블로그 환경에서의 중복문서 핸들링을 위한 프레임워크

이순행, 이상철, 김상욱
한양대학교 전자컴퓨터통신공학과
e-mail:{shlee, korly, wook}@agape.hanyang.ac.kr

A Framework for Handling Duplicate Documents in a Blog Environment

Soon-Haeng Lee, Sang-Chul Lee, Sang-Wook Kim
Dept. of Electronics and Computer Engineering, Hanyang University

요 약

블로그 환경에서의 중복문서는 블로그 검색 서비스 성능의 저하를 초래한다. 기존의 웹 페이지 환경에서와는 달리 블로그 환경에서는 문서의 생성 시점을 알 수 있어 원본 문서와 중복문서를 쉽게 파악할 수 있다는 특징이 있다. 본 논문에서는 이 점에 착안하여 문서를 저장하는 시점에 중복 여부를 판정함으로써 검색 결과에 중복문서가 반영되는 것을 원천적으로 방지할 수 있는 효과적인 중복문서 핸들링 프레임워크를 제안한다. 또한, 성능 평가를 통해 제안하는 프레임워크의 우수성을 보인다.

1. 서론

블로그(blog)는 블로그 소유자인 블로거(blogger)가 자신의 생각을 온라인상에 게시할 수 있는 일종의 개인 웹 사이트이다[7][12].

다수의 블로거들이 다른 블로거의 좋은 문서를 자신의 블로그에 보관하기 위해 중복문서를 생성한다. 중복문서(duplicate document)란 기존의 다른 문서와 내용이 완전히 일치하거나 극히 일부 내용만을 수정한 문서를 말한다. 일부 블로그 서비스 업체들[12]에서는 중복문서를 생성할 수 있도록 타 블로거의 문서를 자신의 블로그로 복사해 올 수 있는 스크랩 기능을 제공한다. 또한, 원본 문서의 내용을 그대로 복사 후 붙여넣기(copy&paste)하거나 복사 후 붙여넣기 한 다음 일부 내용을 수정하여 중복문서를 생성하기도 한다.

블로거들에 의해 생성된 중복문서들은 블로그 검색 서비스 질을 저하시킨다. 검색 서비스 사용자들은 거의 동일한 내용의 문서들로 이뤄진 검색 결과를 열람하게 된다. 또한, 검색 인덱스의 크기를 증가시키기 때문에 검색 시간을 지연시킨다. 이와 같은 문제점을 해결하기 위하여 블로그 환경에서 중복문서를 효과적으로 검출하고, 검출된 문서들을 핸들링 할 수 있는 방법이 필요하다.

중복문서 검출에 관한 대부분의 기존연구들은 웹 페이지 환경에서 진행되었으며, 저자들이 아는 한 블로그 환경을 대상으로 추진한 기존연구는 없다. 또한, 대부분의 연구들은 임의의 두 문서가 주어졌을 때 두 문서간의 중복 여부를 판정하는 기법에 초점이 맞추어져 왔다. 웹 페이지 환경에서의 중복문서 판정 기법은 크게 문서로부터 연속적인 k개의 단어들로 구성된 쉉글을 추출하여 문서간의 중복 여부를 판정하는 쉉글 기반 기법(shingle based algorithms)[2][3][4][9][10]과 문서로부터 의미가 있는 단어만을 추출하여 문서간의 중복 여부를 판정하는 단어 기반 기법(term based algorithms)[5][6]으로 나눠진다[16]. 또한, 웹 페이지 환경의 특성 상 검출한 중복문서들 중 원본 문서를 알 수 없기 때문에 중복문서들끼리 클러스터링하고, 이들 중 대표적인 문서만을 검색 결과에 보여주는 방식을 사용한다[2].

웹 페이지 환경과는 달리 블로그 환경에서는 문서가 작성된 시점 순으로 문서가 작성 시각과 함께 데이터베이스와 검색 인덱스에 저장되는 특징을 갖는다. 따라서 중복된 문서들 중 생성 시각이 가장 앞선 것을 원본 문서로 판단할 수 있으며, 이후에 생성된 중복문서들을 안전하게 검색 대상으로부터 제거시킬 수 있다. 본 논문에서는 이러한 블로그 환경의 특성에 착안하여 중복문서를 생성 시점에 검출하여 검색 대상에서 제거하는 효과적인 프레임워크를 제안한다.

이러한 블로그 환경에서의 특성에 착안하여 제안된 중복문서 핸들링 프레임워크에서는 문서가 생성되는 시점에 Min-hashing을 기반으로 인덱스를 이용하여 중복 여부를 빠르게 판정한다. 또한, 중복으로 판정된 문서들은 검색 서비스를 위한 인덱스에 추가시키지 않음으로써 검색 결과에 반영시키지 않도록 한다.

2. Min-hashing

Min-hashing[2][4][9][10]은 문서로부터 고정 개수의 일부 특징들만 추출하여 두 문서의 유사한 정도를 근사하게 측정하는 기법이다. 먼저, 문서로부터 연속적인 w개의 단

어들로 구성된 시퀀스인 쉉글(shingle)을 추출한다. 문서 내 단어들의 개수가 n일 때, $n-w+1$개의 쉉글들이 추출된다. 또한, 추출된 쉉글은 Rabin's fingerprinting 함수[13]를 사용하여 고정된 크기의 정수 값으로 변환한다.

두 문서로부터 추출된 쉉글들을 비교하여 두 문서간의 유사한 정도를 나타낼 수 있다. 두 문서 간의 유사한 정도를 측정하는 척도로 Jaccard coefficient[9]를 사용한다. 임의의 문서 A와 B의 쉉글들의 집합을 각각 S_A, S_B로 표현할 때, 문서 A, B의 Jaccard coefficient $r(A,B)$는 아래 식 (1)과 같이 정의된다.

$$r(A,B) = \frac{|S_A \cap S_B|}{|S_A \cup S_B|} \quad \text{.....(1)}$$

최소 독립 순열(min-wise independent permutations)[3]을 사용하면 문서 내 모든 쉉글들을 동일한 확률로 추출할 수 있다[4]. 임의의 집합 $X \subseteq [n] (=\{0, \dots, n-1\})$가 주어지고, 임의의 원소 $x \in X$와 $[n]$의 순열들의 집합 F가 주어졌을 때, 랜덤하게 선택된 순열 $\pi \in$F가 식 (2)를 만족하면 F를 최소 독립 순열이라고 한다[4].

$$\Pr(\min\{\pi(X)\} = \pi(x)) = \frac{1}{|X|} \quad \text{.....(2)}$$

이때, 임의의 원소 $x \in X$가 π을 기준으로 집합 X에서 가장 작은 원소로 추출될 확률이 모든 x에 대해서 동일하므로, 다음 식 (3)을 만족하게 된다.

$$\Pr(\min\{\pi(S_A)\} = \min\{\pi(S_B)\}) = \frac{|S_A \cap S_B|}{|S_A \cup S_B|} = r(A,B) \quad \text{.....(3)}$$

따라서 최소 독립 순열을 사용하여 문서 A와 B로부터 순열 개수만큼의 쉉글을 추출하여 r(A,B)를 효과적으로 계산할 수 있게 된다. 그러나 실질적으로는 최소 독립 순열을 만족시키는 적절한 순열들의 집합을 찾는 것은 많은 시간을 많은 시간을 필요로 하기 때문에 최소 독립 순열 대신 선형 순열(linear permutations)을 사용한다[4].

임의의 두 문서 A, B가 주어졌을 때, 두 문서의 중복 여부는 다음과 같이 판정된다. 먼저, 선형 순열 t개를 이용하여 각각의 문서에서 t개의 정수 값들을 추출해낸다. 결국 한 문서는 t개의 순서를 가진 정수 값들의 리스트로 표현된다. 임의의 두 문서를 비교할 경우 t개의 정수 값들을 순서대로 비교함으로써 정수 값들이 일치하는 정도를 계산한다. 또한, 두 문서간의 유사한 정도를 계산할 경우 시간과 공간 복잡도를 줄이기 위해 $t(=s \times k)$개의 정수 값들을 s개씩 k개의 그룹으로 나눈다. 각각의 그룹 내 포함된 s개의 정수 값들을 다시 Rabin's fingerprinting 함수[13]를 사용하여 최종적으로 고정 크기를 갖는 k개의 정수 값을 추출한다. 본 논문에서는 이렇게 추출된 k개의 정수 값들을 문서로부터 추출된 k개의 특징이라고 부르기로 한다. 문서 A와 B의 k개의 특징들 중 r개 이상이 동일하면 두 문서는 중복으로 판정된다.

Min-hashing에서 사용되는 파라메타 k, s, r는 대부분 6, 14, 2를 사용한다[4][8][10]. 이와 같은 실험변수를 적용할 경우 $r(A,B)$가 0.975보다 큰 값을 가질 확률이 1-0.01을 만족하며, $r(A,B)$가 0.77보다 작을 확률이 0.01보다 작다[4]. 따라서 기존 연구의 파라메타를 그대로 적용하여 84개 선형 순열을 이용하여 각각의 문서에서 6개의 특징을 추출하고, 임의의 두 문서가 2개 이상 동일한 특징을 가질 경우 중복문서로 판정한다.

3. 제안하는 프레임워크

3.1 중복문서 판정 시점

본 논문에서는 이러한 기존 웹 페이지 환경에서의 중복문서 검출 시점들이 블로그 환경에서 적합한 지의 여부를 고려하였다. 첫 번째, 오프라인 시점은 오프라인 상태에서 데이터베이스 내 저장되어 있는 모든 문서들을 대상으로 일괄적으로 처리하는 것을 말한다[11]. 대부분의 중복문서 검출 알고리즘들이 오프라인 시점에서 수행되며[2][6][8][10][14][15], 일괄적으로 모든 문서 쌍에 대해 중복을 판정하여 중복된 문서들끼리 클러스터링하고 대표적인 문서만을 검색 결과에 보여준다[2]. 오프라인 상태에서 데이터베이스에 저장되어 있는 문서들로부터 중복문서를 검출하기 때문에 검색 엔진에 별도의 오버헤드를 요구하지 않는다. 그러나 생성된 문서가 이미 검색 인덱스에 저장되어 있기 때문에 중복문서가 질의 결과에 나타날 수 있다는 문제점이 있다.

두 번째, 질의 처리 시점은 온라인상에서 검색 엔진에 의해 처리된 질의 결과 내 문서들을 대상으로 중복을 검출하는 것을 말한다[16]. 이때, 중복된 문서들 중 가장 높은 랭크를 갖는 문서를 제외한 나머지 모든 문서를 실시간으로 검색 결과에서 제외시킨다[16]. 수많은 질의가 처리될 때마다 중복 검출로 인한 오버헤드가 발생하므로 블로그 검색의 성능이 저하된다는 문제점이 있다.

세 번째, 크롤 시점은 크롤러(crawler)에 의해 새로운 문서가 크롤되는 시점에서 해당 문서가 이미 크롤된 문서와 중복인지 판정하는 것을 말한다[11]. 중복으로 판정된 문서에 대해 크롤을 실행하지 않거나 크롤하는 우선순위를 낮추는 것으로 중복문서를 핸들링한다[11]. 블로그 환경에서는 문서가 생성될 때마다 데이터베이스와 검색 인덱스에 바로 저장하기 때문에 크롤이 불필요하다. 따라서 블로그 환경에서는 크롤시점을 적용할 수 없다.

블로그 환경에서는 웹 페이지 환경과는 다르게 문서가 생성되는 시점을 알 수 있으므로, 새로 생성된 문서가 데이터베이스와 검색 인덱스에 저장되기 전에 문서의 중복 여부를 판정할 수 있다. 이것은 기존 웹 페이지 환경에서는 적용할 수 없는 새로운 검출 시점이다. 본 논문에서는 문서가 새로 생성될 때 중복 여부를 판정하는 것을 문서 저장 시점이라 부른다. 문서 저장 시점에 중복문서를 검출할 경우 블로그 검색 엔진에 영향을 주지 않으며, 중복문서가 검색 결과에 반영되는 것을 사전에 방지할 수 있게 된다. 또한, 검출된 중복문서들 중에서 생성된 시각이 가장 앞선 것을 원본 문서로 볼 수 있다. 새로 생성된 문서

가 중복으로 판정될 경우에는 해당 문서를 작성한 블로거가 소장할 수 있도록 데이터베이스에는 반영하되, 검색 서비스를 이용하는 사용자들을 위해서 검색 인덱스에는 반영하지 않는다.

이와 같이 중복문서를 처리함으로써 중복된 문서는 검색 인덱스에서 제외시킬 수 있으므로 인덱스 내 불필요한 공간을 낭비하는 일을 방지할 수 있다. 또한 사용자의 검색 결과에 중복문서가 반영되는 것을 원천적으로 방지할 수 있다. 그러나 문서가 생성되어 데이터베이스에 저장되는 시점에 중복 여부를 판정하므로 문서가 저장되는 시간이 다소 지연된다. 따라서 문서 저장 시점에서 데이터베이스에 저장되어 있는 대용량의 문서들과의 중복 여부를 빠르게 판정할 수 있는 효과적인 기법이 요구된다.

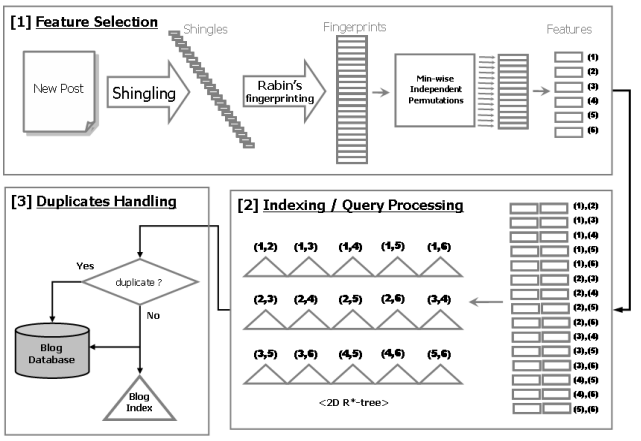
3.2 중복문서 핸들링을 위한 프레임워크

그림 3.1은 본 논문에서 제안하는 중복문서 핸들링 프레임워크를 나타낸 것이다. 첫 번째로 새롭게 생성된 문서로부터 특징을 추출하기 위한 특징추출 과정, 두 번째로 R*-트리[1]를 사용하여 추출된 특징들을 저장하는 인덱싱 및 질의 처리 과정, 그리고 세 번째로 검출된 중복문서 처리 과정으로 구성된다.

그림 3.1의 [과정 1]은 중복문서 판정을 위한 특징추출 과정을 나타낸 것이다. 먼저, HTML 문서 형태로 되어 있는 문서로부터 HTML 태그를 분석하여 블로그 프레임을 제외한 코어 텍스트를 추출한다. 이렇게 추출된 코어 텍스트로부터 w개의 단어들로 구성되는 쉥글을 추출해낸다. 각 쉥글을 Min-hashing 기법[1]에서 사용하는 Rabin's fingerprinting 함수[1][4][13]를 사용하여 고정 크기의 정수 값으로 변환한다. 84개의 선형 순열(linear permutation)을 사용하여 전체 쉥글들과 대응하는 정수 값들로부터 84개의 정수 값들을 추출한다. 추출된 84개의 정수 값은 순서대로 14개씩 6개의 그룹으로 나눈 다음 각각의 그룹 내 포함된 정수 값들을 접합(concatenate)시킨 후, Rabin's fingerprinting 함수를 사용하여 최종적으로 순서를 가진 6개의 고정 크기 정수 값으로 변환한다.

문서 저장 시점에서 저장되는 문서와 이미 데이터베이스에 저장된 대용량의 문서들 간의 중복 여부를 빠르게 판정하기 위해서 인덱싱이 필요하다. 그림 3.1의 [과정 2]는 중복문서 검출을 위한 인덱스 구축 과정을 나타낸 것이다. 문서로부터 순서를 가진 6개의 특징들을 인덱스에 저장해야 하므로 2개씩 가능한 모든 조합을 생성하여 인덱스 키를 생성한다. 6개의 특징의 순서를 고려하여 2개를

1) 중복문서 판정 기법은 기존 웹 페이지 환경에서 연구된 쉥글 기반 기법과 단어 기반 기법을 모두 적용할 수 있다. 그러나 단어 기반 기법은 문서의 구조적인 정보를 고려하지 않으므로 [16] 복사 후 붙여넣기(copy&paste)하여 생성한 중복문서를 판정을 위해서는 쉥글 기반 기법이 적합하다. 따라서 제안하는 프레임워크에서는 대표적인 쉥글 기반 기법인 Min-hashing [5]을 사용하여 새로 생성된 문서로부터 특징을 추출하였다.



(그림 3.1) 제안하는 중복문서 핸들링 프레임워크

선택하는 경우의 수는 ${}_6C_2$이며, 총 15종류의 인덱스 키가 생성된다. 본 논문에서는 이러한 15종류의 인덱스 키들을 위한 15개의 인덱스를 구축한다. 각 인덱스 키는 2개의 서로 다른 특징을 가지므로, 인덱스 구조로서 다차원 인덱스의 하나인 R*-트리[1]를 이용한다.

임의의 문서가 새롭게 생성되면, 인덱스를 이용한 질의 처리를 통해 문서의 중복을 판정한다. 특징추출 과정을 통해 생성된 문서에서 추출된 6개의 특징들로부터 15개의 인덱스 키를 생성한다. 각 인덱스 키가 이미 존재하는 가를 판정하기 위하여 인덱싱 과정에서 생성한 15개의 2차원 R*-트리를 대상으로 인덱스 검색을 차례로 수행한다. 이때, 15개의 2차원 R*-트리 중 적어도 하나로부터 해당 인덱스 키가 검색되면 생성된 문서를 중복문서로 간주하고 이후의 인덱스 검색을 중단한다.

그림 3.1의 [과정 3]은 검출된 중복문서를 처리하는 과정을 나타낸 것이다. 중복문서가 아닌 것으로 판정된 경우에는 검색 인덱스에 반영하고 데이터베이스에 저장한다. 또한, 이후의 중복문서 검출을 위하여 해당 문서의 인덱스 키를 15개의 2차원 R*-트리에 반영시킨다. 반면, 중복문서로 판정된 경우에는 중복문서를 데이터베이스에는 저장하되, 해당 문서는 검색 인덱스에 반영하지 않는다.

4. 성능 평가

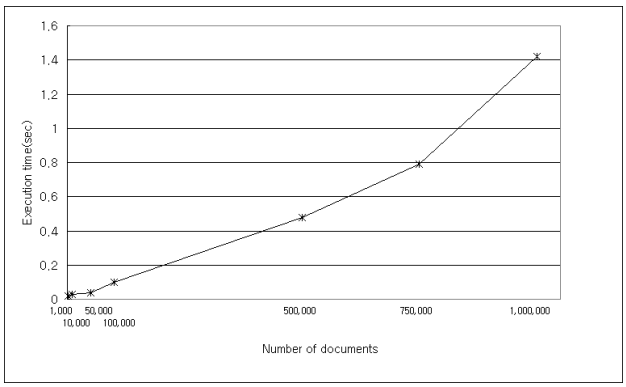
4.1 실험 환경

본 논문에서는 제안한 프레임워크의 성능을 평가하여 인덱싱 및 질의 처리 기법의 우수성을 보인다. 본 논문에서는 Min-hashing의 기본 메커니즘을 그대로 적용하였기 때문에 중복문서 판정의 정확도는 별도로 검증하지 않는다.

본 실험에서는 성능을 평가하기 위하여 블로그 사이트의 하나인 이글루스(egloos)[7]로부터 수집한 1,000,183개 문서를 사용하였다. 데이터 셋의 증가에 따른 제안하는 기법의 성능 변화를 검증하기 위하여 수집한 문서들로부터 각각 1,000개, 10,000개, 50,000개, 100,000개, 500,000개, 750,000개, 1,000,000개를 추출하여 7개의 실험 데이터 집합을 구성하였다. 각 실험데이터 집합을 중복 판정 인덱스

에 저장한 후 중복문서 검출을 위해 각 실험 데이터 집합에서 1,000개의 문서를 무작위로 선정하고, 1,000개의 문서에 대한 평균 중복문서 판정 시간을 측정하였다. 질의 처리 성능 실험을 위하여 본 논문에서는 2G의 메모리 크기와 3 GHz 펜티엄 4 CPU를 가진 윈도우즈 XP 미디어 센터를 이용하여 수행하였다.

4.2 실험 결과



(그림 4.2) 성능 평가 결과

그림 4.2는 인덱스에 저장된 문서 개수의 증가함에 따른 질의 처리 시간의 변화를 나타낸 것이다. 실험 데이터가 1,000,000개 문서일 때, 1.42초가 소요되어 제안한 기법이 효율적으로 중복여부를 판정하는 것으로 나타났다. 또한, 문서의 개수가 증가함에 따라 질의처리 시간이 선형적으로 증가하였다. 따라서 제안하는 프레임워크가 대용량의 웹문서에서 중복문서 검출에 적합한 것으로 나타났다.

5. 결론

블로그 환경에서의 중복문서는 블로그 검색 서비스의 질적 저하를 초래한다. 기존의 웹 페이지 환경에서와는 달리 블로그 환경에서는 문서의 생성 시점을 알 수 있어 원본 문서와 중복문서를 쉽게 파악할 수 있다. 본 논문에서는 이 점에 착안하여 문서를 저장하는 시점에 중복 여부를 판단함으로써 검색 결과에 중복문서가 반영되는 것을 원천적으로 방지할 수 있는 효과적인 중복문서 핸들링 프레임워크를 제안하였다. 또한, 실험에 의한 성능 평가를 통해 제안하는 프레임워크의 우수성을 보였다. 1,000,000여개의 문서들을 가지는 실제 블로그 환경을 대상으로 한 성능 평가 결과, 제안된 방식은 중복 여부를 매우 효율적으로 판정하는 것으로 나타났다.

감사의 글

본 연구는 지식경제부 및 정보통신연구진흥원의 대학 IT 연구센터 지원사업의 연구결과로 수행되었습니다.(IITA-2008-C1090- 0801-0040)

참고문헌

[1] N. Beckmann et al., "The R*-tree: An Efficient and Robust Access Method for Points and Rectangles," In *Proc. ACM Int'l. Conf. on Management of Data*, SIGMOD, pp. 322-331, 1990.

[2] A. Broder et al., "Syntactic Clustering of the Web," In *Proc. Int'l. Web Wide World Wide Web Conference*, WWW, pp. 391-404, 1997.

[3] A. Broder et al., "Min-Wise Independent Permutations," *Journal of Computer and System Sciences*, Vol. 60, No. 3, pp. 630-659, 2000.

[4] A. Broder, "Identifying and Filtering Near-Duplicate Documents," In *Proc. Int'l. Symp. on Combinatorial Pattern Matching*, CPM, pp. 1-10, 2000.

[5] A. Chowdhury et al., "Collection Statistics for Fast Duplicate Document Detection," *Transactions on Information System(TOIS),* Vol. 20, No. 2, pp. 171-191, 2002.

[6] J. Cooper, A. Coden, and E. Brown, "Detecting Similar Documents Using Salient Terms," In *Proc. Int'l. Conf. on Information and Knowledge Management,* CIKM, pp. 245-251, 2002.

[7] (주)SK Communications http://www.egloos. com.

[8] D. Fetterly, M. Manasse, and M. Najork, "On the Evolution of Clusters of Near-Duplicate Web Pages," In *Proc. Int'l. Conf. on the 1st Latin American Web Congress,* LA-WEB, pp. 37-45, 2003.

[9] T. Haveliwala et al., "Evaluating Strategies for Similarity Search on the Web," In *Proc. Int'l. World Wide Web Conference*, WWW, pp. 432-442, 2002.

[10] M. Henzinger, "Finding Near-Duplicate Web Pages: A Large-Scale Evaluation of Algorithms," In *Proc. ACM Int'l.. Conf. on Information Retrieval*, SIGIR, pp. 284-291, 2006.

[11] G. Manku, A. Jain, and A. Sarma, "Detecting Near-Duplicates for Web Crawling," In *Proc. Int'l. World Wide Web Conference*, WWW, pp. 141-149, 2007.

[12] (주)NHN, 블로그홈, http://blog.naver.com.

[13] M. Rabin, "Fingerprinting by Random Polynomials," Technical Report TR-CSE-03-01, Harvard University, 1981.

[14] N. Shivakumar and H. García-Molina, "SCAM: A Copy Detection Mechanism for Digital Documents," In *Proc. Int'l.. Conf. on Theory and Practice of Digital Libraries,* DL, pp. 155-163, 1995.

[15] N. Shivakumar and H. García-Molina, "Finding Near-Replicas of Documents on the Web," In *Proc. Int'l.. Conf. on Web Databases*, WebDB, pp. 204-212, 1998.

[16] S. Ye et al., "A Query-Dependent Duplicate Detection Approach for Large Scale Search Engines," In *Proc. Int'l.. Conf. on Asia-Pacific Web Conference*, APWeb, pp. 48-58, 2004.