(19)日本国特許庁(JP)　(12)公開特許公報(A)　(11)特許出願公開番号
特開2010-39997
(P2010-39997A)
(43)公開日　平成22年2月18日(2010.2.18)

(51)Int.Cl.		F I			テーマコード（参考）
G06F 17/30	(2006.01)	G06F	17/30	330B	5B075
		G06F	17/30	340B	
		G06F	17/30	140	

審査請求 未請求　請求項の数 10　ＯＬ　（全 18 頁）

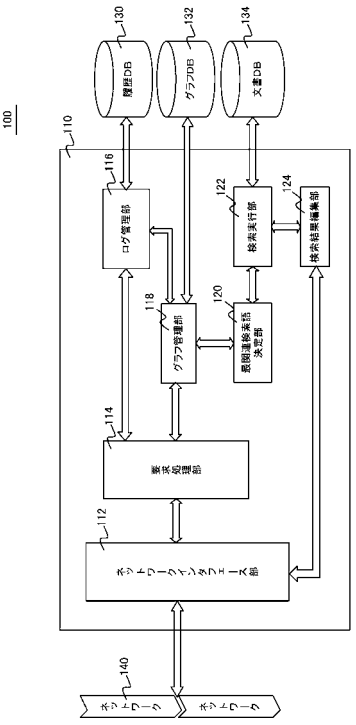
(21)出願番号　特願2008-205582 (P2008-205582)
(22)出願日　平成20年8月8日 (2008.8.8)

(71)出願人　000006747
株式会社リコー
東京都大田区中馬込１丁目３番６号
(74)代理人　100110607
弁理士　間山　進也
(72)発明者　徐　盈輝
東京都大田区中馬込１丁目３番６号　株式会社リコー内
(72)発明者　荒木　禎史
東京都大田区中馬込１丁目３番６号　株式会社リコー内
(72)発明者　池田　哲也
東京都大田区中馬込１丁目３番６号　株式会社リコー内

Ｆターム(参考)　5B075 KK02　ND03　PP23　PR03　UU40

(54)【発明の名称】情報検索装置、情報検索方法、プログラム、および記録媒体

(57)【要約】
【課題】検索履歴を用いて新たな情報検索を行う、情報検索装置、情報検索方法、プログラム、記録媒体を提供する。
【解決手段】情報検索装置１００は、検索要求を受領して検索語または検索語列を抽出する要求処理部と、検索要求の履歴を受領して履歴データベースを更新するログ管理部１１６と、各ノード間のリンクについて定義された重み付け値を計算し、計算された重み付け値を要素とする隣接マトリックスを生成するグラフ管理部１１８と、別の検索要求を受領して、隣接マトリックスに少なくともクエリーノードを追加して最関連検索語で検索クエリーを拡張する最関連検索語決部１２０と、拡張検索クエリーを受領して、文書データベースに照会を実行し、検索結果を受領する検索実行部１２２と、検索結果を編集して構造化文書を作成し、検索要求の送付元に構造化文書を送付させる検索結果編集部１２４とを含んでいる。
【選択図】図１



【特許請求の範囲】
【請求項１】
ネットワークを介して受領した検索要求の履歴を追加して情報検索を実行する情報検索装置であって、
ネットワークを介して外部から送付される検索要求を受領して前記検索要求の履歴を登録し、前記検索要求に含まれる検索クエリーから、検索語または検索語列を抽出する要求処理手段と、
前記要求処理手段からの前記検索要求の履歴を受領して履歴データベースを更新するログ管理手段と、
前記検索要求の履歴および前記検索クエリーを使用して、前記検索要求のユーザノード、クエリーノード、および閲覧履歴の登録された閲覧済文書ノードに関する情報を登録し、前記情報から前記各ノード間のリンクについて定義された重み付け値を計算し、計算された前記重み付け値を要素とする隣接マトリックスを生成するグラフ管理手段と、
外部から別の検索要求を受領して、前記別の検索要求から前記隣接マトリックスに少なくともクエリーノードを追加し、前記追加したクエリーノードに対応する要素を初期設定したベクトルを生成して前記隣接マトリックスを使用してランキングベクトルが収束するまで反復計算して前記別の検索要求に含まれる検索クエリーを拡張するための最関連検索語を決定し、前記別の検索要求に含まれる検索クエリーを拡張した拡張検索クエリーを生成する最関連検索語決定手段と、
前記拡張検索クエリーを受領して、文書データベースに照会を実行し、検索結果を受領する検索実行手段と、
前記検索結果を編集して構造化文書を作成し、前記検索要求の送付元に前記構造化文書を送付させる検索結果編集手段と
を含む情報検索装置。
【請求項２】
前記ノード間のリンクは、少なくとも前記ユーザノード間、前記クエリーノード間、および前記クエリーノードと閲覧済文書ノードとの間に定義され、前記ユーザノード間のリンクは、ユーザが過去に送付した検索クエリーの類似性について重み付け値が与えられ、前記クエリーノード間のリンクは、複数の検索クエリーの間の時系列的間隔を使用して前記検索クエリー間の類似性について重み付け値が与えられ、前記クエリーノードと閲覧済文書ノードとの間に定義されるリンクは、前記クエリーノードに対応する前記検索語と前記閲覧済文書ノードに登録された閲覧済文書の類似度についての重み付け値が与えられる、請求項１に記載の情報検索装置。
【請求項３】
前記最関連検索語決定手段は、直前の反復サイクルのランキングベクトルを、試行ベクトルとして設定し、前記試行ベクトルを使用して前記隣接マトリックスによる反復計算を実行してランキングベクトルを更新する反復計算を実行し、最新のランキングベクトルと、当該反復サイクルでの試行ベクトルとの間の内積が設定された少値以下となった場合に収束を判定して、前記最新のランキングベクトルから前記最関連検索語を決定する、請求項１または２に記載の情報検索装置。
【請求項４】
前記グラフ管理手段は、前記別の検索要求に含まれる検索クエリーを使用して前記隣接マトリックスの前記クエリーノードと前記閲覧済文書ノードとの間の重み付け値を使用して前記隣接マトリックスを更新する、請求項２または３に記載の情報検索装置。
【請求項５】
ネットワークを介して受領した検索要求の履歴を追加してコンピュータが実行する情報検索方法であって、前記コンピュータが、
ネットワークを介して外部から送付される検索要求を受領して前記検索要求の履歴を登録し、前記検索要求に含まれる検索クエリーから、検索語または検索語列を抽出するステップと、

前記検索要求の履歴を受領して履歴データベースを更新するステップと、
前記検索要求の履歴および前記検索クエリーを使用して、前記検索要求のユーザノード、クエリーノード、および閲覧履歴の登録された閲覧済文書ノードに関する情報を登録し、前記情報から前記各ノード間のリンクについて定義された重み付け値を計算し、計算された前記重み付け値を要素とする隣接マトリックスを生成するステップと、
外部から別の検索要求を受領して、前記別の検索要求から前記隣接マトリックスに少なくともクエリーノードを追加し、前記追加したクエリーノードに対応する要素を初期設定したベクトルを生成して前記隣接マトリックスを使用してランキングベクトルが収束するまで反復計算して前記別の検索要求に含まれる検索クエリーを拡張するための最関連検索語を決定し、前記別の検索要求に含まれる検索クエリーを拡張した拡張検索クエリーを生成するステップと、
前記拡張検索クエリーを受領して、文書データベースに照会を実行し、検索結果を受領するステップと、
前記検索結果を編集して構造化文書を作成し、前記検索要求の送付元に前記構造化文書を送付するステップと
を実行する、情報検索方法。
【請求項６】
前記ノード間のリンクが、少なくとも前記ユーザノード間、前記クエリーノード間、および前記クエリーノードと閲覧済文書ノードとの間に定義され、さらに、前記隣接マトリックスを生成するステップは、
前記ユーザノード間のリンクに対して、ユーザが過去に送付した検索クエリーの類似性について重み付け値を計算するステップと、
前記クエリーノード間のリンクに対して、複数の検索クエリーの間の時系列的間隔を使用して前記検索クエリー間の類似性について重み付け値を計算するステップと、
前記クエリーノードと閲覧済文書ノードとの間に定義されるリンクに対して、前記クエリーノードに対応する前記検索語と前記閲覧済文書ノードに登録された閲覧済文書の類似度についての重み付け値を計算するステップと
を含む、請求項５に記載の情報検索方法。
【請求項７】
前記拡張検索クエリーを生成するステップは、
直前の反復サイクルのランキングベクトルを、試行ベクトルとして設定するステップと、
前記試行ベクトルを使用して前記隣接マトリックスを使用した反復計算を実行してランキングベクトルを更新するステップと、
最新のランキングベクトルと、当該反復サイクルでの試行ベクトルとの間の内積が設定された少値以下となった場合に収束を判定して、前記最新のランキングベクトルから前記最関連検索語を決定するステップと
を含む、請求項５または６に記載の情報検索方法。
【請求項８】
前記隣接マトリックスを生成するステップは、前記別の検索要求に含まれる検索クエリーを使用して前記隣接マトリックスの前記クエリーノードと前記閲覧済文書ノードとの間の重み付け値を使用して前記隣接マトリックスを更新する、請求項５または６に記載の情報検索方法。
【請求項９】
情報処理装置が請求項６～８のいずれか１項に記載の各ステップを実行するためのコンピュータ実行可能なプログラム。
【請求項１０】
請求項９に記載のコンピュータ実行可能なプログラムを記録したコンピュータ可読な記録媒体。
【発明の詳細な説明】

【技術分野】
【０００１】
本発明は、情報検索に関し、より詳細には、過去の検索履歴を効率的に利用し、新たな情報検索を行う、情報検索装置、情報検索方法、プログラム、および記録媒体に関する。
【背景技術】
【０００２】
情報処理装置の性能向上およびネットワーク通信速度の向上から、インターネットといったネットワークを介した情報検索が普及している。ネットワークを介した情報検索は、多くの場合、クライアントコンピュータ（以下、クライアントとして参照する。）のユーザが、ウェブブラウザを介して検索要求をウェブサーバに送付することにより実行される。検索要求を受領したウェブサーバは、検索要求に含まれる検索クエリーから検索語または検索語の語列を抽出し、ＳＱＬパーサなどを使用してＳＱＬ文に設定する。そして、ウェブサーバは、ウェブサーバに接続されたデータベースに照会を発行し、データベースでの検索結果を、検索要求の発行元のクライアントに返すことにより、ユーザが、ウェブサーバを使用した情報検索の結果を利用可能とする。
【０００３】
情報検索を考えてみれば、ユーザは、多くの場合、一定の検索エンジンのＵＲＬ(Uniform Resource Locator)を「ブックマーク」や、「お気に入り」に登録して、繰り返し一定の検索エンジンにアクセスして種々の情報検索を実行する。このため、検索エンジンを管理するウェブサーバには、特定のユーザに関する検索履歴が、アクセスログなどとして蓄積されて行く。これは、例えば、商業用の検索エンジンばかりではなく、企業、官公庁、大学、公共施設での検索サービスを提供するウェブサーバでも同様である。
【０００４】
上述した場合、新に受領した検索要求に対し、ウェブサーバに蓄積された検索履歴を使用して検索要求を拡張して検索クエリーを生成することにより、ユーザによる検索効率をより高めることができると考えられる。
【０００５】
これまで、ユーザによるネットワークを介した検索効率を改善する種々の検討がなされている。例えば、特開２００６－１２７５２９号公報（特許文献１）は、ウェブページの検索の効率を向上させるため、ウェブページに階層構造を設け、階層ごとにウェブページのページ重要性ランキングを計算しておき、階層構造にわたるランダムウォークを使用してページ重要性に関連付けて、ウェブページを検索するシステムを開示する。
【０００６】
また、特開２００２－３０４４１１号公報（特許文献２）では、利用者識別情報および検索語を使用して、利用者の過去の利用履歴情報および検索情報の分野を区分して新たな利用履歴情報を生成して検索結果の優先順位を決定する情報検索配信システムを開示する。さらに、特開２００４－１８５３３９号公報（特許文献３）は、ユーザが現在閲覧している文書および文書に付随する情報と、ユーザが過去に指定した検索語の履歴とから検索式を自動的に生成する文書検索システムを開示する。
【特許文献１】特開２００６－１２７５２９号公報
【特許文献２】特開２００２－３０４４１１号公報
【特許文献３】特開２００４－１８５３３９号公報
【発明の開示】
【発明が解決しようとする課題】
【０００７】
特許文献１に記載されたシステムは、ウェブページのページ重要性に関連してランダムウォークを使用してウェブページの検索を実行する点は開示する。しかしながら、多くの文書は、ウェブページのページ単位で重要性が割り与えられていない。このため、特許文献１に記載のシステムは、既存のウェブページについて階層構造を生成しなければ適用できず、現在、インターネットに存在する文書数を考慮すれば現実的なものと言うことはで

きない。また、検索要求の履歴を効果的に利用することを課題とするものではない。
【０００８】
また、特許文献２は、利用者の過去の履歴情報および検索情報の分野を使用して検索結果を生成し、検索結果の優先順位を決定するシステムを開示している。しかしながら、特許文献２に記載されたシステムは、検索結果の優先順位を、利用者の過去の利用履歴情報を使用して決定するものであり、利用履歴を使用するものの、利用履歴を利用して検索範囲を拡張することを課題とするものではない。
【０００９】
さらに特許文献３は、特定ユーザが現在閲覧している文書および文書に付随する情報と、特定ユーザが過去に指定した検索語の履歴とを使用して検索式を自動生成するシステムを開示する。しかしながら、ユーザが閲覧している文書から検索式を自動作成するものであり、検索の多様性や任意性に制限がある。また、特許文献３に記載された技術は、特定のユーザが過去に指定した検索語の履歴を使用して新たな検索語を作製する点で、他のユーザの検索履歴を利用したり、また閲覧済文書についての履歴を含ませて検索効率を向上させることを課題とするものではない。
【００１０】
本発明は、上記従来技術の問題点に鑑みてなされたものであり、本発明では、検索語、ユーザ、閲覧済文書などを含む過去の検索履歴を有効に利用して、検索範囲を拡大させることが可能な情報検索装置、情報検索方法、プログラム、および記録媒体を提供することを目的とする。
【００１１】
また、本発明は、複数の異なるユーザが行った検索の検索履歴を、爾後の検索処理に反映させることにより、検索範囲を自動的に拡張し、検索効率を高めることを可能とする、情報検索装置、情報検索方法、プログラム、および記録媒体を提供することを目的とする。
【課題を解決するための手段】
【００１２】
本発明は、上記課題を解決するために、検索履歴のユーザ、検索クエリー、閲覧済文書の情報を抽出し、検索履歴から隣接グラフを生成する。隣接グラフは、クエリー間、ユーザ間、閲覧文書間、クエリー－ユーザ間、閲覧文書－クエリー間にリンクを定義することにより生成される。また、各リンクには、リンクの端点ノード間の属性により決定される重み付けが定義されていて、隣接グラフを、隣接マトリックスの対応する端点ノード（ｉ，ｊ）についての重み付け値を要素とする隣接マトリックスとして生成する。
【００１３】
隣接マトリックスは、ランダムウォークランキング方法を使用して、クエリー、ユーザ、閲覧済文書の類似性に関連して、それぞれ最関連と推定される最関連検索語が抽出される。各最関連検索語は、ユーザが発行した検索クエリーｑｎｅｗが含む検索語または検索語列に論理和されて、｛ｑｎｅｗ＋ｑｏ｝として、並列検索を実行するための拡張検索クエリーを生成するために利用される。
【００１４】
本発明では、ランダムウォークランキングは、リスタートベクトルを使用して実行され、リスタートベクトルで初期化し、ランキングベクトルを、試行ベクトルとして反復的に使用するランダムウォーキング計算を実行させる。
【００１５】
反復計算は、最新のランキングベクトルと、その反復サイクルでの試行ベクトルとの間の距離、すなわち、内積が設定した小値ε以下となった場合に停止される。反復計算の終了時点では、クエリーに関して、クエリーの類似性の高さに応じてランキングベクトルの要素値が与えられる。また、ユーザおよび閲覧済文書についても、類似度に関連して要素値が与えられる。
【００１６】

検索クエリー、ユーザ、閲覧済文書のそれぞれの種類ごとに最大の要素値を与える端点ノードのうちのクエリーノードが、最関連検索語として抽出され、拡張検索クエリーを生成するために使用される。

【００１７】

拡張検索クエリーは、文書データベースに発行され、情報検索が実行された後に、適切な形式の構造化文書として編集され、クライアントのユーザに検索結果として提示される。

【００１８】

すなわち、本発明によれば、検索語、ユーザ、閲覧済文書などを含む過去の検索履歴を有効に利用して、検索範囲を拡大させることが可能な情報検索装置、情報検索方法、プログラム、および記録媒体を提供することが可能となる。

【００１９】

また、本発明によれば、複数の異なるユーザが行った検索の検索履歴を、爾後の検索処理に反映させることにより、検索範囲を自動的に拡張し、検索効率を高めることを可能とする、情報検索装置、情報検索方法、プログラム、および記録媒体を提供することが可能となる。

【発明を実施するための最良の形態】

【００２０】

以下、本発明を実施形態をもって説明するが、本発明は後述する実施形態に限定されるものではない。図１は、本実施形態の情報検索装置１００の機能ブロックを示す。情報検索装置１００は、ウェブサーバとして構成されており、クライアントコンピュータ（以下、単にクライアントとして参照する。）からの検索要求を受領して文書検索を実行し、検索結果を検索要求の要求元に返す。

【００２１】

情報検索装置１００は、インターネット、ワイアドまたはワイアレス通信を使用するローカルエリアネットワーク（ＬＡＮ）、またはワイドエリアネットワーク（ＷＡＮ）などを含むネットワーク１４０を介して複数のクライアント（図示せず）のユーザから、文書の検索要求を受領する。クライアントは、情報検索装置１００にアクセスするため、Internet Explorer、Mozilla、Opera、NetscapeNavigator（商標）などのブラウザソフトウェアを実装していて、ユーザによる検索要求の発行指令を受領してＨＴＴＰプロトコルなどを使用し、検索エンジンとして構成される情報検索装置１００に検索要求を発行する。

【００２２】

情報検索装置１００は、クライアントが検索要求の作製を容易にするため、クライアントからの要求に応じて検索要求フォームをダウンロードする。クライアントのユーザは、入力フィールドから検索語または複数の検索語を入力して、入力後にSUBMITすると、入力された検索語を含む検索クエリーを、ＧＥＴメソッドまたはＰＯＳＴメソッドにより情報検索装置１００に対して送付する。

【００２３】

クライアントからの検索要求は、ネットワーク１４０を介してネットワークインタフェース部１１２が受領し、ＯＳＩ基本参照モデルにいうところの、データリンク層、ネットワーク層、トランスポート層を経て、本実施形態の情報検索方法を実行するサーバプログラムに検索クエリーを渡している。

【００２４】

本実施形態の情報検索装置１００は、より詳細には、中央処理装置（ＣＰＵ）がＲＡＭなどの実行空間にプログラムを展開してデータを読込、ＣＰＵによるプログラムの実行によりコンピュータ上に各機能部が実現されている。ネットワークインタフェース部１１２が受領した検索クエリーは、要求処理部１１４に送付される。要求処理部１１４では、検索クエリーに含まれるユーザＩＤなどを検査して、情報処理装置１００に当該ユーザが既登録であるか否かを判断し、既登録ユーザでない場合には、新規ユーザアカウントおよびパスワードの登録処理を実行する。

【００２５】
さらに、要求処理部１１４は、ユーザからアクセスを受領して、ログ管理部１１６を呼出し、検索要求を送付したユーザのユーザＩＤ、タイムスタンプ、アクセス回数、特定の文書に対するアクセス開始から、アクセス終了までの時間幅で与えられるアクセス期間などをモニタし、取得した各情報を、履歴データベース（以下、データベースにつき、ＤＢとして略記する。）１３０に登録する。
【００２６】
さらに、要求処理部１１４は、ユーザが新規であると判断した場合、グラフ管理部１１８を呼出してユーザの少なくともユーザＩＤ、ユーザ名などを通知して、隣接グラフの新たなノードとして登録する処理を依頼する。また、要求処理部１１４は、ユーザからの検索要求が含む検索クエリーを検査し、ログ管理部１１６に検索クエリーが新規であるか否かの検査を依頼する。ログ管理部１１６は、要求処理部１１４に対して検索クエリーが新規か、新規でないかの判断結果を通知する。そして検索クエリーが新規であるとの通知を受領した場合、要求処理部１１４は、グラフ管理部１１８に対して、新たな検索クエリーを、隣接グラフの新たなノードとして追加するように指令を発行する。
【００２７】
グラフ管理部１１８は、上述した新規なノードの検出に対応して隣接グラフの新たなノードを追加する処理の他、隣接グラフを構成するためのグラフＤＢ１３２を管理する。グラフ管理部１１８は、隣接グラフが存在しない場合、ノードの蓄積に対応して、ノードの追加および追加されたノードを含む隣接グラフのグラフデータを更新する処理を実行する。このため、グラフ管理部１１８は、ログ管理部１１６に対して、グラフ管理部１１８が検索処理中のノードに関連する履歴を、履歴ＤＢ１３０を参照して取得し、グラフＤＢ１３２の適切なテーブルの項目に登録する処理を実行する。
【００２８】
さらに、グラフ管理部１１８は、履歴ＤＢ１３０を参照してユーザによる文書の閲覧履歴を取得し、グラフＤＢ１３２に登録する。グラフ管理部１１８が管理するグラフデータは、クエリーノードテーブル、閲覧文書ノードテーブル、ユーザノードテーブルに登録されて管理される。さらに、グラフ管理部１１８は、登録された各テーブルのエントリ項目について、項目間のリンクを生成させ、各リンクについての重み付け値を計算し、隣接グラフの要素値として登録する。
【００２９】
また、ユーザからの検索要求を受領し、各ノードに対する更新処理が終了した後、グラフ管理部１１８は、最関連検索語決定部１２０を呼出して、ユーザが現在検索を要求する検索クエリーの拡張処理を指令する。検索クエリーの拡張処理は、本実施形態では、ユーザ、クエリー、または閲覧済文書に最も関連する検索クエリーを、グラフＤＢ１３２に登録された隣接グラフの解析に基づいて抽出し、ＳＱＬパーサなどを使用してユーザが送付した検索クエリーに＜ＯＲ＞属性で追加する処理によって実行することができ、以後、拡張された検索クエリーを、拡張検索クエリーとして参照する。
【００３０】
拡張検索クエリーは、ＳＱＬ文を文書ＤＢ１３４へと送付され、データベースサーバにより、文書の検索が実行された後、情報検索装置１００の検索実行部１２２に対し、文書の抽出結果が返される。文書ＤＢ１３４の検索結果は、検索実行部１２２から検索結果編集部１２４へとい送付され、検索結果からＨＴＭＬやＸＭＬなどの構造化文書が作成され、ネットワークインタフェース部１１２を介して検索要求の要求元のクライアントに返され、ユーザの検索要求に関連する一連のトランザクションが完了する。
【００３１】
図２は、本実施形態の履歴ＤＢ１３０およびグラフＤＢ１３２が管理する、各テーブルのデータ構造を示す。データ構造２００は、履歴ＤＢ１３０が管理するテーブルを示し、データ構造２５０は、グラフＤＢ１３２が管理するテーブルを示す。データ構造２００は、履歴ＤＢ１３０が管理するテーブルであり、テーブル２１０は、ユーザが閲覧した文書

に関連するデータを登録する。また、テーブル２２０は、検索要求が含む検索クエリーとユーザとを関連付けるとともに、当該検索により検索された一致スコアの上位Ｋ番目を意味するｔｏｐＫの文書Id、タイムスタンプ、閲覧時間間隔などが登録されている。また、テーブル２２０には、その他、オプションフィールドなどが設けられ、特定の用途に対するデータの拡張を許容する構成とされている。

【００３２】

また、データ構造２００には、ユーザのユーザId、ログイン名、パスワードなどを格納するテーブル２３０が含まれていて、情報検索装置１００にアクセスするユーザについて隣接グラフのノードとして設定可能とする。なお、後述するデータ構造２５０についても同様であるが、ユーザに関連するユーザId、ログイン名、パスワードなどについては、ユーザ情報を専ら管理するユーザ情報ＤＢに登録し、要求処理部１１４がその処理の必要に応じて、ユーザＤＢにアクセスして、グラフＤＢ１３２での処理のために利用させることができる。

【００３３】

データ構造２５０は、グラフＤＢ１３２が隣接グラフを作製する処理のために管理する情報および隣接グラフ自体の情報を含んで生成される。テーブル２６０は、検索クエリーに関連して文書ＤＢ１３４の文書がどのようにアクセスされたかを登録する文書－検索クエリー間のアクセス履歴を登録する。また、テーブル２７０は、文書ＤＢ１３４内で、文書の閲覧履歴を登録しており、文書が閲覧履歴を有している場合に与えられるｖｉｓＤｏｃId、すなわち閲覧済文書識別値に関連付けて、閲覧済み文書が含む、検索対象のキーワードとして使用される単語または単語リスト、滞在時間、検索日、閲覧頻度などを登録する。また、データ構造２５０は、テーブル２８０としてユーザに関連する情報も登録しているが、テーブル２８０は、別途構成されるユーザＤＢが利用できる場合には、データ構造２５０に含まれなくともよい。

【００３４】

さらに、データ構造２５０は、隣接グラフ２９０を含んで構成されている。隣接グラフ２９０は、２次元配列として定義され、好適にはグラフテーブルとして表現することができる。この隣接グラフ２９０、すなわちグラフテーブルは、データ構造２５０が含む各テーブルを参照し、ノード間に重み付け値を割り当てて生成され、特定の検索要求を受領した場合に、隣接グラフからユーザ、検索クエリー、および閲覧済文書に関連して最関連の検索語を推定し、その時点で受領した検索クエリーを、最関連検索語決定部１２０が決定した最関連検索語で拡張させるために利用される。

【００３５】

また、隣接グラフ２９０を含め、情報検索装置１００が新たな検索クエリー、ユーザを検出した場合、データ構造２００およびデータ構造２５０が更新され、これらに対応して隣接グラフも更新される。なお、本実施形態では隣接グラフは、２次元配列を使用するマトリックス形式で生成され、線形代数の各処理を使用して最関連検索語の探索および決定を実行する。

【００３６】

図３は、本実施形態の情報検索方法の処理についての概略的なフローチャートを示す。図３の処理は、ステップＳ３００から開始し、ステップＳ３０１でユーザから検索要求を受領する。ステップＳ３０２で、当該検索要求を発行したユーザのユーザIdなどを、ユーザ情報をルックアップして検査し、当該ユーザが登録されていない場合（ｎｏ）、ステップＳ３０７で新たなユーザノードとして、ユーザテーブルに登録し、処理をステップＳ３０３に渡す。また、ステップＳ３０２の判断でユーザが既登録であると判断した場合（ｙｅｓ）、ステップＳ３０３で、検索クエリーが新規であるか否かを判断し、検索クエリーが新規な場合（ｙｅｓ）、ステップＳ３０３で検索クエリーをクエリーテーブルに追加する。

【００３７】

なお、検索クエリーは、単一の検索語または複数の検索語を含んでおり、ステップＳ３

０３の処理では、検索クエリーが含む検索語を識別してクエリーテーブルに登録する。

【００３８】

一方、ステップＳ３０４では、検索クエリー、ユーザ、または検索クエリーおよびユーザの両方が新規であった場合、グラフテーブルに新規ノードとして追加する。そして、ステップＳ３０５では、隣接グラフの行列要素として登録するべき重み付け値を計算し、対応する検索クエリーId、ユーザId、閲覧済文書Idなどに対応付けてマトリックスを更新する。ステップＳ３０６では、ユーザ、文書、検索クエリーの各ノードについて生成された重み付け値を使用して、ＲＷＲ(Random Walk Ranking)付けを実行して、最関連検索語を抽出し、ユーザから受領した検索クエリーに倫理和し、拡張検索クエリーを生成する。

【００３９】

ステップＳ３０８では文書ＤＢ１３４に対して拡張検索クエリーを発行し、ステップＳ３０９で、文書ＤＢ１３４から検索結果を受領する。ステップＳ３１０では、検索結果を類似度などを使用してランク付けし、構造化文書として編集し、ユーザに検索結果を送付した後、処理をステップＳ３０１に戻し、以後のユーザからの検索要求の処理を反復する。

【００４０】

図４は、本実施形態で、隣接グラフを構成するために使用する、図２に示したデータ構造２５０の更新処理のフローチャートを示す。図４の処理は、ステップＳ４００から開始し、ステップＳ４０１で、履歴ＤＢ１３０に接続する。ステップＳ４０２で、各テーブルのレコードを検査し、処理対象とするべき各テーブルのレコードに空があるか否かを判断する。ステップＳ４０２の判断で、各テーブルが空のレコードを有していると判断された場合（ｙｅｓ）、ステップＳ４０３で、ユーザノード情報を検索して登録するべき情報を抽出し、ユーザノードテーブルに追加登録する。

【００４１】

ステップＳ４０４では、クエリーノード情報を検索して、登録するべき情報を抽出し、クエリーノードテーブルに追加登録する。さらに、ステップＳ４０５では、閲覧済み文書ノード情報を検索して、登録するべき情報を抽出し、閲覧済み文書テーブルに登録する。その後、処理をステップＳ４０２に戻し、登録するべき各テーブルの情報がある場合、各テーブルのレコードの空きがなくなるまでステップＳ４０３～ステップＳ４０５の処理を反復させ、データ構造２５０を更新してゆく。

【００４２】

一方、ステップＳ４０２でテーブルがすでに空のレコードを有していないと判断された場合（ｎｏ）、ステップＳ４０６で、新に登録されたノードを抽出し、追加ノードリストに一時的に登録する。ステップＳ４０７では、追加ノードリスト中の全ノードについて処理が終了したか否かを判断し、全ノードについて処理を終了した場合（ｙｅｓ）、処理をステップＳ４１２に分岐させ、処理を終了させる。

【００４３】

また、ステップＳ４０７で追加ノードリスト内に未処理のノードが残っている場合（ｎｏ）、ステップＳ４０８で、各テーブルのサイズがしきい値以下かを判断する。各テーブルのサイズがしきい値以下である場合（ｙｅｓ）、ステップＳ４０９で、各テーブルに該当するノードの情報を追加する。一方、ステップＳ４０８の判断で各テーブルのサイズがしきい値を超えると判断した場合（ｎｏ）、処理をステップＳ４１０に分岐させる。ステップＳ４１０では、各テーブルから最古のタイムスタンプを有するノードの情報を削除する。その後、ステップＳ４１１では、各テーブルのトップレコードに処理中のノードの情報を記入し処理をステップＳ４０７に戻し、追加ノードリストの項目全部について処理が終了するまで、処理を反復させる。

【００４４】

図４に示した処理を使用することにより、履歴ＤＢ１３０に新たなノードとして追加するべき情報が追加された場合に、対応してデータ構造２５０をアップデートさせることができる。なお、図４の処理は、検索要求を受領した段階で検索クエリーについてはオンザ

フライでグラフデータに反映される。また、ユーザなどの他のノードについては、情報処理装置１００が例えば、定期メンテナンスや、夜間などアクセス数が低い時間帯に定期的に履歴ＤＢをポーリングして、新規履歴データを検査することによって実行してもよい。なお、テーブルのサイズや隣接グラフのサイズについて設定されるしきい値は、システム制限によるものであって、使用するシステムの能力に応じて変更され、特に制限はない。

【００４５】

図５は、本実施形態で、データ構造２５０を使用して、隣接グラフの要素値を決定する処理のフローチャートを示す。処理は、ステップＳ５００から開始し、ステップＳ５０１で、ユーザノードテーブルからユーザノード情報を抽出し、グラフテーブルのユーザノードを登録するＵＮＴに格納する。ステップＳ５０２では、クエリーノードテーブルからクエリーノード情報を抽出し、グラフテーブルのＱＮＴに登録する。ステップＳ５０３では、閲覧済文書ノードテーブルから閲覧済文書ノード情報を抽出し、グラフテーブルのＶＤＮＴに格納する。なお、ＵＮＴ、ＱＮＴ、ＶＤＮＴは、それぞれ２次元配列として構成することができ、隣接マトリックスを与えるグラフテーブルの部分行列を構成する。

【００４６】

上述のようにして規定されたマトリックスの行および列の要素数は、等しく、この結果、隣接グラフは、正方行列を構成し、その要素は、各ノード間に定義される重み付け値として生成される。以下に説明するステップＳ５０４～ステップＳ５０８は、重み付け値としての要素値を計算してグラフテーブルに登録する処理である。

【００４７】

ステップＳ５０４では、ユーザ間のリンクの定義づけに従い、当該リンクの重み付け値を計算し、これをUser_link_weight(i, j)としてグラフテーブルに登録する。ステップＳ５０５では、クエリー間のリンクの定義付けに従い、当該リンクの重み付け値を計算し、これをQuery_link_weight(i, j)として、グラフテーブルに登録する。ステップＳ５０６では、ユーザ－クエリー間のリンクの定義付けに従い、当該リンクの重み付け値を計算し、これをUser_query_link_weight(i, j)としてグラフテーブルに登録する。また、ステップＳ５０７では、ユーザ－閲覧済文書間に定義されたリンクに従い、当該リンクの重み付け値User_vd_link_weight(i, j)を計算し、グラフテーブルに登録する。さらにステップＳ５０８では、クエリー－閲覧済文書間に定義されたリンクに従い、当該リンクの重み付け値Query_vd_link_weight(i, j)を計算し、グラフテーブルに登録する。

【００４８】

ステップＳ５０９では、ユーザ、クエリー、閲覧済文書を行ノードおよび列ノードとするグラフテーブルとして隣接グラフデータを確定し、ステップＳ５１０で処理を終了する。なお、図４で説明した各ノードの追加があった場合、追加するべき、ユーザ、クエリー、閲覧済文書の各属性の末尾に新規行および新規列を追加し、追加するべきノードについての各重み付け値を登録することにより、ノードの追加に対応付けて隣接グラフを更新する。

【００４９】

図６には、図５の定義済み処理であるステップＳ５０４～ステップＳ５０８を実行するための疑似コードを、例示的にプログラミング言語としてＣ＋＋を使用してコーディングした場合の実施形態を示す。図５の各枠線内の疑似コードが、図５のステップＳ５０４～ステップＳ５０８の処理を実行するための疑似コードに対応する。なお各重み付け値を計算する上で、本実施形態では以下の基準を使用して不正規ノードの登録を排除することで、検索精度を向上させている。

【００５０】

＜閲覧済文書についての不正規インスタンスの登録排除＞

閲覧済文書について不正規インスタンスとして排除する事例を以下に、例示的に列挙する。

（１）同一のユーザからの短時間で発生した多量の検索要求。

（２）同一のユーザからの短時間で発生した多量の検索要求。

（３）長すぎる検索クエリーを含む検索要求または不正な検索要求を含む検索要求。

【００５１】

上述した事例（１）および（２）は、いわゆるスパマーからのアタックを排除する目的であり、事例（３）は、検索クエリーの冗長化による処理効率の低下防止および不正要求に関連するデータが最関連検索語の推定に影響を及ぼさないようにするためである。

＜検索クエリーおよび文書についての不正規・冗長インスタンスの登録排除＞

検索クエリーに関連して不正規インスタンスの登録を排除する基準例を、以下に例示的に列挙する。

（１）同一ユーザによる複数の検索クエリーに対しては、同一の検索語を含む場合にでも異なる検索クエリーIdが割り当てられるが、グラフ生成においては、当該重複登録された検索クエリーのうち、最新のタイムスタンプを有するインスタンスをノードとして採用する。

（２）文書の閲覧を行わなかった検索クエリーについては、グラフ生成から排除する。

（３）設定したしきい値よりも閲覧頻度の低い閲覧済文書はグラフ生成から排除する。

（４）オプション構成として、閲覧頻度が低下する傾向にある文書についてグラフ生成から排除する。

【００５２】

以下、各ノード間について定義される数学的なリンク定義式を説明する。対となる端点ノードｉ、ｊが決定されると、対応する各ノード属性を使用して、リンクについての重み付け値が計算される。この重み付け値が隣接グラフ、すなわち、グラフテーブルの行列要素として登録される。以下、各リンクについての定義式を説明する。

【００５３】

＜$link(q_i \rightarrow vd_j)$＞

$link(\chi_i \rightarrow \chi_j)$（χ＝u、q、vd）は、ノードχi、ノードχjを各端点とするリンクを意味し、以下、ui、uj、vdi、vdjについて同様の表記を採用する。$link(q_i \rightarrow vd_j)$は、閲覧済文書が含む検索語または検索語のセマンティック上での類似性に基づいて与えられる重みである。例えば、閲覧文書が検索クエリーの検索語を含む場合には、Query_vd_link_weight(i, j)=1であり、それ以外の場合には、Query_vd_link_weight(i, j)=0である。ない、セマンティック類似性を利用する場合、文書検索の際に得られた相対類似度（完全に類似する場合に値＝１）の値を重み付け値として与える。

【００５４】

＜$link(q_i \rightarrow u_j)$＞

$link(q_i \rightarrow u_j)$は、検索クエリーの作製者が対象としているユーザか否かの２値判断で割り当てられ、検索クエリーq_iが判断中のユーザにより発行されたものである場合、User_query_link_weight(i, j)=1とされ、それ以外の場合は、User_query_link_weight(i, j)=0が与えられる。

【００５５】

＜$link(q_i \rightarrow q_j)$＞

$link(q_i \rightarrow q_j)$は、検索クエリー間の時系列的関係を含む類似性の重みであり、図７で与えられる関数で定義される。図７は、関数$span(q_i, q_j)$の例示的な実施形態の関数を示した図である。図７に示すように、関数$span(q_i, q_j)$は、対象とされる検索クエリーq_iとq_jとの間に発行された検索クエリーの数であるｋに応じて、単調減少する関数ｆ（ｋ）で与えられる。なお、λは、λ＞－１を満たす実数である。

【００５６】

図７に示されるように$link(q_i \rightarrow q_j)$は、ｉ＝ｊで、０とされ、ｊ＝ｉ±１で、検索クエリーの類似性を与える$Sim(q_i, q_j)$の値と、任意に設定される値｜Ｑ｜の逆数との和に、ｆ（０）の値を乗じて与えられるように定義されている。なお、ｆ（０）は、q_i、q_jが、時系列的に隣接する検索クエリーであることに対応する値であり、ｆ（０）＝（１＋λ）で与えられ、値｜Ｑ｜は、確率的にみて隣接する検索クエリーが類似する程度に対応して設定される値である。例えば、値｜Ｑ｜は、ノードq_iが関連するＱ個のノードを有して

いる場合、検索クエリーの内容を考慮しなければ、当該ノードq_iを受領した後、関連する検索クエリーを受領する確率は、単純計算で、１／Ｑとなる。検索クエリーの内容的な関連性を導入するため、後述する図８で詳細に説明する$Sim(q_i,q_j)$を導入して検索クエリーq_iを受領した後、検索クエリーq_jを受領する確率を調整する。さらに、値＝１／｜Ｑ｜は、ノードq_iとノードq_jとの間の関連づけが０とならないようにするため、適宜設定することができる。

【００５７】

さらに$link(q_i \rightarrow q_j)$は、ｊ＝ｉ、ｊ＝ｉ±１以外については、$span(q_i, q_j)$の大きさおよび検索クエリー間の類似性に関連し、$span(q_i, q_j)$の値が大きくなればなるほど、クエリー間の類似性によりその重みが与えられるように定義される。なお、図８には、関数$Sim(q_i, q_j)$を与える関数の実施形態を例示する。検索クエリー間の類似性は、（ａ）検索の結果抽出された文書のうち、類似性が上位Ｋ番目までの文書Idなどの共通性に基づいて検索クエリーの類似性尺度を与える、検索履歴類似度、（ｂ）各検索クエリーに関連して閲覧された閲覧済文書の類似性を使用する閲覧履歴類似度、（ｃ）検索クエリーq_i、q_jが含む検索語Ｗｓの全種類に対する共通する検索語Ｗｓの割合を使用する内容類似度、および（ｄ）検索類似度、閲覧類似度、内容類似度を適切な定数α（０＜α＜１）を使用して複合化させた複合的類似度として定義できる。

【００５８】

なお、FeedSimの関数としては、検索履歴類似度でも閲覧履歴類似度でも利用することができる。さらに、他の類似尺度を使用することも、検索クエリー間の類似性を与える限り、いかなる関数形式で与えることができる。

【００５９】

＜$link(u_i \rightarrow u_j)$＞

ユーザ間に定義するリンクは、検索クエリーを基準尺度として使用する場合、ユーザが発行した検索クエリーの類似性を重み付け尺度として与えることができ、本実施形態では、検索クエリーを、検索クエリーが含む検索語ベクトルとし、検索クエリーq_iと検索クエリーq_jとの内積として与えることができる。また、ユーザ間の関係は、外部要因を類似性の尺度として使用することもでき、例えば、ＲＳＳなどを介してブックマーク情報にアクセスできる場合には、ユーザ間に共通するブックマーク情報の存在を使用して類似性尺度を計算することもできる。

【００６０】

＜$link(u_i \rightarrow vd_j)$＞

$link(u_i \rightarrow vd_j)$は、ユーザが閲覧した文書について、ユーザと文書間に定義される重み付け値であり、＜$link(u_i \rightarrow vd_j)$は、特定のユーザ$u_i$が閲覧済文書$vd_j$を閲覧した場合には、User_vd_link_weight(i, j)=1として設定し、それ以外の場合には、User_vd_link_weight(i, j)=0を与える。

【００６１】

以上のリンク定義付けを使用して隣接グラフの各ノードについて重み付け値を計算し、グラフテーブルの（ｉ，ｊ）座標値に対応させて重み付け値$W_{i,j}$を登録することで、グラフテーブル、すなわち隣接グラフが完成する。

【００６２】

図９は、本実施形態で生成される隣接グラフ９００の実施形態を示す。図９に示すように隣接グラフ９００は、複数のサブブロックに分割できる。各ブロックは、ノード属性の順序に対応して、図９に示す実施形態では、それぞれ、$link(q_i \rightarrow q_j)$を与え、ブロック９１０と、$link(u_i \rightarrow q_j)$を与え、ブロック９２０と、$link(u_i \rightarrow vd_j)$を与えるブロック９３０と、$link(u_i \rightarrow u_j)$を与えるブロック９４０とされる。なお、対角ブロックを挟んで、（ｉ，ｊ）→（ｊ，ｉ）で与えられる各ブロックの値は、ブロック９１０～ブロック９３０の値と同一である。

【００６３】

また、各要素の値は、０の値が多く存在し、隣接グラフ９００は、このため疎な行列を

構成し、ＣＣＳ(Compressed Column Storage)などの手法を使用してデータ圧縮が可能となる。なお、図９に示した各ノードの配列は、例示的なものであり、行および列のシーケンスが一致していれば、特に制限はない。なお、ブロック９５０は、０行列であり、本実施形態で閲覧済文書間にリンクを生成する必要がないことから、重みを割り当てていないことに対応する。なお、閲覧済み文書間にページ関係など、何らかの重み付けを導入する場合には、ブロック９５０に有為な値を設定することもできる。

【００６４】

本実施形態の最関連検索語決定部１２０は、検索クエリーに対する最関連の過去に登録された検索語を検索する場合に、ランダムウォークを使用して要素のランク付けを、下記式（１）を使用して実行する。

【００６５】

【数１】

$$^{vect}U' = c \times A \times {}^{vect}U + (1-c) \times {}^{vect}V \qquad (1)$$

上記式（１）中、$^{vect}U'$は、ランキングベクトル、ｃは、正の定数、Ａは、隣接グラフに対応する隣接マトリックス、^{vect}Uは、試行ベクトル、^{vect}Vは、リスタートベクトルである。

【００６６】

図１０は、本実施形態の最関連検索語決定部１２０が実行する処理ポリシー１０００を示した図である。図１０には、上記式（１）の関係を、行列要素およびベクトル要素を使用して概略的に示した。隣接マトリックス１０１０は、図９で説明したように、複数のブロックマトリックスを含んで構成され、また０ブロックマトリックスを含んでいる。本実施形態のランキングベクトルは、ランキングベクトルを試行ベクトル^{vect}Uとしてランダムウォークによる反復計算により、^{vect}Uと$^{vect}U'$との距離、すなわち、内積が収束した場合に、^{vect}Uまたは$^{vect}U'$の要素値の大きさを使用して、抽出するべき検索語または検索語列を決定するために利用される。

【００６７】

最関連検索語決定処理は、まず、上記式（１）でＲＷＲ式を定義し、リスタートベクトル$^{vect}V_q$を初期化することから開始する。リスタートベクトル$^{vect}V_q$の実施形態を、$^{vect}V_q$として図１０示す。リスタートベクトル$^{vect}V_q$は、隣接マトリックスに登録されるノード属性のうち、ユーザから検索要求を受領した場合、受領した検索要求が含む検索クエリーを、要素の影響を受けるブロックにつき、クエリーノードを登録する。例えば、修正するべきブロックの最後行および最後列に追加し、対応するマトリックスの列位置ｊに、整数値＝１を設定して初期化される。

【００６８】

なお、ユーザ、閲覧済文書などに関連して行および列を追加する場合、ユーザ、閲覧済み文書に関連する行列要素値の他、クエリーノードについても、類似度を判断して列要素および行要素を取得し、追加することができる。この実施形態では、新に追加されたユーザ、閲覧済文書に関連しても、検索クエリーを類似修正することができるので、より類似検索性を改善させることができる。

【００６９】

影響を受けるブロックは、図９の実施形態では、ブロック９２０、ブロック９３０である。この場合、新に受領した検索クエリーq_{new}を使用して、ＳＱＬ文に含ませてSydney full text searchを実行し、その結果を取得してグラフテーブルに追加するべき値を計算し登録する。登録するべき重み付けの値は、説明する実施形態では、閲覧順位がトップ１０００以内にランキングされていなければ、重み付け値＝０として設定する。この実施形態において使用することができるＳＱＬ文の疑似コードを下記式（２）に示す。

【００７０】
【数２】

```
****************************************
Select v.queryid,v.userid,t.s from (select c_id,c_source,score(c_text) s from catalog where c_text
contains freetext(new query terms') order by score(c_text) desc limit 1000) t,visithistory v where
t.c_id=v.docid and t.c_source=v.docsource limit 1000;
****************************************
```

【００７１】
そして、ＲＷＲ反復計算を開始する時点で、試行ベクトルを、$^{vect}U_q = {}^{vect}V_q$として初期設定する。その後、上記式（１）の計算を実行して更新ランキングベクトル$^{vect}U'$を計算する。その後、$^{vect}U'$を、^{vect}Uの値に設定してさらに$^{vect}U'$の値を更新し、最終的に$^{vect}U'$と、^{vect}Uとの間の距離が収束した段階で、最後の$^{vect}U'$を定常状態ベクトルとして確定する。上記処理は、span(q_i, q_j)を考慮して、最もspanの値が離れたクエリー間での類似度の高さを反復して計算させることに対応し、収束に成功した場合、追加するべき最関連検索語を指定することが適切なためである。
【００７２】
図１１には、本実施形態の最関連検索語決定部１２０が実行する処理のフローチャートを示す。図１１の処理は、ステップＳ１１００から開始し、ステップＳ１１０１で、新たな検索要求を受領する。ステップＳ１１０２では、検索要求内の検索語または検索語列を含ませてグラフデータおよび隣接マトリックスを再構築する。ステップＳ１１０３では、リスタートベクトルを初期化し、ステップＳ１１０４で、ランダムウォークによりランキングベクトルを決定する。
【００７３】
ステップＳ１１０５では、ランキングベクトルの要素を、クエリーノード、ユーザノード、閲覧済文書ノードのタイプごとにソーティングし、ステップＳ１１０６で、それぞれ値がトップの要素に対応する最関連検索語を選択し、それぞれq_0、u_0、vd_0として決定する。その後、ステップＳ１１０７で、検索クエリーとして、$\{q_{new}+q_0\}$、$\{q_{new}+vd_0\}$、$\{q_{new}+u_0\}$として検索クエリーを拡張し、検索エンジンの並列検索を実行する拡張検索クエリーを発行する。その後、ステップＳ１１０８で最関連検索語決定部１２０の処理を終了させる。
【００７４】
図１２は、図１１のステップ１１０４の定義済み処理を、疑似コードとして示す。図１２に示すように、まず、第１行目で、リスタートベクトルを初期化し、第４行目で、上記式（１）にしたがって、$^{vect}U_q'$の値を更新する。第５行目では、$^{vect}U'$、と^{vect}Uとの間の距離を計算し、距離が設定した少値ε以下か否かの判断を使用して、ＲＷＲ計算の収束を判定する。なお、小値εの値は、収束性および精度を考慮して適宜設定でき、コンピュータのアンダーフロー以上の値であれば適宜設定することができる。
【００７５】
収束した場合、第７行目で、収束結果を示す変数IterResult=trueに設定し、さらに第９行目では、反復回数が設定回数を超えたか否かを判断し、超えた場合、収束結果を示す変数IterResult=faultを設定し、処理を検索実行部１２２に渡し、以後の検索を実行する。そして、収束せずまた反復回数を超えていない場合、第１６行目で、$^{vect}U_q = {}^{vect}U_q'$に設定し、収束したか、または反復回数が設定した上限値を超えるまで反復計算を実行させる。
【００７６】
関連する検索語を抽出処理で、例えば、特異値分解を使用して特異値の大きさに応じて最関連検索語を決定することもできる。しかしながら、特異値分解を陽に使用して最関連検索語を抽出処理は、Ｏ（M^2）（Ｍは、隣接グラフのノード数）に対応する計算量を要

し、隣接マトリックスの要素数の増大に対応して計算量が非線形に増加するので、計算効率は、充分とはいえない。しかしながら、本発明のＲＷＲを使用した最関連検索語抽出は、Ｏ（Ｎ×Ｋ）（Ｎは、隣接マトリックスの非ゼロ要素数であり、Ｋは、反復回数である。）程度の計算量増加で済み、より効率的で、検索要求を受領した時点でオンザフライの拡張検索を可能とする。

【００７７】

本実施形態の上記機能は、Ｃ＋＋、Ｊａｖａ（登録商標）、Ｊａｖａ（登録商標）Ｂｅａｎｓ、Ｊａｖａ（登録商標）Ａｐｐｌｅｔ、Ｊａｖａ（登録商標）Ｓｃｒｉｐｔ、Ｐｅｒｌ、Ｒｕｂｙなどのオブジェクト指向プログラミング言語などで記述された装置実行可能なプログラムにより実現でき、本実施形態のプログラムは、ハードディスク装置、ＣＤ－ＲＯＭ、ＭＯ、フレキシブルディスク、ＥＥＰＲＯＭ、ＥＰＲＯＭなどの装置可読な記録媒体に格納して頒布することができ、また他装置が可能な形式でネットワークを介して伝送することができる。

【００７８】

これまで本実施形態につき説明してきたが、本発明は、上述した実施形態に限定されるものではなく、他の実施形態、追加、変更、削除など、当業者が想到することができる範囲内で変更することができ、いずれの態様においても本発明の作用・効果を奏する限り、本発明の範囲に含まれるものである。

【図面の簡単な説明】

【００７９】

【図１】本実施形態の情報検索装置１００の機能ブロックを示した図。

【図２】本実施形態の履歴ＤＢ１３０およびグラフＤＢ１３２が管理する、各テーブルのデータ構造を示した図。

【図３】本実施形態の情報検索方法の処理についての概略的なフローチャートを示した図。

【図４】本実施形態で、隣接グラフを構成するために使用する、図２に示したデータ構造２５０の更新処理のフローチャート。

【図５】データ構造２５０を使用して、隣接グラフの要素値を決定する処理のフローチャート。

【図６】図５の定義済み処理であるステップＳ５０４～ステップＳ５０８を実行するための疑似コードを、例示的にプログラミング言語としてＣ＋＋を使用してコーディングした場合の実施形態を示した図。

【図７】関数span(q_i, q_j)の例示的な実施形態の関数を示した図。

【図８】関数Sim(q_i, q_j)を与える関数の実施形態を例示した図。

【図９】本実施形態で生成される隣接グラフ９００の実施形態を示した図。

【図１０】本実施形態の最関連検索語決定部１２０が実行する処理ポリシー１０００を示した図。

【図１１】本実施形態の最関連検索語決定部１２０が実行する処理のフローチャート。

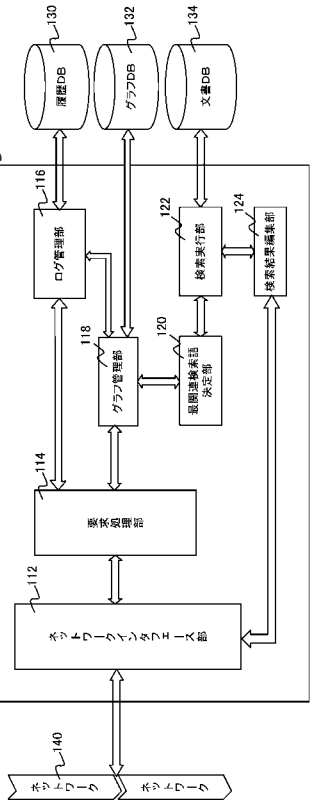
【図１２】図１１のステップ１１０４の定義済み処理を、疑似コードとして示した図。

【符号の説明】

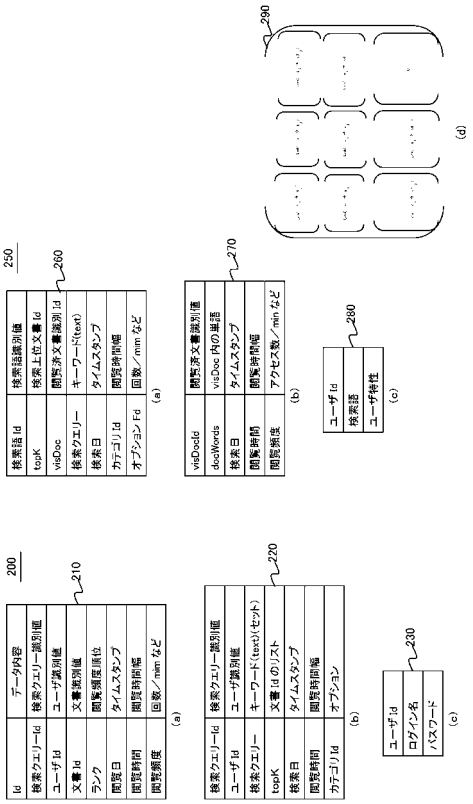
【００８０】

１００…情報検索装置、１１２…ネットワークインタフェース部、１１４…要求処理部、１１６…ログ管理部、１１８…グラフ管理部、１２０…最関連検索語決定部、１２２…検索実行部、１２４…検索結果編集部、１３０…履歴ＤＢ、１３２…グラフＤＢ、１３４…文書ＤＢ、１４０…ネットワーク

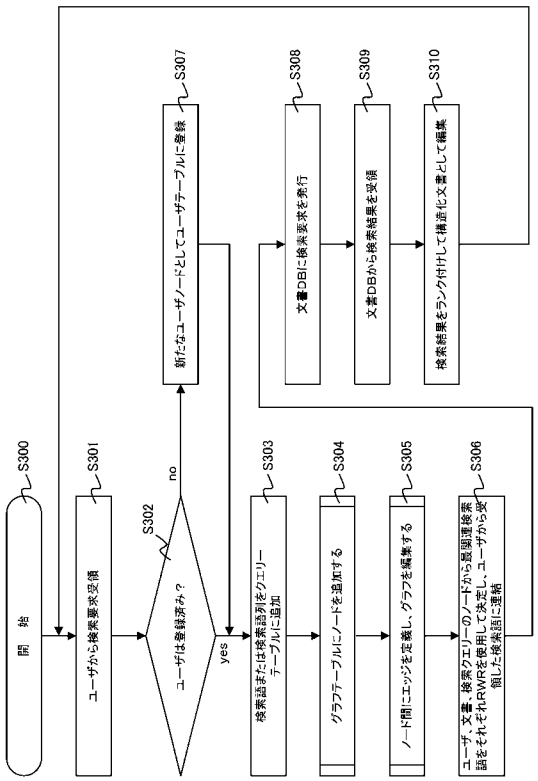
【図１】


100
110
履歴DB
130
グラフDB
132
文書DB
134
ログ管理部
116
検索実行部
122
検索結果編集部
124
グラフ管理部
118
最短連鎖語句決定部
120
要求処理部
114
ネットワークインタフェース部
112
ネットワーク
140
ネットワーク

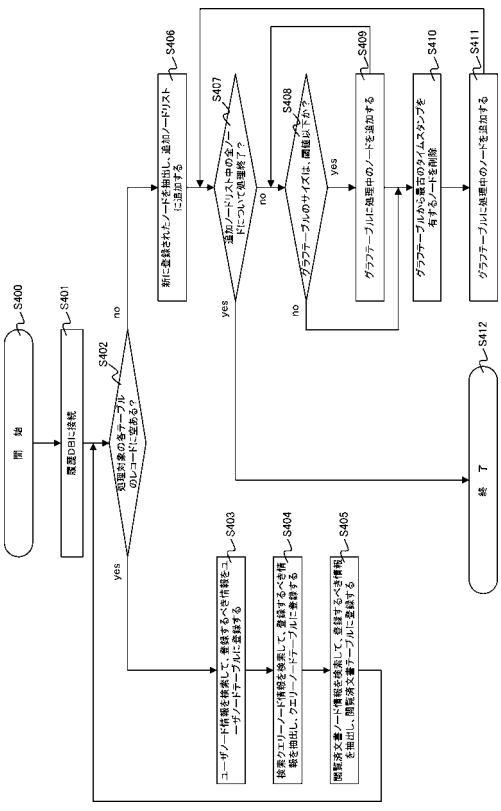
【図２】


200
Id
データ内容
検索クエリーId
検索クエリー識別値
ユーザId
ユーザ識別値
文書Id
文書識別値
ランク
閲覧頻度順位
閲覧日
タイムスタンプ
閲覧時間
閲覧時間幅
閲覧頻度
回数／min など
(a)
210
検索クエリーId
検索クエリー識別値
ユーザId
ユーザ識別値
検索クエリー
キーワード(text)のセット
topK
文書 Id のリスト
検索日
タイムスタンプ
閲覧時間
閲覧時間幅
カテゴリId
オプション
(b)
220
ユーザId
ログイン名
パスワード
(c)
230
250
検索語Id
検索語識別値
topK
検索上位文書 Id
visDoc
閲覧済文書識別 Id
検索クエリー
キーワード(text)
検索日
タイムスタンプ
カテゴリId
閲覧時間帯
オプション Fd
回数／min など
(a)
260
visDocId
閲覧済文書識別値
docWords
visDoc 内の単語
検索日
タイムスタンプ
閲覧時間
閲覧時間帯
閲覧頻度
アクセス数／min など
(b)
270
ユーザId
検索語
ユーザ特性
(c)
280
290
(d)

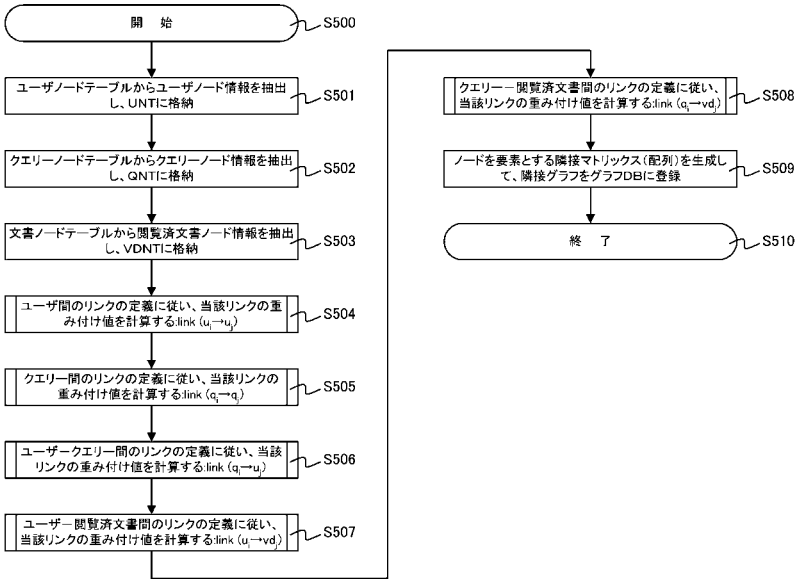
【図３】


開 始
S300
ユーザから検索要求受領
S301
ユーザは登録済み？
S302
no
yes
新たなユーザノードとしてユーザテーブルに登録
S307
検索語または検索語列をクエリーテーブルに追加
S303
グラフテーブルにノードを追加する
S304
ノード間にエッジを定義し、グラフを編集する
S305
ユーザ、文書、検索クエリーのノードから最短連鎖検索語をそれぞれRWRを使用して決定し、ユーザから最も近い検索語に連結
S306
文書DBに検索要求を発行
S308
文書DBから検索結果を受領
S309
検索結果をランク付けして構造化文書として編集
S310

【図４】


開 始
S400
履歴DBに接続
S401
処理対象のグラフテーブルのレコードに変わる？
S402
no
yes
新に登録されたノードを抽出し、追加ノードリストに追加する
S406
追加ノードリスト中の全ノードについて処理終了？
S407
no
yes
グラフテーブルのサイズは、閾値以下か？
S408
no
yes
グラフテーブルに処理中のノードを追加する
S409
グラフテーブルから最古のタイムスタンプを有するノードを削除
S410
グラフテーブルに処理中のノードを追加する
S411
終 了
S412
ユーザ情報を検索して、登録するべき情報をユーザノードテーブルに登録する
S403
検索クエリーノード情報を検索して、登録するべき情報を抽出し、クエリーノードテーブルに登録する
S404
閲覧済文書ノード情報を検索して、登録するべき情報を抽出し、閲覧済文書テーブルに登録する
S405

【図５】



【図６】

600

```
user_user_link ()
{
        FOR (i=0; i<UNT.size; i++)
                {  FOR (j=i; j<UNT.size; j++)
                        {
                                User_link_weight ( );
                        }
                }
}
return;                                                    S504
```

```
user_VD_link ()
{
        FOR (i=0; i<UNT.size; i++)
                {  FOR (j=i; j<VNT.size; j++)
                        {
                                User_vd_link_weight ( );
                        }
                }
}
return;                                                    S507
```

```
query_query_link ()
{
        FOR (i=0; i<QNT.size; i++)
                {  FOR (j=i; j<QNT.size; j++)
                        {
                                Query_link_weight ( );
                        }
                }
}
return;                                                    S505
```

```
query_VD_link ()
{
        FOR (i=0; i<QNT.size; i++)
                {  FOR (j=i; j<VNT.size; j++)
                        {
                                Query_vd_link_weight ( );
                        }
                }
}
return;                                                    S508
```

```
user_query_link ()
{
        FOR (i=0; i<UNT.size; i++)
                {  FOR (j=i; j<QNT.size; j++)
                        {
                                User_query_link_weight ( );
                        }
                }
}
return;                                                    S506
```

【図７】

700

Query_link_weight ()

{

$span(q_i, q_j)$: *number of query nodes between* q_i *and* q_j

$f(k) = 1 + \lambda e^{-k}$

$$link(q_i \rightarrow q_j) = \begin{cases} 0, & j=i \\ f(0) \times \left(Sim(q_i, q_j) + \frac{1}{|Q|} \right), & j=i \pm 1 \\ f(span(q_i, q_j)) \times \left(Sim(q_i, q_j) + \frac{1}{|Q|} \right), & \text{others} \end{cases}$$

}

【図８】

800

Sim () :

(a)検索履歴類似度⇒ $$ObjFeedSim(q_i, q_j) = \frac{topK(q_i) \cap topK(q_j)}{topK(q_i) \cup topK(q_j)}$$

(b)閲覧履歴類似度⇒ $$SubFeedSim(q_i, q_j) = \frac{visDocs(q_i) \cap visDocs(q_j)}{visDocs(q_i) \cup visDocs(q_j)}$$

q_i, q_j : a set of words;

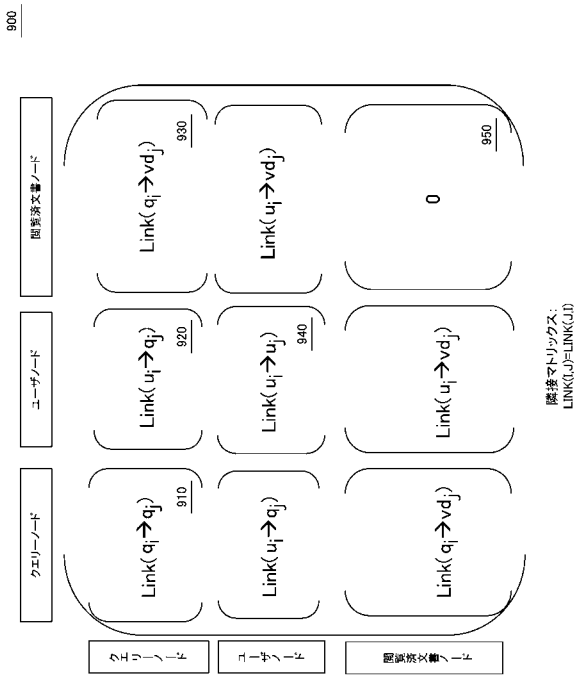
(c)内容類似度 ⇒ $$ConstSim(\vec{q}_i, \vec{q}_j) = \frac{\sum_{w_s \in q_i \cap q_j} w_s}{Max\left(\sum_{w_s \in q_i} w_s, \sum_{w_s \in q_j} w_s \right)}$$

w_s : the term weight which is measured by qtf.idf

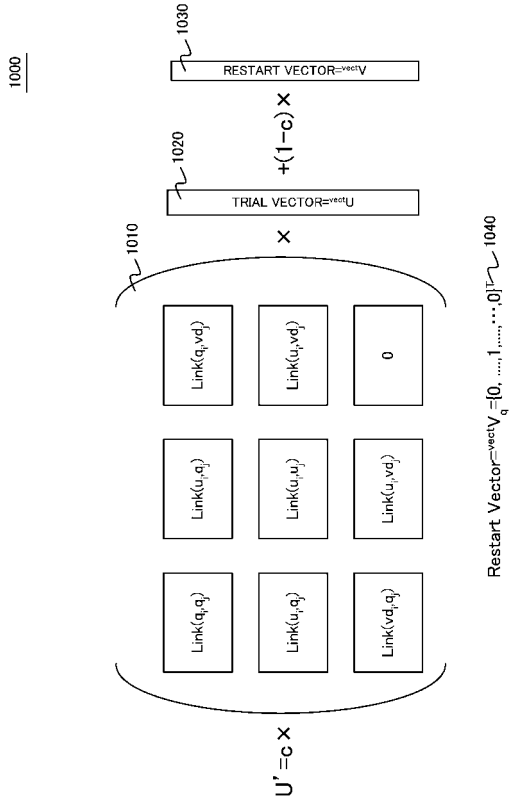
(d)複合的類似度⇒ $sim(q_i, q_j) = \alpha \times ConstSim(\vec{q}_i, \vec{q}_j) + (1-\alpha) \times FeedSim(\vec{q}_i, \vec{q}_j)$

FeedSim could be either SubFeedSim or ObjFeedSim

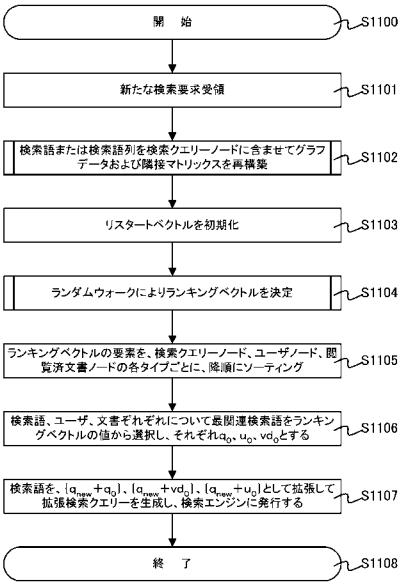
【図９】



【図１０】



【図１１】



【図１２】

1200

```
RWR_rank ()

    {
        Uq=Vq=[0, ....,1,....,・・・,0]T;
                while (true)
                    {
                    Uq'=c×A・Uq+(1−c)Vq;
                    if (dist (Uq', Uq)≦ε )
                                {
                                  IterResult=true;
                                }
                    if (iterStep>maxSteps )
                                {
                                  IterResult=fault;
                                  break;
                                {
                    else
                                {
                                  Uq=Uq';
                                }
                    }

    }
return;
```