全文検索システム Akao の検索手法と性能評価

森本 哲也　　畑中 豊司
株式会社　データ変換研究所
{morimoto,hatanaka}@dehenken.co.jp
http://www.dehenken.co.jp/

あらまし
本論文では弊社が開発した N-gram 型の全文検索システム Akao とフリーウェアながら幅広く導入実績をもつ形態素解析型の Namazu[1]について、インデックス処理・検索レスポンスの性能を定量的に検証した。
測定結果より、検索アルゴリズムの異なる両者の実装において各々の特徴がみられた。インデックス処理に関しては Akao の方や速く、大多数ファイルのインデックス化にその有効性を示した。検索レスポンスに関しては Namazu の方がやや速いが、実運用上、大きな差ではないことを確認した。

1　はじめに

最近、我々は N-gram を検索アルゴリズムに採用した全文検索システム Akao のテストを行った。Akao はイントラネット向けのサービスであり、当初は大規模 LAN 内で Akao の運用性能を測定することを目的に始めた。そして、テストを続けていく内に、さらに検索アルゴリズムの特徴にまで視野を広げた性能評価を追究するに至った。

比較対象として、形態素解析を検索アルゴリズムとして採用している全文検索システム Namazu を取り上げた。Namazu はフリーウェアでありながら官公庁、市役所、企業、大学など幅広く導入実績をもつことから信頼性の高いシステムと言える。1997 年に開発が始まり、2002 年 9 月のバージョン 2.0.12 が最終更新となっている。これは開発がストップしたというより、ソフトウェアとしての完成度が成熟したとみるべきだろう。

そもそも“Akao”という名前は Namazu を超えるソフトウェアにしたい、という願望から、アマゾン河に生息する全長 1.5ｍにもなる大なまず“Redtail Cat”の“Redtail”→“赤尾”→“Akao”に由来する。

Akao には Namazu にはない機能が幾つかある。しかしながら、検索そのものの性能について、N-gram／形態素解析のアルゴリズムの違いはあるが、実際はどのような差があるのだろうか。

本論文では、定量的な評価により、N-gram と形態素解析の特徴、ひいては Akao と Namazu の特徴について述べる。

2　Akao の概要

Akao について簡単に紹介する。Akao は Windows/Solaris/Linux など各サーバに蓄えられた文書ファイルに含まれたキーワードを検索する全文検索システムである。各部門にある分散したファイルサーバのファイルを一括管理し、目的のファイルを探すときのコスト（時間）を大幅に削減することができる。

2．1　Akao の機能

ここで述べる 4 つの機能は Akao の主要な機能であり、全文検索システムとしての操作性や導入の容易さを追求した独自の機能である。

ⅰ）マルチ OS 分散環境の統合

Akao は各種 OS（Windwos / Solaris / Linux など）で構築されたファイルサーバが混在している環境下でも、蓄えられている膨大な文書ファイルを Linux 上に統合し、OS の違いを意識することなく運用することが出来る。

ⅱ）多様なファイルフォーマットに対応

Akao は、MS-Office 系文書（Word、Excel、PowerPoint）、PDF、一太郎など社内で最も使用される文書ファイルのフォーマットに対応している。

また、BMP をはじめとした各種画像ファイルにも対応していて、スキャナなどで取り込んだ OCR 用画像に対しても検索が可能となる。

ⅲ）高速なインデックス作成と検索処理

Akao は大量の文書ファイルに対して高速にインデックスの作成と検索が出来る。MS-Office 系文書、PDF 文書、画像などのテキスト抽出からインデックスまで一括して処理を行う。検索後の結果表示出力についても、日付検索・サーバ指定検索・and/or 検索など目的に応じた出力が行える。内部情報の管理は国際化対応を見据え、日本語処理に依存しないように Unicode で文字情報を取り扱っている。

ⅳ）充実した管理ツール

Akao は Web アプリケーションの一つである。Web ブラウザから GUI でユーザ管理／グループ管理・サ

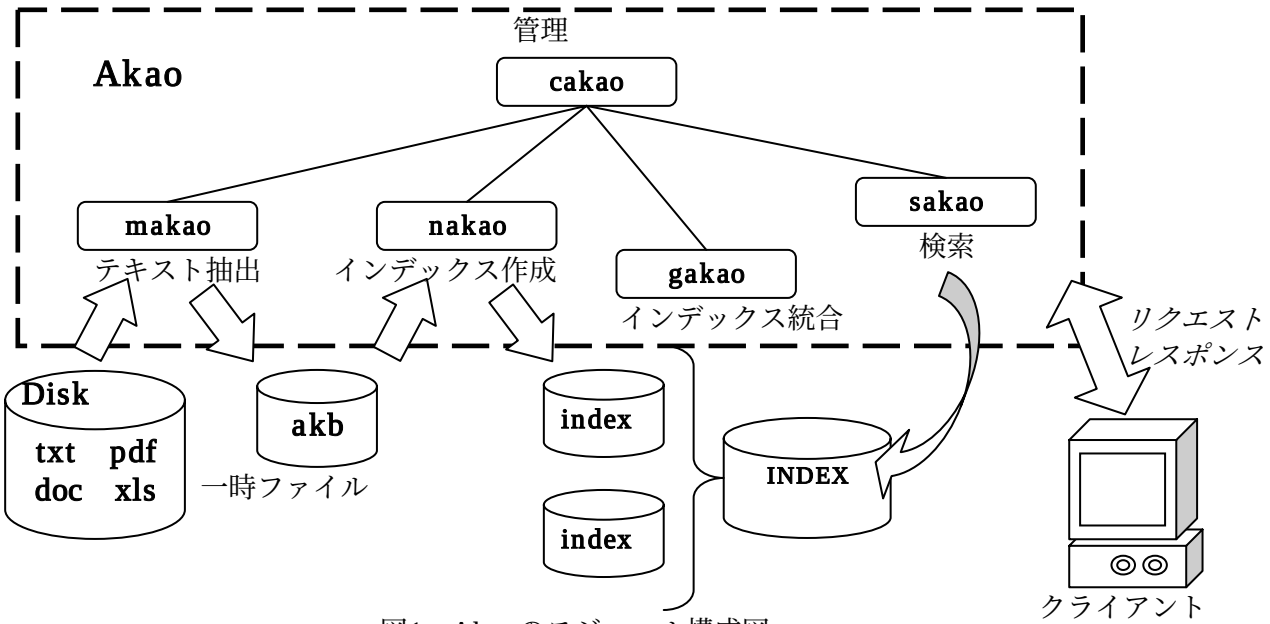

図1　Akaoのモジュール構成図

ーバドライブ接続･インデックス設定･バックアップなどを簡単に行うことができる。

設定された内容は全て CSV 形式にてインポート／エクスポート、フロッピーディスクへのバックアップ／リカバリを行うことが出来るので、マシンの入れ替えや障害の復旧なども迅速に行うことが可能になる。

v）閲覧制御機能

分散されたファイルサーバを統合することは運用コスト面ではメリットであるが、情報漏洩などのセキュリティ面が懸念される。そこで、ユーザや部署毎に検索できるサーバやフォルダを制限する閲覧管理機能も実現している。

2．2　Akao の構成

Akao のモジュールは大雑把に次の 5 つで構成される。それぞれのモジュールに対しての機能は以下の通りである。

makao：テキスト抽出(一時ファイル作成)
nakao：インデックス作成
gakao：インデックス統合
sakao：検索処理
cakao：管理ツール、閲覧管理

モジュールの構成図を図 1 に示す。

3　検索アルゴリズム

AkaoやNamazuは単語型の全文検索システムである。単語で検索するためには、文章（文字列）を単語に区切る作業が必要になる。日本語を意味の通る単語に区切ることを“分かち書き”と呼ぶ。Namazu はこの分かち書きを行って、インデックスファイルを作成している。Namazu の場合、日本語の文章をどうやって分かち書きするかが検索において最も重要な要因の 1 つになる。

一方、Akao では分かち書きを行わず、N-gram という手法で日本語の文章を分割している。

3．1　形態素解析

日本語を分かち書きする手法として、一般的に“形態素解析”という手法が取られる。フリーなツールとして、kakasi[1]や Chasen[2]などがあり、Namazu ではこれらのツールを利用している。

あらかじめ単語登録してある辞書に基づいて文章の中から単語を抽出していく方法である。辞書に存在しない単語は抽出できないので、精度は辞書に依存されてしまうのが大きな特徴である。

例えば、「全文検索システム」のように、「全文」「検索」「システム」と区切ったり、「全文検索」「システム」と区切ったりできる複合語の場合、後者では「検索システム」ではヒットしなくなる。

「インターフェイス」、「インターフェース」、「インタフェイス」、「インタフェース」のような曖昧な外来語(カタカナ文字)も、全てが辞書に登録されていなければヒットしない。一般的でない固有名詞や新語などにも対応できない。

---

[1] kakasi - 漢字→かな(ローマ字)変換プログラム ： http://kakasi.namazu.org/

[2] 態素解析システム　茶筌(ChaSen) : http://chasen.aist-nara.ac.jp/

精度の高い抽出を行うには、常に単語辞書のメンテナンスが不可欠で、さらに日本語の文法の性質上、複合語の分割例からも分かるように必ず「検索漏れ」が生じてしまう。

3．2 N-gram 方式

Akao では N-gram という手法を採用している。N-gram とは、単語を「N 個の文字の組み合わせ」と捉えて、文字列を N 文字ずつ区切っていく方法である。

形態素解析のように辞書で分かち書きしないので辞書に依存せず、検索漏れも理論上はありえない。ただし、検索語によっては全く関係ない単語をヒットさせてしまう可能性もある。フリーなツールでは ngram[3] や morogram[4]などがある。

N-gram は、もともと確率･統計的自然言語処理の分野で用いられていたモデルで、ある文字列の中に N 文字の組み合わせがどのぐらいあるのかを調査する言語モデルである。N-gram は情報理論の創始者であるシャノン[2]によって考案された。

1993 年、長尾真･森信介の両氏によって、任意の文字列から N-gram のテキスト抽出をするソフトウェアが公開された[3]。当時のマシンスペックで比較的簡単に抽出できることから後の日本語テキスト分析に大きな影響を与えることになった。

1999 年、近藤みゆき氏、近藤泰弘氏が国語学･国文学（古典）研究に N-gram を応用した[4][5]。古典文学における表現の共通部分、独自部分について、N-gram を用いて総体的にみることによって、各作品の関連や言語性質に新たな視点を提供した。N-gram が単なる検索を超えて、文献そのものの解析にまで可能性を広げた例と言える。

2000 年、松井くにお氏らは、日本語の全文検索エンジンのデータ構造やテキスト抽出、検索手法を紹介し、テキスト抽出に形態素解析と N-gram を利用することの特徴を述べている[6]。

弊社では、2002 年に Akao の開発を開始した。当時、フリーな全文検索システムとして Namazu が形態素解析を採用して幅広い環境で導入されていた。Namazu との差別化を図るということも視野に入れ、N-gram 方式を採用した。

さて、N-gram の詳細なアルゴリズムを解説する。N-gram では、1 文字ずつ区切るのを 1-gram（uni-gram）、2 文字ずつを 2-gram（bi-gram）、3 文字ずつを 3-gram（tri-gram）と呼ぶ。簡単な例を紹介する。

「明日は晴れかな」という文字列だと、

1-gram：

「明」「日」「は」「晴」「れ」「か」「な」

2-gram：

「明日」「日は」「は晴」「晴れ」「れか」「かな」

3-gram：

「明日は」「日は晴」「は晴れ」「晴れか」「れかな」

といったように単語に分割される。さらに「晴れる」という検索語で検索するときも同様に N-gram において分割される。

1-gram：*「晴」「れ」「る」*

2-gram：*「晴れ」「れる」*

3-gram：*「晴れる」*

1-gram だと、*「晴」「れ」*の 2 単語がヒットする。2-gram だと、*「晴れ」*の 1 単語のみになる。3-gram だとこの場合ヒットしないことになる。

3．3 2-gram による Akao の実装

Akao では 2-gram を採用している。これにはインデックス作成時の実装上の実現性や検索精度などの理由がある。

それを説明するにあたって、まず文字コードについて簡単に述べる。一般的に半角英数字は ASCII コード、日本語の全角文字は JIS コードをシフトした SJIS／EUC コードが使われる。

ASCII コードは、8 ビットで表現される。しかし、標準では最上位ビットを使用せず 7 ビットで 128 文字（94 文字＋34 制御文字）が定義されている。

日本語の場合、ひらがな、カタカナ、漢字など使用する文字の数は英数字に比べて遥かに多いので、16 ビットで割り当てられる。16 ビットだと 2 の 16 乗だから全部で 65536 文字を表すことができる。

しかし、実際にはそれだけの文字数が必要ではなく、94*94=8836 空間の配列内で収まるように配置されている。ひらがなやカタカナなど漢字以外の文字 524 文字、常用漢字である第一水準文字 2965 文字、第二水準文字 3390 文字、計 6879 文字が定義されている。残りの 1957 文字は外字と言い、ベンダやユーザ定義用の拡張領域である。従って、8836 文字以内で日本語が定義されている。

ここでインデックスの作成時のテーブルについて考察する。インデックスを作成するにはファイルに File ID（以後 FID）を割り振り、どの文字（単語）がどのファイルに含まれるかを「文字－FID」の対応関係で表す。

例えば、FID1, FID2, FID3, FID4 のファイルがあるとして、FID1, FID3, FID4 に「晴れ」という単語、FID1,

---

[3] ngram : http://www.jaist.ac.jp/~shigeru/ngram-ja.html

[4] morogram : http://ya.sakura.ne.jp/~moro/resources/ngram/morogram.html

FID4 に「雨」という単語が含まれるとする。このとき、
　晴れ　－＞　FID1, FID3, FID4
　雨　　－＞　FID1, FID4
というようなインデックスファイルが作成される。

　計算を分かりやすくするため、ASCII＋SJIS／EUC を 10000 文字とする。1-gram の場合、1 文字毎に対応関係が必要なので、10000 文字あれば、10000 = 10K のテーブルが必要になる。

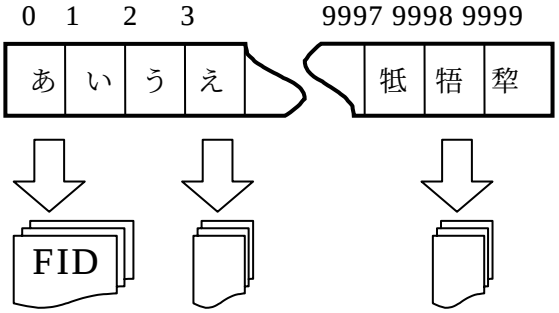

図2　1-gramのテーブル

次に 2-gram の場合、1 文字毎に 10000 文字の組み合わせが存在するから、10000 * 10000 = 100M のテーブルが必要になる。

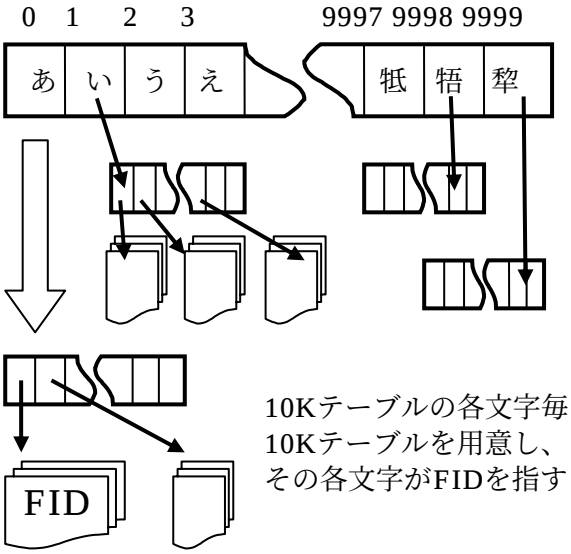

図3　2-gramのテーブル

　同様に、3-gram の場合は 10000 * 10000 * 10000 = 1T のテーブルが必要になる。

　ここで、用意する 1 テーブルにつき平均 4 バイトの FID のデータが存在すると仮定した場合、
　1-gram：4 * 10K　= 40KB
　2-gram：4 * 100M = 400MB
　3-gram：4 * 1T　= 4TB
の容量が必要になる。
さらに検索精度との関係を表 1 にまとめる。

表 1　N-gram と検索文字数の精度の関係

	1 文字	2 文字	3 文字	4 文字
1-gram	○	×	×	×
2-gram	○	○	×	×
3-gram	○	○	○	×

　まず、3-gram では、メモリの使用量が現実的に実装不可である。また仮に実装できたとして、3 文字までの検索の精度しか保証できない。
　次に、1-gram ではテーブルの容量が少なくてすむが、1 文字の検索精度しか保証しないため、ノイズが非常に多くなり、検索にならないと予想される。
　従って、ノイズがある程度は混じってしまうが、最もバランスの取れた 2-gram を採用することが妥当だと考えた。

　日本語のインデックス化に適した N-gram の長さを原田昌紀氏らが評価している[7]。漢字は 2 文字、ひらがなは 3 文字の検索語が多いことから、それぞれ 2-gram、3-gram でインデックス化を行えば良いと推定している。

　Akao では文字種（ひらがな、カタカナ、漢字）によって、N-gram の N を変更するようなことは行っていない。
　ただし、英文字に対しては、2-gram でインデックス化すると、ノイズが多過ぎるため、英文字のみスペースで区切って単語登録する方法で対応している。
　例）book を 2-gram でインデックス化
　　「b」「bo」「oo」「ok」「k」
　　boat, cook, hook, okra,,,
　　などの検索語で全てヒットしてしまう

4　Akao と Namazu の性能評価

　前章では検索アルゴリズムの違いについて述べた。形態素解析と N-gram の評価としては、一般論として表 2 のように言われている。
　本節では、実際に N-gram 型の Akao と形態素解析型の Namazu のインデックス処理、http（CGI）経由のリクエスト－レスポンスのスループットを測定し、その結果から両者のアルゴリズムの特徴を述べる。

　今回、テストを行ったサーバのマシンスペックは、
　CPU ： PentiumII　300MHz

MEM: SDRAM 512MB
HD : Urtra-ATA/33
OS : Redhat 8.0 (Kernel:2.4.18)
である。
ソフトウェアのバージョンは、
検索システム: Akao 1.15
Namazu 2.0.12
Web サーバ : httpd(apache) [5] 2.0.40
ファイル共有: samba[6] 2.2.5
である。

表 2 形態素と N-gram の性能比較

	形態素解析	N-gram
インデックス容量	小さい	大きい
インデックス速度	遅い	速い
検索速度	やや速い	やや速い
検索ヒット	少	多
検索精度	漏れる	漏れない

4. 1 インデックス処理のスループット

Akao や Namazu では、どのファイルにどのキーワードが含まれているかを判別するためのインデックスファイルを作成する。

検索対象とするファイル数やその容量が増加すればするほど、インデックスファイルの容量や処理時間が大きくなる。これらの 2 点について測定し、検証する。

Akao ではテキスト抽出を行う際に akb ファイルという一時ファイルを作成する。この akb ファイルには、各ファイル形式から抽出されたテキスト情報のみが Unicode で保存される。従って、Akao に必要なディスク容量はインデックスファイルと akb ファイルの容量の和になる。

また Namazu の環境設定は、デフォルト状態よりインデックス処理が高速になるように以下の設定がなされている。

Text::Kakasi の Perl モジュールを追加
$ON_MEMORY_MAX 5M -> 50M

またデフォルトでは PDF ファイルをインデックス化できないので PDF フィルタとして xpdf[7]の pdftotext を利用している。

4. 1. 1 Akao と Namazu のインデックス処理のスループットの比較

表 3、表 4 にテキスト･PDF ファイル形式の Akao と Namazu のインデックス処理の測定結果を示す。Akao に対して Namazu がどのぐらいの割合かを評価する (Namazu／Akao)。

ただし、Namazu の PDF ファイル形式のインデックス処理は、xpdf の pdftotext の処理(テキスト抽出)に大部分が依存する。しかしながら、xpdf を利用することが一般的なので、本論文では Namazu の PDF ファイルのインデックス処理として扱う。

まず、インデックス容量 (akb を含まない) について、テキスト、PDF ファイル共に同程度の値なので平均すると、100 ファイルで約 1.4 倍、1000 ファイルで約 0.9 倍、10000 ファイルで約 0.8 倍の容量となっている。ファイル数が増えれば、Akao より Namazu のインデックスファイルの容量が少し小さくなる傾向があることが確認できた。

次にインデックス処理の時間について、テキストファイルでは 100 ファイルで 1.1 倍、1000 ファイルで約 3 倍、10000 ファイルで約 12 倍、PDF ファイルでは 100 ファイルで 0.76 倍、1000 ファイルで約 3 倍、10000 ファイルで約 6 倍の処理時間となっている。ファイル数が増えれば、Akao より Namazu の処理時間が大幅に増加する傾向があることが確認できた。

これらは 3 章で紹介した検索アルゴリズムの違いによる特徴を顕著に表している。

Akao では、2-gram により対象のファイルに保存されている文字列を 2 文字ずつ単語に区切って、その全てに FID を対応付ける。一方、Namazu では、形態素解析により単語に区切るが、2 文字以上の単語やストップワード (助詞や記号など) もあるはずなので、FID を対応付ける単語の絶対数が Akao より少なくなるはずである。従って、絶対的にインデックスファイルの容量は N-gram の方が大きくなる。

さらに、処理時間については大きな差がみられた。インデックス処理のメイン作業が、単語に区切る、単語と FID を対応付ける、の 2 点だとすれば、N-gram の場合、機械的に処理を進めるので、その他のオーバーヘッドは極端に小さくなる。形態素解析の場合、単語辞書を検索、検索を効率化するために単語の並びを頻度順にソート、などのその他のオーバーヘッドが大きくなる。

[5] The Apache Software Foundation : http://www.apache.org/
[6] samba : http://www.samba.org/
[7] xpdf : http://www.foolabs.com/xpdf/

表 3：Akao のテキスト・PDF ファイル形式におけるインデックス処理時間

file count -- format	all file size	1file avg-size	index size	akb size	run time
100 -- txt	5.2 MB	52 KB	3.2 MB	**7.3 MB**	5 分 22 秒
1000 -- txt	51 MB	51 KB	21 MB	**72 MB**	19 分 4 秒
10000 -- txt	505 MB	50.5 KB	197 MB	**712 MB**	55 分 22 秒
100 -- pdf	14 MB	140 KB	1.1 MB	**1.5 MB**	5 分 41 秒
1000 -- pdf	290 MB	290 KB	9.4 MB	**24 MB**	16 分 33 秒
10000 -- pdf	2.9 GB	290 KB	71 MB	**240 MB**	2 時間 1 分 55 秒

表 4：Namazu のテキスト・PDF ファイル形式におけるインデックス処理時間

file count -- format	all file size	1file avg-size	index size	run time
100 -- txt	5.2 MB	52 KB	4.3 MB	6 分 3 秒
1000 -- txt	51 MB	51 KB	18 MB	57 分 13 秒
10000 -- txt	505 MB	50.5 KB	145 MB	11 時間 8 分 23 秒
100 -- pdf	14 MB	140 KB	1.6 MB	4 分 20 秒
1000 -- pdf	290 MB	290 KB	8.9 MB	50 分 49 秒
10000 -- pdf	2.9 GB	290 KB	56 MB	11 時間 58 分 48 秒

表 5：Akao の MS Office 系ファイル形式におけるインデックス処理時間

file count -- format	all file size	1file avg-size	index size	akb size	run time
100 -- doc	25 MB	250 KB	1.02 MB	**1.5 MB**	5 分 7 秒
1000 -- doc	96 MB	96 KB	3.8 MB	**7.2 MB**	5 分 52 秒
10000 -- doc	957 MB	95.7 KB	31 MB	**72 MB**	17 分 6 秒
100 -- xls	37 MB	370 KB	0.6 MB	**1.3 MB**	5 分 5 秒
1000 -- xls	105 MB	105 KB	2.3 MB	**6.7 MB**	5 分 43 秒
10000 -- xls	1.1 GB	110 KB	19 MB	**67 MB**	16 分 20 秒
100 -- ppt	32 MB	320 KB	0.656 MB	**0.728 MB**	5 分 6 秒

よって、今回の測定結果からは、大多数のファイルのインデックス処理が必要な場合、N-gram の方がパフォーマンスは優れていると言える。

また、表 5 では Akao の MS Office 系ファイルのインデックス処理の測定結果を示している。保存されているテキスト情報が異なるため、一概にテキストファイルや PDF ファイルとの比較はできない。

ここで Word の 100 ファイルと 10000 ファイルのインデックス処理時間に着目する。100 ファイルと比べて 10000 ファイルの処理時間は約 3 倍となっている。同様に、表 3 よりテキストファイルの 100 ファイルと 10000 ファイルの処理時間の増加率は約 11 倍、PDF ファイルの場合、約 20 倍である。

Word ファイルと Excel ファイルの増加率は同程度であることから、MS Office 系ファイルの方がテキストファイルや PDF ファイルに比べてインデックス処理が速い傾向にあると分かる。

4．1．2 Akao のインデックス処理

簡単に Akao のインデックス処理に関して紹介する。Akao では以下の 3 つの処理を経てインデックス化がなされる。

ⅰ) makao：対象ファイルからテキストを抽出し、一時ファイル(akb)を作成
ⅱ) nakao：一時ファイルから 5000 ファイル毎に 1 つのインデックスファイルを作成
ⅲ) gakao：複数のインデックスファイルを 1 つのインデックスファイルに再構成

Akao のインデックスファイルそのものの容量は Namazu と比較してもさほど大きいものではないが、実際の運用には一時ファイル(akb)が必要になるのでそのファイル容量も加算する必要がある。表 3、表 5 より、idx＋akb のファイル容量を各ファイル形式別にまとめてみると、インデックス対象の全ファイル容量に比べて、テキストファイルで 2 倍、PDF や MS Office 系ファイルで 0.1 倍程度必要であることが確認できた。

テキストファイルのみ容量が増加しているようにみえる。テキストファイル中には、ASCII コード(半角英数字)と EUC コード(全角文字)がおよそ半分ずつ保存されている。ここで、akb ファイルは Unicode(UCS-2)で保存するため、1 文字当たり 16 ビット必要になる。ASCII コードは 8 ビット、EUC コードは 16 ビットだから、ASCII コードの部分のみ容量が 2

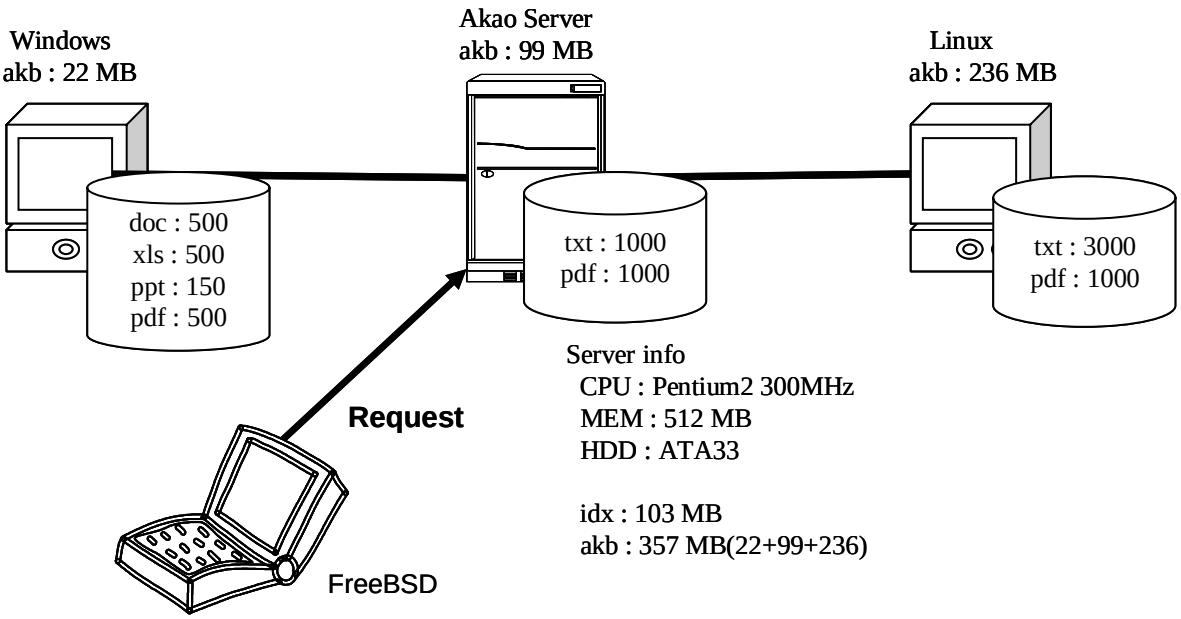

図5　リクエストーレスポンス検証ネットワーク

倍になったためである。その他のファイル形式では、テキスト情報以外にフォント･書式･図形などの情報が容量の大半を占めるために見た目上はこのような結果になったわけである。

また、PDF ファイルの処理時間が他のファイル形式に比べて長い。PDF ファイルでは、あるサイズごとにヘッダで区分して、テキスト情報を FlateDecode 等で圧縮して保存している。インデックス化には圧縮展開してテキスト抽出する必要があるため、処理時間が他のファイル形式に比べて最も必要になる。

さらに 10000 件の PDF ファイルのインデックス処理のサーバ負荷状況を sar コマンドで取得した。図 4 はその負荷状況をグラフ化したものである。

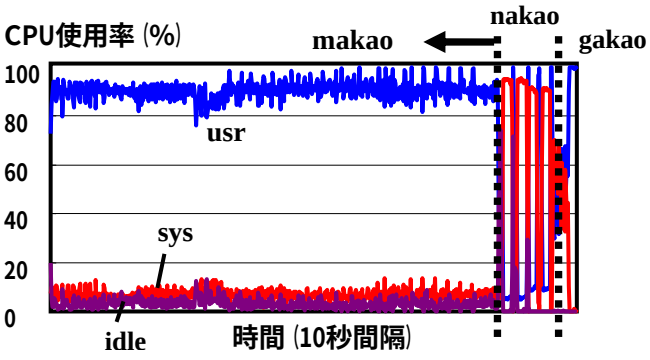

図4　10000-pdf の負荷状況

Akao のインデックス処理は makao(テキスト情報抽出) -> nakao(インデックス書き出し) -> gakao(インデックスの一時ファイルを統合処理) の順番に実行される。

sys 領域の使用率が増加している部分は nakao の処理を表す。そして、その後再度 usr 領域の使用率が増加している部分が gakao の処理になる。全体の処理の約 85%が makao の処理である。

4．2　http のリクエスト ― レスポンスのスループット

Akao は Web サービスの一つである。Web ブラウザから検索リクエストを送信し、Web サーバの CGI 経由で Akao の検索モジュール sakao が呼び出され、出力を HTML 形式で生成する。Namazu も同様に CGI 経由で Web 上から検索できる。

本節では、Web サービスとして、Akao と Namazu の処理能力がどの程度かを検証する。

Akao のテスト用に図 5 のようなネットワークを構築した。Akao サーバを中心に Windows と Linux(Redhat) のファイルサーバをマウントし、Akao サーバ自身のローカルディスクも含めてインデックス化を行った。Namazu では、こういった構成は組めないので、ローカルディスクに 9000 件の PDF ファイルをインデックス化してテストを実行する。

検索リクエストは別の負荷マシン(FreeBSD)から送信する。ベンチマークツールには、apache 付属の ab(apache bench)を用いる。ab は C 言語で記述されていてパフォーマンスも良く、http の GET/POST メソッドにも対応している。最低限のレポート機能も揃っている。

4．2．1　Akao と Namazu のレスポンススループットの比較

表 6：Akao の http リクエストーレスポンスレポート

Process	Connecton(Req)	All-Run-Time	Error	Req/Sec	MinWait	MaxWait
1chara-Search hit:5000	**100**	**1131.389**	**0**	**0.09**	**371.765**	**1131.356**
	250	--	-	--	--	--
	500	--	-	--	--	--
2chara-Search hit:77	**100**	138.504	**0**	0.72	34.208	138.477
	250	346.428	**0**	0.72	116.611	340.16
	500	673.381	**0**	0.74	188.172	452.637
3chara-Search hit:44	**100**	135.799	**0**	0.74	50.133	135.778
	250	339.56	**0**	0.74	112.614	330.03
	500	670.909	**0**	0.75	192.202	458.6931
and-Search hit:126	**100**	143.719	**0**	0.7	49.409	143.695
	250	364.335	**0**	0.69	125.815	351.886
	500	718.729	**0**	0.7	192.202	487.549
or-Search hit:1641	**100**	254.477	**0**	0.39	114.984	254.015
	250	598.186	**0**	0.42	184.421	579.57
	500	1121.824	**0**	0.45	256.203	848.406

表 7： Namazu の http リクエストーレスポンスレポート

Process	Connecton(Req)	All-Run-Time	Error	Req/Sec	MinWait	MaxWait
1chara-Search hit:665	**100**	9.851	**0**	10.15	0.814	9.825
	250	26.78	**0**	9.33	2.152	23.212
	500	51.21	**0**	9.76	15.038	50.044
2chara-Search hit:315	**100**	9.801	**0**	10.2	0.42	9.774
	250	25.533	**0**	9.79	3.407	25.505
	500	49.579	**0**	10.19	16.034	43.018
3chara-Search hit:40	**100**	11.087	**0**	8.47	1.769	11.781
	250	28.369	**0**	8.81	4.239	24.498
	500	55.17	**0**	9.06	19.138	54.207
and-Search hit:110	**100**	10.429	**0**	9.59	0.713	10.4
	25	26.354	**0**	9.49	9.043	25.86
	500	54.339	**0**	9.2	16.393	52.326
or-Search hit:1358	**100**	11.562	**0**	8.65	0.739	11.536
	250	31.47	**0**	7.94	5.175	31.451
	500	60.778	**0**	8.23	19.856	57.27

Connection(Req) ： 同時接続数、1 接続につき 1 リクエスト
All-Run-Time ： 処理が終了するまで総時間（秒）
Error ： レスポンスコード 200 番台が返ってこない
Req/Sec ： 1 秒間あたりのリクエスト処理数
Min(Max) Wait ： 最小（最大）レスポンス待ち時間（秒）
＊ 数値が記入されていない箇所はそれ以上の測定の必要性がないと判断

表 6、7 にレスポンスの測定結果を示す。クライアント側から同時接続数を 100、250、500 と変更して、そのときの処理時間、1 秒間あたりのリクエスト処理数、エラー数、最小ー最大待ち時間を測定した。

検索は Akao、Namazu とも同じ検索語で実行している。インデックスや検索アルゴリズムが異なるため、検索ヒット数には大きな違いがある。

まず、両者の“Req/Sec”の項目に注目してほしい。Req/Sec とは、1 秒間当たりに処理したリクエストの数を表している。Web サービスの評価には最もよく用いられるパラメータである。種々の検索において、Akao

表 8：Akao と Namazu の多数ヒットの http リクエストーレスポンス比較

Process	Connecton(Req)	All-Run-Time	Error	Req/Sec	MinWait	MaxWait
Akao or-Search hit:5000	**100**	**676.705**	**0**	**0.15**	**280.590**	**676.676**
	250	**1393.568**	**0**	**0.18**	**192.202**	**1355.499**
	500	--	-	--	--	--
Namazu or-Search hit:5236	**100**	21.638	**0**	4.62	0.851	21.621
	250	53.634	**0**	4.66	21.011	53.619
	500	110.85	**0**	4.51	31.98	110.82

表 9：namazu コマンドと sakao コマンドの処理時間（秒）の比較

Process	Transaction	namazu (hit)	sakao(hit)	sakao-no-grep
1chara-search	100	6.927 (665)	1015.108 (5000)	899.126
2chara-search	100	6.649 (315)	11.809 (77)	9.948
3chara-search	100	7.153 (40)	7.958 (44)	7.177
and-search	100	7.135 (110)	19.355 (126)	14.935
or-search	100	8.274 (1358)	183.053 (1641)	125.257
or2-search	100	17.484 (5326)	561.516 (5000)	380.263

では、最も高い値で 0.7〜0.8 程度であるのに対し、Namazu では 8〜10 の値がみられる。つまり、Akao の 10 倍以上のパフォーマンスが確認された。

それぞれの値についてさらに詳細にみていく。Akao では N-gram の特性上、1 文字検索が最も負荷のかかる処理となる。1 文字の場合、ヒット数が極端に増大するからである。さらに or 検索が他の検索に比べてパフォーマンスが悪い。ヒット数が 1641 件と他に比べて多いことから、ヒット数が増えれば増えるほど、N-gram では処理時間が長くなる傾向にある。

一方、Namazu でも同様のことが言えるかどうかを確認してみる。表 8 は両者とも or 検索で 5000 件程度ヒットするように検索をかけたときの測定結果である。

Namazu の Req/Sec をみると 4.5 程度と、他の検索に比べて約 0.5 倍に処理能力の低下が確認できた。

Akao では、1 文字検索ほどレスポンスは悪くないが、Req/Sec が 0.15 程度と、他の検索に比べて約 0.2 倍に低下した。

形態素解析でもヒット数が増加すると、パフォーマンスは落ちるが、N-gram ほど顕著ではないことが分かる。

http（CGI）経由のレスポンス比較においては、Akao と Namazu で文字通り、桁違いのレスポンス能力の差が確認された。

4．2．2 コマンドベースの Akao と Namazu のレスポンススループットの比較

Namazu では、CGI を C 言語で記述しているが、Akao では perl で記述している。CGI の構成方法に違いがあるため、それがレスポンス能力に影響を与えた可能性がある。両者に大きな差が確認されたので、それが本当に形態素解析と N-gram の差を表しているものかどうかを検証する。

Namazu では検索コマンドに namazu を、Akao では sakao を用いて検索を実行する。コマンドラインから検索コマンドを実行してその処理時間を比較する。

テスト方法は、それぞれの検索コマンドをスクリプトで 100 回実行し、その処理時間を time コマンドで取得する。測定結果を表 9 に示す。

3 文字検索の項目を注目してほしい。namazu と sakao では、どちらの検索ヒット数も 40 件程度で、ほぼ同じ処理時間になっている。

sakao は、2 文字検索、and 検索においても、CGI 経由のレスポンス時間に比べて極めて処理時間が短いことが確認できる。1 文字検索、or 検索においても、かなりの処理時間の減少がみられる。しかし、ヒット数が増加すれば処理に時間がかかる点は、やはり N-gram の特徴だと言える。

さらに純粋な N-gram のパフォーマンスを測定するために“sakao-no-grep”の項目も追加した。

sakao の実装では、N-gram の検索によりヒットした全ファイルに対して、本当にそのファイルにユーザが指定した検索語があるかどうかを調べるために、シーケンシャルに Unix コマンドの“grep”相当の処理を実行している（便宜上、grep と呼ぶ）。この grep でヒットしたときにそのファイルに重み付けし、検索結果の上位にくるようにヒット率に反映している。つまり、100 同時接続で 40 件ヒットした場合、4000 回 grep していることになる。

これは検索精度を上げるための実装であるが、同

時にパフォーマンスを劣化させることには変わりないのでこの grep をはずした結果も測定した。

当たり前の結果であるが、全ての検索において速度向上がみられた、特に and･or 検索に 30%程度の速度向上がみられた。現時点の Akao では対応していないが、次期バージョンでは、この grep のあり･なしをユーザ選択オプションにする予定である。

まとめると、namazu コマンドのレスポンス能力は検索文字列およびヒット数に依らず、高速である。一方、sakao コマンドでは、1 文字検索やヒット数が多くなる検索においては、レスポンス能力が落ちるが、同程度のヒット数の検索においては、namazu コマンドと比較しても大きな差がみられないことが分かった。

また、Namazu と Akao で CGI の処理能力に大きな差があることが確認された。Namazu では、CGI 処理時間が namazu コマンドの 0.5 倍以下の処理速度であるのに対して、Akao では、例えば、3 文字検索の場合、sakao コマンドは 8 秒程度であることから、CGI 処理時間は 127(135 - 8)秒と、検索コマンドの最大 15 倍以上の CGI 処理時間が必要になる。前節で測定した http のリクエスト－レスポンス比較で、Namazu と Akao で 10 倍もの差がみられたのには、この CGI の処理能力の差が影響したと考える方が妥当である。

5. まとめと今後の展望

本論文では、全文検索システムの検索アルゴリズムに着目して、形態素解析の Namazu、N-gram の Akao について性能評価を行った。具体的に、インデックス処理、検索レスポンスについて定量的に評価した。

インデックス処理時間は、Akao の方が速く、大多数ファイルのインデックス化にその有効性を示した。

検索レスポンスは、Namazu の方がやや速いが、運用方法によっては Akao でもそう大差ないことが確認された。例えば、1 文字検索など運用上ありえない、またヒット数の上限を 5000 件ではなく 1000 件にすれば性能向上が期待される。

また、検証の結果、Akao の機能改善の必要性も確認できた。次期バージョンでは、CGI の改善、検索のオプション選択（grep あり･なし）による細密検索と速度重視検索の実装などに対応する予定である。

弊社、データ変換研究所では、「データ変換」による社会貢献を理念としている。テキスト抽出のフィルタ開発に始まり、そのフィルタを生かした 1 つの形態が Akao である。検索市場において、オールマイティな全文検索システムは存在しない。シェアの高低はあるが、数十もの全文検索システムが存在することからも明白である。

今後は、ユーザのニーズに応じた選択肢の 1 つとして、Akao の位置付けを明確にすることが最大の課題である。

参考文献

[1] 全文検索システム Namazu : http://www.namazu.org/

[2] C.E.Shannon : A mathematical theory of communication, Bell System Tech.J.,Vol.27, pp.379-423, pp.623-656, 1948

[3] 長尾眞･森信介:“大規模日本語テキストの n グラム統計の作り方と語句の自動抽出”, 情報処理学会 自然言語処理 研究報告, NL-96, pp 1-8, 1993

[4] 近藤みゆき:“平安時代和歌資料における特殊語彙抽出についての計量的研究と利用ツールの公開 ―古今和歌集の歌語と表現のジェンダー性について―”, 科学研究費特定領域研究 人文科学とコンピュータ 研究成果報告書 ―コンピュータ支援による人文科学研究の推進―, 1999

[5] 近藤泰弘･近藤みゆき:“平安時代古典語古典文学研究のための N-gram を用いた解析手法”, 言語情報処理学会第 7 回年次大会 発表論文集, 2001

[6] 松井くにお, 他:“全文検索エンジン”, 情報の科学と技術, 50(1), pp 9-13, 2000

[7] 原田昌紀, 他:“Unicode を用いた N-gram 索引の一実現方式とその評価”, 情報処理学会 情報学基礎 研究報告, 57-17, 1999