구매이력 데이터에서 상품 분류 체계를 고려한 시퀀스 유사도 측정 기법

양유정, 이기용
숙명여자대학교 컴퓨터과학과
e-mail : {diddbwjd96, kiyonglee}@sookmyung.ac.kr

A Sequence Similarity Measure Considering the Product Taxonomy in Transaction Data

Yu-Jeong Yang, Ki Yong Lee
Division of Computer Science, Sookmyung Women's University

요 약

본 논문은 구매이력 데이터에서 상품간의 분류 체계를 고려하여 시퀀스 간의 유사도를 계산하는 새로운 방법을 제안한다. 시퀀스란 두 항목간의 순서가 존재하는 데이터를 의미한다. 항목 간의 선후관계가 중요한 시퀀스 데이터에서는 두 시퀀스 간의 유사도를 정확히 정의하는 것이 중요하다. 본 논문에서는 대표적인 시퀀스 유사도 측정 알고리즘인 편집 거리 알고리즘을 활용하여 구매이력 데이터에서 시퀀스 간의 유사도를 정의한다. 상품은 상품의 특성에 따라 항목 분류 체계에서 여러 범주로 분류된다. 이 경우 기존의 편집 거리 알고리즘에서 문자의 일치유무에 따라 단순히 0 또는 1 을 부여하는 것은 부정확하다. 따라서 본 논문은 편집 거리 알고리즘의 수정 연산 중 대체 연산 비용 계산 시 항목 분류 트리를 사용하여 연산 비용이 0 에서 1 사이의 값을 가지도록 세분화하였다. 실험 결과 제안 방법은 대체 연산 비용 계산 시 두 문자가 다르면 단순히 1 을 부여하는 기존의 편집 거리 알고리즘에 비해 시퀀스 간의 유사도를 더 정확하게 계산함을 확인하였다.

1. 서론

매출 증대를 위해 기업에서는 고객들의 특성과 소비 패턴이 담긴 구매이력 데이터에 대한 분석이 활발히 진행되고 있다. 기업은 고객들의 구매이력 데이터에서 새로운 구매 패턴을 찾아 마케팅 전략을 새롭게 구성할 수 있다. 구매이력 데이터는 시간의 흐름에 따른 고객의 구매항목으로 구성되며, 여기서 항목들 간의 순서가 존재하는 데이터를 시퀀스(sequence)라고 한다. 시퀀스는 두 항목 간의 선후관계가 담긴 데이터이기 때문에, 만일 두 시퀀스가 동일한 항목들로 이루어졌더라도 순서가 다르다면 서로 다른 시퀀스로 본다. 따라서 시퀀스에서 항목 간의 선후관계를 고려하여 유사도를 정의하는 것이 중요하다.

본 논문에서는 구매내역 데이터에서 순서를 고려하여 시퀀스 간의 유사도를 계산하는 효과적인 방법을 제안한다. 일반적으로 연관성 분석에서 사용되는 지지도, 신뢰도와 같은 통계적 수치를 활용하지 않고 대표적인 문자열 비교 방법인 편집 거리 알고리즘을 활용하여 두 시퀀스 간의 유사도를 비교한다. 또한 유사도 비교 시 상품 분류 체계를 활용하여 시퀀스 내의 항목을 서로 관련 없는 독립적인 개체로 계산하지 않고 분류 체계 내에서 항목의 위치를 고려하여 보다 의미 있는 결과를 도출한다.

본 논문의 구성은 다음과 같다. 2 장에서는 관련 연구를 간략히 살펴보고, 3 장에서는 항목 분류 체계를 활용한 제안 방법을 상세히 설명한다. 4 장에서는 제안 방법의 성능 평가 결과를 보이며, 5 장에서는 결론을 맺는다.

2. 관련 연구

2.1 구매이력 데이터 분석

고객의 소비활동이 담긴 대규모의 구매이력 데이터에는 소비패턴이 담겨 있다. 이렇게 고객의 구매내역에서 빈번하게 발생한 구매 패턴을 찾아 나가는 과정을 연관성 분석이라고 한다. 연관성 분석은 각 항목을 독립적인 개체로 판단하여 규칙을 생성하며, 여기서 항목 간의 계층 관계까지 고려하는 분석은 장바구니 분석이라고 한다. 이와 다르게 순차 패턴 분석은 구매 내역의 선후관계까지 고려하여 규칙을 생성한다 [1]. 세 가지 분석 모두 지지도, 신뢰도, 향상도라는 평가 기준을 사용하여 항목 간의 관련성을 판단한다.

2.2 시퀀스 유사도 측정

시퀀스란 두 항목 간의 순서가 존재하는 데이터로 대표적인 시퀀스 데이터는 웹 로그 데이터, 단백질

시퀀스 데이터가 있다. 시퀀스 데이터를 분석하여 웹 로그 파일에서 비슷한 사용자들을 그룹화[6]하거나 비슷한 구조를 가지는 단백질 시퀀스들을 그룹화 하여 비슷한 기능을 갖는 단백질 시퀀스를 발견할 수 있다[5]. 시퀀스 간의 유사도를 계산하는 다양한 방법이 있으며, 계산 방법에 따라 다음과 같이 나눌 수 있다.

(1) 편집 기반(edit-based) 유사도 측정 방법
두 개의 문자열이 같아 지기 위한 최소 수정 연산 횟수를 구하는 알고리즘이다. 수정 연산은 추가(Add), 대체(Substitute), 삭제(Delete) 연산을 말하며 가장 대표적인 알고리즘은 레벤슈타인(Levenshtein) 거리 알고리즘이 있다[2]. 수정 연산 횟수가 유사도 판단 척도로 사용되며 그 값이 작을수록 두 문자열이 유사하다고 판단한다.

(2) 정렬(alignment) 유사도 측정 방법
주로 단백질 서열이나 핵산 서열 사이의 상관 관계 분석 시 두 서열 간의 유사한 구역을 찾아낼 때 사용한다. 정렬 범위에 따라 국소(local) 정렬 방법과 전역(global) 정렬 방법이 있으며, 대표적인 국소 정렬 알고리즘으로는 스미스-워터맨(Smith-Waterman) 알고리즘[7]이 있고 전역 정렬 알고리즘으로는 니들만 브니쉬(Needleman-Wunch) 알고리즘[4]이 있다. 두 알고리즘은 정렬 범위에 따라 두 시퀀스가 가장 유사하도록 공백(gap)을 사용하여 정렬한다.

(3) 집합 기반(set-based) 유사도 측정 방법
문자열을 문자의 집합 혹은 토큰(token)의 집합 형태로 바꾸어 계산한다. 집합 관계를 이용하여 연산하며 문자열을 토큰으로 나누는 경우에는 N-gram 개념을 사용하여 문자열을 길이가 N 개의 기준 단위로 절단하여 사용한다. 대표적인 알고리즘으로는 자카드(Jaccard) 유사도[3]가 있다. 자카드 유사도는 집합을 구성하는 원소들 간의 합집합과 교집합 간의 비율을 나타내며 0 에서 1 사이의 값을 가진다.

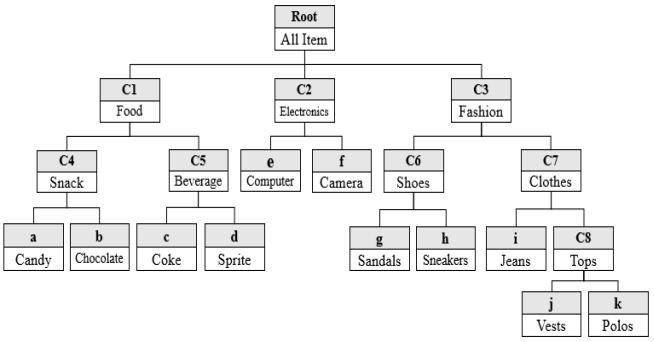
본 논문에서는 시퀀스 간의 유사도 계산시 편집 거리 알고리즘을 사용한다. 편집 거리 알고리즘은 두 시퀀스 간의 항목을 하나씩 비교 가능하며 시퀀스에 공백을 추가하거나 집합 원소로 나누는 등의 별도 연산은 필요하지 않다. 따라서 본 논문에서는 편집 거리 알고리즘의 수행 과정 중 일부분을 변형하여 사용하였다.

3. 제안 방법

본 장에서는 구매이력 데이터에서 항목 분류 체계를 활용하여 두 시퀀스 간의 유사도를 계산하는 제안 방법을 상세히 설명한다. 먼저 구매이력 데이터에서 시퀀스를 $S = < x_1, x_2, \dots, x_n >$라 하자. x_i는 시퀀스 S 의 i 번째 구매 항목을 나타내며, 이 시퀀스는 x_1 부터 x_n 까지 순서대로 구매된 이력을 나타낸다. 시퀀스의 크기는 해당 시퀀스 내의 구매된 항목의 개수를 말하며, 이 시퀀스의 경우 크기는 n 이 된다. 다음은 제안 방법에 대해 상세히 설명한다.

3.1 항목 분류 트리

일반적으로 백화점, 마트와 같은 유통 업체는 모든 상품에 대하여 대분류/중분류/소분류와 같이 계층을 나눈 상품 분류 체계를 가진다. 예를 들어 '바지'라는 상품은 먼저 소분류인 '하의'에 속하며, 여기서 하의는 다시 중분류인 '의류'에 속하게 된다. 이렇게 모든 상품은 항목 분류 체계의 가장 하위 부분을 차지한다. 본 논문에서는 이러한 항목 분류 체계를 트리 구조로 표현하였으며, (그림 1)은 본 논문에서 사용한 항목 분류 트리 중 일부이다.



(그림 1) 항목 분류 트리

본 논문에서 사용한 항목 분류 트리의 체계는 미국 전자 상업 회사인 아마존의 분류 체계를 참고하여 구성하였다. 트리의 각 노드는 이름과 데이터로 구성된다. 노드의 데이터는 실제 상품이나 상품의 상위 범주명이 저장되며, 노드의 이름은 계산 편의상 사용되는 해당 데이터를 나타내는 별칭이다. 만일 노드의 데이터가 상품인 경우 노드의 이름은 임의의 알파벳을 부여하고 범주명인 경우 C_i 형태로 부여한다. 만일 구매내역 시퀀스 내에 'Computer'가 있는 경우 (그림 1)의 트리에 의해 'Computer'라는 항목은 해당 노드의 이름인 'e'로 바꾸어준 후 계산한다. 이렇게 구매이력 데이터에서 시퀀스 내 모든 항목을 각각 대응하는 노드의 이름으로 바꾸어 준다. 해당 과정을 거치면 계산하고자 하는 구매내역 시퀀스는 일종의 문자열 형태로 변환된다.

3.2 항목 분류 체계를 활용한 편집 거리 알고리즘

편집 거리 알고리즘은 레벤슈타인 거리 알고리즘으로도 알려져 있으며 두 문자열 같아지기 위한 최소 수정 횟수를 구하는 알고리즘이다. 문자열 간의 수정 연산은 추가, 대체, 삭제를 말하며 두 문자열 길이만큼의 2 차원 배열을 이용하여 문자열을 한자 씩 비교해 나간다. 계산 후 배열의 가장 마지막 원소의 값이 두 문자열의 최소 편집 거리가 되며 본 논문에서 사용한 알고리즘은 동적 프로그래밍으로 구현하였다. 시퀀스 S_1 과 S_2 가 주어지면 문자열이 없는 경우도 포함하여 2 차원 배열 M 을 행의 개수가 $|S_1|+1$ 이고 열의 개수가 $|S_2|+1$ 인 $(|S_1|+1)\times(|S_2|+1)$ 크기로 생성한다.

M[i][j]의 원소의 값은 S_1[i-1]번째까지의 문자열과 S_2[j-1]번째까지의 문자열의 최소 편집 거리를 의미한다($0 \le i \le |S_1|+1$, $0 \le j \le |S_2|+1$). 배열의 첫번째 행과 첫번째 열은 0 에서부터 두 문자열의 길이만큼 증가시켜가며 초기화 한다. 배열의 두번째 행과 열부터는 이전까지 계산된 배열 값을 사용하여 채워나간다.

M 에서 행에 해당하는 S_1 은 원본 문자열을 의미하며 열에 해당하는 S_2 는 바꾸고자 하는 목적 문자열을 의미한다. M[i][j]의 원소는 추가, 대체, 삭제 비용 중 가장 작은 값을 가진다. 추가 연산은 M[i][j-1]의 값에 문자를 삽입하는 비용 1 을 더한다. 삭제 연산은 M[i-1][j]의 값에 문자를 삭제하는 비용 1 을 더한다. 마지막으로 대체 연산은 M[i-1][j-1]의 값을 이용하여 계산하며, 만일 현재 계산하려고 하는 S_1[i-1]번째 문자와 S_2[j-1]번째 문자가 같다면 대체 비용은 0 이 더해지고, 다르다면 1 을 더해준다. (그림 2)는 본 논문에서 제안하는 시퀀스 유사도 계산 방법 전체를 나타내는 의사코드이다. (그림 2)의 의사코드 중 computeSequenceDistance 함수에서 네모 칸 부분이 1 이 된다면 원래 편집 거리 알고리즘이 수행된다.

```
function computeSequenceDistance(s1, s2)
  int M[|s1|+1][|s2|+1] = 0
  for i in (|s1|+1) do
     M[i,0] = i
  for j in (|s2|+1) do
     M[0,j] = j
  for i in (|s1| + 1) do
    for j in (|s2| + 1) do
      if (s1[i-1] == s2[j-1])
        cost = 0
      else
        cost = computeSubstituteCost(i, j, root)
      M[i][j] = min(M[i][j-1] + 1,    //Insert
                    M[i-1][j] + 1,    //Delete
                    M[i-1][j-1]+cost))//Substitute
    endFor
  endFor
  return M[|s1|+1][|s2|+1]
endFunction

function computeSubstituteCost(i, j, root)
  if (computeCondition) /*(식 1)*/
    longestPath = searchLongestPath(root)
    itemPath = searchPath(s1[i-1],s2[j-1], root)
    cost = itemPath/longestPath
  else
    cost = 1
  return cost
endfunction
```

(그림 2) 제안 방법의 의사 코드

본 논문에서 제안하는 알고리즘은 기존의 편집 거리 알고리즘의 대체 연산 중 더해지는 비용의 값을 0 에서 1 사이로 세분화하여 보다 정확한 편집 거리를 계산한다. 본 논문에서 사용한 데이터는 고객의 구매 항목들이 나열된 시퀀스이다. 이 시퀀스는 항목 분류 트리에 따라 해당 노드의 이름으로 바꾸어 문자열의 형태로 저장된다. 만일 시퀀스 S_1 = <Coke Jeans> 과 S_2 = <Sprite Jeans> 가 주어진 경우 (그림 1)의 항목 분류 트리를 이용하여 S_1 = < c i > 과 S_2 = < d i >로 바꾸어준다. 만일 기존 편집 거리 알고리즘 사용한다면 대체 연산에서 비교하는 S_1 의 c 와 S_2 의 d 가 다르기 때문에 대체 연산 비용은 단순히 1 이 더해진다. 하지만 c 와 d 는 항목 분류 트리 내에서 'Beverage'라는 같은 범주 아래 비슷한 종류의 상품이므로 전혀 상관이 없는 두 상품보다 더 높은 유사도를 부여할 필요가 있다. 따라서 본 논문에서는 대체 연산 비용을 항목 분류 트리 내에서 가장 먼 항목간의 경로를 가질 때만 전혀 관련이 없다고 판단하여 최대 값 1 을 부여하고, 그렇지 않은 경우에는 항목 분류 트리 내에서 서로 다른 두 항목간의 최단 경로의 길이를 이용하여 대체 연산 비용의 값을 세분화 한다. (그림 2)의 의사코드에서 computeSubstituteCost 함수는 M[i][j]의 대체 연산 비용을 계산할 때 사용되는 함수이다. 다음은 대체 연산을 진행할 후보를 선정하는데 사용되는 조건이다.

$$(new\ InsertCost > existing\ SubstituteCost) \wedge (new\ DeleteCost > existing\ SubstituteCost) \quad \text{(식 1)}$$

기존의 대체 연산 값(M[i-1][j-1])이 새로 계산할 추가 연산 값(M[i][j-1]+1)과 삭제 연산 값(M[i-1][j]+1)보다 작은 경우에만 대체 연산 함수를 수행한다. (식 1)의 두 조건을 모두 만족하는 경우에만 항목 분류 트리를 이용하여 해당 배열 값에 대한 대체 연산 비용을 계산한다. 이를 통해 매 대체 연산마다 항목 분류 트리를 탐색하는데 드는 계산 비용을 절약하였다. 본 논문에서는 다음과 같은 식을 사용하여 새로운 대체 연산 비용을 계산한다.

$$cost = \frac{itemPath}{longestPath}\ (0 \le cost \le 1) \quad \text{(식 2)}$$

cost 는 S_1[i-1]번째 문자와 S_2[j-1]번째 문자의 대체 연산 시 더해지는 비용이다. cost 값은 항목 분류 트리 내에서 찾고자 하는 두 항목의 최단 경로의 길이가 가장 먼 두 항목의 길이에서 차지하는 비율을 나타낸다. longestPath 는 항목 분류 트리 내에서 가장 먼 두 노드의 경로 길이의 값을 가지며, itemPath 에는 항목 분류 트리 내에서 S_1[i-1]번째 문자를 이름으로 가지는 노드에서 S_2[j-1]번째 문자를 이름으로 가지는 노드까지의 최단 경로의 길이이다. 두 항목 사이의 최단 경로는 두 항목에서 가장 가까운 범주까지의 거리를 의미한다. 따라서 길이가 작을수록 두 항목은 높은 연관성을 가지며 같은 범주에 분류되어 있을 확률이 높다. itemPath 의 길이가 0 인 경우 두 노드는 같은 노드이며 cost 는 0 이 되고, 가장 먼 길이는 longestPath 와 같은 값을 가지게 되며 cost 는 1 이 된다. 새롭게 계산된 대체 연산 비용은 추가, 대체, 삭제 비용 중 최소 값을 선택할 때 사용된다.

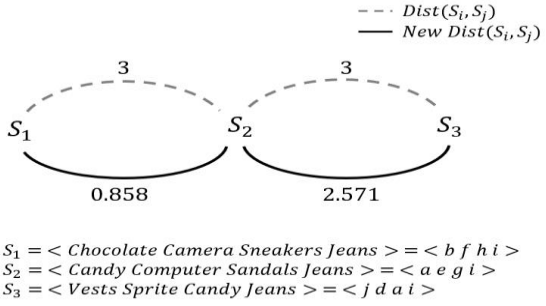
4. 실험 결과

본 장에서는 제안하는 시퀀스 유사도 측정 방법에 대한 실험 결과를 보인다. 실험에서는 시퀀스 크기가 3 에서 10 사이를 가지는 가상의 시퀀스 데이터를 생성하여 사용하였다. 제안 방법은 Python 을 사용하여 구현하였으며, 항목 분류 트리는 anytree 라이브러리

를 사용하여 구현하였다. 실험에서는 Intel i7-5820 3.3 GHz CPU, 8GB 메모리가 장착된 Windows 10 운영체제 환경의 PC 에서 수행하였다.

4.1 정확성 실험

먼저 제안 방법이 기존의 편집 거리 알고리즘에 비해 시퀀스 유사도를 더 정확히 계산할 수 있는지를 평가하였다. (그림 3)은 정확성 측정 실험에서 사용한 시퀀스 데이터와 실험 결과를 나타낸다. 실험에서 사용한 항목 분류 트리는 (그림 1)의 구조를 사용하였으며 각 시퀀스의 구매 항목은 항목 분류 트리에서 대응되는 노드 이름으로 변환된다.

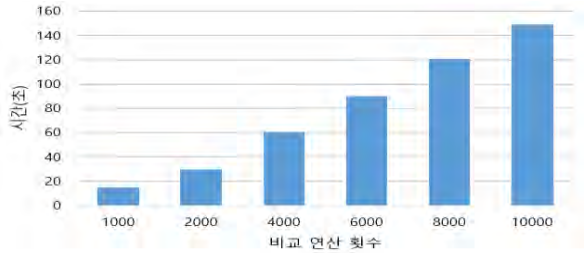


(그림 3) 유사도 비교 결과 예시

(그림 3)은 S_1 과 S_2 에 대한 유사도 계산 값과 S_2 와 S_3 에 대한 유사도 계산 값을 비교한 그림이다. 점선 위의 값은 기존 편집 거리 알고리즘의 결과 값이고, 실선 아래의 값은 제안 방법에 대한 계산 결과이다. 실험 결과 기존 편집 거리 알고리즘을 사용했을 때는 동일한 계산 값을 보이지만 제안 알고리즘을 사용하는 경우에는 S_1 과 S_2 의 거리가 S_2 와 S_3 에 비하여 현저히 낮은 수치를 보인다. S_1 과 S_2 의 경우 서로 구매한 항목은 다르지만 매우 관련이 높은 항목으로 구성되어 있다. 이는 기존 편집 거리에서는 서로 다른 항목에 대해서 단순히 1 을 부여 하여 계산하지만 제안 알고리즘에서는 항목 분류 체계를 고려하기 때문에 S_1 과 S_2 가 서로 항목은 다르더라도 관련성 높은 항목이기때문에 낮은 값이 계산된다. 따라서 제안 방법이 항목간의 서로 다른 관련성을 고려하기때문에 더 정확한 결과를 보인 것으로 판단된다.

4.2 수행 시간 측정

다음으로는 제안하는 시퀀스 유사도 측정 기법의 수행 시간을 측정하여 수용 가능한 수준인지 평가하였다. 실험에서 사용한 항목 분류 트리의 높이는 5 이며 총 36 개의 노드를 가진다. 상품을 의미하는 26 개의 단말 노드(leaf node)와 상품의 범주에 해당하는 11 개의 노드로 구성된다. 두 시퀀스의 비교 연산 횟수를 2000 번에서 10000 번까지 증가시켜가며 수행 시간을 측정하였다. 실험 결과 연산 속도는 시퀀스의 크기와 시퀀스 내 항목 구성에 따라 의존하는 경향을 보인다. (그림 4)는 수행 시간을 측정한 결과로 10 번 측정 후 평균 값을 표현하였으며, 연산 횟수를 증가할수록 수행 시간이 선형적으로 증가하는 모습을 보인다. 하지만 제안 방법의 수행시간은 최대 3 분을 넘지 않으며, 이는 제안 방법의 수행시간이 실제 사용 가능한 수준임을 나타낸다.



(그림 4) 수행시간 측정 결과

5. 결론

본 논문에서는 구매내역 데이터에서 항목 분류 체계를 고려하여 시퀀스 간의 유사도를 측정하는 방법을 제안하였다. 제안 방법은 편집 거리 알고리즘을 활용하였으며 대체 연산 비용 값을 항목 분류 체계를 사용하여 세분화하였다. 이를 통해 항목간의 서로 다른 중요도를 고려할 수 있으며 독립적인 개체로 계산할 때 보다 의미 있는 결과가 도출됨을 실험을 통해 확인하였다. 제안 방법은 구매이력 데이터뿐만 아니라, 계층 구조를 가지는 항목들로 구성된 다른 시퀀스 데이터 간의 유사도 비교 시에도 도움이 될 것을 기대한다.

Acknowledgement

이 논문은 2019 년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구임 (No.2018-0-00269, A research on safe and convenient big data processing methods)

참고문헌

[1] EWT Ngai, L Xiu, DCK Chau, “Application of data mining techniques in customer relationship management: A literature review and classification” Expert systems with applications, Vol. 36, No. 2, pp. 2592-2602, 2009.

[2] Levenshtein, “Binary codes capable of correcting deletions, insertions, and reversals,” Soviet Physics Doklady, Vol.10, pp.707-10, 1966.

[3] P. Jaccard, “Étude comparative de la distribution florale dans une portion des Alpes et des Jura,” Bulletin de la Société Vaudoise des Sciences Naturelles, Vol.37, pp.547-579, 1901.

[4] 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, pp.443-453, 1970.

[5] SF Altschul, W Gish, W Miller, EW Myers, DJ Lipman, “Basic local alignment search tool” Jounal of Molecular Biology, Vol.215, No.3, pp.403-410, 1990.

[6] S Park, NC Suresh, BK 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.

[7] T. F. Smith and M. S. Waterman, “Identification of Common Molecular Subsequences,” Journal of Molecular Biology, Vol.147, pp.195-197, 1981.