新鮮情報検索における情報の新鮮さにもとづくスコア計算

佐藤 永欣† 上原 稔‡ 酒井 義文*
E-mail: jju@toyonet.toyo.ac.jp, uehara@cs.toyo.ac.jp, sakai@biochem.tohoku.ac.jp
†東洋大学植物機能研究センター ‡東洋大学工学部情報工学科
*東北大学大学院農学研究科

集中型アーキテクチャに基づくサーチエンジンでは新鮮な情報検索が困難である。そこで我々は分散型アーキテクチャに基づくイントラネット向けサーチエンジンである協調サーチエンジン (Cooperative Search Engine, CSE) を開発した。CSE は組織内の各 Web サイトでボトムアップにインデックスを更新するため、組織全体の文書のインデックスを数分以内に更新可能である。このように新鮮な文書の検索は可能となったが、スコア計算には文書や情報の新鮮さはまったく考慮されていなかった。そこで、我々は文書内容の新鮮さを考慮した FTF·IDF 法 (Fresh Term Frequency multiplied by Inverse Document Frequency) をもちいる新鮮情報検索の概念を提案し、ランダムに生成した文書による FTF·IDF 法の評価を行った。本論文では、実際の Web 日記を用いた FTF·IDF 法の評価を述べる。

Freshness based Scoring in Fresh Information Retreival

Nobuyoshi Sato†, Minoru Uehara‡, Yoshifumi Sakai*
†Plant Regulation Research Center, Toyo University
‡Department of Information and Computer Sciences, Toyo University
*Graduate School of Agricultural Science, Tohoku University

It is difficult for centralized search engines to retrieve fresh information. So, we have developed a distributed search engine for intranets, Cooperative Search Engine(CSE). CSE realizes fast index updating of all Web documents entire of an organization because each Web sites update indices in bottom up fasion. Although Fresh documents can be retrieved in CSE, however, freshness of documents was not considered on score calclation. So, we proposed FTF·IDF method (Fresh Term Frequency multiplied by Inverse Document Frequency) which considers freshness of doucment contents, and we evaluated FTF·IDF method by radomly generated documents. In this paper, we will describe an evaluation of it in particular Web diary.

1 はじめに

近年、組織内に蓄積された情報の活用が課題になるなど、組織内の情報検索が重要になってきた。組織内の情報検索の大きな特徴として、新鮮な情報を検索するという要求があげられる。ビジネスなどの分野では担当者が肝心な情報を共有していなかった、必要な時期までに情報を知ることができなかったことなどによる機械喪失は問題視されている。Web のような不定型の情報を検索するためには一般的にサーチエンジンが用いられる。組織の規模が大きくなると集中型サーチエンジンでは新鮮な情報検索が困難となる。これは、集中型サーチエンジンでは、ロボットで文書を収集し、インデックスを更新するまでに長い時間がかかるためである。したがって、新鮮な情報検索には分散して文書収集、インデックスの作成、更新が可能な分散サーチエンジンが適している。

そこで、我々は、分散型アーキテクチャに基づく協調サーチエンジン (Cooperative Search Engine, CSE) を開発した [1]。CSE は、各 Web サーバに配置された局所サーチエンジンをメタサーチサーバで統合した大域的サーチエンジンである。CSE はボトムアップで文書収集、インデックス作成等の更新作業を行うので、CSE は更新に関してスケーラブルであり、規模にかかわらず短時間でインデックスを更新できる。また、検索が遅いという問題点があったが、検索結果をキャッシュするなどの高速化技法によりある程度の検索時のスケーラビリティを実現できた。

我々は以前、新鮮な情報を容易に発見するため、新鮮情報検索を提案した。新鮮情報検索では、文書内様の新鮮さに基づく FTF·IDF 法 (Fresh Term Frequency multiplied by Inverse Document Frequency) によってスコアを計算することで新鮮と考えられる文書がより上位にランキングされる。また、ランダムに生成した文書による FTF·IDF 法の評価を行った。FTF·IDF 法によるスコア計算は、TF·IDF 法における語の重みの他に文書の内容が現れてから/更新されてから経過した

時間を考慮する。すなわち、出現・更新されてから時間が経過するごとにスコアが減じられる。ランダムな語をランダムに生成した文書でのFTF·IDF法の有効性は確認できたが、これはFTF·IDF法が有用である可能性を示すに過ぎない。そこで、本論文では、実際のWeb日記を用いてFTF·IDFの評価を行う。

本論文の構成は以下のとおりである。第2章では、新鮮情報検索について述べる。続いて、第3章でCSEの構成と動作について述べ、第4章でFTF·IDF法について述べる。第5章で実際のWeb日記を用いたFTF·IDF法の評価について述べ、最後にまとめと今後の課題を述べる。

2 新鮮情報検索

新鮮情報検索とは、時制情報検索 (Temporal Information Retrieval) の一種である。時制情報検索とは、時間的に変化する情報 (Time-Varying information) を検索することである。文書は作成されてから何度も改変され、時間的に変化する。

文書とデータの違いはあるが、時制情報検索の理論的背景は時制データベースに求められる。時制データベースは、J.F.Allenの提唱した時区間論理 (temporal interval logic)[8] に基づく時区間指定を可能とするデータベースである。よって、時制情報検索も時区間指定ができる必要がある。ここでは、時制情報検索を文書の存在を時間で検索することと定義する。一方、文書の内容の時間に関する記述を検索することは時制情報検索ではない。

文書の存在期間は文書モデルに依存する。文書モデルには不変モデル、可変モデルの2種類がある。不変モデルでは、文書の存在期間は情報の存在期間と等しいとする。文書内容が変更されると異なる情報と考え、文書は変更のたびに消滅・生成が行われると考える。可変モデルでは、文書は何度修正されても同じ文書であると考える。Webの文書は少しずつ変更されることが多いが、同じURLで示される文書が全く違う内容の文書に更新されることもあるため、新鮮情報検索では不変モデルと可変モデルを組み合わせて使う。すなわち、小規模な文書内容の変更には可変モデルを、大規模な更新には文書が新しく作りなおされたと判断して不変モデルを適用する。

文書の生成、修正、および消滅の時点にはいくつかの基準が考えられる。ある人が時刻 t_1 にある情報を持っていて、時刻 t_2 にその情報を記述した文書を作成し、時刻 t_3 に公開し、時刻 t_4 にサーチエンジンに登録されたとする。t_1、t_2 はシステム外の要因で決定される。Web上に文書が存在した時刻は t_3 であるが、全てのWebサーバの時計が正しいとは限らないため信用できない。理想的には $t_3 = t_4$ であることが望ましいが、集中型サーチエンジンでは実現は困難である。CSEにおいては、t_3 と t_4 の差は非常に小さいため、t_4 を使って理想的な時制検索を行うことができる。

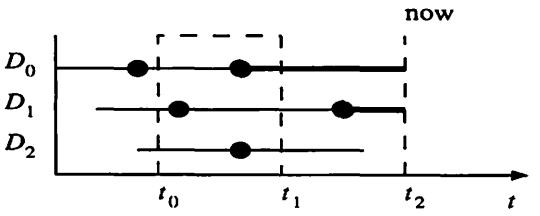

Fig. 1. 新鮮情報検索

完全な時制情報検索を実現するには文書変更の履歴をすべて保持する必要があるが、巨大なストレージサイズが必要となる。そこで、実用的な時制情報検索として、最終更新文書のみを対象とする新鮮情報検索を考える。新鮮情報検索では、文書の最終版のみ保持すればよいため、巨大なストレージサイズは必要ない。新鮮情報検索は、ある現在の内容を持つ現在の文書がある時区間に存在したことを検索することである。また、検索される文書が現在のものに制限される以外は、時間情報検索の機能をすべて持つ。

Fig.1に新鮮情報検索を示す。3つの文書 D_0, D_1, D_2 が存在し、黒丸は変更イベントを表す。太線は文書が語 w を含んでいる状態を示す。非時制情報検索は現在時刻 $now = t_2$ に存在する文書のみを検索できる。Fig.1では文書 D_0、D_1 が新鮮情報検索の対象に該当する。D_2 は削除されているため対象とならない。新鮮情報検索では、現在も存在している文書であれば過去のバージョンであっても検索できる。非時制情報検索は新鮮情報検索で現在の時刻 t_2 を指定した場合と考えられる。t_1 を指定した場合は、t_1 の時点では D_1 は w を含んでいないので、D_0 のみが得られる。

3 協調サーチエンジン

CSEはFig.2に示されるような以下の部品から構成される。

- Location Server (LS): LSはFK (Forward Knowledge) を一元管理する。LSはFKを用いてクエリに基づくサイト選択を行う。LSはサイト選択キャッシュ(Site selection Cache, SC) を持つ。
- Cache Server (CS): CSは、サイト選択の結果と検索結果をキャッシュするサーバである。検索結果をキャッシュすることで継続検索（次の10件の検索）を実現する。また、CSは後述のLMSEを

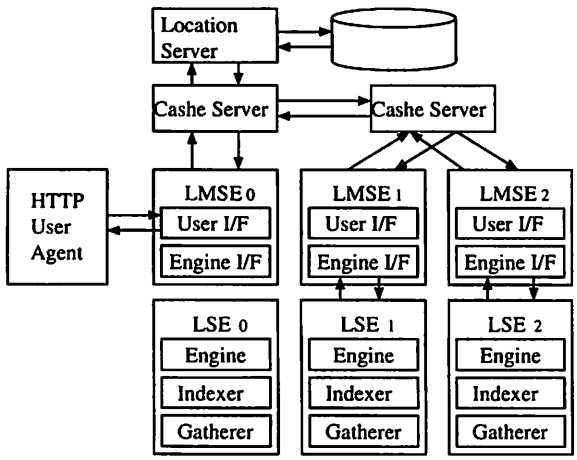

Fig. 2. CSE の概要

並列に呼び出し、並列検索を行う。CS は検索結果キャッシュ(Retrieval Cache, RC) とサイト選択キャッシュを持つ。CS のサイト選択キャッシュは LS のサイト選択キャッシュの部分的、不完全なコピーである。CS のサイト選択キャッシュに検索によって生じた変化は LS のサイト選択キャッシュに反映される。

- Local Meta Search Engine (LMSE): LMSE は、ユーザからの要求を受け付け CS に転送したり (Fig.2 の User I/F)、後述の LSE を呼び出し局所的な検索をしたり (Fig.2 の Engine I/F) する。LSE の差異を吸収するメタサーチエンジンである。
- Local Search Engine(LSE): LSE は局所的な文書収集 (Fig.2 の Gatherer)、インデックス作成 (Fig. 2 の Indexer)、検索 (Fig.2 の Engine) を行う。

更新時の各構成要素の動作は以下の通りである。

1. LMSE は LSE を用いて文書を収集する。
2. LMSE は LSE を用いてインデックスを更新する。
3. LMSE は LS に Forward Knowledge (FK、語の集合、全文書数、語を含む文書数とその最高スコア) を送信する。LS は FK をメタインデックスに記録する。

CSE では、NFS 等のファイルシステム経由で直接収集できる文書はファイルシステムを通して直接収集し、クラスタを用いて並列にインデックス作成を行う事が可能である。また、直接収集できない文書は Web サーバに用意した文書収集用 CGI を用いて一括転送する事も可能である。これらが使用できない場合のみ、通常のロボットによる収集が用いられる。この結果、東洋大学の事例では約 1 分で全文書のインデックスを更新できた。

一方、検索時における各構成要素の振る舞いは以下の通りである。

1. ユーザはブラウザにより身近な $LMSE_0$ に検索を依頼する。
2. $LMSE_0$ は CS に検索を依頼する。
3. もし CS の RC に検索結果の次ページまでキャッシュされていたら 8 へ。もし、該当ページまでしかキャッシュされていなければ 5 へ。
4. CS は SC にサイト選択結果がキャッシュされていたら 5 へ。そうでなければ、LS にクエリに基づくサイト検索を依頼し、スコアの降順に並んだサイトのリストと各語の *idf* を受信する。
5. CS は次ページまでの項目を埋めるのに必要なだけリスト上位からサイト $LMSE_i$ を選び、検索要求を並行に送信する。
6. $LMSE_i$ は LSE に検索を依頼し、検索結果と次の最高スコアを CS に返す。
7. CS は結果をマージし、サイトのリストをスコア順に並び替える。
8. CS は結果を $LMSE_0$ へ返す。
9. $LMSE_0$ は結果を HTML に整形し、ユーザへ返す。

CSE では、検索時に通信による様々な遅延のため応答時間が長くなる。そこで、一回あたりの通信量を極力減らす、不要な通信は行わない等、以下の高速化技法などが提案されている。

クエリに基づくサイト選択 (QbSS) [4]。CSE は積 (AND)、和 (OR)、差 (NOT) の論理検索をサポートしている。ここで、クエリ A、B に対する選択サイトを S_A、S_B とすると、"A and B"、"A or B"、"A not B" はそれぞれ $S_A \cap S_B$、$S_A \cup S_B$、S_A となる。

先読みキャッシュ [3]。「次の 10 件」をバックグラウンドで先読みして応答時間を短縮する。これにより 2 ページ以降の連続した「次の 10 件」検索は常にキャッシュにヒットする。

スコアに基づくサイト選択 (SbSS) [5]。各サイトのクエリに対する最高スコアを収集し、「次の 10 件」検索時にクエリを送信するサイトを多くとも 10 サイトに限定する。これにより論理検索の 1 ページを除く連続した「次の 10 件」検索では、規模に依存せず一定の応答時間を実現した。

永続的キャッシュ [7] 更新間隔の短い CSE ではキャッシュを長く用いることができない。そこで、更新後に再検索を行うことでキャッシュの寿命を永続的にする。これにより一度検索された (論理型) クエリの応答時間も規模に依存せず一定となった。

以上の技法は以下のような場合に適用できる。第一に、検索式が単一キーワードであるかOR演算子のみを含む場合、または2ページ以降の「次の10件」検索では、スコアに基づくサイト選択により一定の応答時間を実現できる。第二に、更新前にクエリが検索されていれば、永続的キャッシュにより、一定の応答時間を実現できる。第三に、更新後にクエリが検索されていれば、大域的共有キャッシュにより即座に応答可能である。最後に、それ以外の場合はクエリに基づくサイト選択を行う。クエリに基づくサイト選択による応答時間は検索対象サイトの数に依存し、検索対象サイトの数は一般にCSEの規模に依存する。

4 FTF·IDF法

第2章で述べたように、我々は新鮮情報検索を提案したが、新鮮情報検索に適合するスコア計算法は存在しなかった。

Googleで用いられるPageRankはWebページの被リンク数をもとに順位を決定する方式である。したがって、word spammingのような順位のつり上げを目的とした攻撃には強く、概ね正しい順位を計算できる。しかし、Webページ間のリンクは手動で行われる場合がほとんどであり、新しいページへのリンクは少ないため、新鮮情報検索において妥当な順位づけを行える可能性は低い。同様な理由から、リンクに基づいたスコア計算法は新鮮情報検索に向いていないと考えられる。

TF·IDF法は伝統的に用いられるスコア計算法である。TF·IDF法によるスコアは、語の頻度(Term Frequency, TF)と語を含む文書の珍しさ(Inverse Document Frequency, IDF)の積で表される。リンクをスコア計算に用いないために新鮮情報検索に向いていると考えられるが、word spammingには弱い。

本論文では、FTF (Fresh TF)をTFの代わりに使用するFTF·IDF法を提案する。FTFはTFに語の新鮮さの考慮を追加したものであり、新鮮なトピックを表す語を古いトピックの語よりも高く評価する。すなわち、古い語の重みは時間の経過とともに減少する。

FTF·IDFの計算には次の3つの方法が考えられる。

第1の方法は、上記の概念に忠実である。語wがn回、文書d中の位置$p_1, p_2, ..., p_n$に現れるとする。位置p_iの語が時刻t_iに追加された場合、その新鮮さf_iは$\exp(-(t-t_i)/\alpha)$で表される。ここで、αはダンピングファクターである。よって、語wのFTF値FTF_wは以下のように表される。

$$FTF_w = \sum f_i = \sum \exp(-(t-t_i)/\alpha)$$

この方法では、インデックス中に全ての語の全てのタイムスタンプを保持する必要があるため、インデックスが巨大化し、実用的ではない。さらに、語が挿入・削除・順序を変更された場合、更新前と更新後の語の順序をマップするのは不可能でないにしても困難である。よって、この方法によるFTFの計算は現実的には不可能である。

第2の方法は、更新前のFTFの値から更新後のFTFの値を計算する方法である。時刻t_iにi番目のインデックス更新プロセスが完了したとすると、その時点でのFTF_iは以下のように計算される。

$$\begin{aligned} FTF_i &= FTF_{i-1} \cdot F + TF_i - TF_{i-1} \\ FTF_i &= TF_0 \end{aligned}$$

ここで、$F = \exp(-(t_i - t_{i-1})/\alpha)$である。$i-1$番目の更新と$i$番目の更新の時間間隔$(t_i - t_{i-1})$が一定の場合、FTFは以下のように再帰的に計算できる。

$$\begin{aligned} FTF_1 &= TF_0 \cdot F + TF_1 - TF_0 \\ &= TF_0 \cdot (F-1) + TF_1 \\ FTF_2 &= TF_1 \cdot F + TF_2 - TF_1 \\ &= TF_0 \cdot F \cdot (F-1) + TF_1 \cdot (F-1) + TF_2 \\ &... \\ FTF_n &= FTF_{n-1} \cdot F + TF_n - TF_{n-1} \\ &= TF_0 \cdot F^{n-1} \cdot (F-1) + ... \\ &\quad + TF_{n-1} \cdot (F-1) + TF_n \\ &= \sum TF_i \cdot F^{n-1-i} \cdot (F-1) + TF_n \end{aligned}$$

この方法では、FTF_iとTF_iを格納するだけでよいため、インデックスの大きさは小さくて済む。しかし、どの語が新鮮であるかを判断することは難しいうえに、削除された語のTFがFTFに影響をあたえる。

3番目の方法は単純である。文書dが時刻t_dに更新されたとすると、FTFは以下のようになる。

$$FTF = F \cdot TF$$

ここでFは2番目の方法と同様に$F = \exp(-(t - t_d)/\alpha)$である。この方法はTFと$F$の乗算を行うだけなので、word spammingに対する耐性を持たない。文書が更新されない場合は、この方法によるFTFの計算結果は1番目の方法と同じである。$TF_i = TF$とすると、FTFは以下のように計算される。

$$FTF_n = TF + (F-1) \cdot TF \sum F^i = TF \cdot F^n$$

これは3番目の方法の式そのものである。

実用性の観点から、我々は2番目の方法を採用することにした。2番目の手法を採用するに当たり、ダンピングファクターの決定が必要である。文献[20]にお

いて、われわれは $\alpha = 2$ を仮に使用した。しかし、実際の文書の更新周期や一度の更新で追加される語の量などは様々なため、日記などの文書にあわせてダンピングファクターを計算すべきである。そこで、更新周期の平均と TF の増加量の平均を元にダンピングファクターを計算する手法を以下のように提案する。

1. 適当なスレッショルド TF_{th} を超える、更新ごとの TF の増加量 $TF_{inc,i}$ を計算する。TF_{th} はトピックや記事の追加以外の微小な修正を無視できる程度の値に決定する。
2. $TF_{inc,i}$ のなかからピークを選び出し、各ピークの時間間隔の平均を Δt、TF 増加量の平均を TF_{avg} とする。
3. 以下の式により α を計算する。

$$\alpha = \frac{TF_{th}}{TF_{avg} \exp(-\Delta t)}$$

すなわち、Δt 経過すると、平均的な更新で増加した TF は F により打ち消され、FTF は TF_{th} に戻る。

最後に、FTF·IDF 法では TF·IDF 法と同様に FTF は IDF と乗算される。

5 Web 日記を用いた FTF·IDF の評価

FTF·IDF の Web 日記における評価を述べる前に、IDF 値を用いた簡易的な stopword について述べる。

通常、stopword は語の品詞によって決定され、冠詞や定冠詞などが対象となる。CSE では、LSE に一般的に利用されている小型サーチエンジンである Namazu を利用しているが、Namazu が日本語文書をインデックスするときには、文書を分かち書きするプログラムによっては品詞情報を利用できない場合がある。また、英語の場合、stopword を利用できない。さらに、文書中にプログラムなどが含まれる場合、無意味な記号のあらゆる組み合わせがインデックス中に現れてしまう。そこで、IDF 値が 1.0 以下の語を stopword とする簡易的な手法を試験的に導入した。その結果、2004 年 1 月 1 日までの日記の全単語のうち、87 語が排除され、TF の合計は 2437347 であったものが、1034149 となった。また、TF の平均も 36.84 から 15.63 になった。排除された語は少ないものの、スコアは大きく変わった。この方法により、ほとんどすべての文書ファイルに含まれる、特に検索する意味のない語を排除できる。

FTF·IDF の有効性を検証するため、Web 日記を対象とした評価を行った。対象となる Web 日記は、ほそかわたつみ氏の「不定期性写真日記」(http://fromto.cc/hosokawa/diary/) を用いた。この日記はデジタルカメラによる写真をメインに文章で構成され、作者が旅行などに行ったときはバースト的に 1 日に数回更新されるが、それ以外の場合は週に 1 から 5 度程度の頻度で更新される。Fig.3 に 2004 年 4 月の更新語数と TF の増化量の変化を示す。

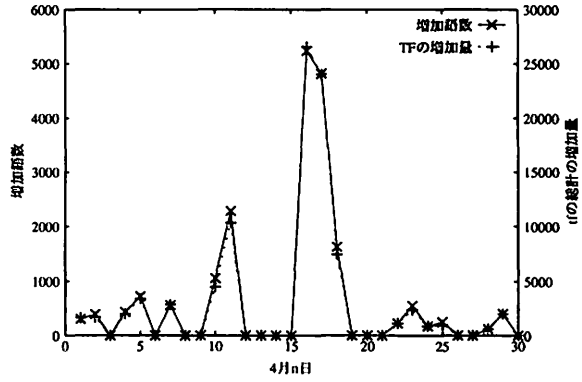

Fig. 3. 2004 年 4 月の語と TF の増加量

Table 1. ダンピングファクターとスレッショルド

TF_{th}	α	TF_{avg}	Δt	ピーク数
10	0.15886	1848.2	3.3796	108
20	0.31772	1848.2	3.3796	108
50	0.79430	1848.2	3.3796	108
100	1.5886	1848.2	3.3796	108
150	2.4379	1864.4	3.4112	107
200	3.4107	1896.2	3.4762	105
250	5.2942	2033.9	3.7629	97
300	7.2008	2109.8	3.9247	93
400	11.895	2232.3	4.1954	87

次に、スレッショルドとダンピングファクターの関係について述べる。スレッショルドを下げるとピーク数が増えるため、TF_{avg} は比較的小さくなる。また、打ち消されるべき TF の増加量が大きくなるため、ダンピングファクターは小さくなる。Table 1 に今回の評価に用いた日記におけるスレッショルドとダンピングファクターの関係を示す。スレッショルド 100 以下では TF_{avg}、t、ピーク数は変わらず、ノイズというべきピークを拾っていることがわかる。したがって、今回の場合は、スレッショルドは 200 から 250 の間がよいと考えられる。

Fig.4 に 2004 年 4 月に追加された語の TF と FTF の平均の推移を示す。TF の平均は日記が更新されるたびに増加するが (TF 計)、FTF は日記の更新が少なければ減少する ($TF_{th} = 20 \sim 300$)。スレッショルドによって、FTF の減少の速度は異なる。

Fig.7 にいくつかの語の ftf の変化を示す。「日吉」、「石垣」、「横浜」は 2004 年 4 月の日記中、数日だけ出現する語、「hosokawa」は毎回出現する語である。

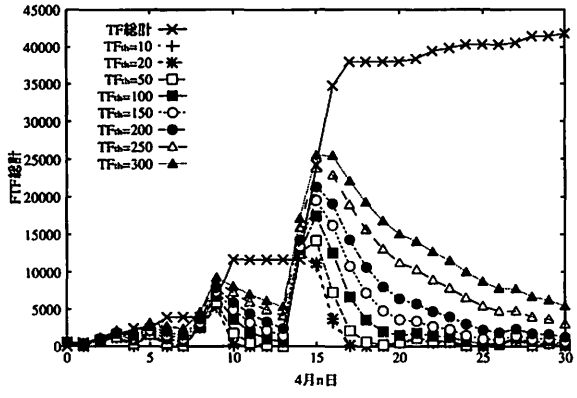

Fig. 4. 2004年4月のFTFとTFの変化

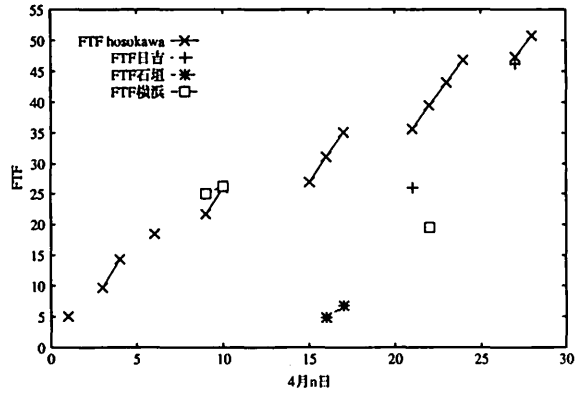

Fig. 5. 2004年4月のいくつかの語のFTFの変化

「hosokawa」はこの日記中に限っては、IDFによるstop-wordに含まれるため、spam wordとみなしてよい。評価に用いた日記は、記事ごとに別のファイルという形式のため、追加された語がすべて新しい内容に関する語とみなされる。したがって、この形式の日記でのspam wordへのFTFによる対応は難しいことがわかる。「横浜」はしばらく日記に使われなかったため、FTFの値が減少し、21日に追加されても以前のFTFの値に達していない。

Table 2にダンピングファクターとある単語を含む文書のうち最新の文書が最上位にランクされる確率との関係を示す。ここでは、インデックスされている全ての語、68551語を1語ずつ検索し、得られた各語の検索結果のうち最新の日付の日記が最上位にランクされる語数を数え、インデックスされている全ての語に対する割合を計算した。$\alpha = 0.31772 \sim 2.4379$では、最上位にランクされる割合は上昇している。$\alpha$がより大きい値をとる場合は、最上位にランクされる割合はほどんど上昇しない。この日記の場合は$\alpha = 3 \sim 7$が適当と思われる。

6 まとめと今後の課題

本論文では、新鮮情報検索において、より新しい文書・情報を浮き立たせるFTF·IDFによるスコア計算法を提案し、実際のWeb日記を対象にFTF·IDFの評価を行った。評価の対象としたほそかわたつみ氏の日記では、FTF·IDFを用いることにより新しいコンテンツが検索結果のより上位に来ることが確認できた。バースト的に更新される特性を持つ日記であれば、FTF·IDFによるスコアリングは十分有効と考えられる。毎日コンスタントに更新される日記や、日記以外のコンテンツでのFTF·IDFの有効性の検証は今後の課題である。

Table 2. ダンピングファクターと新しい日記が上位に現れる率

α	最上位にきた検索語数	最上位に現れる率
0.31772	56188	0.819653
0.7943	62106	0.905982
1.5886	65162	0.950562
2.4379	66190	0.965558
3.4107	66464	0.969556
5.2942	66359	0.968024
7.2008	66303	0.967207
11.895	66148	0.964946

参考文献

[1] 佐藤 永欣、上原 稔、酒井 義文、森 秀樹、"最新情報の検索のための分散型サーチエンジン"、情報処理学会論文誌、 第43巻、第2号、pp.321–331、情報処理学会 (2002)

[2] 西田 喜裕、山本 崇、佐藤永欣、上原 稔、森秀樹 "分散サーチエンジンにおける協調型検索", SWoPP'99, pp.87–92 (1999)

[3] 佐藤永欣、山本 崇、西田喜裕、上原 稔、森 秀樹、"協調サーチエンジンにおける継続検索のための先読みキャッシュ方式"、DPSWS 2000, pp.205–210 (2000)

[4] 酒井義文、上原 稔、佐藤 永欣、森 秀樹、"協調サーチエンジンにおける検索クエリの最適な単調化"、DICOMO'2001, pp.453–458 (2001)

[5] 佐藤永欣、上原 稔、酒井義文、森 秀樹, "協調サーチエンジンにおけるスコアに基づくサイト選択"、DICOMO'2001, pp.465–470 (2001)

[6] 佐藤永欣、上原 稔、酒井義文、森 秀樹、"協調サーチエンジンにおける大域的共有キャッシュ"、DPSWS 2001, pp.219–224 (2001)

[7] Nobuyoshi Sato, Minoru Uehara, Yoshifumi Sakai, Hideki Mori, "Persistent Cache in Cooperative Search Engine", In proc. of The 4th International Workshop on Multimedia Network Systems and Applications, pp.182–187 (2000)

[8] J. F. Allen, "Towards a general theory of action and time," Artificial Intelligence, vol. 23, pp.123–154 (1984)