KIPS Tr. Software and Data Eng.
Vol.4, No.2 pp.71~76 pISSN: 2287-5905

http://dx.doi.org/10.3745/KTSDE.2015.4.2.71

A Similarity Join Algorithm Using a Median as a Filter

Jong Soo Park†

ABSTRACT

In similarity join processing, a general technique employs a generation-verification framework, which includes two phases: the first phase generates a set of candidate pairs from a collection of records; and the second phase verifies each candidate pair by computing real similarity. In order to reduce the number of candidate pairs in the verification phase, the median of one record of each candidate pair is used as a filter in this paper to test whether the other record can has the proper number of overlapped tokens. We propose a similarity join algorithm with the median filter, and show that the proposed algorithm has better performance in execution time than recent algorithms without the filter through extensive experiments on real-world datasets.

Keywords : Similarity Join, Median, Filter, Performance Evaluation

중앙값을 필터로 이용한 유사도 조인 알고리즘

박 종 수†

요 약

유사도 조인 처리에서 일반적인 기법은 생성-검증 구조를 사용하여, 첫 번째 생성 단계는 레코드들의 집합에서 후보 쌍들의 집합을 생성하고 두 번째 단계는 실제 유사도를 계산하여 각 후보 쌍을 검증한다. 검증 단계에서 후보 쌍들의 개수를 줄이기 위하여 본 논문에서는 각 후보 쌍의 한 레코드의 중앙값을 다른 레코드와 공통되는 토큰들의 개수가 적절하게 가질 수 있는지를 검사하는 필터로 사용한다. 중앙값 필터를 가지는 유사도 조인 알고리즘을 제안하고 제안된 알고리즘이 실세계 데이터집합에서 여러 실험을 통해 중앙값 필터를 갖지 않는 최근의 알고리즘들에 비해 실행시간에서 더 좋은 성능을 가진다는 것을 보여준다.

키워드 : 유사도 조인, 중앙값, 필터, 성능 평가

1. 서 론

많은 문서들 중에서 주어진 문서와 유사하거나 거의 동일한 것을 찾아내는 유사도 조인 문제는 데이터 정제나 복사 탐지 등에 응용될 수 있기 때문에 데이터베이스나 데이터 마이닝 분야에서 유사도 조인은 중요한 연산들 중의 하나다[1-4]. 유사도를 측정하는 함수들은 몇 개가 있지만[4-7], 본 논문에서는 주로 Jaccard 유사도를 계산하는 것에 관심을 둔다. 한 객체에 속한 단어들은 일정한 순서에 의해 배열 방식으로 저장된다. 두 객체 사이의 Jaccard 유사도는 공통되는 단어들의 비율로 계산되고, 주어진 한계치 이상의 유사도를 가지면 두 객체는 유사도 조인 쌍으로 출력된다. 유사도 조인 문제는 주로 생성과 검증 단계를 거쳐서 결과를 찾아내고 있다[4-9]. 그 첫 단계는 먼저 유사도 조인 후보 쌍들을 생성하는 단계와 그다음으로 각 후보 쌍의 두 레코드들의 유사도를 계산하는 검증 단계로 구성된다.

본 논문에서 유사도 조인 후보 쌍들의 생성 단계는 기존 알고리즘의 접두 필터링 방법[4, 8, 9]을 이용하고 검증 단계에서는 단순하면서 성능을 개선시킬 수 있는 방법을 모색하였다. 검증 단계에서 각 후보 쌍의 두 객체들에 속한 공통 단어들을 계산하기 전에 먼저 한 객체의 중앙 위치에 있는 단어와 다른 객체에서 그 단어의 위치로 올 수 있는 범위를 정해 유사도 조인 쌍이 될 수 있는 조건식을 찾아낸다. 본 논문에서는 이 조건을 활용함으로 빠르게 유사도 조인 결과를 얻을 수 있는 알고리즘을 제안하고 개선된 성능을 여러 실험을 통하여 보여준다.

2절에서 유사도 함수와 일반적인 유사도 조인 알고리즘의 특성을 설명한다. 3절에서 제안하는 중앙값 필터와 유사도 검증 알고리즘에 대해 서술한다. 4절에서 제안된 알고리즘의 개선된 성능을 실험을 통하여 표와 그래프로 보여주고, 5절에서 결론을 맺는다.

※ 이 논문은 2013년도 성신여자대학교 학술연구조성비 지원에 의하여 연구되었음.
† 종신회원 : 성신여자대학교 IT학부 교수
Manuscript Received : September 12, 2014
First Revision : November 11, 2014
Accepted : November 24, 2014
* Corresponding Author : Jong Soo Park(jpark@sungshin.ac.kr)

2. 유사도 조인 문제

유사도 조인 문제는 주어진 데이터집합에서 서로 유사한 문서들을 찾아내는 것이다. 이 문제와 관련된 기본적인 용어에 대하여 먼저 설명한다. 데이터집합에 속한 모든 문서들의 서로 다른 단어들을 각각의 숫자로 변환시켰을 때 i 번째 순서의 숫자를 w_i로 나타내면, 변환된 숫자들의 집합은 유한한 전체집합 $U=\{w_1, w_2, ...\}$로 표기될 수 있다. 원소 w_i는 또한 토큰이라고도 부른다. 그렇게 되면 각 문서는 집합 $U=\{w_1, w_2, ...\}$로부터 뽑아낸 토큰들의 집합으로 구성된 한 레코드로 볼 수 있다[4, 9]. 유사도 함수(similarity function)는 두 개의 레코드들에서 $[0,1]$ 사이에 있는 유사도 값을 반환한다. 먼저 레코드들의 집합, 유사도 함수 $\sim()$, 유사도 한계치 t가 주어지면, 유사도 조인(similarity join) 문제는 레코드들 $\langle x, y\rangle$의 유사도가 주어진 한계치 t보다 작지 않은 $\mathrm{sim}(x,y) \geq t$인 모든 레코드들의 쌍들의 집합을 찾는 것이다. 본 논문에서 다루는 각 문서나 이메일은 단어들로 파싱되어 토큰들의 다중집합(multiset)인 한 레코드로 변환된다. 토큰들은 한 레코드에서 여러 번 나타날 수 있는데 계속 나타나는 같은 토큰은 새로운 토큰으로 취급하여 토큰 번호를 다르게 배정한다[2, 4, 5, 9]. 각 레코드의 토큰들은 토큰 번호 순서대로 저장되어있다.

본 논문에서는 널리 사용되고 있는 자카드 유사도(Jaccard similarity)를 계산하는 과정에 대해 연구하고, 그것과 관련된 공통부분 유사도(Overlap similarity)에 대해 논의한다. 레코드 x는 $|x|$개의 토큰들로 구성된다. 두 레코드 x와 y를 고려하면 두 유사도 함수는 다음과 같이 정의된다[4-6, 9].

$$\text{Jaccard similarity: } J(x,\ y) = \frac{|x \cap y|}{|x \cup y|}.$$
$$\text{Overlap similarity: } O(x,\ y) = |x \cap y|.$$

$[0,1]$ 사이에 있는 값으로 주어지는 자카드 유사도의 한계치를 t라 하면 자카드 유사도와 공통부분 유사도는 다음과 같은 관계식을 가지므로 공통되는 토큰들의 개수가 α보다 작지 않아야 한다[4-6, 9].

$$J(x,\ y) \geq t \Leftrightarrow O(x,\ y) \geq \alpha = \frac{t}{1+t} \cdot (|x|+|y|) \qquad (1)$$

두 레코드 사이에 유사도를 찾는 방법 중에서 가장 널리 채택되고 있는 기법은 생성-검증(generation-verification) 구조이다[4-9]. 이 구조는 두 단계로 이루어지고 있다. 1) 생성 단계 : 다수의 비슷하지 않은 쌍들을 제거하여 작은 유사도 조인 후보 쌍들을 생성하는 단계, 2) 검증 단계 : 각 유사도 조인 후보 쌍의 실제 유사도를 계산한 후에 한계치 이상이면 결과를 출력하는 단계. 후보 쌍들을 생성하는 단계에서 비슷하지 않은 쌍들을 제거하는 여러 기법들 중에서 접두 필터링(prefix filtering)이 가장 효과적인 기법으로 알려졌고 이것을 최적화하는 많은 알고리즘들이 제안되었다[4, 8, 9]. 각 단계에서 필터를 적용하는 것은 그 자체가 비용을 포함하므로 항상 성능이 개선되지 않으므로 필터 비용과 성능 개선에서 상충되는 면이 있다[7]. 본 논문에서는 생성된 후보를 검증하는 단계에서 간단한 필터를 적용함으로써 필터의 비용을 상쇄하고도 더 성능이 개선되는 알고리즘을 제안한다. 그러므로 본 논문에서는 후보 쌍들을 생성하는 단계로 기존의 알고리즘들[4, 9]을 사용하고 검증 단계에서 효과적으로 유사도 조인 후보 쌍들을 줄이는 방법에 대하여 상세히 서술한다.

3. 중앙값 필터와 유사도 검증 알고리즘

3.1 위치 정보를 이용한 필터 설계

Fig. 1은 유사도 조인 쌍의 후보인 두 레코드 x와 y가 실제 유사도를 계산하기 위한 검증 단계에서 각 레코드의 범위를 나타낸 것이다. 레코드 x의 토큰 인덱스는 min_x와 max_x 사이인 x[min_x: max_x]이고, 레코드 y는 y[min_y: max_y]를 갖는다.

두 레코드 x와 y의 공통부분 유사도가 α보다 작아서 유사도 조인 쌍이 될 가능성이 없는 것을 찾아내기 위하여 새로운 필터를 설계하기로 한다. Fig. 1에서 레코드 y의 중앙 위치인 mid_y에 있는 토큰의 값이 상대 레코드 x에서 나타날 수 있는 위치 범위를 x_low과 x_high 사이라고 하자. 그

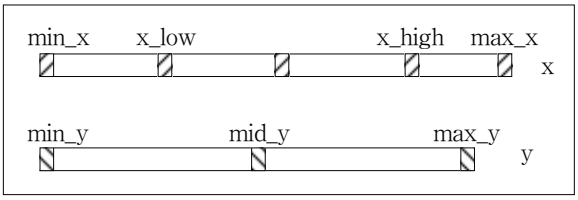

Fig. 1. Positions of tokens of two records x and y

러면 Fig. 1에서 두 레코드 x와 y가 유사도 조인 쌍이 될 수 없는 조건은 다음과 같다.

$$(\text{max\_y} - \text{mid\_y}) + (\text{x\_low} - \text{min\_x}) + O + 1 < \alpha$$
$$\textbf{OR}\ (\text{mid\_y} - \text{min\_y}) + (\text{max\_x} - \text{x\_high}) + O + 1 < \alpha. \qquad (2)$$

여기서 기호 O는 두 레코드가 유사도 조인 후보 쌍으로 선정될 때 공통된 토큰들의 개수이고, α는 두 레코드가 유사도 조인 쌍이 되기 위한 공통 토큰들의 최소 개수이다. Equation (2)는 두 레코드가 가질 수 있는 공통되는 토큰들의 최대 개수가 α보다 작은 경우의 조건식이다. 첫 번째 식은 레코드 y에서 위쪽 부분의 토큰들의 개수, 레코드 x에서 아래쪽의 토큰들의 개수, 현재 공통된 토큰 개수 O, mid_y와 같은 값을 가진 x_low 위치의 값으로 토큰 1개를 합한 것이 α보다 작은 조건식이다. 두 번째 식도 대칭적으로 토큰들의 개수를 합하면 유사도 조인 쌍이 될 수 없는 조건이 된다. Equation (2)를 풀어서 x_low과 x_high의 값을 구하게 되면 다음과 같은 조건식을 얻는다.

$$\text{x\_low} < \alpha - O - \text{max\_y} + \text{mid\_y} + \text{min\_x} - 1$$
$$\textbf{OR } \text{x\_high} > O - \alpha + \text{mid\_y} - \text{min\_y} + \text{max\_x} + 1 \quad (3)$$

Equation (3)을 사용하여 만약 레코드 y의 중앙 위치인 mid_y의 값 v_{mid}와 같은 값을 갖는 레코드 x의 x_low 위치가 $\alpha - O -$ max_y+mid_y+min_x - 1보다 작으면, 두 레코드 x와 y는 유사도 레코드 쌍이 될 수 없게 된다. 그리고 v_{mid}와 같은 값을 갖는 x_high의 위치가 $O - \alpha$+mid_y - min_y +max_x+1보다 크게 되면 역시 두 레코드 x와 y는 유사도 조인 쌍이 될 수 없다. 그러면 유사도 조인 쌍이 될 수 있는 필요조건은 Equation (3)의 조건을 만족하지 않는 조건을 가지면 되고, 이 조건들을 레코드 x와 y의 토큰의 배열로 표현하면 다음 조건식이 된다.

$$\text{y[mid\_y]} \geq \text{x}[\alpha - O - \text{max\_y} + \text{mid\_y} + \text{min\_x} - 1]$$
$$\textbf{AND } \text{y[mid\_y]} \leq \text{x}[O - \alpha + \text{mid\_y} - \text{min\_y} + \text{max\_x} + 1] \quad (4)$$

그러므로 유사도 조인 후보 쌍에서 Equation (4)를 만족하는 경우에만 두 레코드의 토큰들을 비교해서 공통 토큰들의 개수가 α개보다 같거나 큰지를 검사한다. 유사도 조인 쌍이 되는 충분조건은 실제 공통부분 유사도를 계산한 것이 α보다 작지 않으면 된다.

3.2 유사도 검증 알고리즘 VerifyM

```
VerifyM(x, y, O, α)
  input : two records x[p_x:|x|-1] and y[p_y:|y|-1], O and α
          between the two records.
  output: if the number of overlapped tokens between the
          two records is not less than α, then store them as
          a similarity join pair.
1   r = α − O; // the number of tokens
                 to be overlapped between the two records
2   mid = (p_y + |y| − 1)/2; // median position
3   v_mid = y[mid];     // the value of median position
4   if v_mid ≥ x[r − |y| + mid + p_x]
       and v_mid ≤ x[|x| − r + mid − p_y] then
5         while p_y < |y| and p_x < |x| do
6             if x[p_x] < y[p_y] then
7                 p_x++;
8                 if |x| − p_x < r then break;
9             else if x[p_x] > y[p_y] then
10                p_y++;
11                if |y| − p_y < r then break;
12            else then   // overlapped token
13                r−−;  p_x++;  p_y++;
14        enddo  // end of line-5-while-loop
15        if r ≤ 0 then store the records x and y
                    as a similarity join pair;
16  endif  // end of line-4-if-clause
```

Fig. 2. Algorithm VerifyM to compute the similarity of two records

Fig. 2는 유사도 조인 후보 쌍인 두 레코드 x와 y가 실제로 유사도 조인 쌍인지를 검증하는 알고리즘으로 공통 토큰들의 개수를 계산하는 과정에서 Equation (4)를 필터로 이용하였다. 이 알고리즘의 입력은 두 레코드 x와 y, $x[0:p_x-1]$와 $y[0:p_y-1]$ 사이에서 공통되는 토큰들의 개수 O, 유사도 조인 쌍이 되기 위한 최소 공통 토큰들의 개수인 α가 매개변수로 넘어온다. 단계 1의 변수 r은 두 레코드가 유사도 조인 쌍이 되기 위해서 $x[p_x:|x|-1]$와 $y[p_y:|y|-1]$ 사이에 있는 공통된 토큰들의 최소 개수를 의미한다. 다시 설명하면, 이후 단계들에서 r개 이상의 공통되는 토큰들을 발견하면 두 레코드 x와 y는 유사도 조인 쌍으로 결정된다.

단계 3의 v_{mid}는 앞 절에서 설명된 레코드 y에서 검증할 토큰들 중에서 메디안 위치($(p_y+|y|-1)/2$)의 값을 나타내고 있다. 단계 4의 조건은 Equation (4)를 적용한 것으로 중앙값 필터(median filter)에 해당한다. 이 조건을 만족하면 유사도 조인 쌍이 될 가능성이 있기 때문에 단계 5-15를 수행하고, 그렇지 않으면 Equation (4)에 의하여 유사도 조인 쌍이 아닌 것으로 결정된다. 단계 5-14에서는 인덱스 p_x와 p_y를 가진 두 레코드의 토큰들을 비교한 결과에 따라 r, p_x, p_y의 값을 감소 또는 증가시켜나간다. 단계 8과 11에서는 각 레코드에 남아있는 토큰들의 개수가 r보다 작을 때 더 이상 유사도 조인 쌍이 될 수 없으므로 while-loop를 벗어나게 한다[6-9]. 단계 15에서는 r이 0보다 작거나 같으면 공통되는 토큰들의 개수가 α보다 크거나 같은 것을 의미하므로 두 레코드 x와 y를 유사도 조인 쌍으로 저장한다. 중앙값 필터를 검증 단계에서 사용하였지만 유사도 조인 후보 쌍을 생성하는 단계에서도 적용할 수 있다.

4. 실험 결과

대용량의 데이터를 처리하는 알고리즘의 성능을 평가하기 위해 본 논문에서는 기존 연구에서 사용된 주요 데이터집합을 다음과 같이 사용하였다. 각 문자열(string)을 토큰으로 변환하기 위해 구분 문자로 공백과 몇 개의 구두점들을 사용하였다.

DBLP 데이터집합[4-10] : Computer science publications를 포함하고 있는 참고문헌 목록 레코드들의 짧은 정보로 DBLP web site에서 구할 수 있다[1]. 1,916,554개의 논문과 출판물에 대한 정보를 포함하고 있다. 문헌들에서 추출된 서로 다른 토큰들의 개수는 1,293,322개이고, 문헌들의 평균 토큰 개수는 21개이다.

Trec 데이터집합[4, 7] : 정보 검색 분야에서 잘 알려진 벤치마크로 문서들의 집합이다[2]. 348,566개의 문서들을 포함하고, 문서들에서 추출된 전체 토큰들의 개수는 1,776,061개이다. 문서들의 평균 토큰 개수는 158개이다.

1) http://dblp.uni-trier.de/xml
2) http://trec.nist.gov/data/t9_filtering.html

Enron 데이터집합[4, 6, 7, 9, 10] : Enron사의 email을 수집해놓은 것이다[3]. 517,424개의 email들을 포함하고 있다. Email들에서 추출된 전체 토큰들의 개수는 2,362,095개이다. email들의 평균 토큰 개수는 285개이다.

실험에 사용된 컴퓨터는 Intel i7-4820K 3.70GHz CPU를 가진 PC이고 실험 데이터를 충분히 포함하는 메인 메모리를 가진다. PC의 운영체제는 MS Windows 7 64bit이고, MS Visual Studio 2010 C++언어로 프로그램들이 개발되었다.

본 논문에서 제안한 알고리즘의 성능을 평가하기 위하여 기존의 알고리즘들 중에서 성능 평가의 기준으로 채택되는 알고리즘 PPJoin[4, 6-9]과 최근에 발표된 알고리즘인 APJoin [9, 10]을 구현하였다. 앞으로 알고리즘 PPJoin은 PP로 표기하고, PPJoin 알고리즘에 유사도 조인 후보 쌍을 검증하는 단계에서 중앙값 필터를 포함한 Fig. 2의 VerifyM()을 적용한 것은 PPMF로 표기한다. 알고리즘 APjoin은 AP로 표기하고, APJoin 알고리즘에 VerifyM()을 적용한 것은 APMF로 표기하기로 한다.

Table 1은 Enron 데이터집합에 대하여 자카드 유사도 한계치 t가 변화될 때 네 알고리즘에서 얻어진 각각의 유사도 조인 후보 쌍들의 개수를 두 번째 열에서부터 다섯 번째 열까지 표시하고 실제 유사도 조인 쌍들의 개수를 마지막 열에 표시하고 있다. Fig. 3은 네 알고리즘을 세 데이터집합에서 수행한 실행시간을 한계치에 따라 그래프로 보여주고 있고, Fig. 4는 중앙값 필터를 적용한 알고리즘을 세 데이터집합별로 상대적인 성능이익을 그래프로 설명하고 있다. 예를 들면, PPMF-Enron의 성능이익은 다음과 같이 계산한다.

$$Performance\,Gain_{PPMF-Enron} = 100\times(T_{PP-Enron} - T_{PPMF-Enron})/T_{PP-Enron}\%.$$

여기서 $T_{PPMF-Enron}$과 $T_{PP-Enron}$은 Enron 데이터집합에서 PPMF 알고리즘과 PP 알고리즘의 실행시간을 각각 나타낸다.

Table 1의 $|C_{PPMF}|$와 $|C_{APMF}|$는 중앙값 필터를 적용한 후의 후보들의 개수로 $|C_{PP}|$와 $|C_{AP}|$에 비해 크게 감소되는 것을 보여준다. 한 예로, 한계치 $t=0.8$일 때 $|C_{PPMF}|$는 $|C_{PP}|$에 비해 약 40% 정도로 그 개수가 줄어들고, 같은 한계치에서 $|C_{APMF}|$는 $|C_{AP}|$에 비해 48% 정도 더 줄어드는 것을 보여준다. Fig. 3(a)와 Fig. 4에서 그 실행시간을 비교해보면, PPMF의 성능이익은 약 52%이고 APMF는 약 29%가 된다. PPMF는 후보 개수가 줄어드는 비율에 비해서 APMF 보다 상대적으로 성능 개선이 더 우수한 것을 알 수 있다. 이것은 Fig. 2의 중간 단계 8과 11이 PPJoin 알고리즘[6]에서는 적용되지 않아서 중앙값 필터의 비용을 상쇄하고도 더 좋은 성능을 보여주지만, APJoin 알고리즘에서는 두 단계들이 적용되고 중앙값 필터를 계산하는 비용이 추가되어 성능 개선이 후보 수가 줄어드는 비율에 비해 더 적게 이루어졌다. 그런 까닭으로 Fig. 3과 4의 그래프에서 PPMF는 대략 20~70 % 사이의 높은 성능 개선을 보여주고 있다. Trec 데이터집합에서 한계치가 0.6과 0.8 사이일 때 PPMF의 성능이익이 상대적으로 다른 두 데이터집합에 비해서 70% 정도로 우수한 것은 유사도 조인 쌍들의 개수가 다른 것들에 비해서 훨씬 작기 때문이다. 예를 들면, 한계치 $t=0.7$일 때, Table 1의 $|SJoin|_{Enron}$=6,061,086개와 $|SJoin|_{DBLP}$=17,379개이고, $|SJoin|_{Trec}$=562개로 다른 데이터집합에 비해 1/30 이하다.

Fig. 4의 Enron 데이터집합에 대한 대부분의 자카드 유사도 한계치 구간에서 PPMF의 성능이익이 45% 이상이고 APMF의 성능이익은 20% 이상으로 중앙값 필터에 의한 성능 개선이 다른 데이터집합에서 보다 평균적으로 더 좋게 됨을 나타내고 있다. 그 이유를 설명하면, 중앙값 필터로 인한 성능 개선은 유사도 조인 쌍이 될 수 없는 후보 레코드들의 나머지 토큰들을 비교하지 않음으로써 일어나고 개선의 정도는 나머지 토큰들의 개수에 비례하는데 Enron 데이터집합의 평균 토큰 개수가 많기 때문이다. 그러므로 실험결과는 중앙값 필터를 적용한 APMF 알고리즘이 레코드들의 평균 토큰 개수가 아주 많은 데이터집합에서 유사도 조인 쌍을 빠르게 찾아냄을 보여준다.

Table 1. The number of similarity join pairs and the number of candidate pairs by algorithms on Enron dataset

t	$\|C_{PP}\|$	$\|C_{PPMF}\|$	$\|C_{AP}\|$	$\|C_{APMF}\|$	$\|SJoin\|$
0.6	313,215,063	295,449,722	442,460,330	305,159,966	9,809,066
0.65	163,986,648	136,029,026	221,105,699	140,814,632	8,894,058
0.7	79,838,350	58,372,375	104,030,136	60,703,266	6,061,086
0.75	35,734,847	22,964,234	45,196,031	23,992,140	3,477,781
0.8	15,110,283	9,073,627	18,365,578	9,480,527	2,729,318
0.85	6,287,360	4,055,639	7,233,319	4,184,646	2,113,168
0.9	3,108,266	2,467,064	3,292,171	2,505,509	1,437,101
0.95	1,393,473	1,273,183	1,409,082	1,277,615	1,076,620
1.0	951,019	928,509	951,019	928,509	895,144

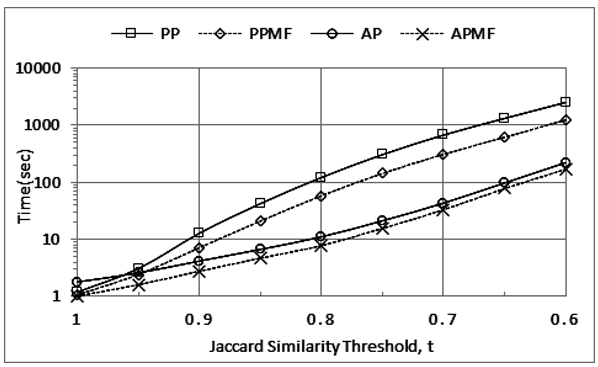

Fig. 3. (a) Running time on Enron

3) http://www.cs.cmu.edu/~enron/

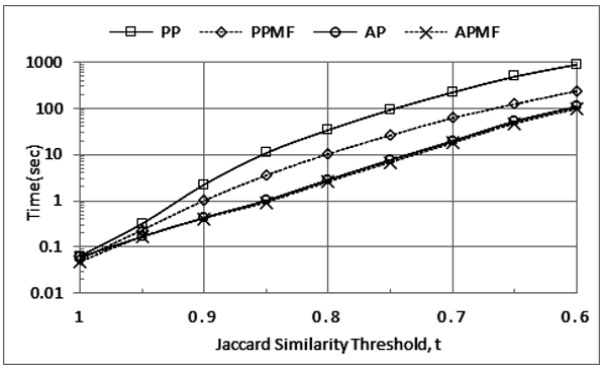

Fig. 3. (b) Running time on Trec

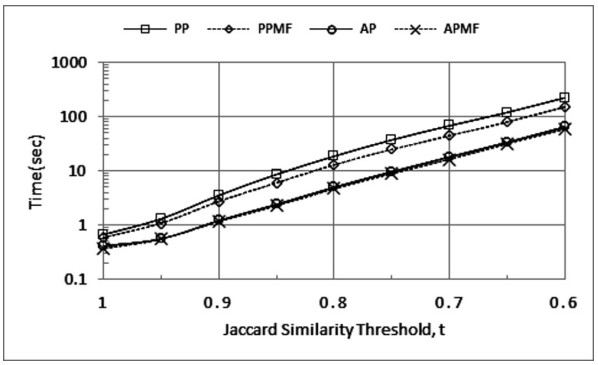

Fig. 3. (c) Running Time on DBLP

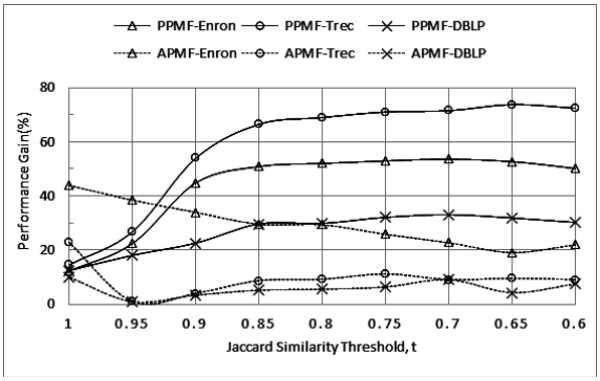

Fig. 4. Performance gain

5. 결 론

본 논문에서는 효율적인 유사도 조인 알고리즘을 개발하기 위하여 유사도 조인 후보 쌍을 검증하는 단계에서 간단한 중앙값 필터를 적용하여 실행 시간을 줄이는 새로운 알고리즘을 제안하였다. 알고리즘에서 사용된 중앙값 필터는 검증하려는 후보 쌍의 두 레코드들 중에서 한 레코드의 중앙값이 다른 레코드에서 올 수 있는 위치의 범위 이내에 공통되는 토큰들의 예상 개수가 주어진 유사도보다 작으면 두 레코드는 유사도 조인 쌍이 될 수 없다는 것이다. 제안된 알고리즘은 유사한 문서 검색이나 논문 표절 검사 등에 활용될 수 있다. 추후 연구과제로 유사도 조인 후보 생성 단계에서 효과적으로 후보들의 개수를 줄이는 색인구조나 필터에 대한 연구가 필요하다.

References

[1] L. Gravano, P.G. Ipeirotis, H.V. Jagadish, N. Koudas, S. Muthukrishnan, and D. Srivastava, “Approximate string joins in a database (almost) for free,” In *Proceedings of the 27th International Conference on Very Large Data Bases,* Roma, pp.491-500, 2001.

[2] S. Chaudhuri, V. Ganti, and R. Kaushik. “A primitive operator for similarity joins in data cleaning,” In *Proceedings of the 22nd International Conference on Data Engineering,* Atlanta, pp.5-16, 2006.

[3] M.R. Henzinger, “Finding near-duplicate web pages: A large-scale evaluation of algorithms,” In *Proceedings of the 29th annual international ACM SIGIR conference,* Seattle, pp.284-291, 2006.

[4] C. Xiao, W. Wang, X. Lin, and J.X. Yu, “Efficient Similarity Joins for Near Duplicate Detection,” In *Proceedings of the 17th international conference on World Wide Web,* Beijing, pp.131-140, 2008.

[5] R.J. Bayardo, Y. Ma, and R. Srikant, “Scaling up all pairs similarity search,” In *Proceedings of the 16th international conference on World Wide Web,* Banff, pp.131-140, 2007.

[6] J. Wang, G. Li, and J. Feng, “Can we beat the prefix filtering? An adaptive framework for similarity join and search,” In *Proceedings of the ACM SIGMOD International Conference on Management of Data,* Scottsdale, pp.85-96, 2012.

[7] Y. Jiang, G. Li, J. Feng, and W.-S., Li, “String Similarity Joins: An Experimental Evaluation,” In *the Proceedings of the 40th International Conference on Very Large Data Bases,* Hangzhou, pp.625-636, 2014.

[8] L.A. Ribeiro, T. Härder, “Generalizing prefix filtering to improve set similarity joins,” *Information Systems,* vol.36, Issue.1, pp.62-78, 2011.

[9] J.S. Park, “Efficient Similarity Joins by Adaptive Prefix Filtering,” *KIPS Transactions on Software and Data Engineering,* Vol.2, No.4, pp.167-272, 2013.

[10] J.S. Park, “A Sampling-based Algorithm for Top-k Similarity Joins,” *Journal of KIIES: Databases,* vol.41, No.4, pp.256-261, 2014.

박 종 수

e-mail : jpark@sungshin.ac.kr

1981년 부산대학교 전기기계공학과(학사)

1983년 1990년 한국과학기술원 전기 및 전자공학과(석사, 박사)

1983년～1986년 국방부 군무설계기좌

1994년～1995년 IBM Watson 연구소 객원연구원

1990년～현 재 성신여자대학교 IT학부 교수

관심분야 : Data mining, Database, Transportation geography