DEWS2004 I-8-06

電子メールの引用関係を用いた話題の検出粒度に関する検討

清水　琢磨†　　上島　紳一†

† 関西大学大学院総合情報学研究科
〒 569-1095 大阪府高槻市霊仙寺町 2-1-1

あらまし　近年，様々な技術情報の共有や相互扶助を目的とした技術系メーリングリストが数多く存在している．このようなメーリングリストでは，1 つの話題は複数の人によって議論されるため，話題は複数の電子メールに分散している．そのため，メーリングリストからある話題に関する情報を検索する場合，回答を複数の電子メールから探す必要がある．本稿では，メーリングリストから 1 つの話題を構成する電子メールの集合を抽出するために，電子メール間の引用関係を利用する．引用関係の特定するために，電子メールのヘッダを用いる手法と，本文の内容を用いる手法を論じ，引用関係をもとに，電子メールを木構造に再構成する．更に，再構成された木構造から，過不足なく話題が含まれる最適な部分経路を探すための，得点付け手法にについて検討する．最後に，本手法をプログラミング言語やツールを題材とするいくつかのメーリングリストに適用し検証する．

キーワード　Web とインターネット，情報検索，情報統合，検出粒度，XML

Detecting relevant granules of emails in Mailling List using quotation relations

Takuma SHIMIZU† and Shinichi UESHIMA†

† Graduate School of Infomations,Kansai University
2-1-1 Ryozenji,Takatuki,Osaka,569-1095,Japan

Abstract　Recently, mailing lists (ML) concerning specific technical issues play a role of shared bulltin boards for infomation exchange among engineers. In such ML's, one single theme is often discussed or commented by many different users, and relevant information is scattered and found on different emails. In case a user searches for answer to his/her question, he/she needs to consult these scattered emails, accordingly. In this paper, we give a method to derive a chain of emails that may have information relevant to a given question, by using quotation relations among them. For this purpose, we first discuss on specification of quotaion relations among emails in a thread, using both email headers and their bodies, and try to rebuild a forest of emails in threads. Next we show a scoring method to derive a subchain of emails of an appropriate granularity as retrieval results. We verify our approach by applying it to ML's , such as those of programming language and tools.

Key words　Web and Internet, infomation retrieval, infomation integration, granules, XML

1. はじめに

近年，プログラム言語やツールに関する技術情報の共有や相互扶助を目的とした技術系メーリングリストと呼ばれるコミュニティが数多く存在している．技術系メーリングリストでは通常，論題とする技術に関して専門的な知識を有する人が主催者となり，交わされる話題も最新のトピックや障害の解決法など貴重なものが多い．しかし，メーリングリストのようなコミュニケーションの中に潜在している情報は，構造化されておらず十分に活用しきれていない．このようなコミュニケーションの中に埋もれた情報を活用するための，コミュニティウェアの研究が盛んに行われている [1] [2]．しかし，その多くはあらかじめ設定されたルールやパターンを用いて特定の情報を抽出し，知識の獲得を行っているため，決められた情報以外はコミュニケーションの中に埋もれたままになっている．

あらかじめルールを設定せずにコミュニケーションの中から情報を検索するためには，Namazu [3] に代表される全文検索システムを用いる．しかし，全文検索システムではファイルを単位として利用しており，メーリングリストから検出される結果はメーリングリスト内の 1 通のメールを単位としている．本稿では，メーリングリストを対象に検索する際，メールを単位とするのではなく，メーリングリストに埋もれている情報を単

位として検出する手法について検討を行う．

メーリングリストでは，複数の人の間で交わされるコミュニケーションの中に情報が隠れているため，情報は複数のメールに分散している．そこで，メール間の引用関係を用いて複数のメールを集約し木構造に再構成を行う．まとめられたメールの集合は，特定の話題に関するメールの集合であり，その中に埋もれる情報は経路として表現でき，キーワードに対する回答を効率的に得ることができる可能性がある．メーリングリストからキーワードに関する情報を効率的に得ることができれば，貴重な情報をメーリングリストに埋没させることなく有効活用することができる．

本稿 2 節では，メーリングリストのモデルに関して述べ，3 節では引用関係の特定手法，4 節で部分経路の抽出手法ついて説明し，5 節でプロトタイプを作成し検証を行う．

2. メーリングリストのモデル

技術系メーリングリストのコミュニティで交わされるメールは，話題を提供するメールと，他のメールに対する返信を返すメールの 2 種類に分類することができる．話題を提供するメールとは，メーリングリストに対して新規に質問や連絡，ニュースの紹介などを行うメールのことである．他のメールに対する返信は，既存の話題を受けて投稿されるメールである．このことから，技術系メーリングリストでは交わされるすべてのメールは，メーリングリスト内のいずれかの話題に属している．

また，技術系メーリングリストに埋もれている情報は，コミュニケーションの成果であり，複数のメールに分散している．そのため，情報を抽出するには，複数のメールを集約しなければならない．(図 1) 情報が分散しているメールは互いに関連性の

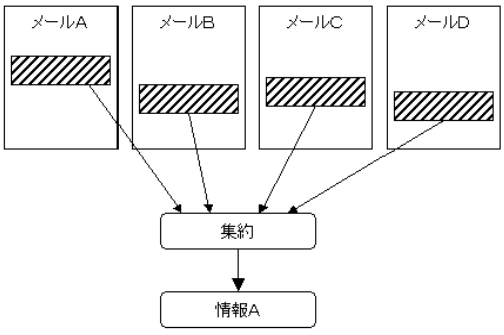

図 1　情報の分散

高いメールであり，それは同じ話題に属していると考えられる．

技術系メーリングリストから情報を抽出するためにメールを話題毎に集約する場合，技術系メーリングリスト内では同時に複数の話題が議論されるため，時間順で並べるだけでは話題を集約することができない．そこで，メールを話題毎に集約するためにメールの引用関係を利用する．ここで引用関係とは，他のメールに対する返事を投稿するときの返信の関係とし，返信の対象となったメール 1 通を引用元メールと定義する．引用関係によって関連付けられたメールは，同じ話題を扱ったメールであると考えることができ，引用関係によってメールを話題毎に集約する．引用関係によって集約された話題は，図 2 のように新規の話題を提供したメールを根，他のメールに対する返信のメールを節もしくは葉とした木構造に再構成できる．

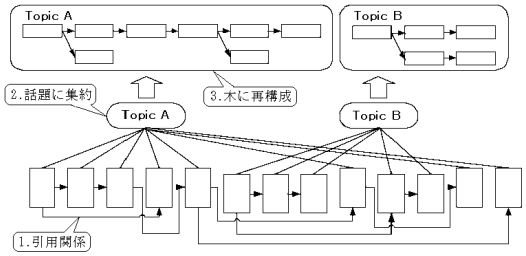

図 2　メーリングリストから木の再構成

また，メーリングリストの構造を BNF で表現したものを表 1 に示す．

表 1　メーリングリストの構造

```
<ML>       :=  <Thread> | <ML>,<Thread>
<Thread>   :=  <RootMail> | <RootMail>,<NodeMail>
<RootMail> :=  <Mail>
<NodeMail> :=  <Mail>,<NodeMail>
```

次に，話題に埋もれている情報を抽出する．

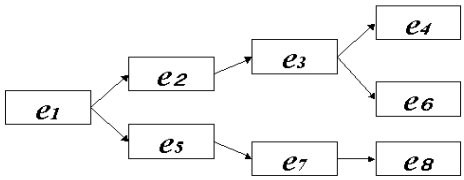

図 3　話題の経路

話題を構成する木は，図 3 にあるメール e_1 と e_2 のような親子関係があるメールでは関連性が高く，情報が埋もれていると考えられる．逆に，e_6 と e_7 のように直接の親子関係が無いものは関連性が低く，情報が埋もれているとは考えにくい．このことから，話題に埋もれている情報は根から葉にまでの経路に埋もれており，情報の埋もれている部分を部分経路と呼ぶ．

部分経路は，その中で最も葉に近いメール e_i を基点とし，部分経路の長さ n を用いて $e_i(n)$ と表す．例えば，図 3 の e_3 を基点とする部分経路は，長さ 2 の部分経路 $e_3(2) = \{e_1, e_2, e_3\}$，長さ 1 の部分経路 $e_3(1) = \{e_2, e_3\}$，長さ 0 の部分経路 $e_3(0) = \{e_3\}$ の 3 種類存在する．このときの部分経路の長さを粒度と呼ぶ．

3. 引用関係の特定

メール間の引用関係を特定する方法として，メールのヘッダ情報を用いる方法とメールの本文の内容から特定する方法を検討する．

3.1 ヘッダ情報を用いた引用関係の特定手法

メールの引用関係は，メールのヘッダ情報の In-Reply-To, References フィールドを用いて特定することができる．しかし，In-Reply-To，References フィールドは，RFC2822 では必須とされておらず，メーラーによってヘッダ情報の扱いに差がある．また，利用者がメールに対する返信を行うときにメーラーの返信機能を使わずに，新規メールとして作成した場合にはメーラーは引用情報を付加することが出来ない．更に，本来引用すべきメールとは異なるメールに対して返信機能を使い作成した場合や，新規メールを返信を用いて作成した場合には間違えた引用情報を付加することになる．

これらの問題点から，ヘッダ情報では引用関係の特定を必ずしも保障できない．そこで，Perl のモジュールのひとつである DBI を論題とした DBI 日本語メーリングリスト (dbi-japan) [4] のメール 100 通と，Java3D を論題とした Java3Djp [5] から，メール 100 通を用いて検証を行った．実験では，正解は人手によって特定し，引用の特定方法は In-Reply-To に記述された Message-ID を用いて特定し，In-Reply-To が無ければ References に列挙された最も新しい Message-ID を用いて特定する．その結果を表 2 に示す．

表 2　ヘッダ情報を用いた引用関係の特定

	正解	返信に新規	新規に返信	引用元間違い
dbi-japan	95	1	4	0
Java3Djp	69	19	4	8

表 2 では，‘ 正解 ’は引用を正しく特定できたもの，‘ 返信に新規 ’は内容は返信であるが新規のメールで送信したもの，‘ 新規に返信 ’は新規のメールにもかかわらず返信で送信したもの，‘ 引用元間違い ’は引用元を間違えたものである．

実験の結果，dbi-japan では高い精度で引用関係を特定することができたが，Java3Djp では精度は低くメーリングリスト毎にばらつきが大きいことが分かる．

3.2 本文を用いた引用関係の特定手法

次に，本文を用いた引用関係の特定を考える．一般にメールを作成する場合，内容のまとまりを空白のみの行で区切り，メールに返事を返す場合には，返事を返す部分を引用符号を用いて引用する．そこで，本文を空白のみの行で分割し，更に引用符号の付いている部分と付いていない部分とで分割し，各々を段落と定義する．

本文からの引用関係の特定は，まず各段落の引用元を特定し，次に段落毎の引用元を集計することでメールの引用関係を特定する．段落毎に引用元を特定する手順は，引用符の付いた段落と比較対象の段落を一定の文字数毎に分割し，分割された文字の共起度をもとに特定する．

このとき，引用元のメールは時間順に並べてた時に，直近から数通のメール内にあることが予想されるため，共起度を計算するメールは直近の数通のみを対象とする．対象とするメールの件数を求めるために，DBI と Java3D のメーリングリストの引用元の正解例を用いて何通前のメールを引用しているのか調べ，結果を図 4 に示す．その結果，DBI では平均して 2.96 通前のメールを引用しており Java3D では平均 5.46 通前のメールを引用していた．累積確率は，DBI と Java3D どちらも約 10 通のメールで 90 %に，約 20 通前のメールで 98 %に達し，20 通以前のメールを引用したものは Java3Djp にあった 21 通前と，63 通前のメールを引用していた 2 通だけであった．

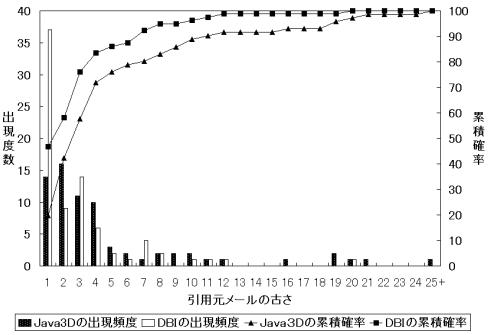

図 4　引用元メールの古さ

この結果から，引用元のメールの候補は直前 20 通のメールを対象とし，また各段落を 3-グラムに分割，閾値は 0.7 としてヘッダからの引用関係の特定とと同様に dbi-japan と Java3Djp のデータを用いて検証を行った．(表 3)

表 3　本文を用いた引用関係の特定

	正解	返信に新規	新規に返信	引用元間違い
dbi-japan	90	2	2	6
Java3Djp	85	0	1	16

実験の結果，ヘッダを用いた引用関係の特定とくらべ Java3Djp では特定の精度に大きな向上が見られた反面，dbi-japan では精度の低下が起きた．しかし，間違いの中で引用元間違いの多くは，本来は木構造の兄弟の関係になるメールを引用関係と特定した間違いであり，話題の展開を考えたとき，重大な間違いではない．そのため，dbi-japan における引用関係を特定する精度の低下の影響は，表に現れる数値より小さいと考えられる．

この結果から，本文を用いる引用関係の特定は，メーリングリストの違いによる精度のばらつきが少なく，本文を用いる引用関係の特定の手法は有効であると考えられる．また，ヘッダからの特定ではメールを単位とした引用関係の特定しか出来ないが，本文を用いた特定では段落毎に特定することができるため，情報を抽出する際に利用することができる．

4. 部分経路の抽出手法

技術系メーリングリストから情報を抽出するしようとするとき，各メールにスコアを付けて比較して抽出しても，情報は断片化されて潜在しているため適切な回答とはならない．情報を抽出するためには複数のメールを集約した部分経路で検出する必要があるため，部分経路に対してスコアを付けて比較することで，質問に対する最適な粒度での検出を目指す．部分経路に対するスコアは，各メールに対するスコアと，話題毎に集約された木構造を用いてスコア付けを行う．

4.1 メールへのスコア付け手法

4.1.1 メールへのスコア付けの基準

メールへのスコアは，3 つの要素をもとにスコア付けを行う．

- キーワードの出現頻度
- 引用元メールのスコア
- メールの投稿者

キーワードを何度も含むメールは，キーワードがそのメールの重要な位置を占めている語だと考えられる．引用箇所にキーワードが含まれていた場合，引用元メールとの関連性が高いと考えられるため，引用元メールのスコアを参考にする．また，メーリングリストに対して何度も投稿をする人は，何度も回答を投稿する人であり，メーリングリストの論題に対して詳しい人であるという考えられる．論題に関して詳しい人の投稿するメールは，的確な指摘を行っていると考えられるため，メーリングリストへの投稿数をメールへのスコア付けに利用する．

4.1.2 メールへのスコア付け手順

メールへのスコア付けは以下の手順とする．

（1） メールの各段落についてスコアを付ける

（2） メール内の段落のスコアを合計する

（3） メールにキーワードが含まれる場合，送信者の評価値を付加する

各段落のスコア $Pg_{(i)}$ は，キーワードが含まれている場合にスコアを加算し，値はその段落が引用箇所か通常の段落かによって変える．もし通常の段落である場合，加算される値は定数 c とし，引用箇所であれば，引用元メールのスコア $Score(e_{parent})$ に一定の重み ω_{quote} を掛けた値を用いる．メール内の各段落にスコア付けを行ったあと，各段落のスコアを合計し，メール内にキーワードが含まれていた場合，更にメールの投稿者のスコアを加算する．メールの投稿者に対するスコアは，投稿者のメーリングリストへの投稿回数をもとにした重み $\omega_{send}(sender)$ を掛けて求める．つまり，メールに対するスコアは，

$$Pg_{(i)} = \begin{cases} Score(e_{parent}) * \omega_{quote} & \text{(引用文)} \\ c & \text{(引用文以外)} \end{cases} \quad (1)$$

$$Score(e_{current}) = \sum_{i=1}^{n} Pg_{(i)} + \omega_{send}(sender) \quad (2)$$

となる．

また，スコア付けに引用元メールのスコアを利用するため，古いメールから順にスコア付けする必要がある．

4.2 部分経路へのスコア付け手法

4.2.1 部分経路へのスコア付け基準

個々のメールに対するスコアを付けた後，部分経路にスコアを付ける．このとき，メーリングリスト内のすべてのメールの組み合わせに対してスコアを付けることは計算量が大きくなりすぎる．また，話題を同じにする木構造の中のメールにも，関連性の深いメールと関連性の薄いメールが存在している．同じ話題の中でも直接の引用関係のあるメールは，直接の引用関係が無いメールよりもより関連性が高いと考えられる．

話題の木構造の中で直接の関連性のあるメールの集合は，根から葉に至る経路となりその経路から部分経路を抽出してスコア付けを行う．この時，すべての話題に対してスコア付けを行うのではなく，各メールにスコアを付けた際に，スコアの付いたメールを含む話題のみを対象とする．

部分経路に対するスコア付けには，下記の項目を基準とする．

- 部分経路を構成するの各メールのスコアの合計
- 部分経路の粒度

各メールのスコアの合計は，キーワードに対する集合の評価値であり，部分経路の粒度は検出結果の情報量に対して過不足が無いようにするための重みである．粒度に対する重み付けをしなければ，粒度が大きいものほど評価値が大きくなり，粒度が大きいものは余分な情報を含む．また，各メールのスコアの合計を粒度で割り，平均を求めると最も評価値の大きなメール 1 通が選ばれることになるため情報を抽出することができない．

4.2.2 部分経路に対するスコア付け手順

部分経路に対するスコア付けの手順を以下に示す．

（1） キーワードを含むメールがある話題の木構造を求める

（2） 各木構造の根から葉へのすべての経路を求める

（3） 経路から連続する部分経路を求める

（4） 部分経路に対してスコアを付ける

まず，キーワードを含むメールが属する話題の他のメールには，キーワードが含まれていなくてもキーワードに関する事柄が記述されている可能性がある．そこで，キーワードの含まれるメールが存在する話題を求め，スコア付けを行う対象とする．また，話題に存在するすべての部分経路にスコアを付けるのではなく，直接の引用関係が存在する部分経路に対してのみスコア付けを行う．部分経路へのスコア $Path(e_a(k))$ は，部分経路に含まれる各メールのスコアを合計し，重み ω_{path} を掛けて求める．部分経路へのスコアを式で表すと，

$$Path(e_a(k)) = \sum_{l \in e_a(k)} Score(e_l) * \omega_{path}(k) \quad (3)$$

となる．

4.3 部分経路の検出アルゴリズム

部分経路の抽出は，話題を集約した木構造内の任意の節を基点とする任意の粒度の部分経路の中の最も高いスコアと，その基点の子を新たな基点として求めた最も高いスコアを再帰的に比較し最も高いスコアの付いた経路を抽出する．

部分経路抽出アルゴリズムを図 5 に示す．なお，図中の $MailPoint$ は部分経路を抽出する基点となるメール，$PathArray$ は部分経路を構成するメールの集合である．$ParentArray, Children$ はそれぞれメールの親と子を記録した配列であり，$MaxScoreValue, MaxPathArray$ はスコアの最大値とそのときの経路を記録する．また，$calc(PathArray)$ は部分経路へのスコア付けを行う．

まず $Scoring$ では，部分経路を抽出する際の基点となるメールを引数として受け取り，経路を記録する配列 $PathArray$ に追加する．この配列が部分経路を表しており，$calc(PathArray)$ によって部分経路に対するスコア $ScoreValue$ を求める．部分経路に対するスコアと，その時点のスコアの最大値 $MaxScoreValue$

```
Function Scoring(MailPoint)
    i = MailPoint
    while(i)
        push(PathArray,i)
        ScoreValue := calc(PathArray)
        if(ScoreValue > MaxScoreValue)
            MaxScoreValue := ScoreValue
            MaxPathArray := PathArray
        end
        i = ParentArray[i]
    end
    foreach Child (Children[MailPoint])
        (ScoreValue,PathArray) = Scoring(Child)
        if(ScoreValue > MaxScoreValue)
            MaxScoreValue := ScoreValue
            MaxPathArray := PathArray
        end
    end
    return (MaxScoreValue,MaxPathArray)
end
```

図 5　部分経路の抽出アルゴリズム

と比較し，求めたスコアが最大値を超える場合，最大値とそのときの部分経路 $MaxPathArray$ を書き換える．その後，基点となるメールの親にあたるメールを探し，親にあたるメールが存在しなくなるまで $PathArray$ にメールを追加し，同様の手順を繰り返す．この手順により，引数によって受け取ったメールを基点とする部分経路の中で最大となる部分経路を特定することができる．

次に，基点となるメールの子にあたるメールを探し，その子を基点として $Scoring$ を再帰的に呼び出す．その子を基点としたスコアの最大値と現時点でのスコアの最大値を比較し，子を基点としたスコアが最大値を超える場合，最大値とそのときの部分経路 $MaxPathArray$ を書き換える．同様の手順を，$MailPoint$ のすべての子に対して行う．

このような手順でスコア付けを行うことで，最終的に得られる結果は，引数として最初に渡されたメールとその子孫を基点とした部分経路の中で最もスコアの高い部分経路となる．そのため，最初に基点として渡す引数を木構造の根となるメールとすることで，木構造に含まれるすべての部分経路の中の最もスコアの高い部分経路を抽出することができる．

5. 検　　討

本稿の手法を用いてプロトタイプシステムを作成し，Java3Djp のデータを用いてその有効性を検討する．プロトタイプシステムは，RedHat8 上で Perl5.8.2 と MySQL を用いて作成した．

5.1 プロトタイプシステム

プロトタイプシステムでは，各メールのデータを特定された引用関係とともに XML に変換し，利用した．引用関係は本文から特定されたものを用い，送信者の情報や引用関係は MySQL に保存し利用する．スコア付けを行う際に用いる重み ω_{quote} には $\frac{1}{\sqrt{Score_{parent}}}$ を，定数 c には 1 を，重み $\omega_{send}(Sender)$ には投稿回数の $\frac{1}{50}$ を用いた．重み ω_{path} には，経験的に質問－回答－補足といった形での展開が多いことから，平均 3，分散 2 の正規分布とする．また，類似した部分経路が多く出力されることを防ぐために，各スレッド毎に最もスコアの高い 1 件を出力する．

図 6　プロトタイプシステムの GUI

プロトタイプの GUI は図 6 の形を持ち，上部にキーワードを入力することで下部左側に検索結果が表示される．検索結果には，スレッド番号，抽出したファイル名，スコアが表示され，スレッド番号をクリックすれば下部左側にその一覧が，ファイル名をクリックすれば下部右側にその内容が表示される．

5.2 評　　価

全文検索システムの Namazu との比較実験を行う．なお，Namazu 2.0.12 for Win32 を用い，GUI として search-s for Namazu 0.9.2 を利用した．実験では Java3Djp のデータを対象に，いくつかのキーワードを Namazu と本稿で作成したプロトタイプを用いて検索した．評価項目は，下記の項目とした．

- 回答への到達の容易性
- 回答への満足度

回答への到達の容易性は，キーワードを入力してから満足する回答を得られるまでにかかった時間と，回答を得るまでに読んだ文書数をもとに判断する．回答への満足度は，被験者に，最終的に得られた回答の満足度を 5 段階で評価してもらった．キーワードは，ランダムに選択した．

回答を得るまでにかかった時間と文書数を図 7 に示す．

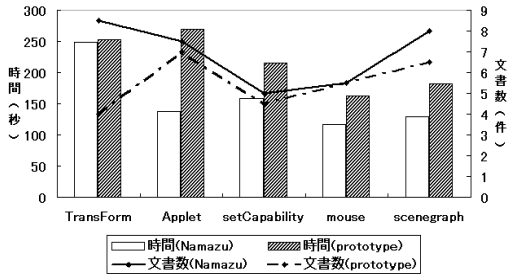

図 7　回答を得るまでの容易性

プロトタイプでは，キーワードによって回答を得るまでの時間にばらつきがあり，最大で約 2 倍，平均して約 1.4 倍の時間がかかった．時間が多くかかる原因として，複数のデータを表示するためデータ量が増え，有用な情報かを判断する情報の取捨選択が困難になったと考えられる．この点は，あいさつや署

名などの不要な情報を取り除き，GUI を改善することによって解決できる可能性がある．また，回答を得るまでに開いた文書数では平均で 1.4 件少なくなり，キーワードに対する回答を少ないステップ数で得られた．次に，回答への満足度を図 8 に示す．

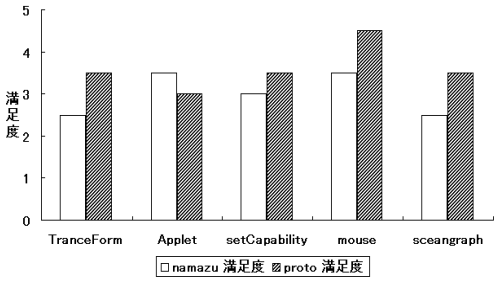

図 8　回答の満足度

回答に対する満足度は，平均で約 0.6 ポイントの向上が見られた．この結果は，情報を単位として検出し一度に提示したことによって前後の話のつながりが理解しやすくなったためだと考えられる．本稿の手法では，情報の取捨選択にかかる時間に問題があることが分かったが，情報までのステップ数や理解度が向上していることが確認できた．

5.3　部分経路の粒度に対する重みの検討

部分経路の粒度に対する重みに対して正規分布を用いたが，その根拠は経験をもとにしたもので，根拠に乏しい．そのため，部分経路の粒度に対する重みにユーザからのフィードバックを利用する．フィードバックは，検索結果に対して'too short','short','just','long','too long' の 5 段階とし，検索結果の粒度毎に集計を行った．(表 4)

表 4　ユーザからのフィードバック

size	too short	short	just	long	too long
2	0	1	2	0	0
3	1	11	23	3	0
4	0	1	3	0	0
5	0	0	0	1	0
6	0	0	1	3	0

集計の結果から，'too short' では $size+2$ を，'short' では $size+1$ を，'long' では $size-1$ を，'too long' では $size-2$ を最適な粒度だと仮定した．フィードバックから最適な粒度の分布を求め，最大の値が同じになるように正規化したものと平均 3，分散 2 の正規分布と比較する．

図 9 において，粒度 3 以上の範囲でフィードバックから求めた分布との相関をとると，0.985 と非常に高い値となった．この結果から，粒度 3 以上の範囲では平均 3，分散 2 の正規分布は粒度に対する重みとして妥当なものだと考えられる．

しかし，粒度 1,2 の場合には，正規分布を用いた重みはフィードバックから求めた分布を大きく上回る結果となった．このとき，ある経路の中に粒度 3 の部分経路が存在する場合，粒度 3 の部分経路は粒度 1,2 の部分経路を含むため，構成するメールのスコア合計は粒度 3 の部分経路より小さくなる．そのため，粒度 1,2 の部分経路に対するスコアは粒度 3 の部分経路のスコアを超えることはない．また，粒度 2 に対する重みが小さすぎると，粒度 2 が最大の粒度となる経路は他の経路に埋もれてしまうため，粒度 2 に対する重みはある程度大きなものである必要がある．

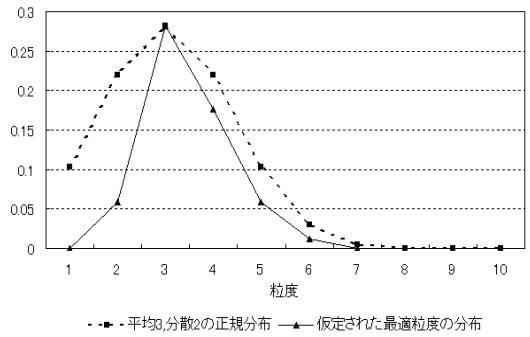

図 9　仮定された最適な粒度と部分経路に対する重みの比較

このことから，粒度に対する重みとして平均 3, 分散 2 の正規分布は妥当なものだと考えられる．

文　献

[1] 梅木秀雄，"コミュニケーションに埋もれた知識を活用するコミュニティウェア，" 情報処理学会会誌，Vol.43，No.10，pp1085-1092，2002

[2] 佐藤円，佐藤理史，篠田陽一，"電子ニュースのダイジェスト自動生成，" 情報処理学会論文誌，Vol.36，No.10，pp.2371-2379，1995

[3] Namazu，http://namazu.org/

[4] dbi-japan，http://member.nifty.ne.jp/hippo2000/dbi/dbi_japan.htm

[5] Java3Djp，http://www.antun.net/tips/java3d/ML/