(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2011-0009098
(43) 공개일자 2011년01월27일

(51) Int. Cl.
G06F 17/21 (2006.01) *G06F 17/27* (2006.01)
G06F 17/30 (2006.01)
(21) 출원번호 10-2010-7022177
(22) 출원일자(국제출원일자) 2009년03월10일
심사청구일자 없음
(85) 번역문제출일자 2010년10월04일
(86) 국제출원번호 PCT/US2009/036597
(87) 국제공개번호 WO 2009/126394
국제공개일자 2009년10월15일
(30) 우선권주장
12/101,951 2008년04월11일 미국(US)

(71) 출원인
마이크로소프트 코포레이션
미국 워싱턴주 (우편번호 : 98052) 레드몬드 원 마이크로소프트 웨이
(72) 발명자
탄코비치, 블라디미르
미국 98052-6399 워싱턴주 레드몬드 원 마이크로소프트 웨이 마이크로소프트 코포레이션 엘씨에이 국제 특허부 내
리, 항
미국 98052-6399 워싱턴주 레드몬드 원 마이크로소프트 웨이 마이크로소프트 코포레이션 엘씨에이 국제 특허부 내
(뒷면에 계속)
(74) 대리인
양영준, 백만기

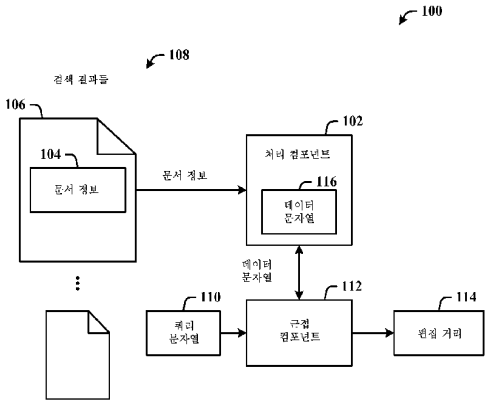
전체 청구항 수 : 총 20 항

(54) 편집 거리 및 문서 정보를 이용한 검색 결과 랭킹

(57) 요 약

쿼리 문자열에 기초하여 검색 결과들로서 수신된 문서들로부터 문서 정보를 추출하고, 데이터 문자열과 상기 쿼리 문자열 사이의 편집 거리를 계산하기 위한 아키텍처. 상기 편집 거리는 전체 쿼리 또는 쿼리의 일부의 거의 매칭하는 것들을 검출함으로써 결과 랭킹의 일부로서 문서의 관련성을 결정하는 데에 채용된다. 상기 편집 거리는 상기 쿼리 문자열이 TAUC(제목, 앵커 텍스트, URL, 및 클릭) 정보 등과 같은 문서 정보를 포함하는 주어진 데이터 스트림에 얼마나 가까운지를 평가한다. 본 아키텍처는 쿼리 용어들의 보다 효과적인 발견을 허용하기 위해 상기 URL 내의 복합 용어들의 인덱스-타임 분할을 포함한다. 또한, 문서 결과들 중 하나 이상 문서 결과들의 최상위 N개의 앵커들을 찾아내기 위해 앵커 텍스트의 인덱스-타임 필터링이 이용된다. 상기 TAUC 정보는 상기 검색 결과들을 랭킹하기 위한 관련성 메트릭들을 개선하기 위해 신경망(예를 들면, 2-계층)에 입력될 수 있다.

대 표 도 - 도1



(72) 발명자

메이어존, 드미트리

미국 98052-6399 워싱턴주 레드몬드 원 마이크로소프트 웨이 마이크로소프트 코포레이션 엘씨에이 국제 특허부 내

수, 준

미국 98052-6399 워싱턴주 레드몬드 원 마이크로소프트 웨이 마이크로소프트 코포레이션 엘씨에이 국제 특허부 내

특허청구의 범위

청구항 1

컴퓨터 구현된 관련성 시스템(100)으로서,

쿼리 문자열에 기초하여 검색 결과들로서 수신된 문서들로부터 문서 정보를 추출하기 위한 처리 컴포넌트(102); 및

데이터 문자열과 상기 쿼리 문자열 사이의 편집 거리를 계산하기 위한 근접 컴포넌트(proximity component)(112) - 상기 편집 거리는 결과 랭킹의 일부로서 문서의 관련성을 결정하는 데에 채용됨 -

를 포함하는 시스템.

청구항 2

제1항에 있어서, 상기 데이터 문자열을 생성하기 위해 채용되는 상기 문서 정보는 제목 정보, URL 정보, 클릭 정보, 또는 앵커 텍스트(anchor text) 중 적어도 하나를 포함하는 시스템.

청구항 3

제1항에 있어서, 상기 처리 컴포넌트는 URL에 관하여 상기 편집 거리를 계산하기 위해 인덱스 타임에 상기 문서 정보의 복합 용어들(compound terms)을 분할하는 시스템.

청구항 4

제1항에 있어서, 상기 처리 컴포넌트는 앵커 텍스트의 최상위 랭킹된 세트를 계산하기 위해 인덱스 타임에 상기 문서 정보의 앵커 텍스트를 필터링하는 시스템.

청구항 5

제1항에 있어서, 상기 문서 정보는 제목 문자들, 앵커 문자들, 클릭 문자들, 또는 URL 문자들 중 적어도 하나를 포함하고, 그 문서 정보는 상기 문서의 관련성을 계산하기 위해 BM25F 함수, 클릭 거리, 파일 유형, 언어 및 URL 깊이의 원시 입력 특징들과 함께 신경망에 입력되는 시스템.

청구항 6

제1항에 있어서, 상기 편집 거리의 계산은 상기 데이터 문자열과 상기 쿼리 문자열 사이의 근접을 증가시키기 위해 용어들의 삽입 및 삭제에 기초하는 시스템.

청구항 7

제1항에 있어서, 상기 편집 거리의 계산은 상기 데이터 문자열과 상기 쿼리 문자열 사이의 근접을 증가시키기 위해 용어들의 삽입 및 삭제와 관련된 비용들에 기초하는 시스템.

청구항 8

관련성을 결정하는 컴퓨터 구현된 방법으로서,

검색 프로세스의 일부로서 쿼리 문자열을 수신하는 단계(1000);

상기 검색 프로세스 동안에 반환된 문서로부터 문서 정보를 추출하는 단계(1002);

상기 문서 정보로부터 데이터 문자열을 생성하는 단계(1004);

상기 데이터 문자열과 상기 쿼리 문자열 사이의 편집 거리를 계산하는 단계(1006); 및

상기 편집 거리에 기초하여 관련성 스코어를 계산하는 단계(1008)

를 포함하는 컴퓨터 구현된 방법.

청구항 9

제8항에 있어서, 상기 편집 거리를 계산하는 단계의 일부로서 용어 삽입을 채용하고 상기 데이터 문자열을 생성하기 위한 쿼리 문자열 내의 용어의 삽입을 위한 삽입 비용을 평가하는 단계를 더 포함하고, 상기 비용은 가중 파라미터로서 표현되는 컴퓨터 구현된 방법.

청구항 10

제8항에 있어서, 상기 편집 거리를 계산하는 단계의 일부로서 용어 삭제를 채용하고 상기 데이터 문자열을 생성하기 위한 상기 쿼리 문자열 내의 용어의 삭제를 위한 삭제 비용을 평가하는 단계를 더 포함하고, 상기 비용은 가중 파라미터로서 표현되는 컴퓨터 구현된 방법.

청구항 11

제8항에 있어서, 상기 편집 거리를 계산하는 단계의 일부로서 위치 비용을 계산하는 단계를 더 포함하고, 상기 위치 비용은 상기 데이터 문자열에서의 용어 위치의 용어 삽입 및/또는 용어 삭제와 관련되는 컴퓨터 구현된 방법.

청구항 12

제8항에 있어서, 상기 편집 거리를 계산하는 단계의 전체적인 비용을 계산하기 위해 상기 데이터 문자열의 문자들과 상기 쿼리 문자열의 문자들 사이에 매칭 프로세스를 수행하는 단계를 더 포함하는 컴퓨터 구현된 방법.

청구항 13

제8항에 있어서, 인덱스 타임에 상기 데이터 문자열의 URL의 복합 용어들을 분할하는 단계를 더 포함하는 컴퓨터 구현된 방법.

청구항 14

제8항에 있어서, 상기 문서에서의 발생의 빈도에 기초하여 앵커 텍스트의 최상위 랭킹된 세트를 찾아내기 위해 상기 데이터 문자열의 앵커 텍스트를 필터링하는 단계를 더 포함하는 컴퓨터 구현된 방법.

청구항 15

제14항에 있어서, 상기 세트 내의 앵커 텍스트에 대한 편집 거리 스코어를 계산하는 단계를 더 포함하는 컴퓨터 구현된 방법.

청구항 16

제8항에 있어서, 상기 편집 거리를 계산하는 단계로부터 얻어진 스코어를 변환 함수의 적용 후에 2-계층 신경망에 입력하는 단계를 더 포함하고, 상기 스코어는 제목 정보, 앵커 정보, 클릭 정보, 또는 URL 정보, 및 기타 원시 입력 특징들 중 적어도 하나와 관련된 상기 편집 거리를 계산하는 것에 기초하여 생성되는 컴퓨터 구현된 방법.

청구항 17

문서의 관련성을 계산하는 컴퓨터 구현된 방법으로서,

검색 프로세스의 일부로서 쿼리 문자열을 처리하여 문서들의 결과 세트를 반환하는 단계(1100);

상기 결과 세트의 문서로부터 추출된 문서 정보에 기초하여 데이터 문자열을 생성하는 단계(1102) - 상기 문서 정보는 상기 문서로부터의 제목 정보, 앵커 텍스트 정보, 클릭 정보, 및 URL 정보 중 하나 이상을 포함함 -;

용어 삽입, 용어 삭제, 및 용어 위치에 기초하여 상기 데이터 문자열과 상기 쿼리 문자열 사이의 편집 거리를 계산하는 단계(1104); 및

상기 편집 거리에 기초하여 관련성 스코어를 계산하는 단계(1106) - 상기 관련성 스코어는 상기 결과 세트에서 상기 문서를 랭킹하기 위해 이용됨 -

를 포함하는 컴퓨터 구현된 방법.

청구항 18

제17항에 있어서, 상기 용어 삽입, 용어 삭제 및 용어 위치의 각각과 관련된 비용을 계산하고, 상기 비용을 상기 관련성 스코어의 계산에 넣는(factoring) 단계를 더 포함하는 컴퓨터 구현된 방법.

청구항 19

제17항에 있어서, 인덱스 타임에 상기 URL 정보의 복합 용어들을 분할하고, 상기 문서에서 앵커 텍스트의 발생의 빈도에 기초하여 상기 앵커 텍스트의 최상위 랭킹된 세트를 찾아내기 위해 인덱스 타임에 상기 앵커 텍스트 정보를 필터링하는 단계를 더 포함하는 컴퓨터 구현된 방법.

청구항 20

제17항에 있어서, 상기 쿼리 문자열의 용어들의 발생들을 판독하여 원본 URL 문자열에서 나타나는 순서로 쿼리 용어들의 문자열을 구성하고 그 용어들 사이의 공간을 단어 마크들(word marks)로 채우는 단계를 더 포함하는 컴퓨터 구현된 방법.

명 세 서

배 경 기 술

[0001] 전형적인 검색 엔진 서비스에서 사용자는 쿼리와 매칭하는 URL들(universal resource locators)의 인덱싱된 컬렉션으로부터 최상의 관련 있는 문서들을 선택함으로써 쿼리를 입력할 수 있다. 쿼리들에 신속히 대응하기 위하여 검색 엔진은 키워드들을 문서들에 매핑하는 하나 이상의 방법들(예를 들면, 역 인덱스(inverted index) 데이터 구조)을 이용한다. 예를 들면, 엔진에 의해 수행되는 제1 단계는 사용자 쿼리에 의해 특정된 키워드들을 포함하는 후보 문서들의 세트를 식별하는 것일 수 있다. 이들 키워드들은 문서 본문 또는 메타데이터에, 또는 실제로는 (앵커 텍스트(anchor text)와 같은) 다른 문서들 또는 데이터 저장소들에 저장되어 있는 이 문서에 관한 추가적인 메타데이터에 위치할 수 있다.

[0002] 큰 인덱스 컬렉션에서 후보 문서 세트의 카디널러티(cardinality)는, 쿼리 용어들의 공통성에 따라서, 클 수 있다(예를 들면, 잠재적으로 수백만). 후보 문서들의 전체 세트를 반환하는 대신에 검색 엔진은 관련성에 관하여 후보 문서들을 랭킹하는 제2 단계를 수행한다. 전형적으로, 검색 엔진은 특정한 쿼리와의 문서의 관련성의 정도를 예측하기 위해 랭킹 함수를 이용한다. 랭킹 함수는 입력으로서 문서로부터 다수의 특징들을 취하고 검색 엔진이 예측된 관련성에 의해 문서들을 정렬하도록 허용하는 수를 계산한다.

[0003] 함수가 문서의 관련성을 얼마나 정확히 예측하는지에 관한 랭킹 함수의 품질은 결국 검색 결과들에 대한 사용자 만족 또는 사용자가 제기된 질문에 대한 응답을 평균하여 몇 회 얻는가에 의해 결정된다. 시스템에 대한 전체적인 사용자 만족은 단 하나의 수(또는 메트릭)에 의해 어림될 수 있는데, 이는 그 수는 랭킹 함수를 변경하는 것에 의해 최적화될 수 있기 때문이다. 통상적으로, 그 메트릭들은 쿼리 로그들(query logs)의 임의 샘플링에 의해 미리 선택되는 쿼리들의 대표적인 세트에 걸쳐서 계산되고, 평가 쿼리들 각각에 대하여 엔진에 의해 반환된 각 결과에 관련성 라벨들을 할당하는 것을 수반한다. 그러나, 문서 랭킹 및 관련성에 대한 이들 프로세스들은 원하는 결과를 제공하는 데는 여전히 효력이 없다.

발명의 내용

[0004] 하기의 내용은 여기에 설명된 일부 새로운 실시예들에 대한 기초적인 이해를 제공하기 위하여 간략화된 개요를 제시한다. 이 개요는 광범한 개관이 아니며, 그것은 중요한/결정적인 요소들을 식별하거나 또는 그의 범위를 묘사하고자 하는 것이 아니다. 그것의 유일한 목적은 뒤에 제시되는 보다 상세한 설명에 대한 서문으로서 일부 개념들을 간략화된 형태로 제시하는 것이다.

[0005] 본 아키텍처는 쿼리 문자열에 기초하여 검색 결과들로서 수신된 문서들로부터 문서 정보를 추출하고 데이터 문자열과 상기 쿼리 문자열 사이의 편집 거리를 계산하기 위한 메커니즘을 제공한다. 상기 데이터 문자열은, 예를 들면, TAUC(제목(title), 앵커 텍스트(anchor text), URL(uniform resource locator), 및 클릭(clicks))와 같은 문서 정보로부터 얻어지는 문서의 짧고 정확한 설명일 수 있다. 상기 편집 거리는 결과 랭킹의 일부로서 문서의 관련성을 결정하는 데에 채용된다. 상기 메커니즘은 전체 쿼리 또는 상기 쿼리의 일부의 거의 매칭하는 것들(near-matches)을 검출하기 위해 근접 관련된 특징들(proximity-related features)의 세트를 채용함으로써

검색 결과 랭킹의 관련성을 개선한다.

[0006] 상기 편집 거리는 상기 쿼리 문자열이 상기 문서 정보를 포함하는 주어진 데이터 스트림에 얼마나 가까운지를 평가하기 위해 처리된다. 본 아키텍처는 쿼리 용어들의 보다 효과적인 발견을 허용하기 위해 상기 URL 내의 복합 용어들(compound terms)의 인덱스-타임 분할(index-time splitting)을 포함한다. 또한, 문서 결과들 중 하나 이상 문서 결과들의 최상위 N개의 앵커들을 찾아내기 위해 앵커 텍스트의 인덱스-타임 필터링이 이용된다. 상기 TAUC 정보를 이용하는 것은 상기 검색 결과들을 랭킹하기 위한 관련성 메트릭들을 개선하기 위해 신경망(예를 들면, 2-계층)에 입력될 수 있다.

[0007] 전술한 및 관련 목적들의 달성을 위해, 다음의 설명 및 첨부 도면들과 관련하여 여기에 특정한 예시적인 양태들이 설명된다. 그러나, 이들 양태들은 여기에 개시된 원리들이 채용될 수 있는 다양한 방법들 중 소수만을 나타내고 모든 그러한 양태들 및 동등물들을 포함하도록 의도된다. 기타 이점들 및 새로운 특징들은 도면들과 함께 고려될 때 다음의 상세한 설명으로부터 명백해질 것이다.

도면의 간단한 설명

[0008] 도 1은 컴퓨터 구현된 관련성 시스템을 예시한다.

도 2는 편집 거리를 계산하기 위한 예시적인 매칭 알고리즘의 순서도를 예시한다.

도 3은 수정된 편집 거리 및 매칭 알고리증을 이용하여 쿼리 문자열 및 데이터 문자열에 기초하여 편집 거리 값들을 처리하고 생성하는 것을 예시한다.

도 4는 수정된 편집 거리 및 매칭 알고리즘을 이용하여 쿼리 문자열 및 데이터 문자열에 기초하여 편집 거리 값들을 처리하고 생성하는 다른 예를 예시한다.

도 5는 문서에 대한 관련성 스코어를 생성하는 데에 도움이 되는 신경망을 채용하는 컴퓨터 구현된 관련성 시스템을 예시한다.

도 6은 쿼리 문자열과 데이터 문자열 사이의 편집 거리를 결정하기 위해 문서 정보에서 채용될 수 있는 데이터의 유형들을 예시한다.

도 7은 인덱스-타임 처리 데이터 흐름을 예시한다.

도 8은 결과 랭킹을 위한 도 7의 인덱스 프로세스로부터 신경망으로의 입력들을 나타내는 블록도를 예시한다.

도 9는 검색 결과들을 계산하고 생성하기 위한 신경망, 편집 거리 입력들 및 원시 특징 입력들의 예시적인 시스템 구현을 예시한다.

도 10은 문서 결과 세트의 문서 관련성을 결정하는 방법을 예시한다.

도 11은 문서의 관련성을 계산하는 방법을 예시한다.

도 12는 개시된 아키텍처에 따른 TAUC 특징들을 이용하여 검색 결과 랭킹을 위한 편집 거리 처리를 실행하도록 동작 가능한 컴퓨팅 시스템의 블록도를 예시한다.

발명을 실시하기 위한 구체적인 내용

[0009] 개시된 아키텍처는 전체 쿼리의 매칭하는 것들 또는 제목, 앵커, URL, 또는 클릭과 같은, 문서에 관한 정확한 메타데이터와 매칭하는 것들을 검출하기 위해 근접 관련된 특징들의 세트를 구현함으로써 검색 결과 랭킹의 관련성을 개선한다. 예를 들면, 쿼리 "company store", 제1 문서의 문서 제목 "company store online" 및 제2 문서의 문서 제목 "new NEC LCD monitors in company store"를 생각해보자. 제1 및 제2 문서들 양쪽 모두에 대하여 다른 속성들이 동일하다고 가정할 때, 본 아키텍처는 선택된 스트림이 쿼리와 매칭하게 하기 위해 얼마나 많은 편집 노력이 바쳐지는지에 기초하여 문서에 대한 스코어를 할당한다. 이 예에서는, 평가를 위해 문서 제목이 선택된다. 제1 문서의 제목은 완전한 매칭(full match)을 만들기 위해 하나의 삭제 동작(용어 "online" 삭제)만을 필요로 하는 반면, 제2 문서의 제목은 5개의 삭제(용어 "new", "NEC", "LCD", "monitors" 및 "in" 삭제)를 필요로 한다. 따라서, 제1 문서는 더 관련이 있는 것으로 추정된다.

[0010] 제목은 복합 용어들로부터 쿼리 용어들이 발견될 수 있도록 어떤 데이터의 스트림들(예를 들면, URL)에 처리가 적용될 수 있는 TAUC(제목, 앵커, URL, 및 클릭) 문서 정보의 한 요소이다. 예를 들면, 다시, 쿼리 "company

store", 및 URL "www.companystore.com"을 생각해보자. 결과는 URL은 4개의 부분들(또는 용어들)로 분할되는 것이다: "www", "company", "store", 및 "com".

[0011] 이제, 같은 참조 번호들이 전체에 걸쳐서 같은 엘리먼트들을 지시하기 위해 이용되는, 도면들이 참조된다. 다음의 설명에서는, 설명을 위하여, 그의 철저한 이해를 제공하기 위해 다수의 특정한 상세들이 제시된다. 그러나, 새로운 실시예들은 이들 특정한 상세들 없이 실시될 수 있다는 것은 분명할 것이다. 다른 사례들에서, 잘 알려진 구조들 및 장치들은 그의 설명을 용이하게 하기 위하여 블록도 형태로 나타내어진다.

[0012] 도 1은 컴퓨터 구현된 관련성 시스템(100)을 예시한다. 이 시스템(100)은 쿼리 문자열(110)에 기초하여 검색 결과들(108)로서 수신된 문서(106)로부터 문서 정보(104)를 추출하기 위한 처리 컴포넌트(102)를 포함한다. 이 시스템(100)은 또한 문서 정보(104)로부터 얻어진 데이터 문자열(116)과 쿼리 문자열(110) 사이의 편집 거리(114)를 계산하기 위한 근접 컴포넌트(proximity component)(112)를 포함할 수 있다. 편집 거리(114)는 검색 결과들(108)의 일부로서 문서(106)의 관련성을 결정하는 데에 채용된다.

[0013] 데이터 문자열(116)을 생성하기 위해 채용되는 문서 정보(104)는, 예를 들면, 제목 정보(또는 문자들), 링크 정보(예를 들면, URL 문자들), 클릭 스트림 정보, 및/또는 앵커 텍스트(또는 문자들)를 포함할 수 있다. 처리 컴포넌트(102)는 편집 거리(114)를 계산하기 위해 인덱스 타임에 문서 정보(104)의 복합 용어들을 분할한다. 처리 컴포넌트(102)는 또한 앵커 텍스트의 최상위 랭킹된 세트를 계산하기 위해 인덱스 타임에 앵커 텍스트로서 문서 정보를 필터링한다.

[0014] 편집 거리(114)의 계산은 데이터 문자열(116)과 쿼리 문자열(110) 사이의 근접을 증가시키기 위한(더 가까워지게 하기 위한) 용어들의 삽입 및 삭제에 기초한다. 편집 거리(114)의 계산은 또한 데이터 문자열(116)과 쿼리 문자열(110) 사이의 근접을 증가시키기 위한(더 가까워지게 하기 위한) 용어들의 삽입 및 삭제와 관련된 비용들에 기초할 수 있다.

[0015] 쿼리 문자열(110)로부터 용어들의 삽입 및/또는 삭제에 기초하여 데이터 문자열(116)(예를 들면, TAUC)을 생성하는 시나리오를 생각해보자. 이 용어 처리는 4개의 동작들에 따라서 수행될 수 있다: 쿼리 문자열(110)에 쿼리가 아닌 단어를 삽입하고; 쿼리 문자열(110)에 쿼리 용어를 삽입하고; 쿼리 문자열(110)로부터 TAUC 용어를 삭제하고; 및/또는 쿼리 문자열(110)로부터 TAUC가 아닌 용어를 삭제한다.

[0016] 편집 거리(114)는 대체가 아닌, 삽입 및 삭제 동작들에 기초한다. 삽입에 대하여 정의된 2가지 유형의 비용이 있을 수 있다. 쿼리 문자열(110)로부터 데이터 문자열(116)을 생성하는 시나리오를 생각해보자. 생성 시에, 최초 쿼리 문자열(110)에 존재하는 단어가 쿼리 문자열(110)에 삽입될 수 있고, 그러면 비용은 1로서 정의되고, 그렇지 않다면, 비용은 $w1(\geq 1)$로서 정의된다. 여기서, w1은 조정되는 가중 파라미터이다. 예를 들면, 쿼리 문자열(110)이 AB이면, ABC의 데이터 문자열을 생성하는 비용은 데이터 문자열 ABA를 생성하는 비용보다 더 높다. 직관은 "관련이 없는 단어"들을 데이터 문자열(116)에 삽입함으로써 전체 데이터 문자열(116)(예를 들면, TAUC)을 더 관련이 없게 만드는 것이다.

[0017] 삭제에 대한 2가지 유형의 비용이 있을 수 있다. 다시, 쿼리 문자열(110)로부터 데이터 문자열(116)을 생성하는 시나리오를 생각해보자. 최초 데이터 문자열(116)에 존재하는 용어를 쿼리 문자열(110)에서 삭제할 때, 비용은 1로서 정의되고, 그렇지 않다면, 비용은 $w2(\geq 1)$로서 정의된다.

[0018] 다른 유형의 비용은 위치 비용이다. 데이터 문자열(116)의 제1 위치에서 삭제 또는 삽입이 일어나면, 추가적인 비용(+w3)이 있다. 직관은 2개의 문자열들(쿼리 문자열(110) 및 데이터 문자열(116))의 처음에 매칭하는 것은 문자열들에서 뒤에 매칭하는 것보다 더 큰 중요성이 주어진다. 쿼리 문자열(110)은 "cnn"이고, 데이터 문자열(116)은 title = "cnn.com - blur blur"인 다음의 예를 생각해보자. 삽입 및 삭제가 제1 위치에서 일어난다면, 그것은 해답의 유효성을 상당히 감소시킬 수 있다.

[0019] 도 2는 편집 거리를 계산하기 위한 예시적인 수정된 매칭 알고리즘의 순서도를 예시한다. 설명의 단순성을 위하여, 예를 들면, 순서도 또는 흐름도의 형태로, 여기에 제시된 하나 이상의 방법들은 일련의 행위들로서 제시되고 설명되지만, 그 방법들은 행위들의 순서에 의해 제한되지 않는다는 것을 이해하고 인식해야 하며, 이는 일부 행위들은, 그것에 따라서, 여기에 제시되고 설명된 것과 다른 순서로 및/또는 다른 행위들과 동시에 일어날 수 있기 때문이다. 예를 들면, 숙련된 당업자들은 방법은 대안적으로 상태도에서와 같은 일련의 상호 관련된 상태들 또는 이벤트들로서 표현될 수 있다는 것을 이해하고 인식할 것이다. 또한, 방법에 예시된 모든 행위들이 새로운 구현을 위해 요구되지는 않을 수도 있다.

[0020] 단계(200)에서는, 쿼리 문자열 및 데이터(또는 대상) 문자열의 요소들이 열거된다. 이것은 n을 쿼리 문자열(여

기서 쿼리 문자열 내의 각 용어는 s[i]임)의 길이인 것으로 설정하고, m을 대상(또는 데이터) 문자열(여기서 대상 문자열 내의 각 용어는 t[j]로 표시됨)의 길이인 것으로 설정하는 것에 의해 달성된다. 단계(202)에서는, 0...m 행들 및 0...n 열들을 포함하는 행렬이 구성된다(여기서 행렬 내의 각 용어는 d[j,i]로 표시됨). 단계(204)에서는, 제1 행은 삭제의 상이한 비용에 의존하는 값으로 초기화되고 제1 열은 삽입의 상이한 비용에 의존하는 값으로 초기화된다. 단계(206)에서는, 단계(208)에 나타내어진 바와 같이, n=0이면, d[m,0]를 반환하고 종료하고, m=0이면, d[0,n]을 반환하고 종료한다. 단계(210)에서는, 쿼리 문자열의 각 문자가 검사된다(1부터 n까지 i). 단계(212)에서는, 대상 데이터 문자열의 각 문자가 검사된다(1부터 n까지 j). 단계(214)에서는, 쿼리 문자열 내의 문자가 데이터 문자열 내의 문자와 같다면, 흐름은 단계(216)으로 진행하고 거기서는 비용은 제로이고 다음 행렬 셀이 처리된다. 환언하면, s[i]가 t[j]와 같다면, 비용은 0이고 d[j,i] = d[j-1,i-1]이다.

[0021] 쿼리 문자열 셀 내의 문자가 데이터 문자열 셀 내의 문자와 같지 않다면, 흐름은 단계(214)로부터 단계(218)로 진행하고 거기서는 현재의 셀은 바로 위의 셀 또는 바로 왼쪽의 셀, 플러스 삽입 또는 삭제 비용으로 설정된다. 환언하면, s[i]가 t[j]와 같지 않다면, 행렬의 셀 d[j,i]를 바로 위의 셀 플러스 대응하는 삽입 비용(d[j-1,i] + cost_insertion으로 표현됨) 또는 바로 왼쪽의 셀 플러스 대응하는 삭제 비용(d[j,i-1] + cost_deletion으로 표현됨)의 최소치와 같게 설정한다. 단계(220)에서는, 완료까지 단계들(210, 212, 214, 216 및 218)이 반복된다. 단계(222)에서는, 셀 d[m,n]에서 발견된 최종 비용이 출력된다. 이 예에서 cost_insertion 및 cost_deletion 양쪽 모두는 2가지 종류의 값들을 갖는다는 것에 유의한다; 예를 들면 삽입 비용의 경우 w1=1, w3=4이고, 삭제 비용의 경우 w2=1, w4=26이다.

[0022] 환언하면, d[j,i]는 문자열들 s[0..i]와 t[0...j] 사이의 편집 거리를 포함한다. 정의에 의해 d[0,0] = 0이다(빈 문자열을 빈 문자열과 같게 만들기 위해 어떤 편집도 필요하지 않다). d[0,y] = d[0,y-1] + (w2 또는 w4)이다. 문자열 d[0,y-1]을 만들기 위해 얼마나 많은 편집이 이용되는지가 알려져 있다면, d[0,y]는 d[0,y-1] + 대상 문자열로부터 현재의 문자를 삭제하는 비용(그 비용은 w2 또는 w4일 수 있음)으로서 계산될 수 있다. 현재의 문자가 s[0...n], t[0...m] 양쪽 모두에 존재한다면 비용 w2가 이용되고, 그렇지 않다면 w4가 이용된다. d[x,0]=d[x-1,0]+(w1 또는 w3). 문자열 d[x-1,0]를 만들기 위해 얼마나 많은 편집이 이용되는지가 알려져 있다면, d[x,0]는 d[x-1,0] + s로부터 t로 현재의 문자의 삽입의 비용(그 비용은 w1 또는 w3일 수 있음)으로서 계산될 수 있다. 현재의 문자가 s[0...n], t[0...m] 양쪽 모두에 존재한다면 비용 w1이 이용되고, 그렇지 않다면 w3가 이용된다.

[0023] 각 (j,i)에 대하여, d[j,i]는 s[i]=t[j]이면 d[j-1,i-1]과 같을 수 있다. 편집 거리는 문자열들 t[j-1], s[i-1] 사이에 계산되고, s[i]=t[j]이면, 편집을 일으키지 않고, 그 문자열들을 같게 만들기 위해 양쪽 문자열들에 공통의 문자가 추가될 수 있다. 따라서, 채용되는 3개의 움직임들이 있고, 여기서 현재의 d[j,i]에 대한 최소의 편집 거리를 제공하는 움직임이 선택된다. 바꾸어 말하면,

```
d[j,i] = min(
      d[j-1,i-1] if s[i]=t[j];
      d[j-1,i] + (w1, if s[j] is present in both strings; else, w3);
      d[j,i-1] + (w2, if t[i] is present in both strings; else, w4)
```

[0024])

[0025] 도 3은 수정된 편집 거리 및 매칭 알고리즘을 이용하여 쿼리 문자열 및 데이터 문자열에 기초하여 편집 거리 값들을 처리하고 생성하는 것을 예시한다. 이 프로세스는 왼쪽-오른쪽, 위-아래, 및 대각선 계산들 중 하나 이상을 수반한다. 용어들 "A B C"의 쿼리 문자열이 용어들 "C B A X"(여기서 X는 쿼리 문자열에 존재하지 않는 용어를 나타냄)의 대상 데이터 문자열과 대조하여 처리된다. 편집 거리를 계산하기 위한 프로세스는 여러 가지 방법들로 수행될 수 있지만; 편집 거리의 수정된 버전을 수행하기 위한 특정한 상세들은 개시된 아키텍처에 따라 계산될 때 상이하다. 쿼리 문자열에 대하여 n = 3이고 데이터 문자열에 대하여 m = 4인 n×m에 기초하여 4×5 행렬(300)이 구성된다. 쿼리 문자열(302)은 행렬(300)의 수평축을 따라서 배치되고 대상 데이터 문자열(304)은 수직축을 따라서 배치된다.

[0026] 이 설명은 4개의 열들(0-3) 및 5개의 행들(0-4)로 나타내어진 행렬(300)을 이용할 것이다. 행 0, 열 0에서 시작하여 왼쪽으로부터 오른쪽으로 도 2에 설명된 편집 거리 매칭 알고리즘을 적용하여, 교차하는 셀 d[0,0]은 "0"을 수신하는데, 이는 쿼리 문자열 ABC의 빈 셀과 대상 데이터 문자열 CBAX의 빈 셀의 비교는 쿼리 문자열을 대상 데이터 문자열과 동일하게 만들기 위한 용어의 삽입 또는 삭제를 일으키지 않기 때문이다. 그 "용어들"은 동일하며 따라서 편집 거리는 제로이다.

[0027] 오른쪽으로 이동하여 쿼리 문자열(302)의 A 용어와 행 0의 빈 셀을 비교하는 것은 문자열들을 동일하게 만들기 위해 하나의 삭제를 이용하고; 따라서, 셀 d[0,1]은 "1"의 값을 수신한다. 다시 오른쪽으로 열 2로 이동하여, 비교는 이제 쿼리 문자열(302)의 용어들 AB와 대상 데이터 문자열 열의 빈 셀 사이에 이루어진다. 따라서, 문자열들을 동일하게 만들기 위해 쿼리 문자열(302)에서 2개의 삭제들이 이용되고 이는 "2"의 편집 거리가 셀 d[0,2] 안에 놓이는 것으로 귀결된다. 동일한 프로세스가 열 3에 적용되고 거기서는 쿼리 문자열(302)의 용어들 ABC가 대상 문자열 열 내의 빈 셀과 비교되어, 그 문자열들을 같게 만들기 위해 3개의 삭제들을 이용하고, 이는 셀 d[0,3]에서 "3"의 편집 거리로 귀결된다.

[0028] 아래로 행 1로 내려가서 왼쪽에서 오른쪽으로 계속하여, 쿼리 문자열의 빈 셀이 대상 데이터 문자열(304)의 제1 용어 C와 비교된다. 그 문자열들을 동일하게 만들기 위해 하나의 삭제가 이용되고, d[1,0]에 "1"의 편집 거리가 삽입된다. 오른쪽으로 열 1로 이동하여, 비교는 쿼리 문자열(302)의 A 용어와 대상 데이터 문자열(304)의 C 용어 사이에 이루어진다. 그 문자열들을 같게 만들기 위해 삭제 및 삽입이 이용되고, 따라서, 셀 d[1,1]에 "2"의 값이 삽입된다. 마지막 셀 d[1,3]로 스킵하여, ABC를 C에 매칭시키기 위한 매칭 프로세스는 2개의 삭제들을 이용하여 셀 d[1,3]에서 "2"의 편집 거리로 귀결된다. 간결을 위해 및 전체적인 편집 거리를 구하기 위해 행 4 및 열 3으로 이동하여, 용어들 ABC를 용어들 CBAX에 매칭시키는 것은 대상 문자열의 제1 용어 C에서 삽입/삭제를 이용하여 "2"의 값, B 용어들 사이의 매칭에 대하여 "0"의 값, 제3 용어들 C와 A의 매칭을 위하여 삽입/삭제를 이용하여 "2"의 값, 용어 X의 삽입을 이용하여 "1"의 값 및 위치 비용에 대한 "3"의 값으로 셀 d[4,3]에서 "8"의 편집 거리로 귀결되고, 이는 셀 d[4,3]에서의 "8"의 최종 편집 거리 값으로 귀결된다.

[0029] 도 4는 수정된 편집 거리 및 매칭 알고리즘을 이용하여 쿼리 문자열 및 대상 데이터 문자열에 기초하여 편집 거리 값들을 처리하고 생성하는 다른 예를 예시한다. 여기서, w1=1의 cost_insertion, 삽입 비용을 위한 w3=4, 및 삭제 비용을 위한 w2=1 및 w4=26에 대한 가중치들에 기초하여 ABC의 쿼리 문자열(402)와 AB의 대상 데이터 문자열(404)을 비교하기 위한 행렬(400)이 생성된다. 환언하면, 왼쪽으로부터 오른쪽으로 행 0에서 작업하여, 쿼리 문자열(402)의 용어 A를 대상 문자열(404) 앞의 빈 셀에 매칭시키는 것은 용어 A의 대상 문자열(404)에서의 하나의 삽입을 이용하여 셀 d[0,1]에서 "1"의 값으로 귀결된다. 쿼리 문자열(402)의 용어들 AB를 대상 문자열(404) 앞의 빈 셀에 매칭시키는 것은 용어들 AB의 대상 문자열(404)에서의 2개의 삽입을 이용하여 셀 d[0,2]에서 "2"의 값으로 귀결되고, 쿼리 문자열(402)의 용어들 ABC를 대상 문자열(404) 앞의 빈 셀에 매칭시키는 것은 용어 AB의 대상 문자열(404)에서의 2개의 삽입에 대한 값 플러스 용어 C에 대한 값 w4=26으로 셀 d[0,3]에서 "28"의 값으로 귀결되는데, 이는 용어 C가 양쪽 문자열들에 존재하지 않기 때문이다.

[0030] 왼쪽으로부터 오른쪽으로 행 1에서 작업하여(d[1,0] = 1인 것을 이해하여), 쿼리 문자열(402)의 용어 A를 대상 문자열(404)의 용어 A에 매칭시키는 것은 대상 문자열(404) 및 쿼리 문자열(402)이 같아서, d[j-1,i-1] = d[0,0] = "0"으로부터 그 값을 취하여, 셀 d[1,1]에서 "0"의 값으로 귀결된다. 쿼리 문자열(402)의 용어들 AB를 대상 문자열(404)의 용어 A에 매칭시키는 것은 용어 B에 대한 대상 문자열(404)에서의 하나의 삽입으로 셀 d[1,2]에서 "1"의 최소 값으로 귀결된다. 셀 d[1,3]에 대하여 쿼리 문자열(402)의 용어들 ABC를 대상 문자열(404)의 용어 A에 매칭시키는 것은 d[j-1,i] = d[0,3]의 값 플러스 w3으로 셀 d[1,3]에서 "28"의 값이 되는 것과 d[j,i-1] = d[1,2]의 값 1 플러스 26(용어 C가 양쪽 문자열들에 존재하지 않기 때문에)으로 27이 되는 것을 비교하여 그와 관련된 최소 값으로 귀결되고, 이는 d[1,3]에서 "27"의 최소 값으로 귀결된다.

[0031] 왼쪽으로부터 오른쪽으로 행 2에서 작업하여, 쿼리 문자열(402)의 용어 A를 대상 문자열(404)의 용어들 AB에 매칭시키는 것은 대상 문자열(404)에서의 삭제로 셀 d[2,1]에서 "1"의 값으로 귀결된다. 셀 d[2,2]에서의 거리를 위하여 쿼리 문자열(402)의 용어들 AB를 대상 문자열(404)의 용어들 AB에 매칭시키는 것은 같음으로 귀결되고, 그에 의해 셀 d[2,2]에 대한 값 "0"으로서 d[j-1,i-1] = d[1,1]로부터의 값을 끌어당긴다. 셀 d[2,3]에 대하여 쿼리 문자열(402)의 용어들 ABC를 대상 문자열(404)의 용어들 AB에 매칭시키는 것은 d[j-1,i] = d[1,3] = 27 플러스 w3=1로 "28"의 값이 되는 것(대상 문자열에 C가 존재하지 않기 때문에)과 d[i,j-1] = d[2,2] = 0 플러스 26으로 26이 되는 것(양쪽 문자열들에 용어 C가 존재하지 않기 때문에)을 비교하여 그와 관련된 최소 값으로 귀결되고, 따라서 d[2,3]에서 "26"의 최소 값으로 된다.

[0032] 도 5는 문서(106)에 대한 관련성 스코어(504)를 생성하는 데에 도움이 되는 신경망(502)을 채용하는 컴퓨터 구현된 관련성 시스템(500)을 예시한다. 이 시스템(500)은 쿼리 문자열(110)에 기초하여 검색 결과들(108)로서 수신된 문서(106)로부터 문서 정보(104)를 추출하기 위한 처리 컴포넌트(102), 및 문서 정보(104)로부터 얻어진 데이터 문자열(116)과 쿼리 문자열(110) 사이의 편집 거리(114)를 계산하기 위한 근접 컴포넌트(112)를 포함한다. 편집 거리(114)는 검색 결과들(108)의 일부로서 문서(106)의 관련성을 결정하는 데에 채용된다.

[0033] 신경망(502)은 문서(106)에 대한 관련성 스코어를 계산하기 위해 입력으로서 문서 정보(104)를 수신하기 위해 채용될 수 있다. 단지 또는 부분적으로 검색 결과들(108)의 일부 또는 전부에 대한 관련성 스코어들에 기초하여, 검색 결과들(108) 내의 문서들이 랭킹될 수 있다. 시스템(500)은 검색 결과들(108) 내의 관련된 문서의 랭킹을 위한 관련성 스코어를 생성하기 위해 신경망(502) 및 코드베이스(codebase)를 채용한다.

[0034] 다음은 쿼리 문자열과 데이터 문자열들의 각 데이터 문자열 사이의 편집 거리를 계산하여 각 쌍에 대한 TAUC 스코어를 획득하기 위한 편집 거리 알고리즘에 대한 설명이다.

[0035] 문서에는 하나의 제목만이 존재하기 때문에, TAUC 스코어는 제목에 관하여 다음과 같이 계산될 수 있다:

[0036] TAUC(Title) = ED(Title)

[0037] 여기서 TAUC(Title)는 변환 함수의 적용 후에 나중에 신경망에의 입력으로서 이용되고, ED(Title)는 제목의 편집 거리이다.

[0038] URL들 및 클릭들뿐만 아니라, 문서에 대한 앵커 텍스트의 다수의 인스턴스들이 존재할 수 있다(여기서 클릭은 그에 대하여 이 문서가 클릭된 이전에 실행된 쿼리이다). 아이디어는 이 문서가 유사한 쿼리들에 대하여 더 관련이 있다는 것이다. 인덱스 타임에, 가장 높은 빈도들을 갖는 N개의 앵커 텍스트들이 선택된다. 그 후 각 선택된 앵커에 대하여 ED 스코어가 계산된다. 마지막으로, 앵커에 대하여 다음과 같이 TAUC 스코어가 결정된다:

[0039] $TAUC(Anchor) = Min\{ED(Anchor_i)\}$ i: 최상위 N개의 앵커들;

[0040] 직관은 앵커들 중 하나와 양호한 매칭이 존재하다면, 그것으로 충분하다는 것이다. TAUC(Anchor)는 변환 함수를 적용한 후에 신경망 입력으로서 이용된다.

[0041] URL 스트링들에 대하여 ED를 계산하기 전에 특별한 처리가 이용된다. 인덱스 타임에 URL 스트링들은 문자들의 세트를 구분 기호들(separators)로서 이용하여 부분들로 분할된다. 그 후 제목 및 앵커 용어들의 딕셔너리(dictionary)로부터 각 부분에서 용어들이 선택된다. 딕셔너리로부터의 용어의 각 발생은 URL 문자열의 처음으로부터 문자들로 측정된 위치와 함께 인덱스에 저장된다.

[0042] 쿼리 타임에 쿼리 용어들의 모든 발생들은 인덱스 타임에 저장된 인덱스로부터 판독되고 중단들(breaks)은 "쿼리가 아닌(non-query)" 용어들로 채워진다. 이 처리 후에 ED가 계산된다. ED 처리의 결과는, 변환 함수의 적용 후의, 신경망 입력이다.

[0043] 처리될 수 있는 또 다른 속성은 주어진 문서 콘텐트에 대하여 사용자가 입력하는 "클릭들"의 수이다. 사용자가 문서에서 클릭할 때마다, 스트림이 데이터베이스에 입력되어 문서와 관련된다. 이 프로세스는 또한 짧은 데이터의 스트림들과 같은 문서 정보 텍스트 내의 스트림 데이터에 적용될 수 있다.

[0044] 인덱스-타임 URL 처리 알고리즘은 문자들의 세트를 구분 기호들로서 이용하여 전체 URL을 부분들로 분할한다. 분할 함수는 또한 urlpart.startpos를 원본 문자열에서의 부분의 위치로 설정한다. 분할 함수는 URL의 무의미한(insignificant) 부분들의 필터링을 수행한다.

[0045] 예를 들면, "http://www.companymeeting.com/index.html"은 "companymeeting/index"로 필터링되고 "companymeeting" 및 "index"로 분할된다.

```
Startpos: 0


Urlparts = split(url, dictionary)

// find terms in different url parts.
For each (term in dictionary)
{
    Int pos = 0;
```

[0046]

```
    For each(urlpart in urlparts)
```

```
    {
        pos = urlpart.Find(term, pos);
        while (pos >= 0)
        {
            // parts_separator is used to distinguish different parts at
query time
            storeOccurrence(term, pos +
urlpart.startpos*parts_separator);
            pos = url.Find(term, pos + term.length);
        }
    }
    setIndexStreamLength(parts_separator * urlparts.Count);
}
```

[0047]

[0048] 딕셔너리가 "company meeting comp"를 포함한다고 가정할 때, 다음의 키들이 생성될 수 있다: Company: 0; Meeting: 7; 및 Comp: 0. 문자열의 총 길이는 parts_separator*2이다.

[0049] ED 전에 쿼리-타임 처리에 관련하여, 쿼리 타임에 쿼리 용어들의 발생들이 판독되고, 쿼리 용어들의 문자열은 원본 URL 문자열에서 나타나는 순서로 구성되고, 용어들 사이의 공간은 "쿼리가 아닌" 단어 마크들로 채워진다. 예를 들면, "company policy"의 쿼리 문자열 및 "company" "쿼리가 아닌 용어" "쿼리가 아닌 용어"의 결과의 문자열을 생각해보자.

[0050] parts_separator, 쿼리 용어 위치들, 및 스트림 길이는 원본 URL 문자열에 얼마나 많은 부분들이 있는지 및 어느 부분이 주어진 쿼리를 포함하는지를 알기 위해 결정된다. 용어들이 없는 각 부분은 "쿼리가 아닌 용어"를 포함하는 것으로 간주된다. 만일 한 부분이 쿼리 용어로 시작하지 않는다면, 그 용어 앞에 "쿼리가 아닌 용어"가 삽입된다. 쿼리 용어들 사이의 모든 공간들은 "쿼리가 아닌 용어들"로 채워진다.

[0051] 도 6은 쿼리 문자열과 데이터 문자열 사이의 편집 거리를 결정하기 위해 문서 정보(104)에서 채용될 수 있는 데이터의 유형들을 예시한다. 문서 정보(104)는 처리 컴포넌트(102)에 의한 처리 및 데이터(또는 대상) 문자열(116)의 생성을 위해, 예를 들면, 제목 텍스트(604), 앵커 텍스트(606), URL(608) 텍스트 또는 문자들, 및 클릭 정보(610)와 같은, TAUC 데이터(602)를 포함할 수 있다. 문서 정보(104)는 또한 사용자가 문서 콘텐트에서 클릭하는 횟수, 사용자가 (클릭을 통해) 선택하는 콘텐트의 유형, 콘텐트에 대한 클릭의 수, 일반 문서 등과 관련이 있는 클릭 정보(610)를 포함할 수 있다.

[0052] 도 7은 인덱스-타임 처리 데이터 흐름(700)을 예시한다. 상부에서, 문서 분석 및 추출에 기초하여 제목(604), 문서 앵커들(606), 클릭 정보(610) 등의 형태의 문서 정보가 수신된다. 제목(604)은 용어 분할 알고리즘(704)을 통하여 처리되고 그 후 딕셔너리(706)로 전달된다. 딕셔너리(706)는 제목(604), 앵커들(606), 클릭 정보(610) 등에서 발견된 여러 가지 용어들의 임시 저장소이다. 딕셔너리(706)는 URL 분할 알고리즘(708)을 통해 URL(608)을 분할하기 위해 이용된다. URL 분할 알고리즘(708)의 출력은 관련성 및 랭킹 처리를 위해 인덱싱 프로세스(710)에 보내진다. 문서 앵커들(606)은 또한 최상위 N개의 앵커들에 대한 필터(712)를 통하여 처리될 수 있다. 클릭 정보(610)는 인덱싱 프로세스(710)를 통해 직접 처리될 수 있다. 다른 문서 정보가 그에 따라서 처리될 수 있다(예를 들면, 용어 분할, 필터링 등).

[0053] 도 8은 결과 랭킹을 위한 도 7의 인덱스 프로세스(710)로부터 신경망으로의 입력들을 나타내는 블록도(800)를 예시한다. 인덱싱 프로세스(710)는 쿼리 문자열(110)에 관하여 URL 편집 거리(ED)(802), 쿼리 문자열(110)에 관하여 최상위 N개의 앵커 ED(804), 쿼리 문자열(110)에 관하여 제목 ED(806), 쿼리 문자열(110)에 관하여 클릭 ED(808)뿐만 아니라, 편집 거리와 관련이 없는 기타 특징들(810)을 계산하기 위해 이용될 수 있고, 그의 일부 또는 전부(URL ED(802), 최상위 N개의 앵커 ED(804), 제목 ED(806), 클릭 ED(808), 및 기타 특징들(810))는, 결국 관련된 문서에 대한 관련성 스코어, 및 그 후 다른 문서 검색 결과들 중에서 그 문서의 랭킹을 구하기 위해, 신경망(502)에의 입력들로서 채용될 수 있다. 신경망(502)은 문서의 관련성을 식별하는 데 기여하는 원시 입력 특징들로서 적어도 TAUC 특징들을 수신하는 2-계층 모델일 수 있다. 신경망은 이들 특징들이 검색 엔진에 의한 정렬을 위해 이용될 수 있는 단일 수로 어떻게 조합될 수 있는지를 결정한다.

[0054] 신경망(502)은 관련성 및 랭킹 처리를 위해 채용될 수 있는 수학 또는 계산 모델의 일례에 불과하다는 것을 인식해야 한다. 나이브 베이즈(naive Bayes), 베이지안 네트워크, 판정 트리, 퍼지 논리 모델, 및 독립성의 상이한 패턴들을 나타내는 기타 통계 분류 모델과 같은 통계적 회귀의 다른 형태들이 채용될 수 있고, 여기서 분류는 랭크 및 우선 순위를 할당하기 위해 이용되는 방법들을 포함한다.

[0055] 도 9는 검색 결과들을 계산하고 생성하기 위한 신경망(502), 편집 거리 입력들 및 원시 특징 입력들의 예시적인 시스템(900) 구현을 예시한다. 신경망(502)의 입력(들) 상의 원시 랭킹 특징들(810)의 세트는 BM25 함수(902)(예를 들면, BM25F), 클릭 거리(904), URL 깊이(906), 파일 유형들(908), 및 언어 매치(910)를 포함할 수 있다. BM25 컴포넌트들은, 예를 들면, 본문(body), 제목, 저자(author), 앵커 텍스트, URL 디스플레이 이름, 및 추출된 제목을 포함할 수 있다.

[0056] 도 10은 관련성을 결정하는 방법을 예시한다. 단계(1000)에서는, 검색 프로세스의 일부로서 쿼리 문자열이 수신된다. 단계(1002)에서는, 검색 프로세스 동안에 반환된 문서로부터 문서 정보가 추출된다. 단계(1004)에서는, 문서 정보로부터 데이터 문자열이 생성된다. 단계(1006)에서는, 데이터 문자열과 쿼리 문자열 사이의 편집 거리가 계산된다. 단계(1008)에서는, 편집 거리에 기초하여 관련성 스코어가 계산된다.

[0057] 이 방법의 다른 양태들은 편집 거리를 계산하는 것의 일부로서 용어 삽입을 채용하고 데이터 문자열을 생성하기 위한 쿼리 문자열 내의 용어의 삽입을 위한 삽입 비용을 평가하는 것을 포함할 수 있고, 그 비용은 가중 파라미터로서 표현된다. 이 방법은 편집 거리를 계산하는 것의 일부로서 용어 삭제를 채용하고 데이터 문자열을 생성하기 위한 쿼리 문자열 내의 용어의 삭제를 위한 삭제 비용을 평가하는 것을 더 포함할 수 있고, 그 비용은 가중 파라미터로서 표현된다. 편집 거리를 계산하는 것의 일부로서 위치 비용이 계산될 수 있고, 그 위치 비용은 데이터 문자열에서의 용어 위치의 용어 삽입 및/또는 용어 삭제와 관련된다. 또한, 편집 거리를 계산하는 것의 전체적인 비용을 계산하기 위해 데이터 문자열의 문자들과 쿼리 문자열의 문자들 사이에 매칭 프로세스가 수행된다.

[0058] 데이터 문자열의 URL의 복합 용어들을 분할하는 것은 인덱스 타임에 일어날 수 있다. 이 방법은 문서에서의 발생의 빈도에 기초하여 앵커 텍스트의 최상위 랭킹된 세트를 찾아내기 위해 데이터 문자열의 앵커 텍스트를 필터링하는 것과 그 세트 내의 앵커에 대한 편집 거리 스코어를 계산하는 것을 더 포함할 수 있다. 편집 거리를 계산하는 것으로부터 얻어진 편집 거리 스코어는 변환 함수의 적용 후에 2-계층 신경망에 입력될 수 있고, 그 스코어는 제목 정보, 앵커 정보, 클릭 정보, 또는 URL 정보 중 적어도 하나와 관련된 편집 거리를 계산하는 것에 기초하여 생성된다.

[0059] 도 11은 문서의 관련성을 계산하는 방법을 예시한다. 단계(1100)에서는, 문서들의 결과 세트를 반환하기 위해 검색 프로세스의 일부로서 쿼리 문자열이 처리된다. 단계(1102)에서는, 결과 세트의 문서로부터 추출된 문서 정보에 기초하여 데이터 문자열이 생성되고, 문서 정보는 문서로부터의 제목 정보, 앵커 텍스트 정보, 클릭 정보, 및 URL 정보 중 하나 이상을 포함한다. 단계(1104)에서는, 용어 삽입, 용어 삭제, 및 용어 위치에 기초하여 데이터 문자열과 쿼리 문자열 사이의 편집 거리가 계산된다. 단계(1106)에서는, 편집 거리에 기초하여 관련성 스코어가 계산되고, 관련성 스코어는 결과 세트에서 문서를 랭킹하기 위해 이용된다.

[0060] 이 방법은 용어 삽입, 용어 삭제 및 용어 위치의 각각과 관련된 비용을 계산하는 것과, 그 비용을 관련성 스코어의 계산에 넣는 것과, 인덱스 타임에 URL 정보의 복합 용어들을 분할하는 것과, 문서에서 앵커 텍스트의 발생의 빈도에 기초하여 앵커 텍스트의 최상위 랭킹된 세트를 찾아내기 위해 인덱스 타임에 앵커 텍스트 정보를 필터링하는 것을 더 포함할 수 있다. 쿼리 문자열의 용어들의 발생들의 판독은 원본 URL 문자열에서 나타나는 순서로 쿼리 용어들의 문자열을 구성하고 그 용어들 사이의 공간을 단어 마크들로 채우기 위해 수행될 수 있다.

[0061] 이 출원에서 사용될 때, 용어들 "컴포넌트" 및 "시스템"은 컴퓨터 관련 엔티티를 지시하기 위해 의도된 것으로, 하드웨어이거나, 하드웨어와 소프트웨어의 조합이거나, 소프트웨어이거나, 또는 실행중 소프트웨어일 수 있다. 예를 들면, 컴포넌트는 프로세서에서 실행하는 프로세스, 프로세서, 하드 디스크 드라이브, (광 및/또는 자기 저장 매체의) 다수의 저장 드라이브들, 개체, 실행 파일, 실행의 스레드, 프로그램, 및/또는 컴퓨터일 수 있지만, 이에 제한되는 것은 아니다. 예시로서, 서버에서 실행하는 애플리케이션 및 서버 양쪽 모두가 컴포넌트일 수 있다. 하나 이상의 컴포넌트들이 프로세스 및/또는 실행의 스레드 내에 존재할 수 있고, 컴포넌트는 하나의 컴퓨터에 국한되거나 및/또는 2개 이상의 컴퓨터들 사이에 분산될 수 있다.

[0062] 이제 도 12를 참조하면, 개시된 아키텍처에 따른 TAUC 특징들을 이용하여 검색 결과 랭킹을 위한 편집 거리 처리를 실행하도록 동작 가능한 컴퓨팅 시스템(1200)의 블록도가 예시되어 있다. 그것의 다양한 양태들에 대한 추가적인 컨텍스트를 제공하기 위하여, 도 12 및 다음의 논의는 다양한 양태들이 구현될 수 있는 적합한 컴퓨팅 시스템(1200)에 대한 간략하고 일반적인 설명을 제공하고자 하는 것이다. 상기 설명은 하나 이상의 컴퓨터들에서 실행할 수 있는 컴퓨터 실행가능 명령어들의 일반적인 컨텍스트에 있지만, 숙련된 당업자들은 새로운 실시예는 또한 다른 프로그램 모듈들과 함께 및/또는 하드웨어와 소프트웨어의 조합으로서 구현될 수 있다는 것을 인

지할 것이다.

[0063] 일반적으로, 프로그램 모듈은 특정 작업을 수행하거나 특정 추상 데이터 유형을 구현하는 루틴, 프로그램, 컴포넌트, 데이터 구조 등을 포함한다. 또한, 숙련된 당업자들은 본 발명의 방법들은 싱글-프로세서 또는 멀티프로세서 컴퓨터 시스템, 미니컴퓨터, 메인프레임 컴퓨터뿐만 아니라, 퍼스널 컴퓨터, 핸드헬드 컴퓨팅 장치, 마이크로프로세서 기반 또는 프로그램 가능한 소비자 전자 장치 등(이들 각각은 하나 이상의 관련된 장치들에 동작 가능하게 연결될 수 있음)을 포함하는, 다른 컴퓨터 시스템 구성들과 함께 실시될 수 있다는 것을 인식할 것이다.

[0064] 예시된 양태들은 또한 통신 네트워크를 통해 연결되어 있는 원격 처리 장치들에 의해 특정한 작업들이 수행되는 분산 컴퓨팅 환경에서 실시될 수도 있다. 분산 컴퓨팅 환경에서, 프로그램 모듈들은 로컬 및 원격 메모리 저장 장치들 양쪽 모두에 위치할 수 있다.

[0065] 컴퓨터는 통상적으로 각종 컴퓨터 판독가능 매체를 포함한다. 컴퓨터 판독가능 매체는 컴퓨터에 의해 액세스될 수 있는 임의의 이용 가능한 매체일 수 있고 휘발성 및 비휘발성 매체, 이동식 및 비이동식 매체를 포함한다. 예로서, 컴퓨터 판독가능 매체는 컴퓨터 저장 매체 및 통신 매체를 포함하지만 이에 제한되는 것은 아니다. 컴퓨터 저장 매체는 컴퓨터 판독가능 명령어, 데이터 구조, 프로그램 모듈 또는 기타 데이터와 같은 정보를 저장하는 임의의 방법 또는 기술로 구현되는 휘발성 및 미휘발성, 이동식 및 비이동식 매체를 포함한다. 컴퓨터 저장 매체는 RAM, ROM, EEPROM, 플래시 메모리 또는 기타 메모리 기술, CD-ROM, DVD(digital video disk) 또는 기타 광 디스크 저장 장치, 자기 카세트, 자기 테이프, 자기 디스크 저장 장치 또는 기타 자기 저장 장치, 또는 컴퓨터에 의해 액세스될 수 있고 원하는 정보를 저장하는 데 이용될 수 있는 임의의 기타 매체를 포함하지만 이에 제한되는 것은 아니다.

[0066] 다시 도 12를 참조하여, 다양한 양태들을 구현하기 위한 예시적인 컴퓨팅 시스템(1200)은 처리 장치(1204), 시스템 메모리(1206) 및 시스템 버스(1208)를 갖는 컴퓨터(1202)를 포함한다. 시스템 버스(1208)는 시스템 메모리(1206)를 포함하지만 이에 제한되지 않는 시스템 컴포넌트들을 처리 장치(1204)에 연결하기 위한 인터페이스를 제공한다. 처리 장치(1204)는 다양한 상업적으로 입수 가능한 프로세스들 중 어느 하나일 수 있다. 듀얼 마이크로프로세서 및 기타 멀티프로세서 아키텍처들도 처리 장치(1204)로서 채용될 수 있다.

[0067] 시스템 버스(1208)는 메모리 버스(메모리 컨트롤러와 함께 또는 메모리 컨트롤러 없이), 주변 버스, 및 각종의 상업적으로 입수 가능한 버스 아키텍처 중 임의의 것을 이용하는 로컬 버스와 더 상호 접속할 수 있는 몇몇 유형의 버스 구조 중 어느 것이라도 될 수 있다. 시스템 메모리(1206)는 비휘발성 메모리(NON-VOL)(1210) 및/또는 휘발성 메모리(1212)(예를 들면, RAM(random access memory))를 포함할 수 있다. 비휘발성 메모리(1210)(예를 들면, ROM, EPROM, EEPROM 등)에는 기본 입력/출력 시스템(BIOS)이 저장될 수 있고, 그 BIOS는 시동 중과 같은 때에, 컴퓨터(1202) 내의 구성요소들 사이의 정보 전송을 돕는 기본 루틴들이다. 휘발성 메모리(1212)는 또한 데이터를 캐싱하기 위한 스태틱 RAM과 같은 고속 RAM을 포함할 수 있다.

[0068] 컴퓨터(1202)는 내부 하드 디스크(HDD)(1214)(예를 들면, EIDE, SATA)(이 내부 HDD(1214)는 적합한 새시에서 외부 사용을 위해 구성될 수도 있음), 자기 플로피 디스크 드라이브(FDD)(1216)(예를 들면, 이동식 디스켓(1218)에 기록을 하거나 그로부터 판독을 함) 및 광 디스크 드라이브(1220)(예를 들면, CD-ROM 디스크(1222)를 판독하거나, DVD와 같은 고용량 광 매체에 기록을 하거나 그로부터 판독을 함)를 더 포함한다. HDD(1214), FDD(1216) 및 광 디스크 드라이브(1220)는 각각 HDD 인터페이스(1224), FDD 인터페이스(1226) 및 광 드라이브 인터페이스(1228)에 의해 시스템 버스(1208)에 연결될 수 있다. 외부 드라이브 구현을 위한 HDD 인터페이스(1224)는 USB(Universal Serial Bus) 및 IEEE 1394 인터페이스 기술들 중 적어도 하나 또는 양쪽 모두를 포함할 수 있다.

[0069] 상기 드라이브들 및 관련된 컴퓨터 저장 매체는 데이터, 데이터 구조, 컴퓨터 실행가능 명령어 등의 비휘발성 저장을 제공한다. 컴퓨터(1202)에 대하여, 드라이브들 및 매체는 적합한 디지털 포맷의 임의의 데이터의 저장의 편의를 도모한다. 비록 상기 컴퓨터 판독가능 매체의 설명은 HDD, 이동식 자기 디스켓(예를 들면, FDD), 및 CD 또는 DVD와 같은 이동식 광 매체를 참조하지만, 숙련된 당업자들은 집 드라이브, 자기 카세트, 플래시 메모리 카드, 카트리지 등과 같은, 컴퓨터에 의해 판독가능한 다른 유형의 매체가 예시적인 운영 환경에서 이용될 수도 있고, 또한 임의의 그러한 매체는 개시된 아키텍처의 새로운 방법들을 수행하기 위한 컴퓨터 실행가능 명령어들을 포함할 수 있다는 것을 인식할 것이다.

[0070] 드라이브들 및 휘발성 메모리(1212)에는, 운영 체제(1230), 하나 이상의 애플리케이션 프로그램(1232), 기타 프

로그램 모듈(1234), 및 프로그램 데이터(1236)를 포함하는 다수의 프로그램 모듈들이 저장될 수 있다. 하나 이상의 애플리케이션 프로그램(1232), 기타 프로그램 모듈(1234), 및 프로그램 데이터(1236)는 시스템(100) 및 관련된 블록들, 시스템(500) 및 관련된 블록들, 문서 정보(104), TAUC 데이터(602), 클릭 정보(610), 데이터 흐름(700)(및 알고리즘들), 및 블록도(800)(및 관련된 블록들)를 포함할 수 있다.

[0071] 운영 체제, 애플리케이션, 모듈, 및/또는 데이터의 전부 또는 일부는 또한 휘발성 메모리(1212)에 캐싱될 수 있다. 개시된 아키텍처는 다양한 상업적으로 입수 가능한 운영 체제들 또는 운영 체제들의 조합들과 함께 구현될 수 있다는 것을 인식해야 한다.

[0072] 사용자는 하나 이상의 유선/무선 입력 장치들, 예를 들면, 키보드(1238) 및 마우스(1240)와 같은 포인팅 장치를 통하여 명령 및 정보를 컴퓨터(1202)에 입력할 수 있다. 다른 입력 장치들(도시되지 않음)은 마이크, IR 리모트 컨트롤, 조이스틱, 게임 패드, 스타일러스 펜, 터치 스크린 등을 포함할 수 있다. 이들 및 기타 입력 장치는 종종 시스템 버스(1208)에 연결되는 입력 장치 인터페이스(1242)를 통하여 처리 장치(1204)에 접속되지만, 병렬 포트, IEEE 1394 직렬 포트, 게임 포트, USB 포트, IR 인터페이스 등과 같은 다른 인터페이스에 의해 접속될 수 있다.

[0073] 모니터(1244) 또는 다른 유형의 디스플레이 장치도 비디오 어댑터(1246) 등의 인터페이스를 통해 시스템 버스(1208)에 접속된다. 모니터(1244) 외에도, 컴퓨터는 전형적으로 스피커, 프린터 등과 같은 다른 주변 출력 장치들(도시되지 않음)을 포함한다.

[0074] 컴퓨터(1202)는 원격 컴퓨터(들)(1248)와 같은 하나 이상의 원격 컴퓨터로의 유선 및/또는 무선 통신을 통해 논리적 접속을 사용하여 네트워크화된 환경에서 동작할 수 있다. 원격 컴퓨터(들)(1248)는 워크스테이션, 서버 컴퓨터, 라우터, 퍼스널 컴퓨터, 휴대용 컴퓨터, 마이크로프로세서 기반 오락 기구, 피어 장치 또는 기타 통상의 네트워크 노드일 수 있고, 전형적으로 컴퓨터(1202)와 관련하여 설명된 구성요소들의 다수 또는 전부를 포함하지만, 간결성을 위하여, 메모리/저장 장치(1250)만이 예시되어 있다. 도시된 논리적 접속들은 LAN(local area network)(1252) 및/또는 더 큰 네트워크, 예를 들면 WAN(wide area network)(1254)으로의 유선/무선 연결을 포함한다. 그러한 LAN 및 WAN 네트워킹 환경은 사무실 및 회사에서 흔한 것이고, 인트라넷과 같은 전사적 컴퓨터 네트워크(enterprise-wide computer network)를 용이하게 하고, 그것들 모두는 글로벌 통신 네트워크, 예를 들면, 인터넷에 접속할 수 있다.

[0075] LAN 네트워킹 환경에서 사용될 때, 컴퓨터(1202)는 유선 및/또는 무선 네트워크 인터페이스 또는 어댑터(1256)를 통하여 LAN(1252)에 접속된다. 어댑터(1256)는 LAN(1252)으로의 유선 및/또는 무선 통신을 용이하게 할 수 있고, LAN은 또한 어댑터(1256)의 무선 기능과 통신하기 위해 그 위에 배치된 무선 액세스 포인트를 포함할 수 있다.

[0076] WAN 네트워킹 환경에서 사용될 때, 컴퓨터(1202)는 모뎀(1258)을 포함할 수 있고, 또는 WAN(1254) 상의 통신 서버에 접속되고, 또는 인터넷을 경유하는 등에 의해 WAN(1254)을 통하여 통신을 설정하기 위한 다른 수단을 갖는다. 내장형 또는 외장형일 수 있고 유선 및/또는 무선 장치일 수 있는 모뎀(1258)은 입력 장치 인터페이스(1242)를 통해 시스템 버스(1208)에 접속된다. 네트워크화된 환경에서, 컴퓨터(1202), 또는 그의 부분들에 관하여 도시된 프로그램 모듈들은 원격 메모리/저장 장치(1250)에 저장될 수 있다. 도시된 네트워크 접속은 예시적인 것이며 이 컴퓨터들 사이에 통신 링크를 설정하는 기타 수단이 사용될 수 있다는 것을 인식할 것이다.

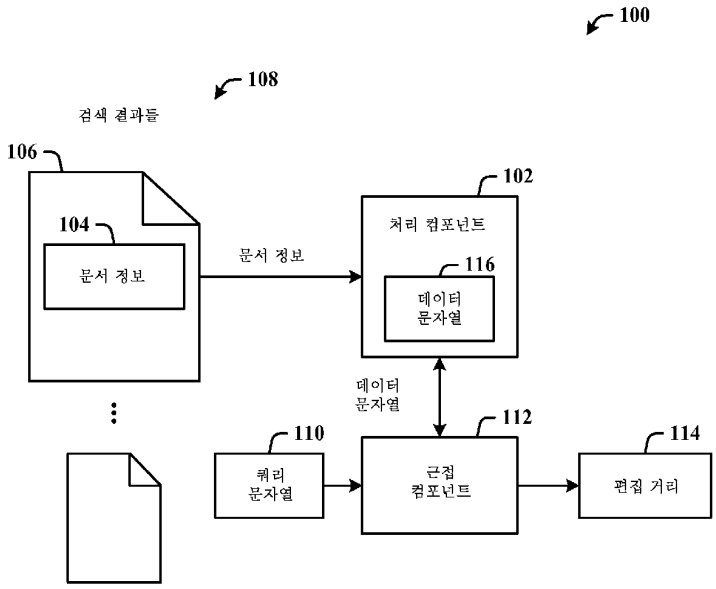
[0077] 컴퓨터(1202)는, 예를 들면, 프린터, 스캐너, 데스크톱 및/또는 휴대용 컴퓨터, PDA(personal digital assistant), 통신 위성, 무선으로 탐지 가능한 태그와 관련된 임의의 장비 또는 위치(예를 들면, 키오스크, 가판대, 화장실) 및 전화기와 무선 통신으로(예를 들면, IEEE 802.11 OTA(over-the-air) 변조 기법) 동작 가능하게 배치된 무선 장치들과 같은, 유선 및 무선 장치들 또는 엔티티들과 IEEE 802 패밀리의 표준들을 이용하여 통신하도록 동작 가능하다. 이것은 적어도 와이파이(Wi-Fi)(또는 Wireless Fidelity), 와이맥스(WiMax), 및 블루투스(Bluetooth™) 무선 기술들을 포함한다. 따라서, 통신은 종래의 네트워크에서와 같이 미리 정의된 구조이거나 또는 단순히 적어도 2개의 장치들 사이의 애드 혹(ad hoc) 통신일 수 있다. 와이파이 네트워크들은 안전하고 신뢰할 수 있고 빠른 무선 연결을 제공하기 위해 IEEE 802.11x(a, b, g 등)라 불리는 라디오 기술들을 이용한다. 와이파이 네트워크는 컴퓨터들을 서로에, 인터넷에, 또는 (IEEE 802.3 관련 매체 및 기능들을 이용하는) 유선 네트워크들에 접속하기 위해 이용될 수 있다.

[0078] 위에 설명된 것은 개시된 아키텍처의 예를 포함한다. 물론, 컴포넌트들 및/또는 방법들의 모든 상상할 수 있는 조합을 설명하는 것은 불가능하지만, 통상의 기술을 가진 당업자는 다수의 추가적인 조합들 및 치환들이 가능하

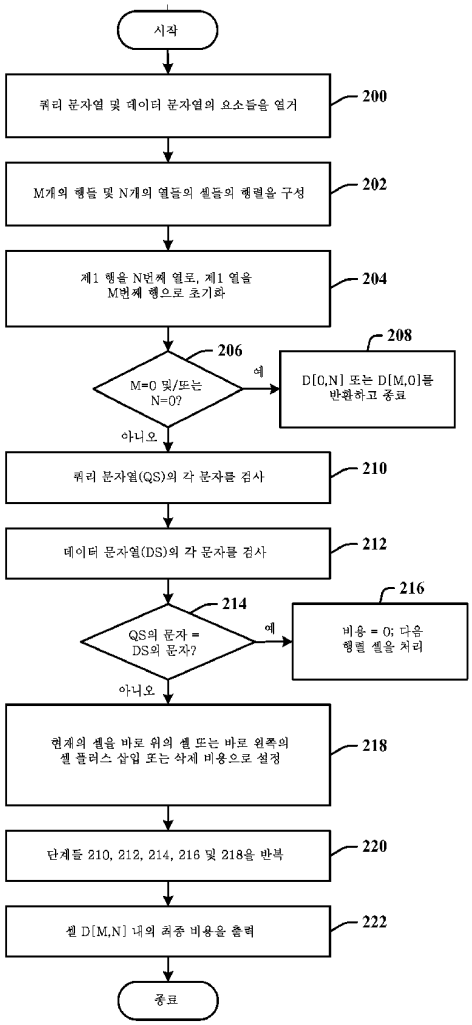
다는 것을 인지할 수 있다. 따라서, 이 새로운 아키텍처는 부속된 청구항들의 정신 및 범위 안에 있는 모든 그러한 변경들, 수정들 및 변형들을 포함하도록 의도된다. 또한, 용어 "includes"가 상세한 설명 또는 청구항들에서 사용되는 한에는, 그러한 용어는 용어 "comprising"이 청구항에서 전이 단어로서 채용될 때 해석되는 것처럼 용어 "comprising"과 유사한 방식으로 포괄적이도록 의도된다.

도면

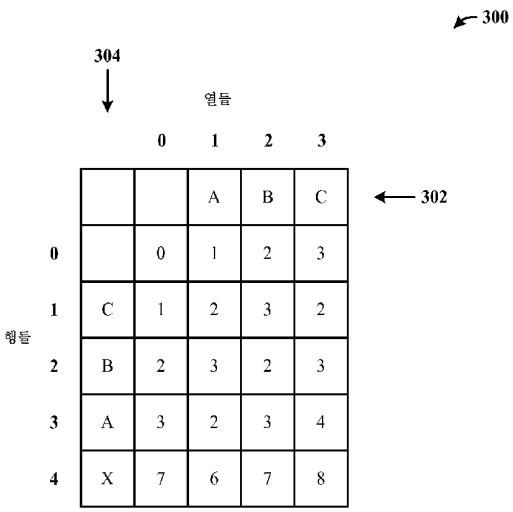
도면1



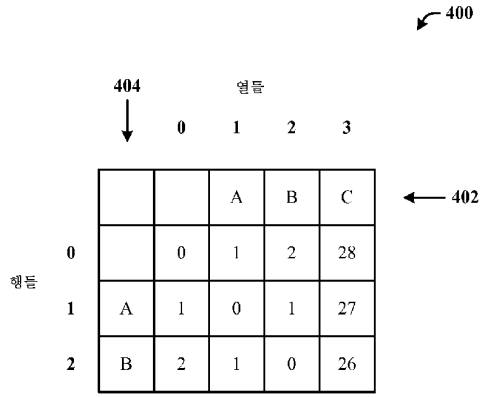
도면2


시작
쿼리 문자열 및 데이터 문자열의 요소들을 얻기
200
M개의 행들 및 N개의 열들의 셀들의 행렬을 구성
202
제1 행을 N번째 열로, 제1 열을 M번째 행으로 초기화
204
206
M=0 및/또는 N=0?
예
208
D[0,N] 또는 D[M,0]를 반환하고 종료
아니오
쿼리 문자열(QS)의 각 문자를 검사
210
데이터 문자열(DS)의 각 문자를 검사
212
214
QS의 문자 = DS의 문자?
예
216
비용 = 0; 다음 행렬 셀을 처리
아니오
현재의 셀을 바로 위의 셀 또는 바로 왼쪽의 셀 플러스 삽입 또는 삭제 비용으로 설정
218
단계들 210, 212, 214, 216 및 218을 반복
220
셀 D[M,N] 내의 최종 비용을 출력
222
종료

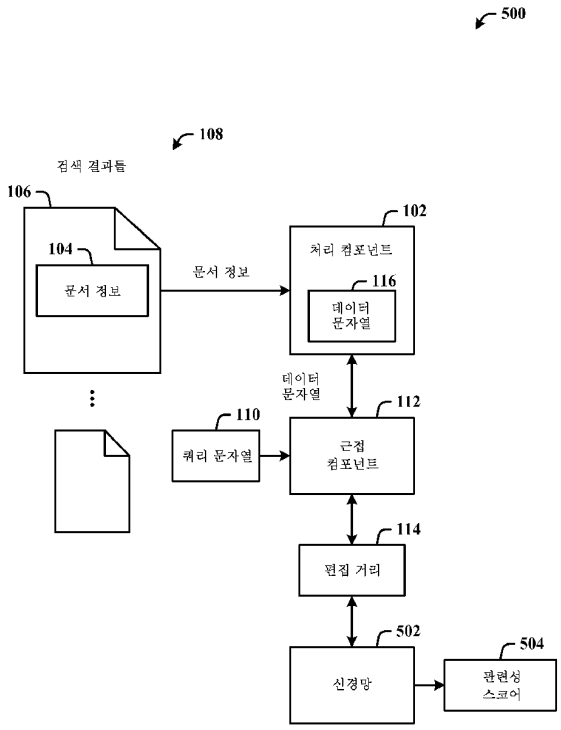
도면3


300
304
열들
0 1 2 3
A B C
302
행들
0 0 1 2 3
1 C 1 2 3 2
2 B 2 3 2 3
3 A 3 2 3 4
4 X 7 6 7 8

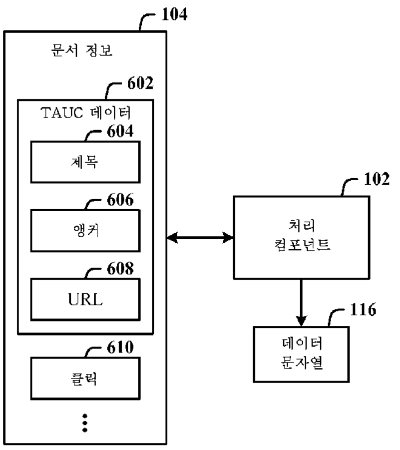
도면4


400
404
열들
0
1
2
3
402
행들
A
B
C
0
1
2
28
A
1
0
1
27
B
2
1
0
26

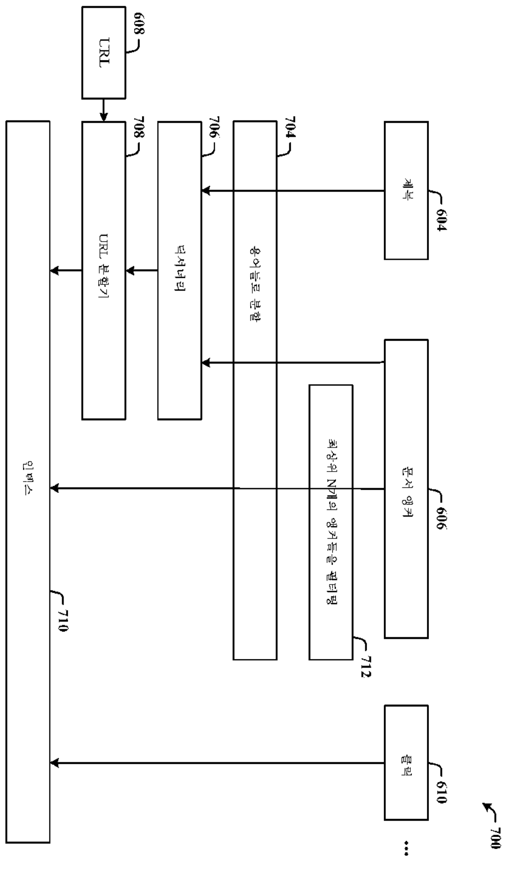
도면5


500
108
검색 결과들
106
104
문서 정보
문서 정보
102
처리 컴포넌트
116
데이터 문자열
데이터 문자열
110
쿼리 문자열
112
근접 컴포넌트
114
편집 거리
502
신경망
504
관련성 스코어

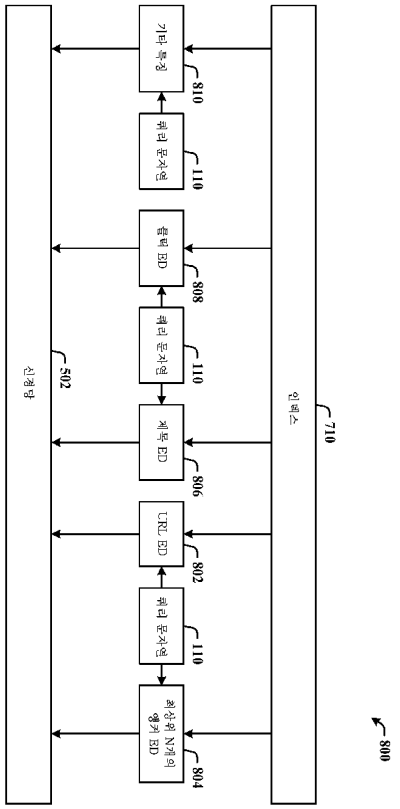
도면6


104
문서 정보
602
TAUC 데이터
604
제목
606
앵커
608
URL
610
클릭
102
처리 컴포넌트
116
데이터 문자열

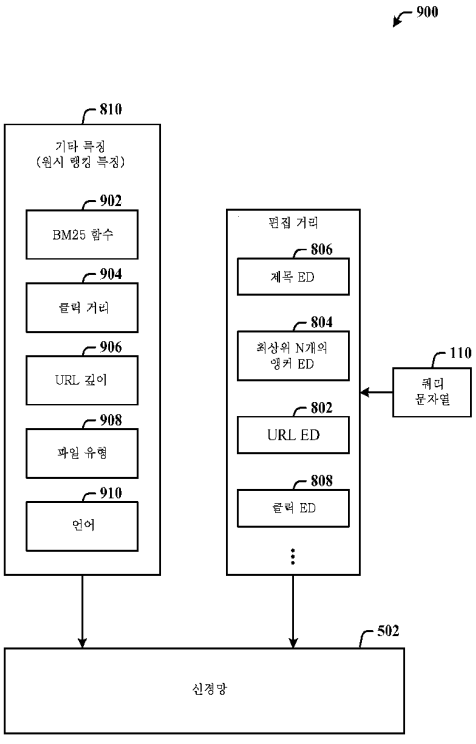
도면7


URL
608
708
URL 분할기
706
딕셔너리
704
용어들로 분할
제목
604
문서 앵커
606
최상위 N개의 앵커들을 필터링
712
인덱스
710
클릭
610
700

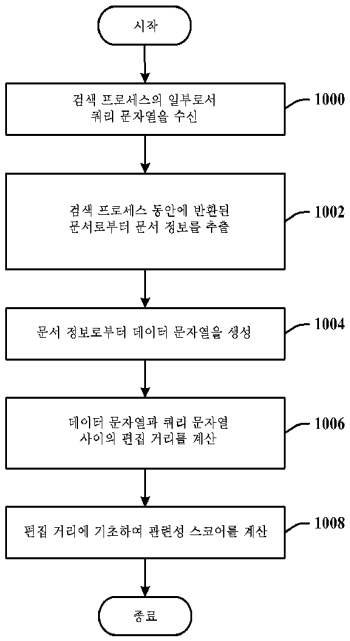
도면8


기타 특징
810
쿼리 문자열
110
클릭 ED
808
쿼리 문자열
110
계목 ED
806
URL ED
802
쿼리 문자열
110
최상위 N개의 앵커 ED
804
신경망
502
순위기
710
800

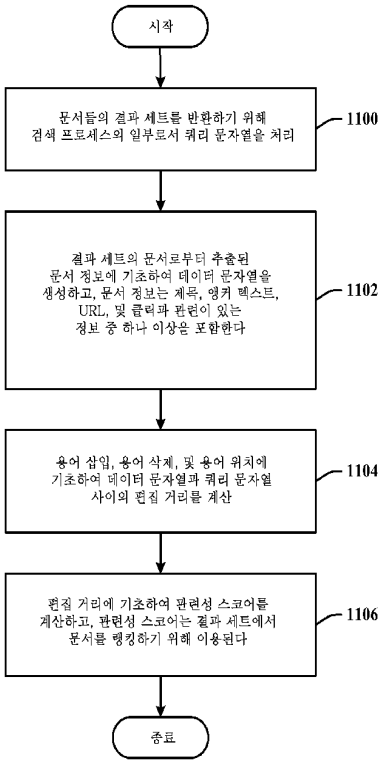
도면9


900
810
기타 특징
(원시 랭킹 특징)
902
BM25 함수
904
클릭 거리
906
URL 깊이
908
파일 유형
910
언어
묶집 거리
806
계목 ED
804
최상위 N개의 앵커 ED
802
URL ED
808
클릭 ED
110
쿼리 문자열
502
신경망

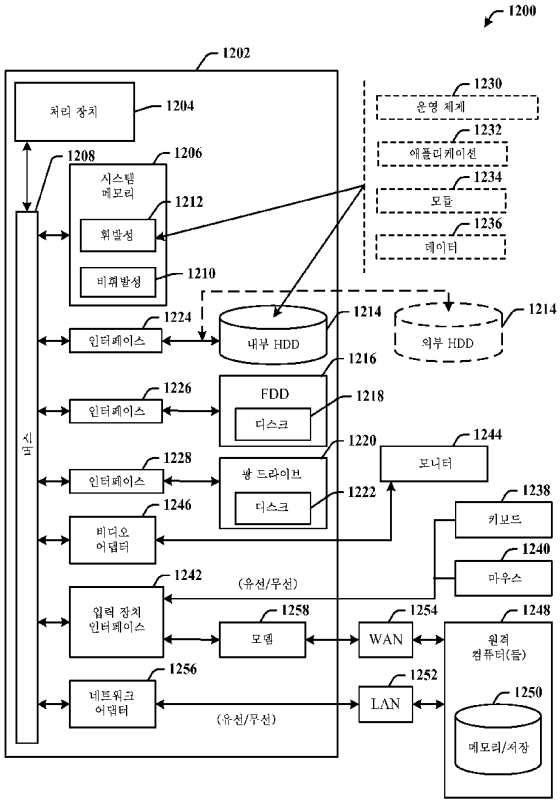
도면10


시작
검색 프로세스의 일부로서 쿼리 문자열을 수신
1000
검색 프로세스 동안에 반환된 문서로부터 문서 정보를 추출
1002
문서 정보로부터 데이터 문자열을 생성
1004
데이터 문자열과 쿼리 문자열 사이의 편집 거리를 계산
1006
편집 거리에 기초하여 관련성 스코어를 계산
1008
종료

도면11


시작
문서들의 결과 세트를 반환하기 위해 검색 프로세스의 일부로서 쿼리 문자열을 처리
1100
결과 세트의 문서로부터 추출된 문서 정보에 기초하여 데이터 문자열을 생성하고, 문서 정보는 제목, 앵커 텍스트, URL, 및 클릭과 관련이 있는 정보 중 하나 이상을 포함한다
1102
용어 삽입, 용어 삭제, 및 용어 위치에 기초하여 데이터 문자열과 쿼리 문자열 사이의 편집 거리를 계산
1104
편집 거리에 기초하여 관련성 스코어를 계산하고, 관련성 스코어는 결과 세트에서 문서를 랭킹하기 위해 이용된다
1106
종료

도면12


1200
1202
처리 장치
1204
1208
1206
시스템 메모리
1212
휘발성
비휘발성
1210
버스
1230
운영 체제
1232
애플리케이션
1234
모듈
1236
데이터
1224
인터페이스
내부 HDD
1214
외부 HDD
1214
1216
FDD
디스크
1218
1226
인터페이스
1220
1244
모니터
1228
인터페이스
광 드라이브
디스크
1222
1238
키보드
1246
비디오 어댑터
1240
마우스
1242
(유선/무선)
입력 장치 인터페이스
1258
모뎀
1254
WAN
1248
원격 컴퓨터(들)
1256
네트워크 어댑터
1252
LAN
1250
메모리/저장
(유선/무선)