, even if their products have similar categories. However, the proposed measure computes the similarity between two purchase histories more accurately by considering the hierarchical classification of products. In Fig. 9(b), we can see that the purchase histories belonging to the same group have higher similarities (i.e., those in the dotted boxes) compared to other purchase histories.

| | $s_1$ | $s_2$ | $s_3$ | $s_4$ | $s_5$ | $s_6$ | $s_7$ | $s_8$ | $s_9$ |
|---|---|---|---|---|---|---|---|---|---|
| $s_1$ | 0 | 3 | 6 | 4 | 6 | 4 | 5 | 7 | 6 |
| $s_2$ | 3 | 0 | 6 | 4 | 6 | 4 | 5 | 7 | 6 |
| $s_3$ | 6 | 6 | 0 | 5 | 7 | 6 | 6 | 6 | 6 |
| $s_4$ | 4 | 4 | 5 | 0 | 5 | 4 | 5 | 7 | 6 |
| $s_5$ | 6 | 6 | 7 | 5 | 0 | 6 | 7 | 7 | 7 |
| $s_6$ | 4 | 4 | 6 | 4 | 6 | 0 | 5 | 7 | 5 |
| $s_7$ | 5 | 5 | 6 | 5 | 7 | 5 | 0 | 7 | 6 |
| $s_8$ | 7 | 7 | 6 | 7 | 7 | 7 | 7 | 0 | 7 |
| $s_9$ | 6 | 6 | 6 | 6 | 7 | 5 | 6 | 7 | 0 |

(a)

| | $s_1$ | $s_2$ | $s_3$ | $s_4$ | $s_5$ | $s_6$ | $s_7$ | $s_8$ | $s_9$ |
|---|---|---|---|---|---|---|---|---|---|
| $s_1$ | 0 | 0.75 | 4.875 | 3 | 5.5 | 2.875 | 4.25 | 6 | 5 |
| $s_2$ | 0.75 | 0 | 4.875 | 3 | 5.5 | 2.875 | 4.25 | 6 | 5 |
| $s_3$ | 4.875 | 4.875 | 0 | 4.5 | 5.625 | 5.125 | 5.125 | 5.625 | 5.25 |
| $s_4$ | 3 | 3 | 4.5 | 0 | 3.5 | 1.375 | 4.125 | 5.875 | 5 |
| $s_5$ | 5.5 | 5.5 | 5.625 | 3.5 | 0 | 4.125 | 5.75 | 5.5 | 5.75 |
| $s_6$ | 2.875 | 2.875 | 5.125 | 1.375 | 4.125 | 0 | 4 | 5.75 | 4.625 |
| $s_7$ | 4.25 | 4.25 | 5.125 | 4.125 | 5.75 | 4 | 0 | 3.5 | 2.75 |
| $s_8$ | 6 | 6 | 5.625 | 5.875 | 5.5 | 5.75 | 3.5 | 0 | 3 |
| $s_9$ | 5 | 5 | 5.25 | 5 | 5.75 | 4.625 | 2.75 | 3 | 0 |

(b)

**Fig. 8.** The distances measured by the Levenshtein distance (a) and the proposed extension (b).

| | $s_1$ | $s_2$ | $s_3$ | $s_4$ | $s_5$ | $s_6$ | $s_7$ | $s_8$ | $s_9$ |
|---|---|---|---|---|---|---|---|---|---|
| $s_1$ | 3 | 0 | -3 | -1 | -3 | -1 | -2 | -4 | -3 |
| $s_2$ | 0 | 3 | -3 | -1 | -3 | -1 | -2 | -4 | -3 |
| $s_3$ | -3 | -3 | 6 | -1 | -1 | -2 | -1 | 0 | 0 |
| $s_4$ | -1 | -1 | -1 | 4 | -1 | 0 | -1 | -3 | -2 |
| $s_5$ | -3 | -3 | -1 | -1 | 7 | -2 | -2 | 0 | -1 |
| $s_6$ | -1 | -1 | -2 | 0 | -2 | 4 | -1 | -3 | -1 |
| $s_7$ | -2 | -2 | -1 | -1 | -2 | -1 | 5 | -2 | -1 |
| $s_8$ | -4 | -4 | 0 | -3 | 0 | -3 | -2 | 7 | -1 |
| $s_9$ | -3 | -3 | 0 | -2 | -1 | -1 | -1 | -1 | 6 |

(a)

| | $s_1$ | $s_2$ | $s_3$ | $s_4$ | $s_5$ | $s_6$ | $s_7$ | $s_8$ | $s_9$ |
|---|---|---|---|---|---|---|---|---|---|
| $s_1$ | 3 | 2.25 | -1.875 | 0 | -2.5 | 0.125 | -1.25 | -3 | -2 |
| $s_2$ | 2.25 | 3 | -1.875 | 0 | -2.5 | 0.125 | -1.25 | -3 | -2 |
| $s_3$ | -1.875 | -1.875 | 6 | -0.5 | 0.375 | -1.125 | -0.125 | 0.375 | 0.75 |
| $s_4$ | 0 | 0 | -0.5 | 4 | 0.5 | 2.625 | -0.125 | -1.875 | -1 |
| $s_5$ | -2.5 | -2.5 | 0.375 | 0.5 | 7 | -0.125 | -0.75 | 1.5 | 0.25 |
| $s_6$ | 0.125 | 0.125 | -1.125 | 2.625 | -0.125 | 4 | 0 | -1.75 | -0.625 |
| $s_7$ | -1.25 | -1.25 | -0.125 | -0.125 | -0.75 | 0 | 5 | 1.5 | 2.25 |
| $s_8$ | -3 | -3 | 0.375 | -1.875 | 1.5 | -1.75 | 1.5 | 7 | 3 |
| $s_9$ | -2 | -2 | 0.75 | -1 | 0.25 | -0.625 | 2.25 | 3 | 6 |

(b)

**Fig. 9.** The similarities measured by the Needleman-Wunsch similarity (a) and the proposed extension (b).

Fig. 10 shows the distances between $s_1$, $s_2$, …, $s_9$ measured by the DTW distance and the proposed extended version. The original DTW distance computes the distance between two purchase histories by adding 1 for each mismatched pair of products. Thus, the original DTW distance cannot differentiate

products with similar categories and those with dissimilar categories. However, the proposed extended version of the DTW distance computes the distance between two purchase histories more effectively by considering the product taxonomy, so it determines that the purchase histories belonging to the same group are more similar than others. In Fig. 10(b), we can see that that the purchase histories belonging to the same group have smaller distances (i.e., those in the dotted boxes) compared to other purchase histories.

From the results of the experiments, we can see that the proposed measures compute the distance (or similarity) between purchase histories more accurately than the existing similarity measures by considering the product taxonomy. In particular, the proposed measures assign a smaller distance to two purchase histories even if they consist of completely different products as long as they are composed of products with similar categories.

| | $s_1$ | $s_2$ | $s_3$ | $s_4$ | $s_5$ | $s_6$ | $s_7$ | $s_8$ | $s_9$ |
|---|---|---|---|---|---|---|---|---|---|
| $s_1$ | 0 | 3 | 6 | 4 | 6 | 4 | 5 | 7 | 6 |
| $s_2$ | 3 | 0 | 6 | 4 | 6 | 4 | 5 | 7 | 6 |
| $s_3$ | 6 | 6 | 0 | 5 | 7 | 6 | 6 | 6 | 6 |
| $s_4$ | 4 | 4 | 5 | 0 | 5 | 4 | 5 | 7 | 6 |
| $s_5$ | 6 | 6 | 7 | 5 | 0 | 6 | 7 | 7 | 7 |
| $s_6$ | 4 | 4 | 6 | 4 | 6 | 0 | 5 | 7 | 5 |
| $s_7$ | 5 | 5 | 6 | 5 | 7 | 5 | 0 | 7 | 6 |
| $s_8$ | 7 | 7 | 6 | 7 | 7 | 7 | 7 | 0 | 7 |
| $s_9$ | 6 | 6 | 6 | 6 | 7 | 5 | 6 | 7 | 0 |

(a)

| | $s_1$ | $s_2$ | $s_3$ | $s_4$ | $s_5$ | $s_6$ | $s_7$ | $s_8$ | $s_9$ |
|---|---|---|---|---|---|---|---|---|---|
| $s_1$ | 0 | 0.75 | 3.75 | 2.75 | 4 | 2.625 | 4 | 5.25 | 4.5 |
| $s_2$ | 0.75 | 0 | 3.75 | 2.75 | 4 | 2.625 | 4 | 5.25 | 4.5 |
| $s_3$ | 3.75 | 3.75 | 0 | 3.5 | 5.625 | 4.375 | 5 | 5.625 | 5.25 |
| $s_4$ | 2.75 | 2.75 | 3.5 | 0 | 1.375 | 1.375 | 3.875 | 5.5 | 4.75 |
| $s_5$ | 4 | 4 | 5.625 | 1.375 | 0 | 2 | 5.25 | 5.5 | 5.625 |
| $s_6$ | 2.625 | 2.625 | 4.375 | 1.375 | 2 | 0 | 3.75 | 5.25 | 4.375 |
| $s_7$ | 4 | 4 | 5 | 3.875 | 5.25 | 3.75 | 0 | 2.25 | 2.375 |
| $s_8$ | 5.25 | 5.25 | 5.625 | 5.5 | 5.5 | 5.25 | 2.25 | 0 | 2.375 |
| $s_9$ | 4.5 | 4.5 | 5.25 | 4.75 | 5.625 | 4.375 | 2.375 | 2.375 | 0 |

(b)

**Fig. 10.** The distances measured by the DTW distance (a) and the proposed extension (b).

## 4.2 Efficiency Evaluation

Next, we evaluated how much the proposed computation method presented in Section 3.4.2 improves the efficiency of computing the proposed similarity measures compared to the naïve method presented in Section 3.4.1. In this experiment, we measured the time taken to compute the proposed similarity measures (i.e., the proposed extended versions of the Levenshtein distance, Needleman-Wunsch similarity, and DTW distance, respectively) for 100 pairs of purchase histories using the proposed computation method and the naïve method, respectively. For this experiment, we randomly generated synthetic purchase histories with different lengths and built a segment tree. When the number of products in the hierarchical classification tree was 10,000, the time taken to build a segment tree was 25 seconds on average. Considering the fact that the segment is built once and used repeatedly for subsequent similarity evaluations, this time is very short and negligible.

**Processing time by varying the average length of sequences**

Fig. 11 shows the time taken to compute the three proposed measures using the proposed computation method and the naïve method, respectively. In this experiment, we varied the average length of purchase histories from 5 to 25. We set the number of products in the hierarchical classification tree to 10,000 and the height of the hierarchical classification tree to 12. In Fig. 11, we can see that the time taken to

compute the proposed measures using the naïve method increases rapidly as the average length of purchase histories increases. This is because the naïve method needs to find the paths from the root node to leaf nodes in the hierarchical classification tree and compare the nodes in the paths repeatedly for each pair of products in purchase histories. On the other hand, the proposed computation method reduces the processing time significantly by using a segment tree, which is built in advance, to quickly obtain the shortest distance between two products in the hierarchical classification tree. Note that all the three proposed measures take almost the same time to compute them either when using the proposed computation method or the naïve method, because all the three measures use similar computation processes. Especially, note also that the processing times of the proposed computation method for the three extended measures are almost the same so they are hard to distinguish in Fig. 11. This is because the proposed computation method can obtain the shortest distance between two products almost immediately so it takes very little processing time for all the three extended measures.

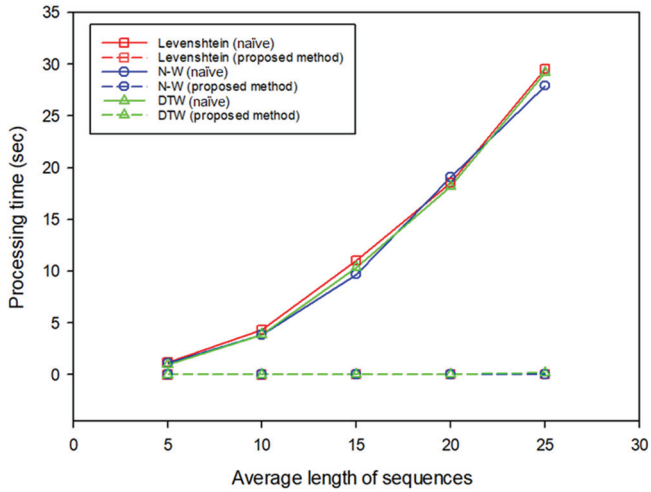


**Fig. 11.** Processing time by varying the average length of sequences.

**Processing time by varying the height of the classification tree**

Fig. 12 shows the time taken to compute the three proposed measures using the proposed computation method and the naïve method, respectively, by varying the height of the hierarchical classification tree from 5 to 17. In this experiment, we set the average length of purchase histories to 20 and the number of products in the hierarchical classification tree to 10,000. In Fig. 12, we can observe that the processing time of the naïve method increases slowly as the height of the hierarchical classification tree increases. This is because the length of paths from the root node to leaf nodes in the hierarchical tree increases. In comparison, the proposed computation method shows almost a constant time regardless of the height of the product classification tree, because the time to obtain the shortest distance between two products in the hierarchical classification tree using the segment tree is not affected by the height of the hierarchical classification tree. Therefore, we can confirm that the proposed computation method is far more efficient than the naïve method to compute the similarity between purchase histories. In Fig. 12, note also that the processing times of the proposed computation method for the three extended measures are almost the same for the same reason as in Fig. 11.

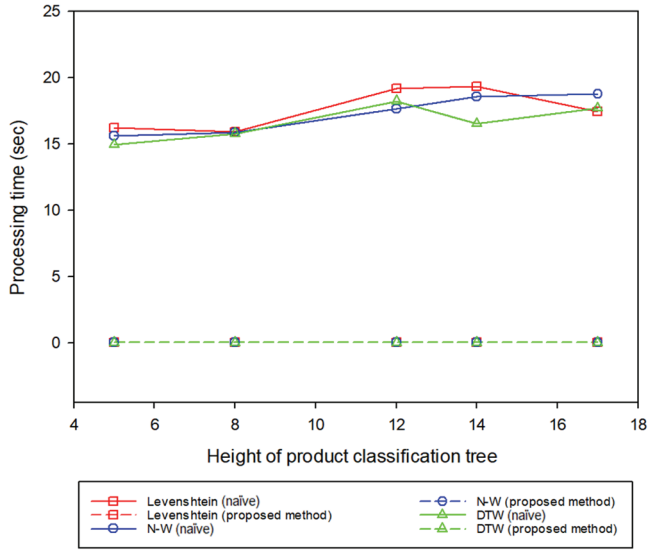


**Fig. 12.** Processing time by varying the height of the hierarchical classification tree.

**Processing time by varying the number of products in the classification tree**

Fig. 13 shows the time taken to compute the three proposed measures using the proposed computation method and the naïve method, respectively, by varying the number of products in the hierarchical classification tree from 7,000 to 25,000. In this experiment, we set the average length of purchase histories to 20 and the height of the hierarchical classification tree to 12. In Fig. 13, the processing time of the naïve method increases linearly as the number of products in the hierarchical classification tree increases. This is because, in the naïve method, the time taken to calculate the shortest distance between two products increases as the size of the hierarchical tree increases. On the contrary, the processing time of the proposed computation method does not increase because the time to obtain the shortest distance between two products using the segment tree is not affected by the size of the hierarchical classification tree. As a result, in Fig. 13, the processing times of the proposed computation method for the three extended measures are almost

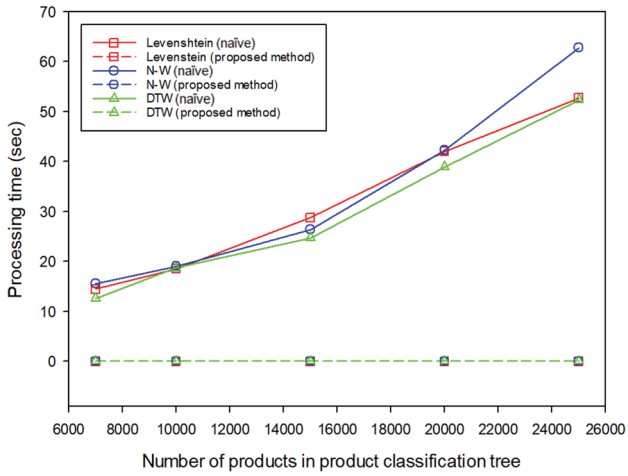


**Fig. 13.** Processing time by varying the number of products in the hierarchical classification tree.

the same and hard to distinguish because it takes very little processing time for all the three extended measures. From the result of this experiment, we can again confirm that the proposed computation method reduces the processing time significantly by using the pre-built segment tree.

Let us close this section by discussing some issues of the segment tree. First, the shape and size of the segment tree do not affect the performance of the proposed computation method because we can obtain the shortest distance between two products almost immediately without traversing the segment tree, as we have empirically shown in this section. Second, if the segment tree is too big to fit in memory, it can lead to frequent disk accesses. However, because we need to store only three numbers for each node in the hierarchical classification tree to build a segment tree (i.e., node ID, depth, and visit order), the amount of memory required to store a segment tree even with 1,000,000 nodes is only about 12 MB, if we use 4 bytes to store a number. Therefore, we can expect a segment tree to fit into memory in most cases.

## 5. Conclusion

In this paper, we have proposed new similarity measures for purchase histories considering a hierarchical classification of products. The proposed measures not only consider the purchase order of products in purchase histories but also the similarity between products in purchase histories. Unlike the existing similarity measures that measure the distance between two products only as 0 or 1 depending on whether they match or not, the proposed measures assign a value between 0 and 1 as the distance between two products considering their distance in the hierarchical classification tree. We have extended three existing representative similarity measures for sequences, i.e., the Levenshtein distance, Needleman-Wunsch similarity, and DTW distance, to consider the hierarchical classification of products.

Furthermore, we have also proposed an efficient computation method to compute the proposed measures. The proposed computation method uses a segment tree to quickly obtain the shortest distance between two products in the hierarchical classification tree. As a result, the proposed computation method reduces the time to compute the distance between two products in the hierarchical classification tree significantly. Using the proposed computation method, the proposed measures can be evaluated very efficiently regardless of the size of the hierarchical classification tree.

In the experiments, we have evaluated whether the proposed measures measure the similarity between purchase histories more effectively than the existing measures. As a result of the experiments, we have confirmed that the proposed measures are more effective than the existing measures in measuring the similarity between purchase histories by considering the hierarchical classification of products. Also, through the experiments measuring the time taken to compute the proposed measures, we have shown that the proposed computation method can reduce the computation time significantly, allowing the proposed measures to be used for a large number of purchase histories.

Therefore, the proposed measures can be effectively used for various marketing strategies, such as customer segmentation, which identifies the groups of customers with similar purchase histories, customer search, which finds customers with similar purchase history, and product recommendation, which recommends products purchased by customers with similar purchase histories.

## Acknowledgement

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (No.2018R1D1A1B07045643).

## References

[1] M. Sforna, "Data mining in a power company customer database," *Electric Power Systems Research*, vol. 55, no. 3, pp. 201-209, 2000.

[2] C. Rygielski, J. C. Wang, and D. C. Yen, "Data mining techniques for customer relationship management," *Technology in Society*, vol. 24, no. 4, pp. 483-502, 2002.

[3] M. Kaur and S. Kang, "Market Basket Analysis: identify the changing trends of market data using association rule mining," *Procedia Computer Science*, vol. 85, pp. 78-85, 2016.

[4] C. Yin, S. Ding, and J. Wang, "Mobile marketing recommendation method based on user location feedback," *Human-centric Computing and Information Sciences*, vol. 9, article no. 14, 2019.

[5] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals," in *Soviet Physics Doklady*, vol. 10, no. 8, pp. 707-710, 1996.

[6] S. B. Needleman and C. D. Wunsch, "A general method applicable to the search for similarities in the amino acid sequence of two proteins," *Journal of Molecular Biology*, vol. 48, no. 3, pp. 443-453, 1970.

[7] D. J. Berndt and J. Clifford, "Using dynamic time warping to find patterns in time series," in *Knowledge Discovery in Databases: Papers from the 1994 AAAI Workshop, Seattle, Washington*. Melon Park, CA: AAAI Press, 1994, pp. 359-370.

[8] S. Park, N. C. Suresh, and B. K. Jeong, "Sequence-based clustering for Web usage mining: a new experimental framework and ANN-enhanced K-means algorithm," *Data & Knowledge Engineering*, vol. 65, no. 3, pp. 512-543, 2008.

[9] E. Zorita, P. Cusco, and G. J. Filion, "Starcode: sequence clustering based on all-pairs search," *Bioinformatics*, vol. 31, no. 12, pp. 1913-1919, 2015.

[10] M. A. Alqarni, S. H. Chauhdary, M. N. Malik, M. Ehatisham-ul-Haq, and M. A. Azam, "Identifying smartphone users based on how they interact with their phones," *Human-centric Computing and Information Sciences*, vol. 10, article no. 7, 2020.

[11] M. H. Pandi, O. Kashefi, and B. Minaei, "A novel similarity measure for sequence data," *Journal of Information Processing Systems*, vol. 7, no. 3, pp. 413-424, 2011.

[12] X. Sun and J. Zhang, "miRNA pattern discovery from sequence alignment," *Journal of Information Processing Systems*, vol. 13, no. 6, pp. 1527-1543, 2017.

[13] P. Senin, "Dynamic time warping algorithm review," Information and Computer Science Department, University of Hawaii at Manoa, Honolulu, HI, 2008.

[14] I. Boulnemour and B. Boucheham, "QP-DTW: upgrading dynamic time warping to handle quasi periodic time series alignment," *Journal of Information Processing Systems*, vol. 14, no. 4, pp. 851-876, 2018

[15] F. P. Preparata and M. I. Shamos, *Computational Geometry: An Introduction*. New York, NY: Springer Science & Business Media, 1985.

[16] M. A. Bender, M. Farach-Colton, G. Pemmasani, S. Skiena, and P. Sumazin, "Lowest common ancestors in trees and directed acyclic graphs," *Journal of Algorithms*, vol. 57, no. 2, pp. 75-94, 2005.

[17] C. F. Su, "High-speed packet classification using segment tree," in *Proceedings of IEEE Global Telecommunications Conference (Cat. No. 00CH37137)*, San Francisco, CA, 2000, pp. 582-586.

**Yu-Jeong Yang** **https://orcid.org/0000-0002-7155-5367**

She received the B.S. degree from the Division of Computer Science, Sookmyung Women's University, Korea, in 2019. She is now M.S. student in the Department of Computer Science of degrees at Sookmyung Women's University, Korea. Her current research interests include databases, data mining and big data.

**Ki Yong Lee** **https://orcid.org/0000-0003-2318-671X**

He received his B.S., M.S., and Ph.D. degrees in Computer Science from KAIST, Daejeon, Korea, in 1998, 2000, and 2006, respectively. From 2006 to 2008, he worked for Samsung Electronics, Suwon, Korea as a senior engineer. From 2008 to 2010, he was a research assistant professor of the Department of Computer Science at KAIST, Daejeon, Korea. He joined the faculty of the Division of Computer Science at Sookmyung Women's University, Seoul, in 2010, where currently he is a professor. His research interests include database systems, data mining, big data, and data streams.