DEWS2007 B7-8

自己組織化マップを利用する分類済み階層の自動設定

澁谷慧一郎† 遠山 元道††

† 慶應義塾大学 理工学部 情報工学科 〒 223–8522 横浜市港北区日吉 3–14–1
†† 慶應義塾大学 理工学部 情報工学科 〒 223–8522 横浜市港北区日吉 3–14–1
E-mail: †shibuya@db.ics.keio.ac.jp, ††toyama@ics.keio.ac.jp

あらまし ショッピングサイトやオークションサイトにおいてジャンルやトピックで分類する階層的なディレクトリ構造が多く見られるようになった。ディレクトリ構造は大量の商品データに対して、ユーザが意図した検索条件を持たない曖昧な検索をする際に便利である。しかし、階層の末端ディレクトリにおいても商品データ件数が 1000 件を超えるような場合が多数見られる。こうした状況に対してユーザの嗜好に合った必要な商品データだけを効率的に得る手段として、本論文では，Kohonen の自己組織化マップ（self-organization maps）を利用しディレクトリ構造の最下層における再分類を行う. 実装したシステムでは、各商品名に含まれる頻出単語をキーワードとして商品特徴ベクトルを作成する。そして、各末端ディレクトリにおける全商品特徴ベクトルから自己組織化マップを利用しクラスタリングを行う。そして、ユーザが選択した商品データからユーザ特徴ベクトルを作成し、商品特徴ベクトルから作成されたマップにユーザ特徴ベクトルをマッピングする。ユーザの嗜好を表すユーザ特徴ベクトルを含むクラスタを提示しユーザが各商品データを選択することでユーザは嗜好に合った商品データを得ることが出来る。
キーワード 自己組織化マップ，WEB

Automatic setting of a hierarchy finished with a classification using a self-organization maps

Keiichiro SHIBUYA† and Motomichi TOYAMA††

† School for Open and Envionmental Systems, Graduate School of Science and Technology, Keio University
Hiyoshi3–14–1, Kouhoku-ku, Yokohama-shi, 223–8522 Japan
†† Department of Information and Computer Science, Faculty of Science and Technology, Keio University
Hiyoshi3–14–1, Kouhoku-ku, Yokohama-shi, 223–8522 Japan
E-mail: †shibuya@db.ics.keio.ac.jp, ††toyama@ics.keio.ac.jp

Abstract In a shopping site and an auction site, hierarchical directory structure to classify in a genre and a topic came to be generally seen. When I do the vague search that does not have the search condition that a user aimed at, directory structure is convenient for a large quantity of article data. However, a lot of cases that the article data number is beyond 1,000 cases in an end directory of a hierarchy are seen. As means to get only the necessary article data which matched taste of a user for such situation effectively, I use a self-organization maps of Kohonen in this article and perform reclassification in a bottom layer of directory structure. By the system which I implemented, I make an article characteristic vector as keyword by a frequent appearance word to be included in each brand name. And I use a self-organization maps from all articles characteristic vector in each end directory and perform clustering. And I make a user characteristic vector from the article data which a user chose and map a user characteristic vector onto a map made from an article characteristic vector. Because I show a cluster including a user characteristic vector expressing taste of a user, and a user chooses each article data, a user can get the article data which matched taste.
Key words SOM, WEB

1. はじめに

近年，Amazon.co.jp [1] や楽天 [2] のようなショッピングサイトや Yahoo!オークション [3] のようなオークションサイト等，大量の商品データを扱ったサイトが一般的に見られるようになった．そうしたサイトではジャンル等で分類する階層的なディレクトリ構造により大量の商品データを分類している場合がほとんどである．このようなディレクトリ構造は大量の商品データに対して、ユーザが意図した検索条件を持たない曖昧な検索をする際に便利である。しかし，ディレクトリの深さはジャンルやトピックにより千差万別であり，ディレクトリ末端における商品データ件数が 1000 件を超えるものも多数存在している場合もある．いわゆる情報過多と呼ばれる状況である。商品データ件数が非常に多い中から欲しい商品を探すのはユーザにとって大きな負担である．このような末端ディレクトリに対しては，新しいディレクトリを作成する必要があると考えられるが，商品データは頻繁に誕生またはなくなるために，更新の維持が難しくなる．そしてディレクトリ数の増加により，商品データ件数が非常に多いディレクトリと非常に少ないディレクトリが出来てしまい，ディレクトリ末端における対象数を常に均等にするのは困難になる．さらに，ディレクトリの作成・更新の作業は人手によるので，ジャンルごとの知識をもつ多くの熟練した編集者と多大な費用が必要になる．そこで，こうした負担を軽減し，かつユーザが簡単に欲しい情報を得やすくするために，本論文では，自己組織化マップを利用してジャンルやトピック等で分類された既存のディレクトリの末端の再分類を行う手法を提案する．

提案手法では以下の 3 つの手順により実現する．(1) ユーザが選択した商品データからユーザの嗜好に応じたユーザ特徴ベクトルを作成する。(2) 商品特徴ベクトルの自己組織化マップによるクラスタリングを行う。(3) 各クラスタをディレクトリ群として考え、ユーザ特徴ベクトルをマップにマッピングし、ユーザが近傍の各商品データを選択する。

以下，本稿の構成を示す．2 章では Kohonen の自己組織化マップの原理について述べる．次に，3 章では商品データ分類のための各商品の特徴ベクトルと，ユーザの欲しい商品の特徴を表したユーザ特徴ベクトルの生成について述べる．これらを実装，評価，検討した結果を 4 章で述べた後，最後に 5 章でまとめを述べる．

2. 自己組織化マップ（Self-Organizing Map：SOM）

自己組織化マップ（SOM）は人間の脳の仕組みを模倣した情報処理機構である人工ニューラルネットワークの 1 つである．教師無しの学習アルゴリズムをとる．多次元の入力層と出力層の 2 層からなるネットワークである．通常出力層は 2 次元に配列される．学習を繰り返すことで特徴の似ているデータ同士が出力マップ上で近い位置に配置され，類似したデータかどうか直感的に理解しやすく，視覚的な情報処理が可能である．SOM には多くの種類があるが，ここでは最も基本的名ものについて述べる．

SOM で用いられるネットワークは，セルを 2 次元に六角格子状に配置したものである．それぞれのセル i はセルの特徴ベクトル $m_i(t) \in R^n$（R は実数）を持っており（t は時間を表し $m_i(0)$ は適切な方法で初期化されている），これらのセルの特徴ベクトルを，入力である特徴ベクトル $x_j \in R^n (j = 1, 2, \cdots, d)$ に選択的に近づけることによって学習は進行する．このとき，SOM では入力となる特徴ベクトルに一番近いパターンを持つ出力セルおよびその近傍のセルの集合のみが入力ベクトルに近づくことができるようなアルゴリズムをとる．

SOM のアルゴリズムを以下に示す．

（1） 各入力特徴ベクトルを生成し，その集合を X とする．

$$X = x_j | x_j \in R^n, j = 1, 2, \cdots, d$$

（2） 出力層にある各ユニットの持つパターンを初期化する．

$$M = m_i | m_i \in R^n, i = 1, 2, \cdots, d$$

（ただし，$m_i(0) = [0, 0, \cdots, 0]$ とした）

（3） T をあらかじめ設定された学習回数とする．このとき，$t = 0, 1, \cdots, T$ について以下を繰り返す．

（a） $\|x_j - m_x(t)\|$ を最小にするセル c を求める．

$$c = argmin \|x - m_i\|$$

（b） 探し出したセル c の特徴ベクトル m_c を更新し，さらにその近傍のセルの集合 N_c も入力パターンに近づける．

$$m_i(t+1) = \begin{cases} m_i(t) + \alpha(t)[x_k(t) - m_i(t)] & (i \in N_c(t)) \\ m_i(t) & (i \notin N_c(t)) \end{cases}$$

N_c の中央はセル c である．N_c の半径は，学習の初期段階ではたいてい大きく，学習を繰り返していくうちに単調に減少させる．また，$\alpha(t) \in (0, 1)$ は「学習率」を表し，これもまた時間と共に単調に減少させる．

$$\alpha(t) = \alpha_0(t) exp(-\frac{\|r_c - r_i\|^2}{\sigma(t)^2})$$

ただし，r_c と r_i はそれぞれセル c とセル i のもつベクトルを表す．

（c） $j = 1, 2, \cdots, d$ について上記 (a), (b) を繰り返す．

3. 商品データの分類のための特徴ベクトル生成

商品データのジャンルやトピックにより分類してユーザに推薦する研究 [7] や，商品データの値段や作成年月日などの集約型の情報による分類を行う研究 [5] [6] は数多くなされている．本論文で扱う商品データはジャンルやトピック等で分類されたディレクトリ末端のデータを対象としておこなう．図 1 に本手法の概要を示す．ディレクトリ末端において，ユーザはランダムに選ばれた商品の中から嗜好に合う商品を複数選択し，ユーザ選択商品の商品名に形態素解析をおこないユーザ特徴ベクトルを生成する．次いで，ディレクトリ末端の全商品名ごとに形態素解析をおこなった商品特徴ベクトルから自己組織化マップで利用する初期ベクトルを生成する．そして，自己組織化マップによりクラスタリングをおこなう．ユーザ特徴ベクトルを自己組織化マップにマッピングをおこない，ユーザ特徴ベクトルの近傍の商品データをユーザが選択することにより，ユーザは独自の嗜好に合う商品を得ることができる．以下に各処理につ

いて示す．

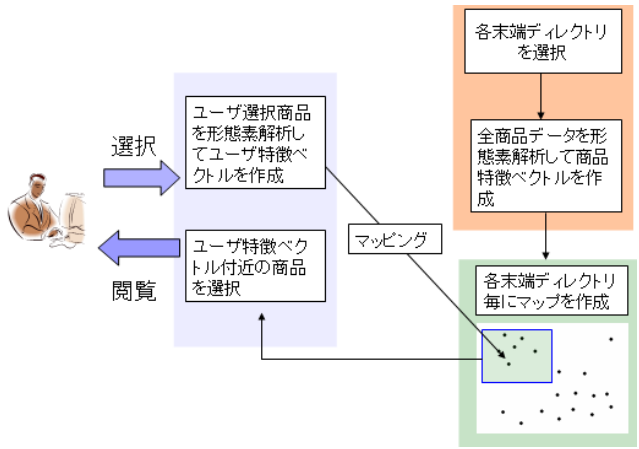

図 1　システムの全体構成

3.1　商品特徴ベクトルの作成

各ディレクトリ末端において，商品特徴ベクトルを生成する．各商品名を茶筅により形態素解析をおこない，名詞部のみを抽出する．そして，抽出された単語を出現回数でソートをおこない，それぞれ単語 $w_1, w_2, \cdots, w_n$ として特徴ベクトルの属性とする．　各商品 D_i における属性 $w_1, w_2, \cdots, w_n$ の重要度を求め，商品の特徴ベクトル $\mathbf{M}(D_i)$ を生成する．

$$\mathbf{M}(D_i) = (\frac{f_1^i}{N_1}, \frac{f_2^i}{N_2}, \cdots, \frac{f_n^i}{N_n})$$

ここで $f_j^i (j = 1, 2, \cdots, n)$ は商品 D_i における単語 w_j の出現頻度を表し，$N_k (k = 1, 2, \cdots, n)$ は f_j^i の全商品に対する単語の総出現頻度を表す．全商品において出現頻度が高い属性は，分類済みの商品のジャンルやトピックを表している可能性が高いので属性値が低くなるようにしている．全商品において出現頻度が低い属性は，その商品の特徴をよく表していると考えられるので属性値が高くなるようにしている．

3.2　ユーザ特徴ベクトルの作成

ユーザの商品に対する嗜好は商品データのジャンルやトピック，数ヶ月，数年という時間の経過にともない大きく変わっていくものであり一概には決定し難い．そこで，ディレクトリ末端においてランダムに選ばれた商品データをユーザに複数選択させることにより，そのディレクトリにおけるユーザの特徴ベクトルを決定する．まず，ユーザの選択した商品データの商品名を茶筅により形態素解析をおこない名詞部のみを抽出する．そして，抽出された単語からディレクトリ末端におけるユーザ特徴ベクトル $\mathbf{U}$ を作成する．

$$\mathbf{U} = (\frac{g_1}{N_1}, \frac{g_2}{N_2}, \cdots, \frac{g_n}{N_n})$$

ここで $g_j (j = 1, 2, \cdots, n)$ はユーザの選択した各商品 D_i における属性 $w_1, w_2, \cdots, w_n$ の総出現頻度，$N_k (k = 1, 2, \cdots, n)$ は g_j の全商品に対する総出現頻度を表す．商品特徴ベクトルと同様に，全商品において出現頻度が高い属性は，分類済みの商品のジャンルやトピックを表している可能性が高いので属性値が低くなるようにしている．全商品において出現頻度が低い属性は，その商品の特徴をよく表していると考えられるので属性値が高くなるようにしている．

3.3　属性数の決定

商品特徴ベクトルを作成する際，頻出単語を表す属性 $w_1, w_2, \cdots, w_n$ を決定するが，出現回数の低い単語には商品データの特徴を表していない無意味な単語が含まれてしまう場合が多い．特に茶筅による形態素解析において未分類語を含めた場合は無意味な単語が多数含まれてしまう．そこで，出現回数に閾値を設けて無意味な単語を切り捨てることにより属性の精度が高くなるようにする．

実際に Amazon.co.jp において，本/コンピュータ・インターネット/プログラミングのディレクトリ下の末端ディレクトリである Java ディレクトリと C ディレクトリについて検討してみる．各ディレクトリに含まれる商品データを茶筅により形態素解析して名詞・未分類語を抽出した．Java ディレクトリでは，691 個の単語が存在し，C ディレクトリでは，791 個の単語が存在した．各ディレクトリにおいて頻出単語の出現回数に閾値を設けて，ある一定以上の出現回数の単語のみを属性とし，その閾値と属性数を図 2 に示す．

出現回数の閾値が低い範囲では属性数の減少率が大きいが，閾値が 8 回（Java）と 10 回（C）を超えた付近から緩やかになっている．実際の頻出単語を確認してみても出現回数が 8〜10 以下の単語は商品データの特徴とは関係ない無意味な単語が多数含まれていた．このような単語を属性として利用すると分類の精度が落ちてしまうと考えられる．

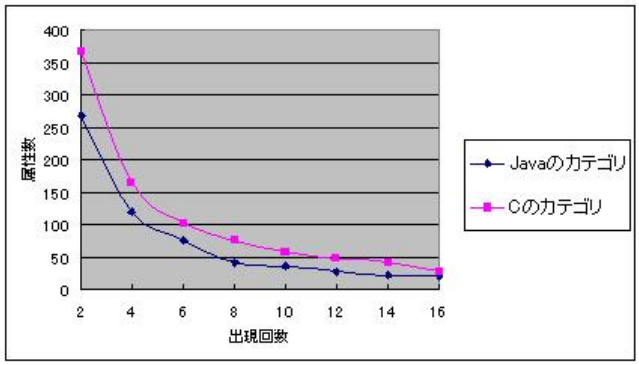

図 2　頻出単語閾値と属性数

3.4　自己組織化マップの実行

各末端ディレクトリにおいて、ディレクトリに含まれる全商品データの商品特徴ベクトルで自己組織化マップを実行する。自己組織化マップは各末端ディレクトリごとに用意しておく。

3.5　ユーザ特徴ベクトルのマッピング

各末端ディレクトリごとに用意された自己組織化マップにユーザ特徴ベクトルのマッピングをおこなう。商品特徴ベクトルの学習後のコードブックベクトルにラベル付けをおこなう際にユーザ特徴ベクトルのラベル付けをおこなう。ユーザ特徴ベクトルの近傍の商品データをユーザが選択することで嗜好の商品データを得ることが出来る。

4. 実装・評価・検討

4.1 実装の概要

特徴ベクトル生成と SOM によるクラスタリングを PHP の関数として実装する．本節では実装した関数の仕様と利用例について述べる．なお，ここで扱う分類済みである商品データは，各末端ディレクトリごとに DB 内にあるものとする．

4.2 関 数 仕 様

以下は実装した PHP の関数の仕様である．実装した関数のそれぞれの役割について表 1 に示す．

表 1　追加した関数およびその役割

関数名	役割
makevector	特徴ベクトル作成用の関数
inivector	商品特徴ベクトル作成用の関数
makesom	SOM 作成用の関数
viewdata	ユーザ選択データ表示用関数
chasen	chasen による形態素解析実行用の関数

4.2.1 makevector 関数

int **makevector** (string *sql*, string *string*)

makevector 関数は，ユーザが選択した商品データ *string* として SQL 質問文 *sql* からユーザ特徴ベクトルを生成する．データベースには末端ディレクトリごとの商品データが格納されているとする．*sql* は文字列型であり *string* は配列である．*sql* により対象の末端ディレクトリの商品データを指定する．makevector 関数の戻り値は整数型で，すべての処理が問題なく終了した場合に 1 を返し，エラーが生じた場合には −1 を返す．

ユーザ特徴ベクトル作成の簡単な例を示す．表 3 の商品においてユーザが b0 の初心者のための JAVA 入門と b1 の Eclipse で JAVA 入門の 2 つの商品を選択した場合のユーザ特徴ベクトルは表 2 のようになる．

4.2.2 inivector 関数

int **inivector** ()

inivector 関数は，各末端ディレクトリにおける商品特徴ベクトルを生成する．実行時間短縮のため，商品特徴ベクトルはサイト作成者がこの関数により前もって作成しているものとする．inivector 関数の戻り値は整数型で，すべての処理が問題なく終了した場合に 1 を返し，エラーが生じた場合には −1 を返す．実際に，商品特徴ベクトル作成の簡単な例を示す．表 3 のような 4 つの商品データがあった場合，商品特徴ベクトルは表 4 のようになる．

4.2.3 makesom 関数

int **makesom** (int *xdim*, int *ydim*, int *rlen*)

makesom 関数は，初期ベクトルから SOM を実行する関数である．*xdim* と *ydim* と *rlen* は全て整数型である．実行結果から得られるマップの横幅を表す *xdim* と縦幅を表す *ydim* と SOM の学習回数を表す *rlen* をユーザがフォームから指定できる．また，学習近傍形状は 6 角格子型，近傍関数はステップ関数，初期近傍半径は 6，乱数は毎回異なったものを使用する．makevector 関数の戻り値は整数型で，すべての処理が問題なく終了した場合に 1 を返し，エラーが生じた場合には −1 を返す．

実際に，SOM 作成の簡単な例を示す．表 2 のユーザ特徴ベクトルに対して 50 件の商品データで SOM を実行すると図 3 のようになる．ユーザ特徴ベクトルを含むクラスタ内にある商品データのいくつかを表 5 に示す．クラスタ上部では初心者または入門の属性値を持つ商品データが多く，クラスタ下部に行くと Eclipse の属性値を持つ商品データが多く見られる．

4.2.4 mappingsom 関数

int **mappingsom** ()

mappingsom 関数は，商品特徴ベクトルから作られた SOM

表 2　ユーザ特徴ベクトル例

	属性 1	属性 2	属性 3	属性 4	属性 5	属性 6	属性 7
	java	入門	初心者	Eclipse	簡単	難解	プログラム
U	(2/4)	(2/3)	1	1	0	0	0

表 3　商　品　例

b0	初心者のための JAVA 入門
b1	Eclipse で JAVA 入門
b2	簡単な JAVA 入門
b3	難解な JAVA プログラム
…	…

表 4　商品特徴ベクトル例

	属性 1	属性 2	属性 3	属性 4	属性 5	属性 6	属性 7
	java	入門	初心者	Eclipse	簡単	難解	プログラム
b0	(1/4)	(1/3)	1	0	0	0	0
b1	(1/4)	(1/3)	0	1	0	0	0
b2	(1/4)	(1/3)	0	0	1	0	0
b3	(1/4)	0	0	0	0	1	1
…	…	…	…	…	…	…	…

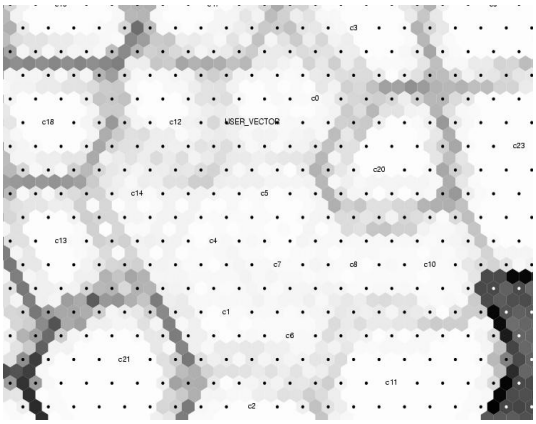

図 3　SOM 実行例

表 5　SOM 実行結果による商品

b0	初心者のための JAVA 入門
b12	初心者でも JAVA アプリケーション作成
b5	Eclipse による JAVA 入門
b14	分かり易い JAVA 入門
b4	Eclipse で JAVA
b7	簡単 JAVA 入門
b8	Eclipse を使った簡単な JAVA
b10	Eclipse+JAVA でプログラミング
…	…

図 4　表　示　例

に対してユーザ特徴ベクトルをマッピングする関数である．商品特徴ベクトルの学習後のコードブックベクトルにユーザ特徴ベクトルを追加して，ラベル付けを行う．

4.2.5　viewdata 関数

int **viewdata** (string *sql*, string *string*)

viewdata 関数は，ユーザ特徴ベクトルを作成するための SQL 質問文 *sql* から商品選択画面を表示する関数である．データベースには末端ディレクトリごとの商品データが格納されているとする．*sql* は文字列型であり *string* は配列である．*sql* により対象の末端ディレクトリの商品データを指定する．通常は商品データをランダムで 50 件表示し，ユーザはフォームにより嗜好データを決定する．viewdata 関数の戻り値は整数型で，すべての処理が問題なく終了した場合に 1 を返し，エラーが生じた場合には −1 を返す．商品データ表示例を図 4 に示す．

4.2.6　chasen 関数

int **chasen** (string *chasen*, string *str*, int *i*)

chasen 関数は，商品データ名の頻出単語を決定するための形態素解析をおこなう関数である．*chasen* により chasen の実行ファイルがあるパスを指定する．*str* は文字列型であり，商品データ 1 件あたりの商品名である．*i* はその商品データが何番目のものであるかを意味している．通常は形態素解析をした結果より名詞のみを選択するが，場合により未定語や動詞等を含める必要がある．chasen 関数の戻り値は整数型で，すべての処理が問題なく終了した場合に 1 を返し，エラーが生じた場合には −1 を返す．

```
1   /* sample.php */
2   <html>
3   <head>
4   <title>書籍リスト</title>
5   <meta http-equiv="Content-Type"
          content="text/html;charset=Shift_JIS" />
6   </head>
7   <body>
8   <form method="POST" action="sample.php">
9   <?php
10      include("viewer.inc");
11      include("makevector.inc");
12      include("makesom.inc");
13      $sql = "SELECT b.name, b.author, b.publisher,
                  b.price, b.pdate FROM book b";
14      if($_POST[select]){
15
16          makevector($sql, $_POST[select]);
17          mappingsom();
18
19      }elseif($_POST[name]){
20
21          viewdata($sql, $_POST[name]);
22
23      }else{
24          viewdata($sql);
25
26          }
27  ?>
28  </body>
29  </html>
```

図 5　PHP ファイル例

4.2.7　利　用　例

4.2 節のとおり，viewdata 関数でユーザ選択画面を表示し makevector 関数で商品特徴ベクトルとユーザ特徴ベクトルを作成し mappingsom 関数で SOM にユーザ特徴ベクトルのマッピングを行う．サイト作成者はデータベースの SQL 文を指定するだけでプログラミングに関する知識をあまり要求しない．例として，図 5 にサイト作成者が用意すべき PHP ファイルの内容を示す．

4.3　評　　価

ここでは，本システムの有用性を検証するための実験と評価について述べる．

4.3.1　実 験 環 境

コンピュータリテラシーの高い被験者，そうでない被験者を含めた 4 人に本システムを用いた Web ページを閲覧してもらった．Web ページに用いたデータは AmazonWebService から得た書籍データを利用し，各ユーザが興味があり，対象のデータ

表 6　評価に使用したディレクトリ

	ディレクトリ名（データ件数）
A	コンピュータ・インターネット / プログラミング / java (812)
B	コンピュータ・インターネット / プログラミング / C (1439)
C	歴史・地理 / 日本史 / 戦国・安土桃山 (672)
D	医学・薬学 / 病気別 / 消化器・循環器病 (1040)

数が一定値以上の末端ディレクトリを選択してもらい，表 6 に示すディレクトリを得た．

4.3.2　パラメーターを求める実験

ここでは，自己組織化マップを実行したときの商品推薦精度から最適な属性数と頻出単語の出現回数の閾値を決定する．属性閾値を 30 以下、30〜40、40〜50、50〜60 の 4 つになるような出現回数の閾値を決め、その分類結果に対する評価実験を行った．

評価方法はそれぞれの方法に対し，推薦の精度，実行速度を考慮した場合の妥当性の観点で 5 段階評価をしてもらった．1:非常に不適切，2:不適切，3:妥当，4:適切，5:非常に適切，として点数をつけてもらった．表 6 の各ディレクトリにおける実験結果を図 6 に示す．

実験結果より、属性数に関しては、属性数が多いほど細かい分類が可能であると考えられるが通常属性数 40 個を超えた辺りから商品データの特徴とは関係ない無駄な属性を含むケースが多い。また、属性数が少なすぎると大まかな分類しかできず、推薦の精度が落ちてしまうと考えられる。評価実験より、属性数は 40 個近辺が適切であると考えられる。

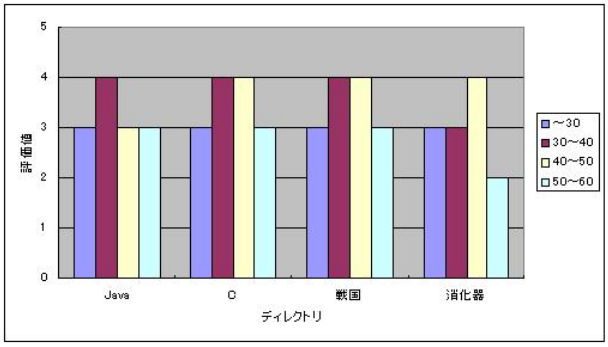

図 6　属性数とカテゴリ別の評価

4.3.3　評 価 実 験

実験では，被験者に以下の 4 つの方法で評価をしてもらった．

（1）　被験者に表 6 に示した既存の末端ディレクトリにおいて自分の興味のある商品データを探してもらった．

（2）　表 6 に示した既存の末端ディレクトリの下にさらに新たにディレクトリを作成した．このディレクトリは対象となる末端ディレクトリに含まれる全商品名を，形態素解析した結果の頻出単語をディレクトリ名とした．そのディレクトリ下には，各ディレクトリ名を商品名の一部に含む商品データを置いた．出現回数の閾値を被験者が自由に設定することでディレクトリ数を調整できるようにした．各末端ディレクトリにおける作成されたディレクトリ例を表 7 に示す．表 7 では例として各ディレクトリにおける上位 15 件の頻出単語を作成したディレクトリ名とする．そして，被験者に自分の興味のある商品データを探してもらった．

（3）　表 6 に示した既存の末端ディレクトリに含まれる商品データをランダムに表示し，被験者に自分の興味のある商品データを複数選択してもらった．選択された商品データに形態素解析を行い名詞部を抽出し，その単語を含む商品データから閲覧者に自分の興味のある商品データを探してもらった．

（4）　SOM を利用した本手法により，被験者に自分の興味のあるデータを探してもらった．

それぞれの方法に対し，自分の興味のあるデータを探す場合の妥当性の観点で 5 段階評価をしてもらった．1:非常に不適切，2:不適切，3:妥当，4:適切，5:非常に適切，として点数をつけてもらった．

各実験の結果を図 7 に示す．各実験における評価値の平均を表 8 に示す．商品探索における妥当性において，実験 4 の SOM を利用した手法が優れているのが分かる．実験 2 と実験 3 の比較ではディレクトリにより評価値の優劣違う．これは商品データ名の単語の出現頻度がデータ件数に対して非常に高いものが存在しているディレクトリでは，実験 2 のほうが優れた結果になっている．

表 8　評価値の平均

	平均値
実験 1	1.25
実験 2	3
実験 3	2.8
実験 4	4

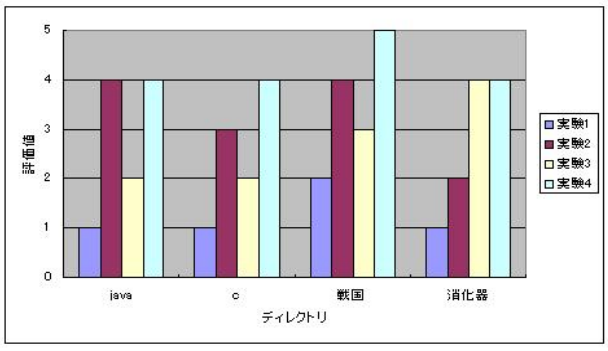

図 7　実験 1 による各ディレクトリの評価

4.4　検　　討

以下では，実験の際に被験者から得られた意見や現在考えられる本手法の問題点や検討すべき点について議論を行う．

4.4.1　ベクトルの生成方法

現段階でのベクトルの生成方法は形態素解析した結果の単語の出現回数のみを考慮している．全商品データ及び各商品にお

表 7　頻出単語によるディレクトリ

A のディレクトリ	出現頻度	B のディレクトリ	出現頻度	C のディレクトリ	出現頻度	D のディレクトリ	出現頻度
Java	303	C	418	戦国	171	器	210
プログラミング	120	言語	260	日本	51	消化	195
入門	98	プログラミング	256	武将	44	疾患	150
開発	41	入門	152	史	44	治療	115
基礎	31	C++	126	合戦	37	循環	103
JavaScript	30	基礎	58	時代	34	病	80
Web	25	編	51	秀吉	35	診断	75
アプリケーション	24	Visual	50	城	33	心臓	70
Eclipse	22	アルゴリズム	39	戦	31	管	59
編	22	C#	35	期	27	診療	50
対応	21	実践	35	大名	27	最新	47
オブジェクト	19	標準	26	信長	26	臨床	45
言語	16	オブジェクト	25	歴史	26	性	43
実践	14	指向	24	天下	25	マニュアル	30
設計	13	プログラム	24	国	20	心	30
活用	12	版	23	人	20	大腸	25
完全	12	C/C++	23	謎	19	病態	25
徹底	11	演習	22	研究	18	法	25
J2EE	10	構造	22	本能寺	15	肝炎	25

ける全データの出現確率において閾値を決め出現確率が非常に高い単語属性を使わないことにより属性数を削減することができる．属性の決定には商品名の単語だけを使用したが，値段や発売月日などを含める方法もある．また，ユーザの閲覧コストを定義し，コストに基づいたアプローチを組み合わせることで分類の向上ができると考えられる．

4.4.2　自己組織化マップの作成頻度

現在のシステム実装では、各末端ディレクトリ毎に自己組織化マップを用意している。商品データは頻繁に更新または無くなるため、商品データが変更される度に自己組織化マップを作成する必要がある。商品データの更新頻度が高くなるにつれサイト作成者の負担が大きくなってしまう。自己組織化マップの更新頻度も考慮する必要がある。

5. ま と め

本稿では，ショッピングサイトやオークションサイトに見られる階層的なディレクトリ構造における末端ディレクトリにおいて，自己組織化マップを利用して自動的に再分類をする手法を提案した．評価実験を通して，サイト作成者が新しいディレクトリ作成を考慮する必要なしに，閲覧者が自分の嗜好の商品データを得ることが出来，データ検索が容易になったことを示した．今後は 4.4 節の検討事項にあげた点の改善を中心に研究を進めていく予定である．

文　献

[1] Amazon.co.jp: http://www.amazon.co.jp/

[2] 楽天市場: http://www.rakuten.co.jp/

[3] Yahoo!オークション: http://auctions.yahoo.co.jp/

[4] 茶筅: http://chasen.naist.jp/hiki/ChaSen/

[5] Chakrabarti, K., Chaudhur, S. and won Hwang, S, “Automatic Categorization of Query Results”, *Proceedings of ACM SIGMOD ’04 International Conference on Management of Data*, pp. 755-766, 2004

[6] Nemoto, J. and Toyama, M, “Automated SuperSQL Query Formulation Based on Statistical Characteristics of Data”, *Proceedings of DEXA ’05 International Conference on Database and Expert Systems Applications*, 2005

[7] 加藤由花, 川口賢二, 箱崎勝也, “オンラインショッピングを対象とした正確性と意外性のバランスを考慮したリコメンダシステム” , 情報処理学会論文誌:データベース, *Vol.46, No.SIG 13(TOD27), pp.53-64(2005)*

[8] 波多野賢治, 佐野綾一, 段一為, 田中克己, “自己組織化マップと検索エンジンを用いた Web 文書の分類ビュー機構” , 情報処理学会論文誌:データベース, *Vol.40, No.SIG 3(TOD1), pp.47-59(1999)*