社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2008－DBS－145 (11)
2008－ＦＩ－ 91 (11)
2008／6／20

Wikipediaへの関連単語抽出アルゴリズムの適用とその評価

堀 憲太郎† 大石 哲也† 峯 恒憲†† 長谷川 隆三†† 藤田 博†† 越村 三幸††

†九州大学大学院システム情報科学府

††九州大学大学院システム情報科学研究院

抄録 本稿では，ユーザのクエリから，その意図に沿った拡張クエリを作成する際に，Web上で最大の百科事典であるWikipediaを用いるシステムを提案する．Wikipediaからの拡張クエリの抽出には，関連単語提案アルゴリズム[1]を用いる．関連単語提案アルゴリズムとは，あるキーワードとそれに関連するテキストが与えられた時，テキスト内で出現する単語間の距離に着目して，与えられたキーワードに関連し尚且つ重要と思われる単語を抽出するアルゴリズムである．このシステムをWeb検索時に用いることにより，クエリ拡張し，ユーザの目的とする文書を見つけるための支援技術とすることを目的としている．

Applying a related word extraction algorithm to Wikipedia for query extention

Kentaro Hori†, Tetsuya Oishi†, Tsunenori Mine††, Ryuzo Hasegawa††, Hiroshi Fujita††, Miyuki Koshimura††

†Graduate School of Information Science and Electrical Engineering, Kyushu University

††Faculty of Information Science and Electrical Engineering, Kyushu University

Abstract In this paper, we propose a method to extend the original query by applying a related word extraction algorithm to Wikipedia that is an encyclopedia. The algorithm extracts words related to given key words from any text and calculates a score for each extracted word according to distance between words in the text. The method helps users to obtain their desired documents in web search by extending a query with highly scored words.

1 はじめに

近年，インターネットの進歩により，一般家庭からでも容易にWeb(World Wide Web)にアクセスすることができる環境になっている．また，目的のページを見つけるための手段としてサーチエンジンを利用することが普及した．

大手サーチエンジンにgoogle[2]，yahoo[3]，goo[4]といったものがある．これらはユーザが自分の興味ある事柄について，単一，或いは複数のキーワードを入力するだけで，膨大なデータベースから最適なページを取捨選択してくれるものである．例えば，googleはページの評価に各ページのリンクに基づいたPageRankアルゴリズム[5]を用いて，ユーザの必要とするページを検索上位に提示することに成功している．

しかし，Web上のデータが膨大であること，サーチエンジンが得ることができるのはユーザから与えられた数少ないキーワードしかないこと，同じキーワードが個人，或いは場合によって異なる意味合いを持つ場合があることなどが原因で，ユーザが望むような検索結果が得られないことが多々ある．サーチエンジンが返す検索結果の上位にユーザが求めるページが現れない時，ユーザは残りのページから必要なページを見つける作業や，キーワードを変えての検索作業の繰り返しを強いられる．

検索結果を絞り込む手段として，複数のキーワードを与えるAND検索がある．ユーザの目的に適したキーワードを追加すると，単一のキーワードの時に比べて検索結果を格段に絞り込むことが可能である．しかし，ユーザは検索する際に必ずしも適切なキーワードを思いつくとは限らない．キーワードを追加しても，想定したほど検索結果を絞り込めない可能性がある．また，入力したキーワードに関する新たな知識を獲得したいとユーザが考える時もある．彼らは入力したキーワードに対してある程度の知識を持っているが，そのキーワードに対して知らない知識もある．このためユーザの知らない知識で絞込検索できないのは，新たな知識を得る際には非常に不便である．

そこで，ユーザの入力したキーワードに関連した単語(関連語)を自動的に提案するシステムがいくつか研究されている[1][8][12]．この関連語を提案するためには，ユーザが入力したキーワードに加えてさらに他の情報源が必要となる．その情報源としてユーザからのキーワードを一旦サーチエンジンに入力し，そこから得られた検索結果を用いる手法がある．例えば擬似フィードバックと呼ばれる手法は，検索結果の上位10件をユーザの意図に沿った文書(以降適合文書と呼ぶ)，それ以下を不適合文書とし，それらの文書集合から関連語を抽出する（以降このシステムを旧システムと呼ぶ.）．この手法の利点は全自動化できるためユーザに負担を与えない点，そして入力したキーワードに対して何かしらの検索結果が得られる点，検索意図に縛られないという点である．しかし昨今，blogや掲示板，オンラインショップが発達しており，サーチエンジンから得られた検索結果の上位にこれらが表示されると，有用でない情報が引き出だされてしまうことがある．さらに，情報を抽出するため何ページにもアクセスするので時間がかかる．

本論文では関連単語抽出の情報源としてWeb上の

百科事典，具体的にはWikipedia[6]を使った関連単語提案システムを提示する．百科事典を使うことにより，ユーザが入力したキーワードの検索結果である情報源に含まれるblogや掲示板などの不必要な情報を排除し，より未知のものに対してWebを検索した際の検索精度を向上させる．さらに，多量の情報が含まれたWebページから情報を抽出するという工程を，関連した単語が密集している短い文章を事典から抽出することに置き換え，システムの処理時間を短縮できる．しかし，百科事典はWebページ全ての情報量と比べると非常に少なく，ユーザが入力したキーワードが辞典の中に含まれていない可能性がある．そこで，百科事典がどの検索単語なら含まれているのか，またどういったジャンルの検索ならばユーザの意図に沿った関連語を提案できるのか，も明確にする．

2 関連研究

疑似フィードバックの他には，ユーザが直接文書に点数をつける明示的フィードバック，スクロールおよびページ拡大，ページのクリックなどの操作に関して，ユーザがいずれかの操作を行ったかどうかを調査して，ユーザの意図にあった文書を収集する暗黙的フィードバックといったものもある．

他の情報源として，ユーザ個人の特性をあらわす情報(個性情報と呼ぶ)を用いるシステムもある．個性情報は一般にユーザプロファイルと呼ばれ，例えばスケジュールであったり趣味趣向をあらかじめデータベース化しておいたり，と様々なものがある．これらを用いることで，ユーザの意図にあった関連単語を提案することができる．例えばユーザの住んでいる付近の天気予報を調べるときなど，このシステムがユーザの住んでいる地域に限定してくれる．個性情報に沿った検索をしている限りは，Webという様々な情報が混濁している場から関連した単語を見出すより，ユーザプロファイルから関連語を導くほうがより有効な手段であると言える．しかし，このシステムはユーザが新しいことについて知識を得ることが困難である．知らない外国の文化や歴史，見たことがない単語について検索するときに個性情報を参照しても，そこに有用な情報はないからである．

3 関連単語提案システムの概要

全体的な流れは，図1のように行う．

まず，システムはユーザから検索キーワードを受け取る．それを元にユーザの意図に沿った文書（以降適合文書と呼ぶ）を収集，文書を解析して単語の品詞ごとに分解（以降形態素解析と呼ぶ），単語の評価を関連単語抽出アルゴリズム[1]で行って，その結果の上位をユーザに返す．これらを全てシステムで自動的に行う．

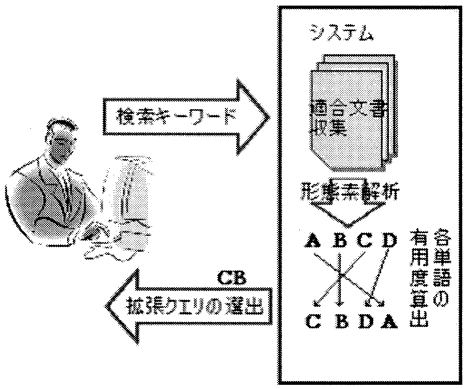

図1: システムのイメージ

3.1 各構造の説明

はじめに，ユーザに元となる検索キーワード（以降クエリと呼ぶ）を入力してもらう．このクエリに関連した単語を返すことがシステムの目的である．以降，システムが返す関連語を特に拡張クエリと呼ぶ．

次に，拡張クエリを見つけ出すための情報源として適合文書の収集を行う．旧システムではサーチエンジンにクエリを投げ，その検索結果の上位R件を適合文書としている．本研究では情報源としてWeb上の百科事典を用いる．具体的には，百科事典内のクエリに関連する文章を抜き出し，それを適合文書として用いる．

収集した適合文書を，高速形態素解析システム「MeCab」[7]を利用して形態素解析を行う．このときMeCabを通すことで現れる単語を精錬する．まず名詞のみに絞り，その上で不必要な単語，例えば「こと」「もの」といった検索に適していない単語は本システムで極力削除する．また,「卒業論文」といった連結語は「卒業」「論文」と分かれてしまうので，連結語であると判断した場合は本システムで「卒業論文」と戻すようにしている．

以上のような形態素解析の後，精錬した単語について次は関連語としての有用度を算出する．この計算手法についても様々な研究が行われている．例えば，RSV(Robertson Selection Value)[8]と呼ばれるアルゴリズムがある．RSVは適合文書か不適合文書のどちらかに偏って出現している単語を重要とするものである．しかし，提案手法は百科事典から適合文章を抜き出すという構造上，適合文書数は1であり，不適合文章も収集できず，どの単語が偏っているかという算出はできない．そこで今回使用したのは我々の研究室で提唱している関連単語抽出アルゴリズムである．これについては後述する．

アルゴリズムを適用して，単語ごとに評価値が計算されたら，それを元にソートし，結果として評価値の高いほうから順にn件の単語を，拡張クエリとして返す．その拡張クエリを用いて，通常のサーチエンジンで検索をし，よりよいページを上位に表示させるのが最終目標である．

4 関連単語抽出アルゴリズム

4.1 アルゴリズムの概要

このアルゴリズムは，あるキーワード群 K とそれに関するテキスト T に現れる単語から，K に関連すると思われる単語を抽出，出力するものである．単語の関連度の算出には，単語間の距離に着目し，それを元に評価を行う．

このアルゴリズムの根幹となっている考え方が，単語間の距離である．具体的には，文章中に出現する単語の順番に注目し，単語 A について A の付近に出現している単語ほど A に関連性があるという考え方である．

4.1.1 定義

アルゴリズムの説明の前に，以下の記号を定義する．

- K ･･･ 関連単語を抽出するための基となるキーワード群
- T ･･･ キーワード群 K に関するテキスト
- $k_i(i = 1 \cdots m)$ ･･･ テキスト K で出現する m 個の単語 (出現順に $k_1, k_2, k_3, \cdots k_m$)
- $s_h(h = 1 \cdots n)$ ･･･ テキスト T で出現する n 個の文章 (出現順に $s_1, s_2, s_3, \cdots s_n$)
- $t_j(j = 1 \cdots o)$ ･･･ テキスト T で出現する o 個の単語 (出現順に $t_1, t_2, t_3, \cdots t_o$)

但し，t_j はテキスト T を MeCab で形態素解析し，その結果得られた名詞のみ抽出し，それらを出現順に並べたものである．また，形態素解析した結果，同一単語が複数出現した場合もそれらは別のものとみなす．

4.2 テキスト内の単語の評価

テキスト T 内で出現する単語の評価は以下のように行う．

1. $k_i(i = 1 \cdots m)$ を基準に $s_h(j = 1 \cdots n)$ の基礎評価値 (BasicValue)$BV(s_h)$ を計算
2. 文章の単語への分解，平滑化
3. 単語の出現頻度による最終評価値 $V(t_j)$ を計算

4.2.1 $BV(s_h)$ の算出

本アルゴリズムはキーワード群 K を基に，K とテキスト T で出現する単語間の距離を中心に単語の評価を行うことを目的としており，はじめにその基本となる評価値を求める．

文章 s_h のキーワード k_i に関する評価値 $BV_{k_i}(s_h)$ を次のように定める．

$$B_{k_i}(s_h) = n - d \tag{1}$$

ここで，d は k_i と s_h の距離である．すなわち，文章 s_q に k_p が出現するとき d は 0 で，$BV_{k_p}(s_q) = n$ となり，その一つ隣の文章 s_{q+1} と s_{q-1} は $d = 1$ で，$BV_{k_p}(s_{q+1}) = BV_{k_p}(s_{q-1}) = n - 1$ となる．k_i と s_h の距離が近いほど $BV_{k_i}(s_h)$ は大きな値をとることに注意されたい．

以上の計算をすべての $k_i(i = 1 \cdots m)$ で行い，それぞれの s_h において $BV_{k_i}(s_h)$ の和を $BV(s_h)$ とする．すなわち $BV(s_h)$ は次式により求める．

$$BV(s_h) = \sum_{i=1}^{m} BV_{k_i}(s_h) \tag{2}$$

表 1 に $BV(s_h)$ を求める例を示す．例より，キーワード群 K に含まれている単語を含む文章中心に，それに近い文章ほど高評価を得ていることがわかる．

表 1: 距離による文章の評価例

キーワード K	A	B			
テキスト T	AFB	ED	AFC	FE	DE
$BV_A(s_h)$	5	4	3	2	1
$BV_B(s_h)$	5	4	3	2	1
$BV_A(s_h)$(2回目)	3	4	5	4	3
$BV(s_h)$	13	12	11	8	5

4.2.2 $BV(s_h)$ の平滑化

前節で求められた $BV(s_h)$ は，文章 s_h のテキスト T で出現する位置を考えると不公平である．テキスト T の端に出現する文章 $s_h(h = 1, n)$ については $1 \leq BV_{k_i}(s_h) \leq n$ であるのに対し，テキスト T の中央に出現する文章 $s_{n/2}$ については $n/2 \leq BV_{k_i}(s_{n/2}) \leq n$ である．よって $BV(s_h)$ のとり得る値の期待値は文章 s_h の出現位置 h によって異なり，このままでは T の中央に出現する文章は必然的に与えられる評価値が大きくなってしまい，公平な評価はできない．

この問題を解決するために，求めた $BV(s_h)$ を文章 s_h の出現位置 h での評価値の期待値を用いて平滑化を行う．文章 s_h の出現位置 h での評価値の期待値 $EBV(h)$ は以下の式で求められる．

$$EBV(h) = \frac{1}{2n} n(n + 2h - 1) - 2h(h - 1) \tag{3}$$

ここで，n はテキスト T 内で出現する文章ののべ総数である．

本アルゴリズムでは，平滑化後の評価値を $EBV(s_h)$ とし，以下の式で計算する．

$$EBV(s_h) = \frac{BV(s_h)}{EBV(h)} \tag{4}$$

表 2 に $EBV(h)$ を計算し，$EBV(s_h)$ を求めるまでの例を示す．$EBV(h)$ は出現位置 h での評価値の期待値であるので，中心付近 ($h = n/2$ 付近) の値が大きくなっている．中心をピークとして左右対称になっている．このように $BV(s_h)$ では出現位置により不公平な評価を得ていたものが平滑化されていることがわかる．

表 2: 出現位置 h での期待値 $EBV(h)$ とそれを用いた評価値 $EBV(s_h)$ の例

キーワード K	A	B			
テキスト T	AFB	ED	AFC	FE	DE
$BV(s_h)$	13	12	11	8	5
$EBV(h)$	3	3.6	3.8	3.6	3
$EBV(s_h)$	4.33	3.33	2.89	2.22	1.67

4.2.3 $V_T(t_j)$ の算出

本アルゴリズムは単語間の距離を最重要視して評価値計算を行っているが，それに加えて TF/IDF 法 [9] などで用いられるテキスト内での単語の出現頻度 TF(TermFrequency) の概念も考慮する．つまり，テキスト T 内で複数回出現した単語はある程度重要視すべきである，ということである．

まず，文章から単語に分解する．そのとき，テキスト T 内で複数回出現した単語についてはその単語の $EBV(t_j)$ の平均値 $AveEBV(t_j)$ を計算する．例えば文章 s_a と文章 $s_b (a \neq b)$ の 2 つに単語 t_c が 1 回ずつ現れたとすると，$AveEBV(t_c) = (EBV(s_a) + EBV(s_b))/2$ となる．無論，T 内で出現が 1 回のみの単語 $t_j (t_j$ は s_h内 の単語とする) に関しては $AveEBV(t_j) = EBV(s_h)$ である．

さらに単語 t_j の tf 値 (T 内での出現回数) を用いて以下の式で表される重み $W_T(t_j)$ を計算する．

$$W_t(t_j) = 1 + \frac{tf(t_j)}{n} \log tf(t_j) \tag{5}$$

ここで，$tf(t_j)$ はテキスト T 内での単語 t_j の出現回数である．

そして得られた $AveEBV(t_j)$ と $W_T(t_j)$ を用いて以下の式でテキスト T で出現する単語 t_j の評価値 $V_T(t_j)$ を計算する．

$$V_T(t_j) = AveEBV(t_j) * W_T(t_j) \tag{6}$$

こうして求められた $V_T(t_j)$ を，本アルゴリズムでのテキスト T 内で出現する単語 t_j の評価値とする．表 3 に $V_T(t_j)$ 算出までの例を示す．テキスト T 内では単語 C が 2 回出現しているので $tf(C)$ は 2 となっており，その他の単語については tf 値は 1 となっている．重み $W_T(t_j)$ は tf 値を基に計算しているので，tf 値が 1 の単語については 1，2 回出現している単語 C について 1 より大きな値をとっている．さらに，単語 C に関しては $EBV(C)$ の平均も計算する．そして最終的な単語の評価値 $V_T(t_j)$ が求められ，本アルゴリズムでは，テキスト T 内の単語は，評価値の大きい F，A，B，E，D，C の順でキーワード群 K へ関連度が高いとみなす．

表 3: $V_T(t_j)$ 算出までの各値の例

キーワード K	A	B				
テキスト T	AFB	ED	AFC	FE	DE	
$EBV(s_h)$	4.33	1.67	2.11	2.22	2.67	
単語	A	B	C	D	E	F
$AveEBV(t_j)$	3.61	4.33	2.11	1.67	2.50	2.56
$tf(t_j)$	2	1	1	2	3	3
$W_T(t_j)$	1.28	1	1	1.28	1.66	1.66
$V_T(t_j)$	4.62	4.33	2.11	3.20	4.00	5.23

5 提案手法

5.1 適合文書

このシステムでは，適合文書に百科事典を使う．事典は Wikipedia[6] を使用した．この理由については主に 3 点が挙げられる．具体的には

- Wikipedia が Web 上で最大の百科事典であり，日本語だと現在約 48 万語載っている．(2008 年 5 月現在)
- 普通の事典にはない，充実した内容が書かれている [10].
- 誰でも更新ができるというその特性から，多数のユーザが頻繁に更新し常に最新の内容が維持される．

という点から決めた．

5.2 抽出方法

適合文章の収集方法は，クエリを構成するキーワード数によって異なる．

- クエリが 1 つのキーワード (クエリ A とする) からなる場合

Wikipedia 内に A についての記事があるかどうか検索を行う．どの記事にもページ上部に単語についての概要が書かれているので，A の該当記事があれば，その部分を適合文章として抽出する．図 2 は，「イチロー」というクエリで検索したときに，Wikipedia のイチローの記事から適合文章を抜き出す様子である．

- クエリが 2 つのキーワード (クエリ「A B」とする) からなる場合

Wikipedia にも AND 検索は実装されているが，ただ両方のキーワードが載っていれば表示されるだけで，結果の妥当性が低い．すなわち有用な関連語を抽出できる文書であるかどうかはわからない．よって代わりに以下の方法を用いることで，関連語が確実に載っている文書を探し出す．まずクエリ A についての記事を検索し，該当記事があればその記事内について

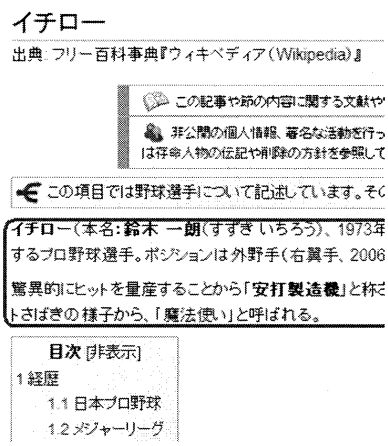

図 2: 実際の抽出イメージ

B というクエリが現れる文章を捜索する．発見したらその文章がある段落を抽出する．次に，クエリ B についての記事を検索し，再び該当記事があればその記事内について A というクエリが現れる文章を捜索し，見つかれば合わせて抽出する．例えば，「卵焼き 味付け」というクエリで検索したときに，まず Wikipedia の「卵焼き」のページを探す．「卵焼き」のページは Wikipedia に存在するので，次は「味付け」という単語で「卵焼き」ページ内を捜索する．見つかったら「味付け」がある段落を適合文章と抽出する．次に，「味付け」のページを探す．ところが，Wikipedia に「味付け」というページは存在しないので抽出は行えない．よって「卵焼き」ページから抽出した適合文章のみを適合文書として，形態素解析に移る．図 3 は，卵焼きページ内から味付けという単語のある文章を探し出し，その段落を抜き出す様子である．

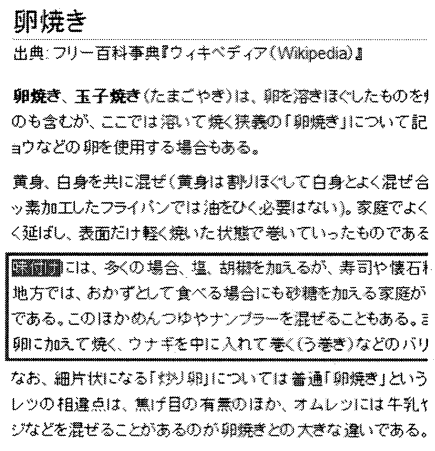

図 3: 実際の抽出イメージ

- クエリの入力が 3 つ以上からなる場合

当然クエリの入力が 3 つ以上であることもありえるが，百科事典上で擬似的に 3 単語以上の AND 検索を実現するのはあまり現実的ではない．仮に実現できたとしても，クエリの入力数が 3 以上で Wikipedia 内に該当文章が存在することはほとんどないと予想される．このため本研究ではクエリの入力が 2 つまでの場合のみを扱う．

さらに，抽出した文章について関連単語抽出アルゴリズムを適用した後，抽出された単語について評価値の操作を行う．具体的には，ユーザから与えられたクエリの評価値を最高値に設定し，かならず提案の先頭に現れるように設定する．なぜなら，ユーザが直接入力した語というのは，入力の間違いをしていない限り，検索意図の方向性を決める重要な単語であるといえるが，Wikipedia の文章の構成上の問題でユーザの与えたクエリが必ずしも上位に現れるというわけではないからである．例えば，元クエリが 2 つ (A B) だった場合に，まず A のクエリに関する記事内で B の単語が出現する文章を拾ってくるが，その文章に A という単語が含まれていない可能性がある．よって上記のような操作を行った．また，これに関しては具体的なデータとともに，精度が確かに上がっているかどうかの確認も行う．

6 実験と考察

6.1 Robertson selection value

実験方法を説明する前に，実験の比較対象として使う RSV(Robertson selection value)[8] について説明する．RSV では，適合文書に与えられる適合度の期待値と不適合文書に与えられる適合度の期待値を比較し，その差が大きくなる語ほど関連語として適しているとみなす．即ち，擬似フィードバックを反映した重みを用いて，語 t_j の関連語としての有用度に RSV を用いて以下の式で求められる．

$$RSV(t_j)=(\frac{df^+}{R^+}-\frac{df^+ + df^-}{R^+ + R^-})$$

$$*\{\alpha \log \frac{R^+ + R^-}{df^+ + df^-} + (1-\alpha) \log \frac{(df^+ + 0.5)/(R^+ - df^+ + 0.5)}{(df^- + 0.5)/(R^- - df^- + 0.5)}\} \quad (7)$$

上記の式で使われているパラメータは以下の通りである．

- R^+…適合文書数
- df^+…語 t_j を含む適合文書数
- R^-…不適合文書数
- df^-…語 t_j を含む不適合文書数
- α…制御パラメータ (α=0.0)

α については，対数が取れるようにするための調整値であるが，今回は不適合文書だけに含まれる語は必要がないので，α=0.0 とした．

表 4: 提案手法に対する調整の評価 (単位：%)

	単語数 1	単語数 2	平均
調整前	24.7	25.3	25.0
調整後	26.7	38.7	32.7

6.2 評価方法

それぞれ用意した手法を用いて，ユーザから与えられたクエリに関する単語を抽出する．抽出された単語のそれぞれの評価値の上位 5 語を再びクエリとして与え，検索エンジンによる検索結果の精度を評価値としている．ここで精度とは，検索結果上位 10 件のうち何件がユーザの検索意図に沿ったページであるか，を百分率で表したものとする．検索意図に沿っているかどうかは，実際に検索結果を見てユーザが判断する．

6.3 提案手法に対する調整の評価

提案手法に対する調整の有効性を調べる．第 4 章で述べたように，今回提案した手法ではユーザから与えられたクエリの評価値を最高値に設定し，提案する関連単語の中に必ず含まれるように調整している．そこで 1. 調整前，2. 調整後とし，それぞれ

1. ユーザから与えられたクエリの評価値を調整しない手法
2. ユーザから与えられたクエリの評価値を最高値に調整する手法

の 2 つを用意し，精度が上昇しているかどうかを確認する．実験に使用したデータは，ユーザから与えられたクエリの単語数が 1 つの場合を 15 件，単語数が 2 つの場合を 15 件，合計 30 件でジャンルはランダムとした．ただし，Wikipedia 内文書にユーザから与えられたクエリが存在することを前提条件としている．ユーザから受け取ったクエリの単語数が 1 のときを「単語数 1」，受け取ったクエリの単語数が 2 からなる AND 検索のときを「単語数 2」と表現する．

表 4 を見ればわかるように，単語数 1，2 ともに精度の上昇が見られる．単語数 1 の上昇値は低いが，この理由は調整をしなくても元のクエリが上位 5 件内に含まれていることが多いからである．単語数 2 のときは，調整なしでは元のクエリ 2 つともは含まれにくいため，精度の上昇も著しい．以降の実験では，提案手法は調整後のものを使用する．

6.4 速度比較

検索結果上位 R 件からの抽出を，Wikipedia からの抽出にすることで，実行速度にどの程度変化があったかを [Wikipedia][RSV][RWS] という 3 つの手法について実験を行い，結果を比較する．[Wikipedia] は Wikipedia から適合文書を取得し，形態素解析して抽出した単語を関連単語抽出アルゴリズムで評価値計算したもの，[RSV] は元クエリによる検索結果上位 5 件を適合文書として取得し，形態素解析の結果を RSV で評価値計算したもの，[RWS] は元クエリによる検索結果上位 5 件を適合文書として取得し，形態素解析の結果を関連単語抽出アルゴリズムで評価値計算したものである．それぞれの対応を表に示すと，

表 5: 各手法の対応

手法	適合文書	単語の評価方法
Wikipedia	Wikipedia	関連単語抽出アルゴリズム
RSV	検索結果上位 5 件	RSV
RWS	検索結果上位 5 件	関連単語抽出アルゴリズム

となる．この RWS が旧システムに該当し，Wikipedia が提案手法に該当する．実験に使用したデータは 6.3 節と同じものを使用した．元クエリによる検索結果上位 5 件は，goo より結果を取得した．なお，測定時の環境は，CPU に Pentium M 1.60GHz，メモリは 768MB の PC を用いた．

表 6: 単語数 1 の時の速度比較 (単位：秒)

	情報取得時間	演算処理時間	平均
Wikipedia	0.285	0.194	0.479
RSV	3.181	1.744	4.925
RWS	2.960	0.289	3.249

表 7: 単語数 2 の時の速度比較 (単位：秒)

	情報取得時間	演算処理時間	平均
Wikipedia	1.168	0.8052	1.973
RSV	5.258	1.292	6.550
RWS	5.510	1.866	7.376

表 6，表 7 を見ると，提案手法がかなり高速で動作していることがわかる．理由としては，情報取得時間の面では，かならず上位 R 件を見なければならない旧システムに比べ，提案手法はクエリの数だけしかページを見なくてよいからである．そして演算処理時間は，R 件分の情報を処理するより，辞書内の一部分だけの情報処理でいい提案手法が高速に動作する．また，単語数が 1 になると提案手法は Wikipedia ページ内部の単語捜索も省くため，非常に高速になる．

6.5 ジャンル別 Wikipedia のヒット率

6.4 節と同じ，[Wikipedia][RSV][RWS] という 3 つの手法について実験を行い，結果を比較した．ユーザから与えられるクエリは，ODP(Open Directory Project)[11] に基づいたジャンル別 14 種類ごとに各 10 クエリ (クエリの単語数が 1 つの場合を 5 クエリ，単語数が 2 つの場合を 5 クエリ) で，全 140 クエリについて行う．この ODP とは，人の手によって編集されているウェブ最大の包括的なディレクトリである．

表 8: ジャンル別 Wikipedia の単語ヒット率 (%)

	アート	オンラインショップ	ゲーム	コンピュータ	スポーツ	ニュース	ビジネス	レクリエーション
単語数 1	100	100	100	100	100	80	80	100
単語数 2	80	20	40	60	60	20	40	60
平均	90	60	70	80	80	50	60	80

	家庭	科学	各種資料	健康	社会	地域	平均
単語数 1	100	100	80	100	100	100	95.7
単語数 2	60	80	20	40	100	100	55.7
平均	80	90	50	70	100	100	75.7

表 9: 単語数 1，Wikipedia 内のクエリの有無を考慮しない精度 (単位：%)

	アート	オンラインショップ	ゲーム	コンピュータ	スポーツ	ニュース	ビジネス	レクリエーション
RSV	42	48	20	12	28	22	44	16
RWS	40	32	24	12	40	24	54	22
Wikipedia+RWS	54	2	20	36	32	32	46	16

	家庭	科学	各種資料	健康	社会	地域	平均
RSV	28	28	38	26	28	12	28.00
RWS	22	28	32	48	28	20	30.43
Wikipedia+RWS	24	42	28	36	30	18	29.71

表 8 は，Wikipedia から何らかの適合文書が抽出できれば 1，できなければ 0 としてカウントしたものである．数値が高いほど関連単語の提案に成功する確率が高く，低いほど提案に失敗する確率が高いことになる．

平均を算出すると，元クエリの単語数 1 は 95.7%，単語数 2 は 55.7%となった．これらより，単語数 1 の場合は，ほとんどのクエリは Wikipedia に載っていることがわかった．引っかからない単語があった [ニュース][ビジネス][各種資料] について見ると，それぞれ，近々発売される予定の「ティファニー携帯」，ビジネス用語から「クリティカルチェーン」，資料として利用するための「英和辞書」であった．これらより 2 つの単語を合成した語に関して弱い傾向が見られる．例えば「英和辞書」に関して，「英和」「辞書」という 2 単語として検索すればヒットしていたが，英和辞書は辞書として一くくりに書かれているので，つなげた一単語としての記事はない．

次に，単語数 2 のときについて見ると，特に高い数値を出しているのが [地域][社会] の項目である．社会が高くなった原因は，Wikipedia の特徴として単語とそれに関するしきたりや歴史，成り立ちを詳しく載せる傾向があるからである．また，地域が高くなったのは，Wikipedia には世界中全ての国名・地名が記載されているからである．特に低い数値が見られるのは [オンラインショップ][ニュース][各種資料] の項目である．[オンラインショップ] が低くなった原因として考えられるのは，事典という公平な立場からは，安さや値段といった店によって違いすぐに変動する事項については記載できないからである．また，記載してしまうとそれは宣伝効果や営業妨害になり，それは百科事典の意図するところではない．[ニュース] については，Wikipedia はニュースを記載できない点が理由である．[各種資料] については，地図や時刻表といった事典では扱わない情報があるからである．全体的に単語数 1 にくらべて単語数 2 が大幅に下がってしまっている．1 単語目は確かに 45 万件の記事から探せばほぼ見つかるが，2 単語目は 1 単語目の該当記事内に含まれる単語の中からしか見つけられないことが原因である．

6.6 Wikipedia による拡張クエリ選出の精度

次に，ユーザから与えられたクエリをそれぞれの手法で拡張し，その精度を求める．(表 9，表 10)．Wikipedia+RWS とは，Wikipedia 内にユーザから与えられたクエリが存在しない場合に，RWS に切り替えた時のデータである．実験に使用したデータは 6.5 節と同じものである．これによって，実際に検索手段として使用したときに，どの程度 Wikipedia を適合文書収集元とすることが有効なのかを測ることができる．また，関連単語抽出アルゴリズムは RSV と比較して有効なのかどうかも見る．

まず単語数 1 については，RWS と同程度の精度でしか関連語を選出できない．あまり伸びなかった一番の原因は，精度が低いジャンルでも，1 単語であればほぼ Wikipedia に存在するからである．実際に，一番苦手なジャンルであるオンラインショップのときは RWS に全て代わってもらうというシステムに変更したところ，平均は 31.86%と RWS を上回った．

単語数 2 で，Wikipedia+RSV についてみると，得意ジャンルと不得意ジャンルの差がはっきり出ている．すなわち [アート][コンピュータ][スポーツ][地域] が得意で，[オンラインショップ][ニュース][レクリエーション][各種資料] は不得意である．RSV と RWS の結果のみに注目すると，単語数 1 の時は若干有効なだけであるが，単語数 2 のときはほとんどのジャンルで RWS がよい結果を返している．よって関連単語抽出アルゴリズムは有効であるといえる．また，Wikipedia+RWS は他 3 つと比べて非常に精度が向上する結果となった．

表 10: 単語数 2，Wikipedia による拡張クエリ選出の精度 (単位：%)

	アート	オンラインショップ	ゲーム	コンピュータ	スポーツ	ニュース	ビジネス	レクリエーション
RSV	22	12	6	32	20	30	2	4
RWS	24	12	10	24	26	34	40	30
Wikipedia+RWS	56	6	14	46	44	34	28	16

	家庭	科学	各種資料	健康	社会	地域	平均
RSV	20	8	6	10	18	8	14.14
RWS	22	30	34	24	24	8	24.43
Wikipedia+RWS	44	18	34	16	26	20	28.71

Wikipedia にクエリがある場合は Wikipedia を使用したほうが、良い情報が抽出できていることがわかった.

7 おわりに

処理速度の面では，収集文書を Wikipedia にすることで，情報取得，演算処理の両方が高速化した．クエリ 1 つ 1 つの演算速度を見ても全て提案手法の方が速い．精度面では，旧システムより提案手法のほうがよりよい結果を返すことがわかった．特に，[アート][コンピュータ] というジャンルに対して良い拡張クエリが選出できる．これは，アートやコンピュータの知識に長けている人間が Wikipedia の編集をするため，自然と内容が充実するからである．また，Wikipedia にクエリが存在しなかった場合は別のシステムに変わることで，総合的に他のシステムより良い結果を返すことができた．

今後の研究に関しては，提案手法ではクエリによる wikipedia のページが取得できない場合の対処法を考えている．一般的なサーチエンジンに「ja.wikipedia.org」という単語を拡張クエリとして追加し，擬似的に wikipedia 内で 2 単語以上のクエリによる全文検索（文書に含まれる全文からの検索）を行う方法がある．これにより，元クエリから何らかの wikipedia ページを取得できるようになるが，本来 wikipedia に情報が載っていなかった場合でも，代わりのページが提示されてしまう．今後，全文検索の実装をし，この問題点がどう影響するのか今回の提案手法と比較評価していきたい．

8 謝辞

本論文の基礎は，九州大学（現 日本電気通信システム株式会社）倉元俊介により行われた。

参考文献

[1] 倉元俊介，長谷川隆三，藤田博，越村三幸，峯恒憲: “関連単語提案アルゴリズムを用いた関連語提案システム”，合同エージェントワークショップ&シンポジウム 2007

[2] google, http://www.google.co.jp/

[3] yahoo, http://www.yahoo.co.jp/

[4] goo, http://www.goo.ne.jp/

[5] Lawrence Page, Sergey Brin, Rajeev Motwani, Terry Winograd, “The PageRank Citation Ranking: Bringing Order to the Web”, 1998,

[6] Wikipedia: http://ja.wikipedia.org/

[7] 高速形態素解析システム「MeCab」: http://mecab.sourceforge.net/

[8] S.E.Robertson and K.Spark, J.: “On relevance weights with little relevance information”, the 20th Annual International ACM SIGIR Conference(SIGIR'97), pp.16-24

[9] 石川博: “次世代データベースとデータマイニング”, Web とデータマイニング, P182-183, 2005, CQ 出版社

[10] Web of the Year 2007: http://woy2007.sbcr.jp/

[11] Open Directory Project, http://www.dmoz.org/World/Japanese/

[12] 真野博子，伊東秀夫，小川泰嗣: “文書検索におけるランキング検索技術”, http://www.ricoh.co.jp/about/businessoverview/report/29/pdf/A2902.pdf