オンライン学習による新聞記事からのキーワード抽出とリンク

小山内一由† 杉浦 遼一† 青柳 拓哉† 岡部 正幸†† 梅村 恭司†

† 豊橋技術科学大学 情報・知能工学系 〒 441–8580 愛知県豊橋市天伯町雲雀ケ丘１－１
†† 豊橋技術科学大学 情報メディア基盤センター 〒 441–8580 愛知県豊橋市天伯町雲雀ケ丘１－１
E-mail: †{osanai,sugiura,aoyan}@ss.cs.tut.ac.jp, ††okabe@imc.tut.ac.jp, †††umemura@tut.jp

あらまし 自然言語文書からキーワードとなる文字列を抽出する方法の 1 つに，多種類の文書頻度を特徴量として機械学習分類器を使いキーワード抽出をする方法がある [1]．Ogata の研究では SVM を使用したが，SVM のスケーラビリティに問題があり，また応用例が明確でないという課題があった．本研究では，分類器としてオンライン学習アルゴリズムを使用することでスケーラビリティ改善できることを確認し，応用システムを開発した．学習アルゴリズムには Passive Aggressive を使用し，応用システムとして多数の新聞記事からキーワードを抽出して同じキーワードを持つ記事をリンクする記事ビューアを作成した．

キーワード キーワード抽出，機械学習，オンライン学習，Adaptation

Keyword Extraction and Linking from Newspaper Articles

Kazuyoshi OSANAI†, Ryoichi SUGIURA†, Takuya AOYANAGI†, Masayuki OKABE††, and Kyohi UMEMURA†

† Department of Computer Science and Engineering, Toyohashi University of Technology
1-1 Hibarigaoka, Tenpakucho, Toyohashi, Aichi, 441–8580 Japan
†† Information & Media Center, Toyohashi University of Technology
1-1 Hibarigaoka, Tenpakucho, Toyohashi, Aichi, 441–8580 Japan
E-mail: †{osanai,sugiura,aoyan}@ss.cs.tut.ac.jp, ††okabe@imc.tut.ac.jp, †††umemura@tut.jp

1. はじめに

文書からキーワードとなる文字列を自動で見つけ出す方法について，すでに多くのものが提案されてきている．その 1 つに日本語の文書から複合語を抽出する方法について扱った Ogata らの研究がある [1]．Ogata らの用いた方法の特徴の 1 つとして，キーワードの抽出にあたって文法の情報を必要としないため，形態素解析器などを用いないシンプルなキーワード抽出器を構成することができるというものがある．

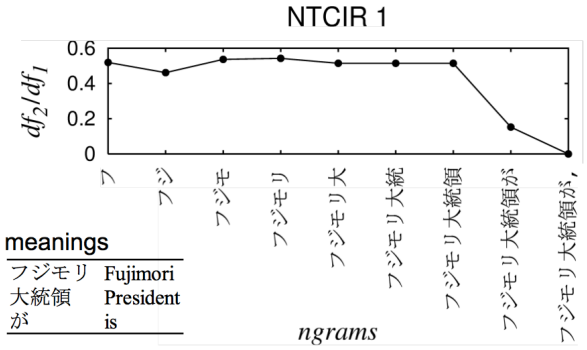
具体的な方法としては，文字列の文書頻度などから得た値から教師データを構築し，分類器に学習させることで複合語を抽出するというものである．しかし Ogata らの研究 [1] では，分類器に使った Support Vector Machine(SVM) のスケーラビリティの問題によって，扱えるデータの量に制限があった．また具体的にどういった場面で使えるのか，応用例が明確でなかった．

そこで本研究では Ogata の研究について，1) 分類器のスケーラビリティの改善，2) 具体的な応用システムの開発，を試みる．

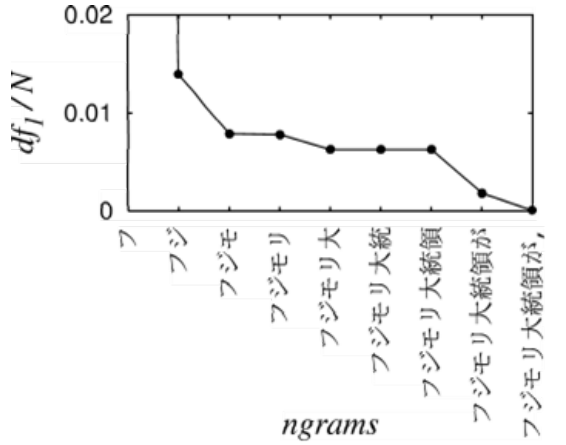
分類器のスケーラビリティの問題に対処するにあたって，より省メモリで高速な分類アルゴリズムを使用することが考えられる．その要求を満たすと期待されるものにオンライン学習と呼ばれる一群の機械学習アルゴリズムがある．オンライン学習はデータを 1 つ 1 つ見て学習モデルを更新していく学習アルゴリズムの総称で，あくまで 1 つ 1 つデータを処理するためデータを一度にメモリに展開する必要がない．この特徴によりメモリの使用量を抑えることができる．また一般にオンライン学習は収束が速いとされる．

オンライン学習アルゴリズムにおいて今までいくつかのものが提案されており，その 1 つに Passive Aggressive と呼ばれるアルゴリズムがある [2]．Passive Aggressive アルゴリズムの，特に PA-I というアルゴリズムは Ogata[1] が用いた SVM に近い性能を持つ．

以上から，本研究は Ogata の方法 [1] のスケーラビリティを SVM の代わりに PA-I[2] を使用することで改善できることを確認し，具体的な応用システムを開発するものである．本論文では，結果としてスケーラビリティが改善されたことを示し，また開発した応用システムを紹介する．



(a) Adaptation の変化



(b) df/N の変化

図 1　語の境界での変化

2. Adaptation とその他の推定量

Adaptation の定義と推定

まず N をコーパスに含まれる文書数とする．文書の確率変数を D とすると，$P(tf(D,s) \geq 1)$ は，ランダムに選ばれた文書 D について，ある文字列 s が 1 つ以上出現する確率である．$P(tf(D,s) \geq 1)$ の確率変数 D に対しての期待値は $df(s)/N$ から推定できる．$df(s)$ は文字列 s を 1 つ以上含む文書の数である．本論文で扱う $df(s)$ は，単語ではなく文字列の関数である．

文字列 s に対する Adaptation は，$P(tf(D,s) \geq 2 | tf(D,s) \geq 1)$ の確率変数 D に対して求められる期待値である [3]．この確率は，$df_2(s)$ を文字列 s が 2 つ以上含む文書の数としたとき，$df_2(s)/df(s)$ から推定できる．

$$
\begin{aligned}
&E\{P(tf(D,s) \geq 2) | tf(D,s) \geq 1\} \\
=&E\{(P(tf(D,s) \geq 2 \& tf(D,s) \geq 1)/P(tf(D,s) \geq 1)\} \\
=&E\{P(tf(D,s) \geq 2)/P(tf(D,s) \geq 1)\} \\
\simeq& (df_2(s)/N)/(df(s)/N) \\
=&df_2(s)/df(s)
\end{aligned}
$$

Church はほぼすべての英単語が高い Adaptation を持つことを示した [3]．日本語の文書については Takeda らの研究 [4] があり，日本語の専門用語も同様に高い Adaptation を示すことを報告している．図 1(a) は語の境界で Adaptation が大きく変化していることを示している．日本語の複合語の抽出について Adaptation を用いたものが Ogata らの研究 [1] であり，Adaptation を複合語の抽出に役立たせることができることを示している．

Adaptation の問題

キーワードの判別にあたって，Adaptation がうまく働かない状況が考えられる．例えば文書中に 1 度しか出現しないキーワードがあった場合，df_2 が 0 になるため df_2/df(Adaptation の推定値) も必ず 0 になってしまう．

そのような場合にキーワード判別のヒントを学習器に与えるために，Ogata ら [1] は df_2/df に加えて df/N も用いている．図 1(b) に示すように，df/N も語の境界で変化する性質がある．また図からも分かるとおり，かな文字 1 文字などはあまりに高い df を持っているので df/N の値も高くなっていて，それらをキーワードから除外するヒントを与える意味もある．

表 1　文字を加える操作の例

... 対 す る 中 国 の 反 応...　(1)

(2)

(3)

(1)　文字列 s ．df, df_2 をとる

(2)　s に直前の 1 文字を加えた文字列．${}^{+}df, {}^{+}df_2$ をとる

(3)　s に直後の 1 文字を加えた文字列．df^{+}, df_2^{+} をとる

表 2　文書頻度の例

	df	${}^{+}df$	df^{+}
中国	7959	259	1420
る中国	259	(省略)	(省略)
中国の	1420	(省略)	(省略)

表 3　教師データファイルの一部 (小数第 4 位を四捨五入)

文字列	$\frac{df}{N}$	$\frac{df^{+}}{N}$	$\frac{{}^{+}df}{N}$	$\frac{df_2}{df}$	$\frac{df_2^{+}}{df^{+}}$	$\frac{{}^{+}df_2}{{}^{+}df}$	Y/N
中	0.494	0.046	0.025	0.571	0.509	0.064	no
中国	0.046	0.011	0.003	0.509	0.306	0.097	yes
中国の	0.011	0.000	0.001	0.306	0.192	0.063	no
中国の反	0.000	0.000	0.000	0.192	0.000	0.000	no
中国の反応	0.000	0.000	0.000	0.000	0.000	0.000	no

3. 推定量から得られる境界情報

使用する統計値

Adaptation と df/N の変化を機械学習のためのデータに落としこむために，いくつかの統計値を用いる．まず ${}^{+}df, {}^{+}df_2$ は，文字列の直前にある 1 文字を加えたときの df, df_2 とする．同様に df^{+}, df_2^{+} は，文字列の直前にある 1 文字を加えたときの df, df_2 とする．

文字を加える操作の例を表 1 に示す．ここで,「中国」が現在見ている文字列で，これについて df, df_2 を計算するとする．

そのとき「中国」に対する ${}^{+}df, {}^{+}df_2$ は「る中国」についての df, df_2 となる．同様に「中国」に対する df^{+}, df_2^{+} は「中国の」についての df, df_2 となる．

そのことを表 2 に示す.「中国」の ${}^{+}df$ は「る中国」の df に等しい．同様に「中国」の df^{+} は「中国の」の df に等しい．同じ関係が $df_2, {}^{+}df_2, df_2^{+}$ についてもある．

表 3 は学習器に学習させた教師データファイルの一部抜粋である．教師データについての詳細は後述するが，ここでも文書頻度の間の関係が表われていることが分かる．df/N のカラムのセルと，そのすぐ右上の df^{+}/N カラムのセルを見ると同じ値になっている．同様に df_2/df カラムのセルと，そのすぐ右上の df_2^{+}/df^{+} カラムのセルの値は等しい．一例を挙げると，文字列「中」の df^{+}/N と，それに直後の 1 文字を追加した「中国」の df/N が等しい．

また，表 3 のカラム (df_2/df のカラム) に着目すると，図 1(a) に示した Adaptation の特性も表れていることが分かる．表の中でキーワードと言える文字列は「中国」だが,「中」-「中国」間の Adaptation の差が $0.571 - 0.509 = 0.062$ なのに対して,「中国」-「中国の」間の Adaptation の差は $0.509 - 0.306 = 0.203$ である．図 1(a) に示した Adaptation はキーワードの境界で大きく変化するという特性が表 3 にも表れていると言える．

Adaptation などから得られる情報

図 2(a) は横軸に df_2/df(文字列の Adaptation)，縦軸に ${}^{+}df_2/{}^{+}df$(文字列に直前の 1 文字を加えたときの Adaptation) を取ったとき，いくつかの専門用語と単なる文字列の断片をプロットしたものである．専門用語は図の右上に多くプロットされ，文字列断片はもう少し広い範囲に散らばってプロットされている．この図から，専門用語と文字列断片が異なる特性を持っていることがわかる．分類はこの違いを利用して行うことになる．ただし，両者がオーバーラップする右上の部分においては，完全には区別できないことも図から言える．

また，縦軸を df_2^{+}/df^{+} とした場合も図 2(a) と同様の特性が得られる [1].

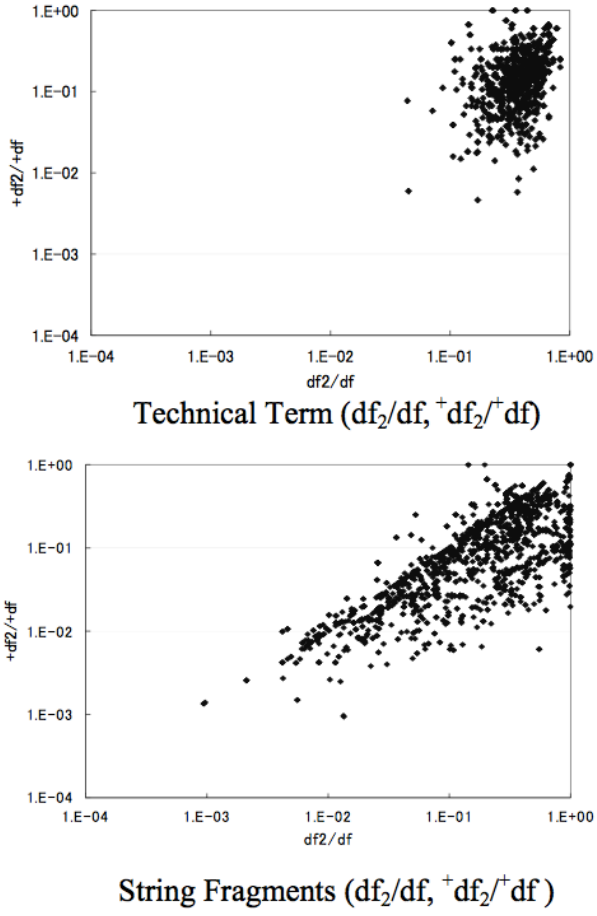
図 2(b) は ${}^{+}df/N$ と df/N について同様のプロットを行ったものである．図 2(a) よりもオーバーラップする部分が大きいものの，専門用語と文字列断片で特性の違いが見られる．

また，縦軸を df^{+}/N とした場合も図 2(b) と同様の特性が得られる [1].

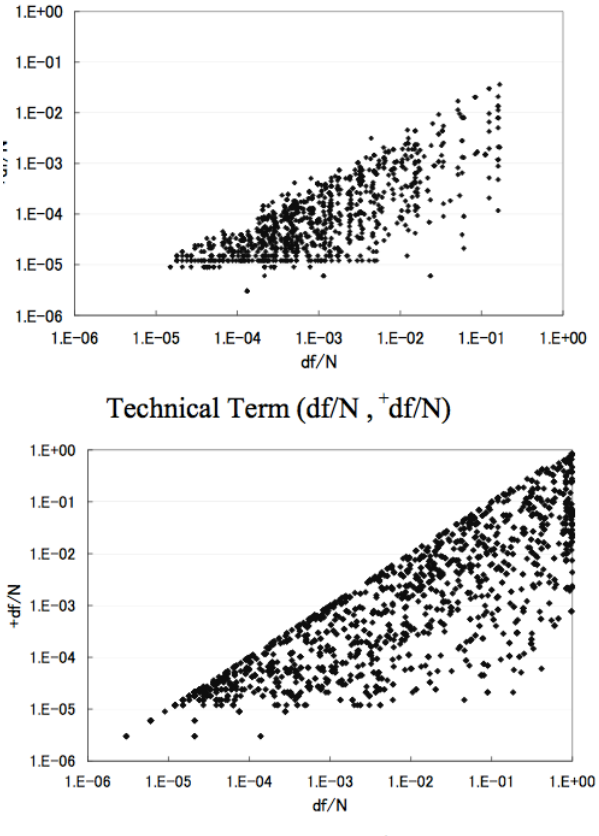
4. 学 習 方 法

機械学習と扱うデータ

ここでは，機械学習とは新たな入力に対する出力を予測するものだとする．つまり，新たな入力 x が与えられたとき出力 y を返す関数 $y = f(x)$ を決めることと言える．この論文ではキーワードか否かの判別を行うので y は yes か no の 2 値になる．また，x として以下の 2 種類のデータを作り，それぞれについて実験した．



(a) 専門用語と文字列断片の Adaptation(Ogata et al. 2004)



(b) 専門用語と文字列断片の df/N(Ogata et al. 2004)

図 2　専門用語と文字列断片のプロット

$$\frac{df}{N}, \frac{{}^{+}df}{N}, \frac{df^{+}}{N}, \frac{df_2}{df}, \frac{{}^{+}df_2}{{}^{+}df}, \frac{df_2^{+}}{df^{+}} \tag{1}$$

$$\begin{aligned}&\frac{df}{N}, \frac{{}^{+}df}{N}, \frac{df^{+}}{N}, \frac{df_2}{df}, \frac{{}^{+}df_2}{{}^{+}df}, \frac{df_2^{+}}{df^{+}},\\ &\left(\frac{df}{N}\right)^2, \left(\frac{{}^{+}df}{N}\right)^2, \left(\frac{df^{+}}{N}\right)^2,\\ &\left(\frac{df_2}{df}\right)^2, \left(\frac{{}^{+}df_2}{{}^{+}df}\right)^2, \left(\frac{df_2^{+}}{df^{+}}\right)^2, 1\end{aligned} \tag{2}$$

この (1)，(2) は確率の推定値からなるベクトルである．(1) の実際の値は表 3 に示している．(2) は (1) の値に加えて，それらを 2 乗した値と，定数 1 を付加したものである．

(2) は非線形なものとして (1) を扱おうとするものである．本論文で用いた学習アルゴリズムは線形モデルを扱うが，図 2 から考えて，今回の判定が線形分離可能とは言いきれない．線形モデルの学習アルゴリズムを用いて非線形なデータを扱う方法としてカーネルトリックがあるが，本論文で用いた機械学習フレームワークは実験を行った時点ではカーネル関数の使用に対応していない．機械学習フレームワークはオープンソースソフトウェアのためソースコードを書き換えることも可能だが，実装やアップデートへの追従にかかる労力を鑑み (2) のような次元数を増やす方法を試みることとした．

オンライン機械学習

オンライン機械学習とは，訓練例を 1 つ見るごとに，上記で説明した関数 $y = f(x)$ を決めるパラメータを変更するような機械学習のことである．ここでは関数 $y = f(x)$ を決めるパラメータは 1 つの実数ベクトル $\boldsymbol{w}$ であるとし，y は+1 か-1 のいずれかの 2 値分類として説明する．

単純なオンライン学習アルゴリズムを疑似コードで表したものが以下である．

```
入力：{(x_i, y_i)} (i = 1..n)   y_i ∈ {1, −1}     //1, −1の2値分類
w = {0, 0, .., 0}                 // 初期化
for i in [1, .., n]               // 訓練データを順番に取ってくる
  s := y_i w^T x_i                //w^T x_iの符号が現在の予測の正誤指標
  if(s ≦ 1)                       // 誤り(負) or 自信がない(小さな正)
    w = w + α y_i x_i             // 特徴ベクトルを足す
  endif
until(訓練データが全部正解)
ただし，α は十分小さい正の値
```

このコードでは，$\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x_i}$ の符号を現在の予測とし，それと y_i の積をとることで予測の正誤を表現している．予測が誤っているか，予測に自信がないとき $\boldsymbol{w}$ の更新を行っている．1 つの訓練例 $(\boldsymbol{x_i}, y_i)$ を見るごとにパラメータ $\boldsymbol{w}$ を更新しているのでオンライン学習になっていると言える．

$\boldsymbol{w}$ の更新式は直感的には次の式で説明できる．

$$\begin{aligned}&\boldsymbol{w}_{\boldsymbol{i+1}}^{\boldsymbol{T}}\boldsymbol{x_i}\\ =&\left(\boldsymbol{w_i} + \alpha y_i \boldsymbol{x_i}\right)^{\boldsymbol{T}} \boldsymbol{x_i}\\ =&\boldsymbol{w_i}^{\boldsymbol{T}}\boldsymbol{x_i} + \alpha y_i \boldsymbol{x_i}^{\boldsymbol{T}}\boldsymbol{x_i}\end{aligned}$$

第 1 項 $\boldsymbol{w_i^T x_i}$ は直前の誤った予測結果である．また $\boldsymbol{x_i^T x_i}$ は 2 乗なので常に正になり，第 2 項の符号は正解 y_i の符号になる．誤った予測の符号を正しい符号の方向に修正していくので学習ができる．

Passive Aggressive アルゴリズム

ここで，更新の幅を表すパラメータ α は大きすぎるとマージンがうまくとれなくなり，小さすぎると収束が遅くなってしまう．そこで，刻み幅 α を適切に設定することが求められる．

今回用いたオンライン学習アルゴリズム Passive Aggressive[2] は刻み幅を都度計算するアルゴリズムである．疑似コードで表すと以下になる．

```
入力：{(x_i, y_i)} (i = 1..n)   y_i ∈ {1, −1}     //1, −1の2値分類
w = {0, 0, .., 0}                 // 初期化
for i in [1, .., n]               // 訓練データを順番に取ってくる
  s := y_i w^T x_i                //w^T x_iの符号が現在の予測の正誤指標
  if(s ≦ 1)                       // 誤り(負) or 自信がない(小さな正)
    τ = l(x_i, y_i; w)/x_i x_i^T  // 適切な刻み幅を計算し
    w = w + τ y_i x_i             // 特徴ベクトルを足す
  endif
until(訓練データが全部正解)
ここで，l(x_i, y_i; w) = max(0, 1 − y w^T x)
```

今回用いたものは Passive Aggressive アルゴリズムの中でも PA-I[2] というもので，$\boldsymbol{w}$ を以下の式で更新する．τ の計算方法が上記の疑似コードと少し異なっている．ここで C は正の値をとり，高い値であるほど積極的な更新を行う．

$$\begin{aligned}\boldsymbol{w_{i+1}} &= \boldsymbol{w_i} + \tau_i y_i \boldsymbol{x_i}\\ \tau_i &= \min\left\{C, \frac{l_i}{||\boldsymbol{x_i}||^2}\right\}\\ l_i &= \max\left\{0, 1 - y_i(\boldsymbol{w_i} \cdot \boldsymbol{x_i})\right\}\end{aligned}$$

これを一種の SVM として見ることができることは [2] で示されている．上式のように単純な更新で実装できるアルゴリズムでありながら SVM と同様な結果を得ることができる．

5. システム構成

コーパス

コーパスとしては毎日新聞の 1991 年から 1996 年の記事データを選択した．これを選択した理由は，第一に文書の数が多いためスケールアップを目指す今回の目的に適合すること，第二に多くの文書がキーワードとなる文字列を持っていると考えられるためキーワード抽出の問題に適していると判断したことである．

コーパスは次の 3 つの用途で，それぞれコーパスの別の部分を使用した．1) 学習データ，2) 文書頻度の初期値を作るためのデータ，3) システムの記事データベースに挿入して記事ビューアに表示するデータ．

1 の学習データとしては表 3 に代表されるデータを 1 万行用意した．そのうち Y/N カラムが yes のデータは 100 行で，ほかは no のデータである．

2 の文書頻度の初期値を作るためのデータとは，少数の記事

について部分文字列をとり文書頻度管理サーバに最初の文書頻度を計算させるためのデータである．このシステムでは文書頻度をもとに分類を行うため，文字列の正確な文書頻度が取得できることが要求される．しかしシステムに登録された記事数が少ない段階では正確な値からかけ離れた文書頻度が得られてしまう恐れがある．そこでシステムに記事を追加する前に少数の記事について文書頻度だけを計算して初期値を作った．これには 500 件程度の記事を使用した．

3 のシステムの記事データベースに挿入して記事ビューアに表示するデータとしては，10 万行，11359 件の記事を使用した．これらは学習と文書頻度初期値の作成が終わってからシステムのデータベースに挿入される．

これを用いてインターネットのニュースサイトのようなウェブページを作成した．

なお，今回はキーワードの最大の長さ (*df* などを取得する部分文字列の最大の長さ) は 10 文字に制限した．

ユーザーインターフェースと学習サーバ

今回のシステムにおけるユーザーインターフェースはウェブページになる．ウェブページの HTML の生成には Java Servlet を使用し，記事やキーワードデータの管理には軽量な SQLite データベースを使用した．Java Servlet を使用した理由は SQLite データベースへのアクセスができて動的にウェブページを生成できること，Passive Aggressive アルゴリズムの実装として利用した Jubatus が Java クライアントをサポートしていることなどによる．

分類器は Jubatus フレームワークを用いて作成した．Jubatus は分類問題をはじめ，回帰・近傍探索・グラフ学習などの様々な機械学習をサポートする機械学習フレームワークである．その分類器のアルゴリズムとして Passive Aggressive アルゴリズムを選ぶことができる．

Jubatus は単なる機械学習ライブラリではなく，Jubatus の学習器がサーバとして動作し，利用する側がクライアントとなるという特徴を持つ．Jubatus 自身は C++で実装されているが，Java を含む他の何種類かの言語をクライアントにできる．これは，Jubatus が MessagePack-RPC による Remote Procedure Call(RPC) の仕組みを持っていることによって実現される．

さらに Jubatus サーバとなりうるものは標準で提供されている学習器サーバだけに限定されるわけではなく，ユーザが Jubatus サーバを作成することもできる．その際，MessagePack-IDL という Interface Definition Language (IDL) ファイルを記述することにより，Jubatus サーバとして動作するサーバのひな形を自動生成する機能もある．

今回のシステムでは非常に多くの文字列についての df, df_2 という大量のデータを管理する必要があり，その管理のために IDL をもとに作成した Jubatus サーバを用いている．

計算機環境

システムの実行に用いた計算機環境は以下のとおりである．Jubatus は分散処理もサポートするが，今回のシステムでは PC1 台で動作させた．

OS

Ubuntu Server 11.10
(Linux kernel 3.0.0-13-server)

CPU

Intel Core i7 3.30GHz, 6 Cores, 12 Threads

Memory

64GB

6. 実 行 結 果

実行の様子

開発したシステムを実行している様子を図 3 から図 8 に示す．

図 3 はウェブサイトのトップページで，最新の記事 5 件とキーワードの上位 5 件が表示される．図 4 は全記事の一覧，図 5 は全キーワードの一覧のページである．

全キーワード一覧のページでなんらかのキーワードをクリックすると，図 6 のようなそのキーワードを持つ記事の一覧が表示される．図 6 などから記事名をクリックすると図 7 のように記事の内容が表示される．記事の文章中でキーワードと判定されたものにはまたリンクが張られており，クリックすると図のようなページにジャンプする．これによって同じキーワードを持つ記事どうしを自動でリンクする仕組みができている．

図 8 は記事を追加するための管理ページである．ここに特定の形式 (毎日新聞コーパスの形式) に沿った記事テキストを入力して「送信」ボタンをクリックすると，数秒でキーワード抽出とデータベース挿入が完了して記事が追加される．もちろんここで追加した記事はトップページや全記事一覧にも表示される．

図 3　トップページ

図 4　記事一覧ページ

(a) 特徴量に 2 乗値を含めない場合

(b) 特徴量に 2 乗値を含めた場合

図 5　キーワード一覧ページ

図 6　「大阪」を持つ記事一覧 (2 乗値を含めた場合)

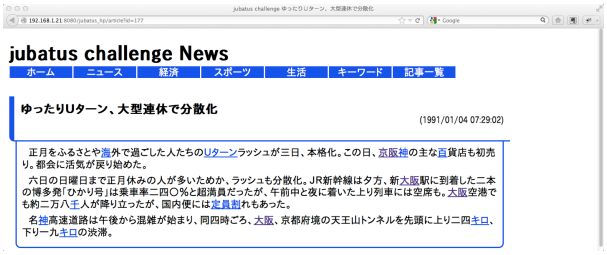


図 7　個別記事ページ

図 8　記事追加用ページ

表 4　11359 件の記事 (10 万行) 追加処理の実行結果

	抽出キーワード数	総実行時間
2 乗値を使わない場合	101,813	55 分 54 秒
2 乗値を使う場合	28,613	44 分 3 秒

表 5　上位 10 件のキーワード

(a) 2 乗値を含まない場合

順位	キーワード	文書数
1	共通	8
2	月から	7
3	下	6
4	メキシコ市	6
5	違い	6
6	の組	6
7	源地は	5
8	本	5
9	任期は	5
10	公立大入試の二次試	5

(b) 2 乗値を含む場合

順位	キーワード	文書数
1	大阪	1052
2	大統領	700
3	車	537
4	疑	530
5	銀	518
6	首相	517
7	ソ連	508
8	党	478
9	イラク	425
10	選	416

性能評価

記事データを分類器にかけて記事データベースに挿入する前の段階で、1 万行の教師データを分類器に学習させた後、文書頻度の初期値を作るために約 500 件の記事の部分文字列についての文書頻度を文書頻度管理サーバに計算させた。Ogata らのシステム [1] では教師データが 1 万件程度になると学習が現実的な時間で終わらないという問題があったが、本システムでは学習と初期文書頻度の計算は，データに 2 乗値を含めない場合と含める場合の両方で合わせて 3 秒程度で終了した。よって、Ogata の研究 [1] に比べてスケーラビリティを向上させることができたと言える。

システムのデータベースに挿入する記事データとしては，10 万行，11,359 件のデータを用いた．表 4 はそれらからキーワードを抽出・データベース挿入をしたときの実行時間と抽出されたキーワードの数を示している。表から，約 1 万件の記事データについて，1 文字から 10 文字までの長さのすべての部分文字列データを取り扱い，それをデータベースに挿入する処理が 1 時間以内で終了している．また，管理ページから新たな記事を挿入する際にも 5 秒ほどで処理が完了した．それらのことから実用的な時間で処理を行うシステムを構築できたといえる．

また，表 4 から (1) に示される 2 乗値を含まないデータを使った場合の方が，より抽出キーワード数が多く，実行時間がかかっていることが分かる．次元数が少ないデータを使用したときの方が実行時間がかかるというのは奇妙である．この理由として，学習・分類のフェーズでは 2 乗値を使わない方が高速に処理できているが，抽出キーワード数が多いためにデータベースへの挿入のフェーズが遅くなっていると考えられる．

表 5 は抽出されたキーワードをリンク記事数が多い順に 10 件リストアップしたものである．抽出されたキーワードを見てみると，キーワードと呼べるものはデータに 2 乗値を含まない場合で “メキシコ市” 1 つなのに対し，データに 2 乗値を含む場合では “大阪，大統領，首相，ソ連，イラク” の 5 つある．データに 2 乗値を含まない場合に比べて，2 乗値を含む場合の方がよりキーワードらしい文字列を抽出できていることが分かる．サンプル調査の結果として，キーワードでないものも多く出力されるものの，2 乗値を使用した方がその割合が減少しており分類が線形分離可能でなかったことが示唆される．

また 1 つのキーワードにつき，同じキーワードが抽出された記事も多くなっている．抽出キーワードが少なくなった分 (表 4 参照)，記事中でキーワードの候補となる文字列が少なくなり，同じ文字列がキーワードとして選ばれることが多くなったと考えられる．

7. ま と め

本研究では，Adaptation[3] を利用してキーワードを抽出する方法を提示した Ogata らの研究 [1] のスケールアップと応用システムの提案を試みた．スケールアップについてはオンライン学習アルゴリズムである Passive Aggressive[2] を用いて改善できることを示した．また応用システムとして新聞記事のキーワードを上記の方法で抽出し，同じキーワードを持つ記事同士を相互にリンクする新聞記事ビューアを開発した．

文　献

[1] Tomomi Ogata, Kenichiro Terao, and Kyoji Umemura. “Japanese Multiword Extraction Using SVM and Adaptation”. LREC-2004 Workshop on methodologies and a Evaluation of Multiword Units in Real-word Applications, pp.1–4, 2004.

[2] Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. “Online Passive-Aggressive Algorithms”. JOURNAL OF MACHINE LEARNING RESEARCH, 6:551–585, 2006.

[3] Kenneth W. Church. “Empirical Estimates of Adaptation”. Coling 2000 Proceedings of the 18th conference on Computational linguistics, 1:180–186, 2000.

[4] Yoshiyuki Takeda and Kyoji Umemura. “Selecting Indexing Strings Using Adaptation”. Proceedings of the 25th Annual International ACM SIGIR Conference, pp.427–428, 2001.