日本ソフトウェア科学会第 27 回大会 (2010 年度) 講演論文集

研究室向けのファイル管理ウェブシステム

村口 洋士　榎原 博之

本研究では，研究室向けのウェブアプリケーションとして利用できるファイル管理システムを構築し，ファイルを容易に探し出せるように，全文検索エンジンを導入する．全文検索を行うためにはテキスト化されたデータが必要であるため，バイナリ形式のデータをテキスト化するために，各ファイル形式に対応した文書フィルタの作成を行う．テキスト化する際，論文などの形式を考慮して，文書中で重要と思われる箇所を自動的に判断し，それらに検索結果が良くなるように重み付けを行い，研究者にとって良い検索結果が出力されるように改良を加えている．

1　はじめに

近年の情報化社会の中で，様々な電子データは増加の一途をたどっている．電子データの中には文書データだけではなく，音声データ，画像データ，動画データも存在する．しかし，現在もっとも利用されているものは文書データである．ウェブ上には文書データが無数に存在しているが，Google[†1]などのウェブ検索エンジンを用いることで必要なものを容易に探し出すことができる．

現在，様々な研究を行う上で，様々な種類の電子化された文書データが大量に必要である．しかし，個人でそれらのデータを保有すると管理が大変な手間となり，複数人でのデータの共有を行うことは非常に困難である．また，管理していないデータから必要なデータを探し出す場合，ウェブ上にあるデータと比べると時間が掛かり効率的ではない．

研究に必要なデータを管理する方法の 1 つに，ウェブアプリケーションでデータを管理する方法が挙げられる．これにより，個人でデータを管理するより手間がかからず，容易に複数の人とデータを共有することができる．また，大量に存在するデータからある特定のデータを探し出すための方法に，全文検索システムを導入することが考えられる．その際，研究者にとって必要なデータを検索結果の上位に持ってくることで，研究の効率化を行うことができる．

本論文では，ウェブアプリケーションとして利用できるファイル管理システムを構築し，その利用状況を基に，研究室向けに最適化した全文検索エンジンを導入する．全文検索を行うためにはテキスト化されたデータが必要である．バイナリ形式のデータをテキスト化するために，各ファイル形式に対応した文書フィルタを作成する．さらに，文書中で重要な箇所と考えられる部分に重み付けを行い，検索結果を改善する．検索結果については重み付けしたものと，重み付けしていないものについての比較のアンケートを行い，評価を行う．さらに，重み付けの有効性を確認する手段として実験を行う．

本論文の構成は以下のようになっている．本章を第 1 章とし，第 2 章ではファイル管理ウェブシステムの開発手法について述べ，第 3 章では全文検索エンジンの導入手法，第 4 章では導入の結果と評価について述べる．第 5 章で関連研究について述べ，第 6 章

File management web system for laboratories.

Hiroshi Muraguchi, 関西大学 大学院 理工学研究科, Graduate School of Science and Engineering, Kansai University.

Hiroyuki Ebara, 関西大学 システム理工学部 電気電子情報工学科, Department of Electrical and Electronic Engineering, Faculty of Engineering Science, Kansai University.

†1　Google, http://www.google.co.jp/

でまとめについて述べる．

2 ファイル管理ウェブシステムの開発

本研究では，ウェブブラウザから利用できるファイル管理ウェブシステムを構築する．本研究で構築したシステムを PoleManage と名付ける．PoleManage の Pole は **P**hoto and file **o**rrganizer for **l**aboratory **e**nvironment の略である．

2.1 概要

本研究で構築したシステムでは開発言語に Ruby1.8.7，ウェブサーバに Apache2.2，データベースに SQLite3 を使用している．アップロードされたファイル自体はデーターベースで管理するのではなく，ファイルとして HDD に保存している．データベースにはファイルのアップロードされた時間やサイズなどの属性情報を登録している．

本システムのトップページは図 1 のようになっている．本システムではアカウントを利用者個人ごとに作成する．また，ファイル共有のためにグループフォルダを作成し，利用者ごとのアカウントが所属する形となっている．それにより，所属するグループフォルダ以外のファイルを見ることができないようにしている．また，利用者の個人専用のグループフォルダも作成でき，個人的なファイルもアップロードすることができる．各グループフォルダごとに個別にフォルダを作成することができる．

ファイルをアップロードする際には，コメントをつけることができる．また，オリジナルのファイル名を維持したままにするかどうかを選択することができる．維持されなかった場合はシステムがつける管理名で保存される．アカウントの設定を行う際にメールアドレスが登録でき，アップロード時にそれを知らせるために同じグループに所属している利用者にメールを一斉送信することができる．また，複数のファイルを同時にダウンロードできる機能を搭載している．

アップロード時に一部のファイル (JPEG，PNG，PDF，OpenDocument Format) ではサムネイルを生成している．サムネイルはダウンロード時に表示される．また，通常使用されるファイルモードの他にフォトモードがある．このモードではアップロードされた画像ファイルのサムネイルのみを表示することにより，素早く写真を見ることができる．

ファイルの削除，移動，名前変更，コメント変更は，アップロードした利用者と，グループごとに存在する管理者以外できないようになっている．複数人が共有して使用できるゲストアカウントが設定でき，ゲストアカウントでは IP アドレスやアップロード後の経過時間を判断して，他のゲストアカウントを利用する人がアップロードしたファイルの削除などができないようになっている．

本システムでは利用者が所属している全グループのアップロード状況を容易に確認するため，利用者ごとに RSS(Atom) フィードを配信している．

2.2 ファイルウェブ管理システムの構築

本節では，ファイル管理ウェブシステムの構築手法について述べる．

2.2.1 フレームワーク

ファイル管理ウェブシステムのフレームワークとして，Rack[†2] と Phusion Passenger[†3] を使用している．Rack とは Ruby のためのウェブサーバインターフェースである．Rack を用いることによってアプリケーションの開発者はウェブサーバとのやりとりの詳細を意識しなくてよくなる．また，Phusion Passenger は Ruby on Rails[†4] などのフレームワークを実行させるための Apache モジュールである．Rack はこれに対応している．これを用いることにより，アクセス時にアプリケーションを毎回起動させるのではなく，常に起動させたままにしておくことができる構造となっている．結果的に，毎回起動する時に掛かる負荷を抑えることができる．また，HTML 生成に必要なファイルや設定ファイルを読み込む回数を少なくすることにより，高速化が実現できる．

2.2.2 認証

認証は，アカウント名とパスワードを入力して照合を行うことで実現している．またユーザのログイン情

†2 Rack, http://rack.rubyforge.org/
†3 Phusion Passenger, http://www.modrails.com/
†4 Ruby on Rails, http://rubyonrails.org/

図 1　トップページの表示

報は Cookie に保存される．Cookie 内の情報を用いることによってページ遷移後にもログイン情報が保たれるようにしている．ログイン情報は一度認証した後 60 分経過するか，ウェブブラウザが閉じられるまで有効である．

生成された RSS(Atom) へフィードリーダからアクセスした場合，フィードリーダによっては先程と同じ認証方式を行うことができない．よって，アカウントごとに固有に生成した鍵を用いて認証を行っている．

2.2.3　ファイル管理

アップロードされたファイルには固有の管理名をつけて管理を行っている．ファイルをハッシュ関数 SHA-1 にて計算を行った結果の先頭 12 文字と，アップロードされた時間をハッシュ関数 SHA-1 で計算を行った結果の先頭 8 文字を組み合わせて 20 文字の管理名としている．また，万が一管理名が重複した時は，時間を一秒ずつ増やして計算し直している．

管理名に拡張子をつけたものをファイル名として HDD に保存している．アップロードされたファイルは，一時ファイルとして HDD に保存されている．これをデータベースに保存する場合は，一度ファイルをオープンして読み込まなければならない．しかし，HDD に保存する場合はファイルを移動するだけでよいので，データベースを使用する時よりも高速に処理を行うことができる．

また，ファイル名やコメント，アップロードされたフォルダなどの情報は，データベースにて管理している．

2.2.4　Ajax の効果的な使用

Ajax とは，JavaScript による非同期通信を用いて動的にウェブページを書き換えるためのいくつかの技術の総称である．

本システムでは Ajax を効果的に用いることにより，ウェブページの遷移を少なくしている．Ajax と JavaScript を用いることでタブを表示し，機能の切り替えを行っている．

また Firefox[†5] などの一部のブラウザでは，Ajax と HTML5[†6] の API を利用してドラッグ&ドロップでアップロードができるようになっている．

2.3　利用状況

本研究で構築したシステムを関西大学システム理工学部電気電子情報工学科アルゴリズム工学研究室のサーバに導入する．アルゴリズム工学研究室のサーバの OS は Debian GNU/Linux lenny である．

アルゴリズム工学研究室のメンバーに使用してもらったところ，2009 年 6 月 24 日から 2010 年 1 月 20 日までの 211 日間で 431 個，1071MB のファイルがアップロードされた．

ファイルを種類別に見ると図 2 のようになる．アップロードされたファイルの中で一番多かったものは Portable Document Format(PDF) ファイルである．これは，論文や発表会での配付資料が多くアップロードされたためである．

†5　Mozilla Firefox, http://mozilla.jp/firefox/
†6　HTML5, http://www.w3.org/TR/html5/

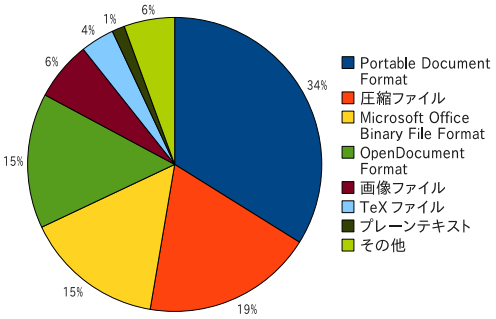

図 2　ファイル種類別一覧

次章では，これらのファイルを全文検索できるように全文検索エンジンを導入する．

3　全文検索エンジンの導入

全文検索エンジンの導入に当たって既存の全文検索エンジンを比較し，本研究では Ruby から容易に利用できる Senna[†7] を採用することにする．

3.1　全文検索エンジン Senna

Senna は完全転置インデックスを採用する全文検索エンジンで、単語の分割手法として N-gram と形態素解析のどちらかを選択できる。Senna では形態素解析で作成したインデックスを用いても、単語境界よりも短い文字列で適合されることができ、検索結果の漏れを防ぐように制御することができる。また、インデックスを更新する際にバッファをあらかじめ確保することにより、インデックスの作成更新処理を高速化している。その際、インデックスを自動再配置することにより、検索性能の低下も防いでいる。

本研究では Senna を用いて転置インデックスを作成している．その際，2.1 で述べたように利用者が所属するグループごとに転置インデックスを作成している．また本研究では単語の分割手法として形態素解析を用いている．形態素解析のプログラムとして MeCab[†8] を，辞書には IPA 辞書を用いている．

Senna は上記で述べた他に，様々な特徴がある．

†7　Senna, http://qwik.jp/senna/FrontPageJ.html
†8　MeCab, http://mecab.sourceforge.net/

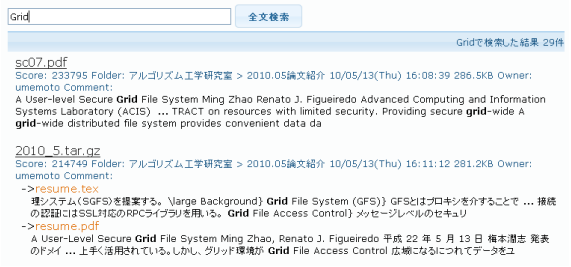

図 3　本システムでの検索結果

3.1.1　セクション単位での転置インデックスの登録

Senna では転置インデックスを作成する際，1 つの文書 ID に複数のセクションを登録することができる．文書中のテキスト，ファイルに対するコメント，アップロードされた際につけられるファイル名をそれぞれセクションとし，登録している．これにより，ファイル名やコメントも検索対象とすることができる．

3.1.2　文字列に対する重み

Senna は文書中の任意の文字列の重みを変えることができる．重みとは文書中の文字列がどの程度重要かということを表す値である．検索時に重みの大きい文字列に適合した文書の順位を上げることができる．これについては 3.3 で述べる．

3.1.3　組込型

Senna はデータベース管理システムやスクリプト言語処理系に組み込んで使用する組込型である．また，組み込み先が持っているデータ管理システムとの重複管理を避けるために，転置インデックスを作成する際に必要となるテキスト化された文書を管理しない．しかし，ファイル管理システムからファイルを削除した際に，転置インデックスから削除したファイルの情報を削除する場合は，登録した際に用いたテキスト化された文書が必要である．そこで，本研究では 3.2 で述べる文書フィルタを用いてテキスト化を行う．文字列をどのように重み付けを行うのかというデータは XML 形式にて作成し，保存している．また，検索時に XML ファイルから文書中のテキストを抽出し用いることで，図 3 の様に結果の内容に本文の適合した箇所を表示することができる．

3.2 文書フィルタ

文書フィルタとはバイナリ形式のファイルから文字列を抽出するものである．文書フィルタはバイナリ形式ごとに必要である．どの文書フィルタを用いるかをファイルの拡張子で判断している．対応している拡張子の一覧を表 1 に示す．また，文書フィルタはウェブアプリケーションの動作に影響がないように別プロセスで処理を行っている．

以下ではそれぞれの形式に対応した文書フィルタについて述べる．

3.2.1 圧縮ファイル

圧縮ファイルにはいくつかのファイルが含まれている．しかし，文書フィルタのない形式のファイルまで解凍すると無駄である．そこで，拡張子をチェックして文書フィルタが存在する場合のみファイルとして解凍し，それぞれの文書フィルタを用いてテキスト化を行っている．

3.2.2 Portable Document Format(PDF)

Portable Document Format(PDF) は，論文，取扱説明書などの電子文書として用いられている形式であり，扱う環境によらずほぼ同じ状態で表示，印刷できるファイル形式である．

表 1 文書フィルタ対応拡張子一覧

圧縮ファイル	.zip .gz .bz2 .tar .tar.gz .tgz .tar.Z .taz .tar.bz2 .tbz .cpio .cpio.gz .cpio.Z .cpio.bz2
Portable Document Format (PDF)	.pdf
Microsoft Office Binary File Formats	.doc .xls .ppt .csv
Office Open XML	.docx .xlsx .pptx
OpenDocument Format	.odt .ods .odp
プレーンテキスト	.txt
Tex 形式	.tex

本システムでは Poppler[†9] というライブラリを用いてテキスト化を行っている．

3.2.3 Microsoft Office Binary File Formats

Microsoft Office で作成されたファイル形式は 2007 年以降に発売されたバージョンとそれ以前のバージョンではまったく異なる．ここでは，2007 年までに発売された Micrsoft Word(拡張子が doc)，Microsoft Excel(拡張子が xls)，Microfsoft PowerPoint(拡張子が ppt) で作成されたファイルを対象とする．

対象となるファイルの仕様は公開されている [1]．しかし，構造が複雑であり容易に解析できないため，本研究では catdoc[†10] と言うプログラムを用いている．

catdoc を用いることにより，拡張子が doc と ppt であるファイルはテキスト化を行うことができる．また，拡張子が xls であるファイルは catdoc を用いることにより，カンマ区切りのテキストファイルである CSV ファイルにでき，カンマを除去することによりテキスト化を行っている．

3.2.4 Office Open XML

Office Open XML は国際標準化機構にて標準化されたオフィスソフト用のファイルフォーマット規格の一つである [4]．2007 年以降に発売された Micrsoft Word，Microsoft Excel，Microfsoft PowerPoint の標準保存形式となっている．拡張子はワードプロセッサファイルが docx，表計算ファイルが xlsx，プレゼンテーションファイルが pptx となっている．

フォーマットの中身は XML ファイルや画像ファイルを ZIP 形式で圧縮したファイルとなっている．これらの中からテキストコンテンツが含まれているファイルを解析しテキスト化を行っている．

3.2.5 OpenDocument Format

OpenDocument Format は国際標準化機構にて標準化されたオフィスソフト用のファイルフォーマット規格の一つである [3]．OpenOffice.org[†11] 2.0 以降のバージョンの標準保存形式となっている．この中で，

†9 Poppler, http://poppler.freedesktop.org/

†10 catdoc and xls2csv: http://wagner.pp.ru/~vitus/software/catdoc/

†11 OpenOffice.org, http://www.openoffice.org/

ワードプロセッサファイル (拡張子が odt)，表計算ファイル (拡張子が ods)，プレゼンテーションファイル (拡張子が odp) を対象にしている．

フォーマットの中身は XML ファイルや画像ファイルを ZIP 形式で圧縮したファイルとなっている．この中でテキストコンテンツが含まれているファイルを解析しテキスト化している．

3.2.6 プレーンテキスト

プレーンテキストにはテキストデータしか含まれていないが，さまざまな文字コードが使用されている．日本で利用される文字コードには主要なものだけでも，Shift_JIS，EUC-JP，ISO-2022-JP，UTF-8 と様々なものがある．そこで，nkf[†12]を用いて文字コードを転置インデックスで使用している UTF-8 に変換している．

3.2.7 TEX 形式

TEX[†13]とは組版処理用のソフトウェアであり，専用形式のファイルを処理し組版を行っている．この形式のファイルはテキストファイルの中に，文章と構造を指定する命令が書かれている．そこで本システムでは文字コードを変換し，構造を指定する命令の削除を行っている．

3.3 文書に対する重みの導入

研究者にとって必要な検索結果を得られるように，文書中で重要な箇所と考えられる部分に重み付けを行っている．重み付けは研究者が検索する上でキーワードとなる可能性が高い箇所に対して行っている．本システムでは文書フィルタでテキスト化を行うのと同時に重み付けを行っている．

以下では，重み付けの方法について述べる．

3.3.1 ファイル名とコメント

アップロードされた際につけられるファイル名とコメントは文書を検索する際の重要な手がかりとなる．そこで，アップロードされているすべてのファイルについてファイル名とコメントに重み付けを行っている．

3.3.2 Portable Document Format(PDF)

Portable Document Format(PDF) は主に論文の配布形式として使用されている．そこで論文形式に適する重み付けを行っている．Portable Document Format(PDF) では Poppler に存在するテキストの検索機能を用いて重み付けを行っている．文書中の文字列をスペースの連続した箇所で分割し検索機能を用いることで，その文字列が存在する箇所を四角で囲った領域の座標が返される．それらから計算を行うことで，文字列の高さを推定することができる．文字列の高さを各ページで集計する．1 ページ目では文字列の高さが一番大きいものを論文のタイトル，二番目に大きいものをサブタイトルや節タイトルと見なし，重み付けを行っている．そのほかのページでは文字列の高さがもっとも大きいものを章タイトルや節タイトルと見なし重み付けを行っている．

3.3.3 Microsoft Office Binary File Formats

3.2.3 で述べた Microsoft Office Binary File Formats では，外部のプログラムである catdoc を用いてテキスト化を行っている．これにより文書中の構造が把握できないので Microsoft Office Binary File Formats では重み付けを行っていない．

3.3.4 ワードプロセッサ形式

Office Open XML と OpenDocument Format のワードプロセッサ形式のファイルでは，図 4 のような構成を取っているものが多い．そこで，1 行目にタイトルが含まれていると仮定している．また，タイトルと，図形や飾り付けされた文字は重要なものであると仮定し，それらに重み付けを行っている．

3.3.5 表計算形式

Office Open XML と OpenDocument Format の表計算形式のファイルではセルのフォントデータを参照している．シートごとにもっともフォントサイズが大きいセルで，かつ 15 ポイント以上のセルの文字列をタイトルと見なし重み付けを行っている．

3.3.6 プレゼンテーション形式

Office Open XML と OpenDocument Format のプレゼンテーション形式のファイルでは，図 5 のような構成を取っているものが多い．そこで，文字列の

†12 nkf Network Kanji Filter, http://sourceforge.jp/projects/nkf/

†13 TEX, http://www.tug.org/

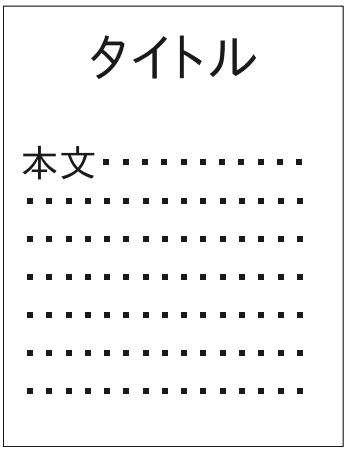

図 4　ワードプロセッサ形式の例

図 5　プレゼンテーション形式の例

フォントデータを参照し，各スライドのもっともフォントサイズが大きいものをタイトルと見なし，重み付けを行っている．また，1 枚目のスライドには重要な事項が書いていることが多いのでそれらにも重み付けを行っている．

3.3.7　プレーンテキスト

プレーンテキストには決まった書式が存在しないために，任意の書き方を行うことができる．また，文書中に文字の大きさや色付け情報が一切存在しない．これらにより文章中の構造が判断できないため，重み付けを行っていない．

3.3.8　TEX 形式

TEX 形式では構造を指定する命令が含まれている．それにより容易にタイトルなどを識別することができる．本システムでは題名と章題，節題，項題，著者を指定している文字列に重みを加えている．

4　導入の結果と評価

全文検索システムを導入した結果，2.3 で述べた同一期間にアップロードされた 431 個のファイルの内，379 個である 88%のファイルの文書中のテキストを検索できるようになる．しかし，文書フィルタが対応していない形式については，文書中のテキストを検索することはできない．

4.1　アンケートによる評価

検索結果に関する評価を行うために，関西大学システム理工学部電気電子情報工学科アルゴリズム工学研究室のメンバー 12 人に対してアンケートを行う．アンケートは次のような項目について行っている．

1. 検索機能が使いやすいかどうか?
2. 検索結果が見やすいかどうか?
3. 検索機能が役に立っていると思うか?
4. 重み付けを行っていない検索結果に自分にとって必要なものが上位にあるか?
5. 重み付けを行った検索結果に自分にとって必要なものが上位にあるか?

アンケートは各項目 5 段階で評価している．もっとも良い結果を 4 点，もっとも悪い結果を 0 点と換算し平均をとると，図 6 のようになる．

検索機能の使いさすさ，検索結果の見やすさ，検索機能が役に立つかという項目についてのアンケート結果は，検索機能の使いやすさという項目では平均 3.5 点，検索結果の見やすさという項目では平均 3.5 点，

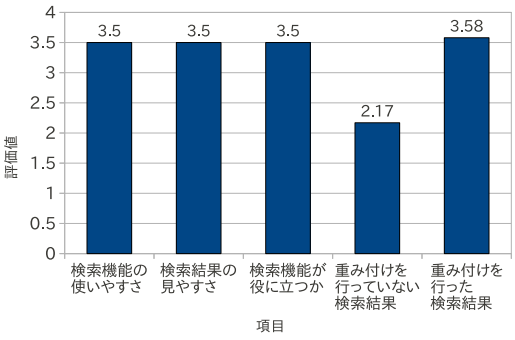

図 6　アンケート結果

検索機能が役に立つがどうかという項目では平均 3.5 点という評価になった．これらは，それぞれ平均である 2 点を大きく越えている．よって，この 3 点に関しては利用者が満足していると考えられる．

重み付けを行っていない検索結果と重み付けを行った検索結果において，対象者が必要なものが上位に来ているかという項目についてのアンケート結果は，重み付けを行っていない検索結果が平均 2.17 点，重み付けを行っている検索結果が平均 3.58 点となった．これらを比較すると重み付けを行っている検索結果の方が 1 点以上も高く，重み付けを行うことに効果があることが分かる．

4.2 実験による評価

重み付けの有効性を定量的に確認する手段として実験を行う．一定数のファイルがアップロードされているグループに対して Wikipedia 日本語版[†14]に存在する約 126 万ページの内，ランダムに選択した 10%にあたる約 12 万ページを追加する．追加したファイルは全体のファイルのうち，99%以上を占めている．

それらの中から重み付けを行っていないものと重み付けを行っているものそれぞれをに対して，研究者が使用する頻度が高い単語と考えられる表 2 の 12 個のキーワードで検索を行う．重み付けは既存のファイルに対してのみ行う．検索結果の中で既存ファイルが先頭から 10 番目までに出現した場合，1 番目から順位に応じて 10 点から 1 点の得点を与える．これらを合計し満点が 55 点となるようにして 12 個のキーワードで平均を取ると，図 7 のようになる．

表 2 検索キーワード一覧

検索キーワード
開発，環境，管理，技術
計算，研究，システム，構築
情報，統計，発表，論文

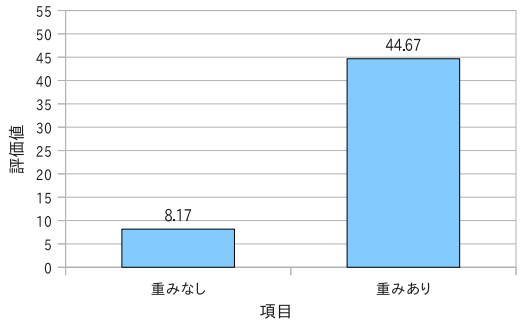

図 7 実験結果

重み付けを行っていない検索結果では平均 8.17 点，重み付けを行っている検索結果では平均 44.67 点となった．重み付けを行っていない検索結果についても既存ファイルの一部が検索結果に現れているが，重み付けを行っている検索結果の方がより多くの既存ファイルが検索結果の上位に現れている．この結果から重み付けの有効性がわかる．

5 関連研究

ファイル管理に関する関連研究として，Mikan というソフトウェアがある [2]．このソフトウェアでは SMTP[†15] /POP3[†16]を用いてファイル管理を行っている．クライアント側は専用のプログラムが必要で Windows[†17]でしか動作しないが，サーバ側では既存のメールシステムがあればよい．それに対して，本研究では，HTTP[†18]を用いてファイル管理を行い，サーバ側には本研究で開発したソフトウェアが必要であるが，クライアント側はウェブブラウザがあればよい．また，Mikan にはファイルの検索機能は付いていないが，本研究では検索機能があり，容易にファイルを探し出すことができる．

全文検索に関する関連研究として，Google で使用されている PageRank とよばれているランキング技術がある [5]．PageRank はウェブページ間のリンクを利用したスコアリングを行う．あるページの PageRank は

†14 Wikipedia, http://ja.wikipedia.org/

†15 RFC 5321 - Simple Mail Transfer Protocol, http://tools.ietf.org/html/rfc5321

†16 RFC 1939 - Post Office Protocol - Version 3, http://tools.ietf.org/html/rfc1939

†17 Microsoft Windows, http://www.microsoft.com/japan/windows/

†18 RFC 2616 - Hypertext Transfer Protocol - HTTP/1.1, http://tools.ietf.org/html/rfc2616

そのページへのリンクがある複数ページの PageRank から計算することができる．人気のあるページでは多くのページからリンクされるため，PageRank が高くなり，人気のあるページからリンクされたページも PageRank が高くなる．しかし，本研究でアップロードされるファイルはファイル間のリンクがなく，この技術は使用することができないため，ファイルの重要度を決める指標として重みを用いている．

6 おわりに

様々な研究を行う上で電子化されたデータは重要である．しかし，それらを手作業で管理することは手間であり，必要なデータを探し出すことに時間がかかりすぎるため，ウェブアプリケーションとして利用できるファイル管理システムを構築し，そしてその利用状況を基に全文検索エンジンの導入を行った．

バイナリ形式のファイルをテキスト化できるように各ファイル形式に対応した文書フィルタを作成した．また，検索結果を向上させるため，文書中で研究者が検索する際にキーワードとして使用する可能性が高い箇所に重み付けを行い，評価を行うためにアンケートと実験を行った．アンケートの結果より，検索機能が使いやすいこと，検索結果が見やすいこと，検索機能が役に立つことが確認でき，研究者にとって必要なデータを検索結果の上位に持ってくる手法として，重み付けを行うことに効果があることを確認できた．さらに，実験の結果より，全体のうち 99%以上が研究に不必要であるものに対して検索を行うと，研究に必要なファイルがあまり検索の上位に現れないが，重み付けを行うことで，研究に必要なファイルを検索の上位に現れるようにすることができ，このことから重み付けの有効性を確認できた．

今後の課題として，文書フィルタの追加と性能向上，重み付けの改良が挙げられる．

最後に，この研究で作成したソフトウェアは公開する予定である．

謝辞 アンケートに協力いただいた関西大学システム理工学部電気電子情報工学科アルゴリズム工学研究室のメンバーに感謝の意を表す．

参 考 文 献

[1] Microsoft Office Binary (doc, xls, ppt) File Formats: http://www.microsoft.com/interop/docs/OfficeBinaryFormats.mspx

[2] 西村 祐貴：SMTP を利用したファイル共有システムに関する研究, 修士論文, 慶應義塾大学大学院, 2003

[3] OpenDocument Format for Office Applications v1.0: http://www.oasis-open.org/committees/download.php/12572/OpenDocument-v1.0-os.pdf

[4] Standard ECMA-376 Office Open XML File Formats: http://www.ecma-international.org/publications/standards/Ecma-376.htm

[5] Page, L. and Brin, S. and Motwani, R. and Winograd, T.: The pagerank citation ranking: Bringing order to the web, Technical report, Stanford Digital Library Technologies Project, 1998