文書と保存ディレクトリ構造からの個人的な概念・語彙階層木の生成とWeb検索の個別化への利用

大島　裕明† 　小山　聡† 　田中　克己†

† 京都大学大学院情報学研究科社会情報学専攻
〒606–8501 京都市左京区吉田本町
E-mail: †{ohshima,oyama,tanaka}@dl.kuis.kyoto-u.ac.jp

あらまし　個人の知識や，もののとらえ方，考え方は，個人ごとに異なるものである．個人のローカルコンピュータには様々な文書が存在し，人が見た場合その人の知識などを推測することが可能である．しかし，そのような個人ごとの知識や考え方は，コンピュータが利用できるような状態になっていない．本研究では，ディレクトリ構造に分類された文書群を基にして特徴ベクトルによって概念を表現した概念階層木というものを作成する．さらに，概念階層木の概念の一つに着目したときにある語にとって他の語がどのような関係であるかということを，その概念における語彙の偏在性を基に評価し，語彙どうしの関係性を表した語彙階層木というものを作成する．そして，それらを利用したウェブのパーソナライゼーションとして，クエリ拡張と検索結果の再ランキングについて提案を行う．

キーワード　パーソナライゼーション，知識発見，情報検索，知識処理

Creating Personal Concept/Term Tree based on Documents and Directory Structure and Applying for Web Search Personalization

Hiroaki OHSHIMA†, Satoshi OYAMA†, and Katsumi TANAKA†

† Department of Social Informatics, Graduate School of Informatics, Kyoto University
Yoshidahonmachi, Sakyo-ku, Kyoto, 606–8501 Japan
E-mail: †{ohshima,oyama,tanaka}@dl.kuis.kyoto-u.ac.jp

Abstract　Each person has different knowledge, ideology, and concept. Nowadays, there are much personal informatioh in PC. Althogh it is easy for people to guess how the user of the PC is, computer can not estimate it. This paper proposes a method to represent personal knowledge and concept based on documents classified in directory structure. They are called “Personal Concept Tree” and “Personal Term Tree”. “Personal Concept Tree” indicates concept classification in which each concept is represented by a feature vector. “Personal Term Tree” indicates relationship between terms, and it is created from “Personal Concept Tree” based on deviation of term appearance. Furthermore, thie paper proposes Web search personalization; query keyword expansion and re-ranking of search result using the created trees.

Key words　Personalization, Knowledge Discovery, Information Retrieval, Knowledge Processing

1. はじめに

現在，個人のローカルコンピュータ上には，様々な個人的な情報が存在する．電子メールを始めとして，ローカルに保存された様々な文書，ウェブの閲覧履歴などは，そのコンピュータの利用者がどのような知識を持っているか，どのような考え方をしているか，ということを推測するために十分な質と量をもっているといえる．しかし，例えば，ウェブの検索エンジンを用いる際に，そのような情報を生かすことはできない．つまり，あるキーワードにたいして，その人がどのような考え方をもっているのかということ，その人がどのような情報については既に知っているかということ，といったことを検索エンジンに伝えることはできないのである．

本研究の目的は，そのような個人の知識や考え方を，コン

ピュータに利用可能な形に表現し，それを利用して様々な分野におけるパーソナライゼーションを実現することである．今回，我々が個人的な情報として利用したものは，ローカルコンピュータ上において，ディレクトリのツリー構造上に分類して保存された様々な文書群である．ディレクトリ構造は，その人にとっての物事の分類体系，概念体系といったものが反映されていると考えられる．

本研究では，そのような情報を利用して，個人のおおまかな概念の構造を表現した概念階層木というものを作成した．さらに，概念階層木の概念の一つに着目したときにある語にとって他の語がどのような関係であるかという，語彙どうしの関係性を表した語彙階層木というものを作成した．

以下，2. 章では関連研究について，3. 章では概念階層木と語彙階層木とその作成アルゴリズムについて，4. 章では，試作システムによる実例と，それらを用いたウェブ検索におけるパーソナライゼーションについて，5. 章でまとめと今後の課題について，それぞれ述べる．

2. 関 連 研 究

すでにいくつかの研究において，個人的な情報からの知識を生成するような研究が行われている．それらについて述べる．

Haystack [1], [2] は MIT が開発した，個人的な情報管理システムである．扱う情報は，e-mail やカレンダー，文書，Web ページなど多岐にわたり，それらを RDF で管理する．本研究では，Haystack と同様に，さまざまな個人的な情報を管理するが，蓄積された情報そのものをより効果的に利用しようとする Haystack とは異なり，蓄積された情報から新たに知識を作り出して利用することを目的としている．

WorkWare++ [3], [4] は富士通研究所が開発した，会社などのグループで用いられるビジネス文書の蓄積と再利用のための情報管理システムである．さまざまな文書が登録され，その登録時には時間などのメタ情報が自動的に付加される．また，人やイベントの情報も同時に管理されている．ユーザは蓄積されたメタ情報を元に，ある研究分野に関してどのような技術が蓄積されているかや，ある事柄を知っていそうな人が誰であるかなどの情報を取得可能である．本研究では，文書群から知識を抽出するという点では同じだが，より汎用的に利用できるような知識を得ることを目的としており，WorkWare++で行っている，グループによる情報共有や，蓄積された知識の獲得とは異なる．

Hyperclip [5] は NTT が開発した，知識流通プラットフォームである．ユーザが利用した複数のコンテンツの間の関係を表現することができ，そこで作成された RDF をピア・ツー・ピアネットワークで共有することによって，ある文書と関連する文書を検索することができるようになる．Hyperclip で検索できる文書はピア・ツー・ピアネットワーク上の誰かによってメタ情報が付加されたものである．本研究では，文書群から知識を抽出してそれを利用することを目的としており，ユーザが RDF という利用可能な形の知識を作成する Hyperclip とは異なる．

湯川ら [6] は，個人が所有する文書に出現する単語の隣接度合いから，それぞれの単語同士の関連度合いを表す概念ベース，パーソナル・リポジトリを個人ごとに作成した．ユーザがコミュニティーのピア・ツー・ピア型システムの他の人が保有する情報を検索するときには，エージェントが検索キーをパーソナル・リポジトリによって拡張し，他人のパーソナルリポジトリ内でどのような情報が検索結果として適当であるかを判断することが可能になる．本研究と同様に，個人が所有する文書をもとに個人の概念を表しているが，その目的がピア・ツー・ピア型のネットワークで文書を共有することであり，より利用範囲の広い知識を作成することを目的とする本研究とは異なる．

これらの研究は，ある特定の環境やコミュニティーの中で利用可能な知識を作成しようとしている点において，我々のものとは異なっている．

3. 概念階層木と語彙階層木の生成

3.1 概　　要

本研究で，個人の知識や考え方を表現するものとして作成したものは 2 つあり，それぞれを

- 概念階層木
- 語彙階層木

と名づけた．

概念階層木とは，個人の概念体系をおおまかに表したもので，概念をノードとするツリー構造である．概念はタームの出現頻度を基にした特徴ベクトルによって表現される．

語彙階層木とは，概念階層木の概念の一つに着目したときに，その概念における語彙どうしの関係性を表したものである．つまり，ある語に対して他の語が，より上位の概念を説明するのに用いられる語であるか，より下位の概念を説明するのに用いられる語であるか，また，概念を説明する際に共起的に用いられる語であるか，兄弟概念を説明するのに用いられるような排他的関連性を持った語であるか，といったことを評価する．

以下，本節では，概念階層木と語彙階層木の概要について，3. 1. 1 節で概念階層木の作成アルゴリズムについて，3. 1. 2 節で語彙階層木の作成アルゴリズムについて述べる．

3.1.1 概念階層木の概要

普通，概念体系とは，一般的な概念についての体系である．しかし，個人はその人ごとに異なる考え方をもっており，得意とする分野や分野ごとにおける知識量なども異なっている．つまり，その人にとっての概念体系というものは，人それぞれ異なっていると考えられる．本研究の目的は，そのような個人ごとにおける概念体系を表現することである．

一般的な概念体系は数十万という数の概念を扱い，各概念は見出しや説明文によって表されており，また，上位概念や下位概念が定義されている．このような概念体系は，多くの人と時間をかけて初めて作ることができるものであり，個人個人の概念をそれぞれの人が作る，といったことは不可能である．

しかし，個人が保有するローカルコンピュータ上には，

- 電子メール
- アドレス帳
- ウェブの閲覧履歴

- ウェブのブックマーク
- その他様々な文書

など，非常に多くの情報が存在しており，人間にとっては，その人がどのような考え方をしているか，ということを理解するために十分な質と量を備えている．

本研究では，その中でもディレクトリのツリー構造上に分類して保存された様々な文書群に着目した．当然，文書の分類のやり方は人それぞれであり，人によっては全く分類を行っていないという場合もあるが，ここでは，ユーザによってディレクトリのツリー階層構造上に文書が分類されている場合を想定する．そのようなディレクトリのツリー階層構造は，個人が持っていると考えられる概念体系における上位下位階層が非常に色濃く反映されたものであると考えることができる．

まず，ある 1 つのディレクトリに分類されている文書群について考えると，そこには何らかの共通性があると考えられる．例えば,「京都」という名前のディレクトリがあり，そこにいくつかの文書が分類されているとすると，その中には，その人にとっての「京都」を説明するような文書が多く存在すると考えられる．そこで，1 つのディレクトリは，その人にとっての何らかの概念 1 つを表しているととらえる．1 つのディレクトリが 1 つの概念を表しているとすると，その概念を表現する必要がある．普通の概念体系においては，概念の見出しや説明文が使われるが，ここでの概念を説明するものは，そのディレクトリに分類されている文書群である．ディレクトリの名前も概念の見出しとして使える可能性はあるが，必ずしも適切な名前がつけられているとは限らないため，ここでは無視する．よって，文書群から得られる情報で概念を説明するのだが，本研究では，タームの出現頻度を基にした特徴ベクトルを用いることとする．

1 つのディレクトリを 1 つの概念としてとらえるとすると，サブディレクトリは，親ディレクトリが表す概念の子概念と考えることができる．

ローカルコンピュータ上の文書群が分類されたディレクトリのツリー階層を以上のようにとらえ，1 つのディレクトリを 1 つの概念として考え，各概念を文書群におけるタームの出現頻度を基にして作成した特徴ベクトルによって表したものを，ここでは概念階層木と呼ぶ．つまり，概念階層木とは，ツリー構造の各ノードにおいて，概念を説明する特徴ベクトルを持ったものである．

概念階層木を作成する方法については，3.2 節で述べる．

概念階層木を作成することができれば，

- 新規取得文書の自動分類
- 概念をクエリとする Web 情報検索
- 語彙階層木の作成

といったことに利用可能であると考えられる．

本稿では，概念階層木を作成する主目的を，次に説明する語彙階層木を作成することしているため，他の利用については特に述べない．また，実装についても，語彙階層木を作成するための実装としており，上に列挙した他の目的については本稿では扱わない．

3.1.2 語彙階層木の概要

我々はこれまで，概念階層木を用いたウェブ検索結果の再ランキングによる個別化を行ってきたが [7]，そこでは，

（1） 入力されたキーワードからもっとも近いと考えられる概念を推定する

（2） 推定された概念を表現する特徴ベクトルと検索結果の文書の特徴ベクトルの類似度を基に検索結果のアイテムを評価する

という手法を用いていた．しかし，概念階層木に含まれる概念の数は，ディレクトリの数と等しく，せいぜい数十しかないのに対して，検索キーワードは無限に存在する．結果として，無理に検索キーワードに近い概念を推定しようとしており，検索キーワードと評価に用いる特徴ベクトルの関係が必ずしも強いとは言えなかった．そこで，今回は，様々な語彙に対して，個人がどのような考え方をしているか，ということを説明する，語彙階層木というものを提案する．

語彙階層木とは，理想的には一つの語彙をノードとした語彙の上位下位関係を木構造によって表したグラフ，一種のシソーラスである．しかし，そのような完全なシソーラスは自動的に作成することは不可能である．また，語彙は，様々な意味や観点において用いられることがあり，それは，その語彙が用いられるコンテキストに依存するものである．

そこで，本研究では，概念階層木におけるある概念に着目し，その概念を語彙が用いられるコンテキストしたときに，ある語と他の語がどのような関係を持っているかということを評価し、それによって理想的な木構造の概念階層木を近似的に表す。評価は直行する 2 つの軸，

- 上位的か下位的か
- 共起的か排他的関連か

で行う．

実際に，語彙階層木を作成する手法については，3.3 節で述べる．

3.2 概念階層木の作成

3.2.1 概念階層木作成の基本的アルゴリズム

概念階層木の作成方法は，目的や基になる文書群の構成によって様々な手法が考えられるため，まず基本的なアルゴリズムについて述べる．

はじめに，説明に用いる記号について定義する．

- 各ディレクトリを D_i とする．
- 概念階層木の D_i に相当する概念を表現する特徴ベクトルを C_i とし，語彙 $term$ に対するベクトルの値を $C_i(term)$ とする．
- D_i に直接分類されている文書の集合を $DirectDoc(D_i)$ とする．これには，サブディレクトリに分類されている文書は含まない．
- D_i 以下のサブディレクトリも含んだすべての文書の集合を $Doc(D_i)$ とする．
- D_i のすべてのサブディレクトリ集合を $SubDir(D_i)$ とする．これには，サブディレクトリのサブディレクトリもすべて含む．

- D_i と同じ親ディレクトリを持つ兄弟ディレクトリの集合を $SiblingDir(D_i)$ とする．

概念階層木作成の基本的アルゴリズムは，以下のような流れである．

（1） $DirectDoc(D_i)$ を基にして，特徴ベクトル $V_1(D_i)$ を作成する．

（2） $V_1(D_j)$（ただし，$D_j \in SubDir(D_i)$）を合成して特徴ベクトルを作成し，それにさらに $V_1(D_i)$ を合成して特徴ベクトル $V_2(D_i)$ を作成する．

（3） $V_2(D_k)$（ただし，$D_k \in SiblingDir(D_i)$）を基にして，兄弟ディレクトリとの違いを強調するため，$V_2(D_i)$ を修正する．

（4） さらに，後処理を行って作成された特徴ベクトルを，C_i とする．

概念階層木における概念と，もとのディレクトリ構造のディレクトリは，一対一に対応する．この時，概念を説明するものは，ディレクトリに分類されている文書群である．(1) ではまず，仮に，ディレクトリに直接分類されている文書群から特徴ベクトルを作成する．しかし，ディレクトリによっては，直接分類されている文書がなく，サブディレクトリがあり，その中に文書が分類されている，という場合があり，その場合はサブディレクトリ内の文書群を考慮せざるを得ない．そこで，(2) においてサブディレクトリの仮の特徴ベクトルと，自身の仮の特徴ベクトルをすべて何らかの手法によって合成する．この際の，特徴ベクトルを合成する手法によって，ある文書が親ディレクトリに分類されたか，子ディレクトリに分類されたか，ということに差異をつけること可能である．同様に，複数のサブディレクトリのどれに文書が分類されたか，ということについて差異を持たせるためのプロセスが (3) であり，兄弟ディレクトリにおいて作成される特徴ベクトルを用いて，あるディレクトリの特徴ベクトルを修正する．(4) は，後処理で，一般的な語を排除する，といったことをここで行うことができる．

3.2.2 概念階層木作成の実装

基本的アルゴリズムに従って行った実装について述べる．

まず，前小節で述べた，基本的アルゴリズムの流れにしたがって，簡単に説明する．

（1） 文書の DF 値を基にして，$V_1(D_i)$ を作成する．

（2） ディレクトリに直接分類されている文書も，サブディレクトリに含まれている文書も，同様に扱い，結果として，すべての文書からの DF 値を用いて $V_2(D_i)$ を作成する．

（3） 兄弟ディレクトリについては何も考慮しない．

（4） すべての文書から作成した IDF を特徴ベクトルに掛け合わせる．

概念を表現するための特徴ベクトルを作成する際に用いられるのが，その概念に相当するディレクトリに属する文書群である．このアルゴリズムでは，あるディレクトリに属する文書群として，そのディレクトリに分類されている文書群と，すべてのサブディレクトリに分類されている文書群を等しく扱う．その中の多くの文書にある語彙が存在するとすると，その語彙はその概念を説明するものとして重視すべきだと考えられる．そこで，DF 値を特徴ベクトルの作成に用いる．

ある対象文書集合 Doc において，語彙 $term$ に対する DF 値は，以下のように定義される．

$$DF(term, Doc) = \frac{Num(term, Doc)}{Num(Doc)}$$

ただし，

$Num(Doc) =$ Doc の総文書数,

$Num(term, Doc) =$ Doc 内で $term$ を含む文書の数

DF 値を用いると，そのディレクトリの特徴を出すことが可能であるが，一般的な語の DF 値は常に高くなってしまう．そこで，すべての文書を対象として求めた IDF 値を掛け合わせることによって，一般的な語の評価を下げる．

ある対象文書集合 Doc において，語彙 $term$ に対する IDF 値は，以下のように定義される．

$$IDF(term, Doc) = \log\left(\frac{1}{DF(term, Doc)}\right)$$

結果，この実装において，ディレクトリ D_i に相当する概念を表現する特徴ベクトル C_i は，C_i における語彙 $term$ の値を $C_i(term)$ として，以下のように定義される．

$$C_i(term) = DF(term, Doc(D_i)) \cdot IDF(term, Doc(D_{root}))$$

今回行った実装は，作られた概念階層木から語彙階層木を作成するということを前提としており，文書の自動分類のために用いたりする場合には，別の実装のほうが良いと考えられる．

3.3 語彙階層木の作成の基本的アルゴリズム

3.3.1 語彙階層木作成の基本的アルゴリズム

語彙階層木は，概念階層木のある概念における語彙どうしの関係性を定義することによって得られる．その手法には，様々な実装の方法が考えられるが，ここではまず，語彙階層木の作成の基本的な考え方を述べながら，基本的アルゴリズムについて説明する．

まず，説明に用いる記号についてに定義する．

- 対象とする概念階層木の概念を C_i とする．
- 対象とする語彙を X とし，X と他の語彙 $term$ について関係性を定義する．
- X に対する $term$ の関係において，
  - 上位性を $V_{上位}(term)$
  - 下位性を $V_{下位}(term)$
  - 共起性を $V_{共起}(term)$
  - 排他的関連性を $V_{排他}(term)$

  とする．
- C_i の下位の概念を表す特徴ベクトルの集合を $Sub(C_i)$ とする．

語彙階層木作成の基本的アルゴリズムは，以下のような流れである．

（1） C_i と $Sub(C_i)$ を比較し，X について，出現がどの程度偏っているかという**偏在性**を求める．

（2） C_i に含まれるすべての語彙 $term$ に関しても同様に，偏在性を求める．

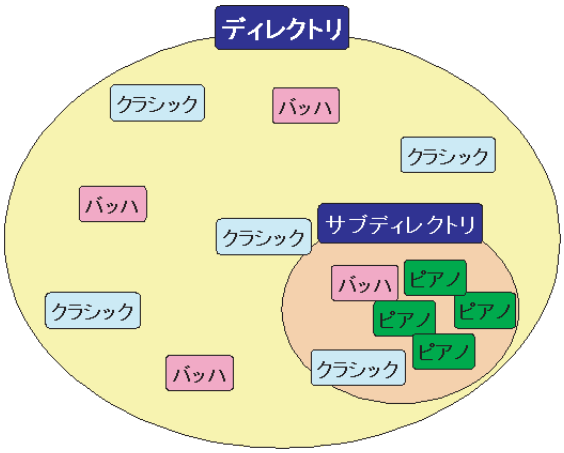

図 1　語彙の偏りの例

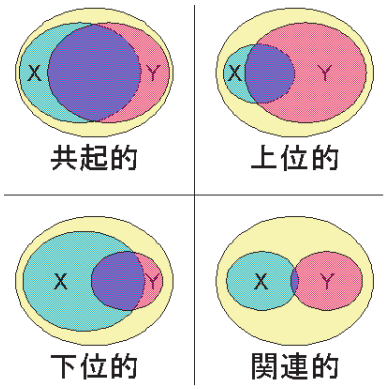

図 2　語彙 X にとっての語彙 Y との関係

（3）　X の偏在性と $term$ の偏在性の差異から，上位的か下位的かの判断を行う．

（4）　X の偏在性と $term$ の偏在性の共通性から，共起的か排他的関連かの判断を行う．

（5）　それを基に，C_i についてのみ考慮したときの，X に関する $term$ の関係性，$V_{\text{上位 }i}(term)$，$V_{\text{下位 }i}(term)$，$V_{\text{共起 }i}(term)$，$V_{\text{排他 }i}(term)$ を求める．

（6）　$C_i(X)$ を基に，重みを決定する．

（7）　$Sub(C_i)$ の各概念について，再帰的に X に関する $term$ の関係性を求める．

（8）　再帰的に求められた関係性を，重みを基にして合成し，$V_{\text{上位}}(term)$，$V_{\text{下位}}(term)$，$V_{\text{共起}}(term)$，$V_{\text{排他}}(term)$ を求める．

語彙階層木を作成する際に注目するのが，語彙の偏りである．図 1 は，あるディレクトリに文書がいくつか分類されており，その文書に現れる語彙の分布を表したものである．サブディレクトリもいくつか存在しているものとする．このディレクトリにおいて,「クラシック」や「バッハ」といった語は，まんべんなく分布している．それに対して,「ピアノ」という語はあるサブディレクトリにおいて頻出していて，偏っているといえる．このような分布の時に，例えば,「クラシック」と「ピアノ」という語の関係は,「クラシック」という語に対して,「ピアノ」は下位的な語である，ととらえることが可能である．また,「クラシック」と「バッハ」という語の関係は，お互いに広く分布する状況であり，お互いに共起する語である，ととらえることが可能である．

このように，ある範囲における 2 つの語の偏りの仕方によって考えられる例えば，語彙 X と語彙 Y がある時に，偏在性が異なるとすると，X と Y は上位下位関係になっているととらえることができる．すなわち，X にとっての Y との関係は，

（1）　語彙 X 偏り大 ∩ 語彙 Y 偏り小 → 上位的

（2）　語彙 X 偏り小 ∩ 語彙 Y 偏り大 → 下位的

と考えられる．また，偏在性が共通しているときには，

（1）　語彙 X 偏り小 ∩ 語彙 Y 偏り小 → 共起的

（2）　語彙 X 偏り大 ∩ 語彙 Y 偏り大 → 排他的関連

と考えることができる．

図 2 はこれら 4 つの関係性を図示したものである．例えば，左上では，語彙 X，語彙 Y ともに広く分布していて偏りが小さく，このときは共起的関係であり，右上では，語彙 X の偏りは大きく，語彙 Y の偏りは小さく，語彙 X にとって語彙 Y は下位的関係と考えられる．

このような，語彙の出現の偏在性は，概念階層木において，C_i と $Sub(C_i)$ を調べることによって算出することが可能である．

次に，重みだが，C_i は語彙 X を説明するために，$C_i(X)$ の値が大きければより重要で，小さければあまり重要ではない，ということができる．そのため，合成の際にそのことを考慮する必要がある．

C_i の下位のすべての概念 C_j において，$V_{\text{上位 }j}$，$V_{\text{下位 }j}$，$V_{\text{共起 }j}$，$V_{\text{排他 }j}$ を計算することが可能であるため，これを再帰的に計算して，重みを考慮しながらそれらを合成する必要がある．

以上が，概念階層木から，語彙階層木である，$V_{\text{上位}}(term)$，$V_{\text{下位}}(term)$，$V_{\text{共起}}(term)$，$V_{\text{排他}}(term)$ を求める基本的なアルゴリズムである．

3.3.2　語彙階層木作成の実装

基本的アルゴリズムに従って行った、今回の実装について述べる．

今回は，語彙の偏りの程度を算出するために，ジニ係数という統計的指標を用いた．ジニ係数は，主に経済学において，所得分配の不平等さや富の偏在性などを表す指標として用いられる指標である．係数の範囲は $[0,1]$ であり，例えば富の偏在性を表すときには，富が完全に平等に分配されているときには 0 になり，富を一人が独占しているときには 1 となる．

ジニ係数を用いたのは、確立された統計的指標であることと、偏りの程度が $[0,1]$ の範囲で突出した特徴なくばらつくためである。しかし、ジニ係数以外にも偏在性の計算のために用いることができる式はいくらでも考えられるため、評価の基準に従って別の実装も行ってみる必要があるが、それについては今後の課題とする。

ジニ係数の一般的な式について説明する．

ある N 個のデータが与えられ，それを

$$x_1, x_2, \ldots, x_N$$

とする．このとき，すべての 2 つのデータの差の絶対値を全

データにわたって総平均したものが平均差と呼ばれ，

$$\text{Mean Difference} = \frac{\sum_i \sum_j |x_i - x_j|}{N(N-1)}$$

という式で表される．すべてのデータの組み合わせの差を足しあわせ，データの組み合わせ総数で割っている．

平均差は，データの単位に応じた値となる．ジニ係数は，平均差をデータの単位に依存しない相対的な指標にしたもので，下記の式で表される．

$$GC = \frac{\frac{\sum_i \sum_j |x_i - x_j|}{N \cdot (N-1)}}{2\overline{X}} = \frac{\sum_i \sum_j |x_i - x_j|}{2N(N-1)\overline{X}}$$

ただし，

$$\overline{X} = x_i \text{ の平均値} = \frac{\sum_i x_i}{N}$$

この式によって得られる値は，元のデータの単位にかかわらず，$[0, 1]$ の値になる．

この，ジニ係数を語彙の出現の偏りの計算のための計算に用いることを考える．まず，ディレクトリ D_i のサブディレクトリ集合 $SubDir(D_i)$ を考える．このとき，D_i に直接分類されている文書群について，仮想のサブディレクトリに分類されているものとして扱う．つまり，$SubDir(D_i)$ は実際より 1 つ多いサブディレクトリの集合となる．この時，$D_j (\in SubDir(D_i))$ において，ある 1 つの文書に $term$ が現れる確率は，

$$P_j(term) = \frac{Num(term, Doc(D_j))}{Num(Doc(D_j))} = DF(term, Doc(D_j))$$

となり，そのような確率を持った文書が $Num(Doc(D_j))$ だけ存在していることになる．この $P_j(term)$ をジニ係数の計算におけるデータとして扱うと，ディレクトリ構造における語彙の出現の偏りが求められることになる．つまり，語彙 $term$ の D_i におけるジニ係数 $GC(term)$ の計算式は，以下のように表すことができる．

$$GC(term) = \frac{\sum_m \sum_n (|P_m(term) - P_n(term)| \cdot N_m N_n)}{2N_i(N_i - 1) \cdot (P_i(term)/N_i)} = \frac{\sum_m \sum_n (|P_m(term) - P_n(term)| \cdot N_m N_n)}{2(N_i - 1)(P_i(term))}$$

ただし，

$$D_m, D_n \in SubDir(D_i),$$
$$N_m = Num(Doc(D_m)),$$
$$N_n = Num(Doc(D_n)),$$
$$N_i = Num(Doc(D_i))$$

ここで，概念階層木の今回の実装を見てみると，3.2.2 小節で述べたように，概念 C_i の特徴ベクトルは，

$$C_i(term) = DF(term, Doc(D_i)) \cdot IDF(term, Doc(D_{all}))$$

と表される．これを $P_i(term)$ を使って表して，先ほどのジニ係数の計算式に代入して整理すると，

$$GC(term) = \frac{\sum_m \sum_n (|C_m(term) - C_n(term)| \cdot N_m N_n)}{2(N_i - 1)(C_i(term))}$$

となる．この式は，各概念を表現するのに用いた文書数 $Num(Doc(D_i))$ があれば，概念階層木からジニ係数が計算できることを示している．

今回の概念階層木はこのような関係を示すために，非常に単純な実装であったが，基本的にはどのように概念階層木が作られても，$Num(Doc(D_i))$ があれば，同様にして語彙の出現の偏りを計算できることになる．

ジニ係数を用いて，概念 C_i における，ある語彙 X に対する $term$ の関係性は，上位的か下位的か，ということについては，

$$V_{\text{上位 } i}(term) = (GC(X) \cdot (1 - GC(term))) \cdot C_i(term)$$
$$V_{\text{下位 } i}(term) = ((1 - GC(X)) \cdot GC(term)) \cdot C_i(term)$$

と表され，共起的か排他的関連か，ということについては，

$$V_{\text{共起 } i}(term) = ((1 - GC(X)) \cdot (1 - GC(term))) \cdot C_i(term)$$
$$V_{\text{排他 } i}(term) = (GC(X) \cdot GC(term)) \cdot C_i(term)$$

と表される．

これらの語彙 $term$ における関係性は，始めにコンテキストで与えられた概念の下位の概念においても求めることができ，最終的には合成しなくてはならない．今回の実装においては，各ノードにおける重みを以下のように定義した．

$$Weight_i(X) = \frac{Num(X, Doc(D_i))^k}{Num(Doc(D_i))^{k-1}} = Num(Doc(D_i)) \cdot \left(\frac{C_i(X)}{IDF(X, Doc(D_{root}))} \right)^k$$

現在は，経験的な観点から $k = 3$ を用いている．

最後に，これらを合成するのだが，重みを掛けてすべて足しあわせる，という方法をとった．すなわち，$V_{\text{上位}}$ の場合では，以下の式となる．

$$V_{\text{上位}}(term) = \sum_i (V_{\text{上位 } i}(term) \cdot Weight_i(X))$$

以上が，語彙階層木を作成するアルゴリズムの実装である．今回は，文書群とディレクトリ構造から直接作るように説明したが，どのような実装の概念階層木からも

- $Num(Doc(D_i))$
- $IDF(term, Doc(D_{root}))$

の 2 つが計算可能であれば，本実装を適応可能であることが数式から明らかである．

4. 試作システムと Web 検索の個別化への利用

4.1 試作システムの概要

これまで述べたアルゴリズムの実装に従って試作システムを作成し，実際に概念階層木と語彙階層木を作成した．

本システムは，Windows 上で動作する．扱える文書のタイプは，

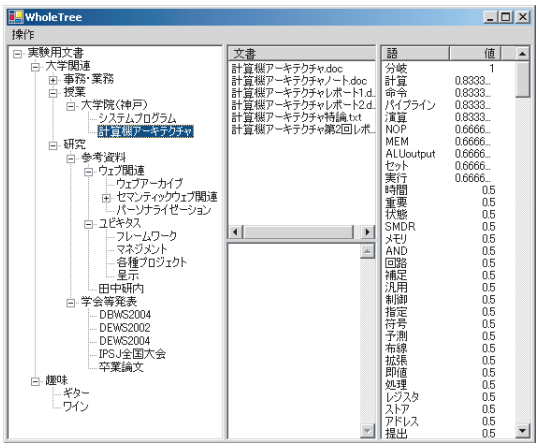

図 3　概念階層木を提示するフォーム

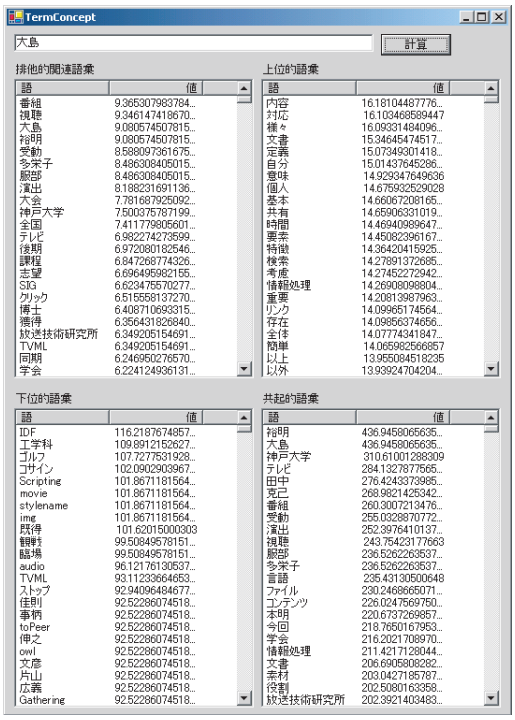

図 4　語彙階層木の語彙の関連性を提示するフォーム

- テキストファイル
- PDF ファイル
- Microsoft Word ファイル
- Microsoft PowerPoint ファイル

である．

今回，システムに読み込ませたのは，1 つのディレクトリツリーで，

- 全ファイル数：149
- 全フォルダ数：37

である．図 3 が，概念階層木を提示するフォームである．左がディレクトリツリーで，ディレクトリを選択すると，右にそのディレクトリの DF 値が提示される．

図 4 が，語彙階層木のフォームである．コンテキストとして概念階層木の概念の 1 つをあらかじめ指定しておき，語彙を入力すると，語彙どうしの関連性を算出して提示する．図では，

図 5　Web 検索の個別化を行うフォーム

コンテキストとしての概念をルートの概念とし，そのときの「大島」という語に関して関係性を求が求められ，関係性の高い順に語彙とそれぞれの関連の評価値を提示している．

図 5 が，Web 検索の個別化を行うフォームである．詳しくは次節で説明するが，現時点では，

- 検索キーワードの追加
- 検索結果の再ランキング

が行えるようになっている．

4.2　Web 検索の個別化

今回作成した語彙階層木を用いて行った，Web 検索におけるパーソナライゼーションについて述べる．

4.2.1　検索キーワードの追加

何か欲しい文書があるときに，どのような検索キーワードで検索すれば良いかという判断が難しい場合がある．そのようなときに，語彙階層木を用いて検索キーワードを拡張することができる．

単にキーワード拡張をするのであれば，これまで様々な研究がされている．しかし，語彙階層木を用いることによって，ユーザは検索キーワードを投げるとともに，欲しい文書が自分が保有しているディレクトリ構造の中で，どのあたりを考慮しながらキーワードを拡張して欲しいかということを指定することができる．なぜなら，ディレクトリ構造と概念階層木は同一視でき，その概念のもとで出現するすべての語彙に対して，4 つの関連性を基にして，関係が深い語彙が求まっているからである．

つまり，ユーザはシステムに対して，

- 検索キーワード
- コンテキストとしての概念階層木の概念

を与えることで，よりそのユーザが求めている文書を絞り込むための検索キーワード拡張が行える．

システムでは，与えられたコンテキストとしての概念のもとで，検索キーワードに対する語彙階層木を求めて，関連性の高い語を求めて，検索キーワードに AND の関係で追加して，既存の検索エンジンで検索結果を求める，ということを行っている．

図 5 において追加キーワードというところの「通常」という

ボタンを押すことで、共起的関連が強い語が追加され、「緩やか」というボタンを押すことで、上位的関連が強い語が追加されるようになっている。ユーザは場合に応じて検索語と欲しいページとの関係からどのような検索キーワード拡張を行うか，ということを指定することが可能になっている．

4.2.2 検索結果の再ランキング

Web 上の検索エンジンで検索を行うと，検索結果は順番に並んで返される．このとき，検索エンジンの内部ではページに対して何らかの評価が行われており，それによってページの順序が決定されている．例えば Google では，検索結果を PageRank という評価を用いて並べ替えて提示している．

そのような検索エンジンによる評価は，といったような社会的・一般的に広く必要とされると考えられるページに対して高くつけられるものである．しかし，ある人が本当に求めているページが必ず検索結果の上位に現れるということはなく，しばしば，様々なページを実際に読んでみて，本当に探していたページを見つける作業が必要になる．

概念階層木や語彙階層木は，その人の考え方を表しており，これらを用いて検索結果の再ランキングを行えば，各個人に適応したランキング評価になるのではないかと考えた．

今回，実際に行ったのは，語彙階層木と結果ページの類似度を基にした再ランキングである．流れを説明する．まず，ユーザは検索キーワードを指定する．同時に，検索のコンテキストとして，概念階層木における概念の 1 つを指定する．検索キーワードに対して語彙階層木を求め，4 つの関連性の 1 つを基に，語彙を次元とし，関連性の評価値をその次元の値とするベクトルを生成する．検索キーワードで既存の検索エンジンから検索ページをいくつか取得する．検索で得られたページの特徴ベクトルと語彙階層木から作成されたベクトルの類似度を計算し，類似度が高い順に並べ替える．

個人がある語彙に対してどのような考えを持っているか，ということを表すのが語彙階層木である．検索で得られたページの中には，検索キーワードがその人の意図するような使われ方をしているものもあれば，そうでないものも含まれている．検索キーワードが意図とより近い使われ方をしている文書を見つけるためには，検索キーワードに対する概念階層木から 1 つの関連性に着目してベクトルを生成し，それと検索で得られたページの特徴ベクトルの類似度を求めれば良いと考えられる．

特徴ベクトルの類似度としては，コサイン類似度を用いた．特徴ベクトル V_1 と V_2 のコサイン類似度は以下のように定義される．

$$Similarity(V_1, V_2) = \frac{V_1 \cdot V_2}{|V_1| \cdot |V_2|}$$

着目する関係性は 4 つあるが，通常はその中でも対象の語彙と共起しやすいと考えられる語彙が高い評価値を持つ，共起を用いるのが良いと考えられる．ただ，検索で得られたページの内，より概要的な文書やより詳細な内容の文書のランクを高くしたいときには，上位や下位といった関係性も利用できると考えられる．図 5 において、評価ベクトルとされているところで、どのような関係性に着目するかを指定することが可能となっている。

5. まとめと今後の課題

本研究では，個人が保有する文書とそれらが分類されているディレクトリ構造を基にして，

- 概念階層木
- 語彙階層木

の作成を行った．これらは，個人の概念体系や，あるコンテキストにおいて語彙どうしの関連性を表すものである．さらに，それらを Web 検索のパーソナライゼーションにおいて利用する手法について提案を行った．

個人的な情報はコンピュータ上にさらに増えつつあり，今後の課題の一つとしては，電子メールや Blog といった情報源も対象とすることが考えられる．そのとき，重要になってくるのは時間の概念である．現在は，個人のスタティックな知識は表現できたが，個人の考え方などは時間とともに変化するものであり，電子メールや Blog などを考えたときには時間についての考える必要がある．

また，今回行った，Web 検索における概念階層木・語彙階層木の利用において，実験が十分でないことも課題である．個人的に良いものが得られるようになるかということは評価が困難であるが，どのように評価するかということも含め，早急に実験・評価を行う必要がある．

謝　　辞

本研究の一部は，文部科学省科学技術振興費プロジェクト「異メディア・アーカイブの横断的検索・統合ソフトウェア開発」（代表：田中克己），および，平成 16 年度科研費特定領域研究 (2) 「Web の意味構造発見に基づく新しい Web 検索サービス方式に関する研究」（課題番号：16016247，代表：田中克己），および，21 世紀 COE プログラム「知識社会基盤構築のための情報学拠点形成」によるものです．ここに記して謝意を表すものとします．

文　　献

[1] E.Adar, D. Karger and L. Stein, ”Haystack: Per-User Information Environment”, Proc.1999 Conference on Information and Knowledge Management, pp.413-22, 1999.

[2] Haystack: http://haystack.lcs.mit.edu/

[3] 片山佳則，小櫻文彦，井形伸之，渡部勇，津田宏，セマンティックグループウェア WorkWare++と KnowWho 検索への応用，情報処理学会 研究報告「情報学基礎」No.071, 2003.

[4] 内野寛治，津田宏，松井くにお，WorkWare:WEB を用いた文書の時間順整理の試み，情報処理学会 研究報告「情報学基礎」No.051, 1998.

[5] Hiroyuki Sato, Yutaka Abe and Atsushi Kanai, ”Hyperclip: a Tool for Gathering and Sharing Meta-Data on Users’ Activities by using Peer-toPeer Technology”, WWW2002 Workshop on Real world RDF and Semantic Web applications, 2002.

[6] 湯川高志，吉田仙，桑原和宏，パーソナル・レポジトリに対するピア・ツー・ピア型協調検索機構の提案，電子情報通信学会 信学技報 AI2001-48，2001.

[7] 大島裕明, 小山聡, 田中克己: 個人コンテンツからの概念体系の生成とこれに基づく Web 検索のパーソナライゼーション. 情報処理学会 DBS 研究会技術報告 (2004), Vol.2004, No.72, 2004-DBS-134, pp.345–351