筑波大学 情報学群 情報メディア創成学類

卒業研究論文

クエリ依存型リンク解析手法における スコア近似に関する研究

佐藤 豪

指導教員　古瀬 一隆　陳 漢雄

2016 年 1 月

概要

私たちが普段使用しているWEBページの検索エンジンに用いられている手法の1つに、リンク構造に基づいて得点を与えるSALSAアルゴリズムがある。SALSAアルゴリズムはクエリ依存型のリンク解析手法であり、よりクエリに適した検索結果を得ることができる。一方で、クエリが与えられてからWEBグラフの抽出を行いリンク構造を取得し、各ページのスコア計算を行うため多くの応答時間がかかってしまう。

先行研究では、SALSAアルゴリズムの一部をクエリが与えられる前に処理することで高速化を図る手法が提案された。従来のSALSAアルゴリズムに比べて高速化することに成功したものの、ランキング精度においては著しく低い結果となり、クエリに適した検索結果が得られるというクエリ依存型リンク解析手法の長所が失われてしまった。そこで、本論文では先行研究の手法を改善し、ランキング精度を保ちつつ、高速化を行う手法を提案する。

本研究では、クエリ依存型リンク解析手法SALSAにおいて、クエリに適した検索結果が得られるという長所を保ちつつ、検索の高速化を行うことを目的とする。

目 次

図 目 次

第1章　序論

世界中に 10 億件以上存在する WEB サイトの中から、必要な情報を手探りで探すことは困難である。そのような場合、検索エンジンを用いることが有効である。検索エンジンは、ユーザーが必要とする情報に関連した検索キーワードを受け取り、そのキーワードに関連する WEB ページの集合を抽出し、検索結果としてユーザーに掲示する。検索結果として掲示される WEB ページの集合は、検索キーワードとの関連度が高く、ユーザーにとって役に立つ WEB ページが上位に表示されることが望ましい。このように、ある検索ワードに関連する WEB ページの集合に対して、ユーザーに掲示するための順位付けを行う手法を WEB ランキングアルゴリズムと呼ぶ。

WEB ランキングアルゴリズムは大きく分けて 2 種類ある。1 つ目は、WEB ページのテキストや HTML 構造を解析することで内容得点を計算し、順位付けを行う手法である、2 つ目は、WEB ページ間のリンク構造に基づいてスコア計算を行い、順位付けを行うリンク解析手法である。本論文では後者のリンク解析手法を扱うものとする。

リンク解析手法のアルゴリズムで有名なものとして PageRank アルゴリズムや HITS アルゴリズムが挙げられる。さらに近年の研究では、PageRank アルゴリズムと HITS アルゴリズムの長所を取り入れた SALSA アルゴリズムと呼ばれるリンク解析手法が、ランキング精度において他の WEB ランキングアルゴリズムより優れているという報告がされている。

PageRank アルゴリズムは、検索キーワードに依存せずスコア計算を行い、順位付けをすることができる。このようなリンク解析手法をクエリ独立型と呼ぶ。一方で HITS アルゴリズムや SALSA アルゴリズムは、検索キーワードが与えられてからスコア計算と順位付けを行う。このようなリンク解析手法をクエリ依存型と呼ぶ。したがって、これらのアルゴリズムを検索エンジンで用いる場合、検索キーワードが与えられる前にスコア計算を行う PageRank アルゴリズムは高速に応答することができるのに対し、検索キーワードが与えられてからスコア計算を行う HITS アルゴリズムや SALSA アルゴリズムは応答時に無視できないほどの時間がかかってしまう。

この問題の解決策として、クエリ依存型リンク解析手法において時間のかかるスコア計算を前処理化することで、応答時間を減らすという手法が過去に提案された。これらの手法では、検索キーワードが与えられる前にあらかじめ WEB ページのクラスタリングを行い、クラスタ内での各ページのスコア計算を行う。検索キーワードが与えられた後は、そのクラスタとスコアを用いて結果を近似し、WEB ランキングを作成する。しかし、これらの手法では近似したスコアと本来の SALSA アルゴリズムによるスコアが大きく異るため、応答時間は早くなるがランキング精度が下がる結果となった。

そこで本研究では、クエリ依存型リンク解析手法 SALSA の前処理化におけるクラスタリングとスコア近似を改善することで、従来の SALSA アルゴリズムにより近い WEB ランキングを作成することができ、かつ応答時間の少ない手法を提案する。

第2章　関連研究

本章では、クエリ依存型リンク解析手法 SALSA に関する技術について、特に関連のある研究について紹介する。

2.1　WEB ランキングアルゴリズムの歴史

巨大化した WEB の中から目的の WEB ページを見つけるには、検索エンジンを使うのが有効である。検索エンジンに与えた検索キーワード、別名クエリと関連のあるページに対して順位付けをするのが WEB ランキングアルゴリズムである。

1998 年まで WEB ランキングアルゴリズムは、WEB ページのテキストや HTML 構造といった内容得点で順位付けを行っていた。この内容得点というのは、各 WEB ページにどこにどれほどクエリの文字列が含まれているのか計算したものであった。しかし、WEB が巨大化し限りなく増大していったことや、内容得点がスパムの影響を受けやすかったことから、1998 年までには従来の内容得点は有効でないことが明らかになっていた。

そこで、内容得点だけではなく、WEB のハイパーリンク構造に対して有向グラフを作成し、その有向グラフを用いて WEB ページの人気得点を計算するリンク解析手法が登場した。

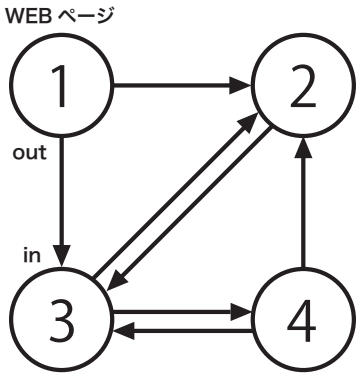

図 2.1: WEB グラフ

これによって従来の内容得点によるランキングよりも品質が向上し、検索エンジンの利用者は劇的に増え、リンク解析手法は WEB ランキングアルゴリズムの主流となった。

2.2 PageRank

PageRank は、当時 Stanford University に在学していた Sergey Brin と Larry Page の 2 人の計算機科学科の学生によって開発されたものであり、開発者の名前がアルゴリズムの由来となっている。これがのちの WEB 検索エンジン Google であり、このアルゴリズムの基本的な概念は「多くの良質なページからリンクされているページは、やはり良質なページである」という再帰的な関係をもとに、全てのページの重要度を判定したものである。

以下の図 2.2 は PageRank の概念図である。あるページのスコアを、そのページに存在する出リンク数で割った数が、それぞれの被リンク先のスコアに加算されるという関係になっている。

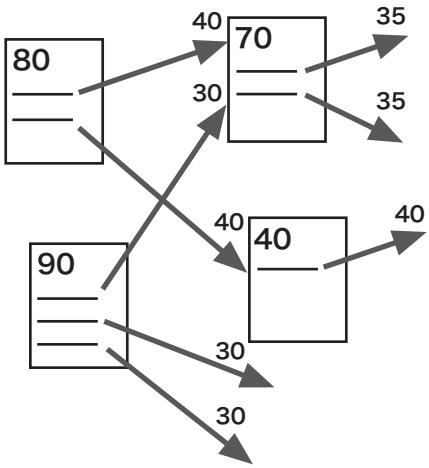

図 2.2: PageRank の概念図

PageRank は、WEB グラフにランダムウォークモデルを適応し、各ページへの遷移確率によって WEB ランキングを作成する。あるページからそのページが指している全リンク先へとランダムに遷移する確率と全ての WEB ページへとランダムに遷移する確率を足したスコアが PageRank のスコアになる。

PageRank のスコア計算は以下の式 (2.1) で定義される。

$$\pi^T = \pi^T(\alpha S + (1-\alpha)E) \tag{2.1}$$

ここで π^T は PageRank ベクトルであり、α は 0.85 程度のスカラー値であり、行列 S はページ i からそのリンク先であるページ j への遷移確率行列である。また行列 E は全ての WEB ページへのレポーテーション行列である。

ここで行列 S は、ページ i からそのリンク先ページ j への遷移確率を示すので以下のように定義することができる。

$$S_(i,j) = \begin{cases} 1/out(i) \quad (out(i) \text{ はノード } i \text{ の出リンク数}) \\ 0 \quad (\text{ノード } i \text{ からノード } j \text{ へのリンクがないとき}) \end{cases} \tag{2.2}$$

また、行列 E はページ i から全てのページへのランダムな遷移を示すので、以下 (2.3) のように定義することができる。

$$E_(i,j) = 1 \tag{2.3}$$

PageRank の特徴の 1 つとして、クエリ独立型のリンク解析手法であるという点が挙げられる。これは、クエリに関係なく、最初から全ての WEB ページに対する WEB ランキングが決定しているということである。クエリが与えられてからは、既に作成されている WEB ランキングに適切なフィルタリングを施し、検索結果として WEB ランキングを表示するだけである。このため PageRank は、HITS や SALSA のようなクエリ依存型のリンク解析手法に比べ、応答時間が短い。

2.3 HITS

HITS は、1998 年に Jon Kleinberg によって発明された。PageRank と同様、WEB ページに関連した人気得点を作るのに WEB のハイパーリンク構造を用いる。しかし、PageRank と違い HITS は各ページに対して権威スコアとハブスコアという 2 種類のスコアを作成し、権威ページとハブページを定義する。

権威ページはたくさんの入リンクを持つページであり、ハブはたくさんの出リンクを持つページである。また、良い権威ページたちは良いハブページたちによってリンクされており、良いハブページたちは良い権威ページたちをリンクしているという循環的な性質を持っている。

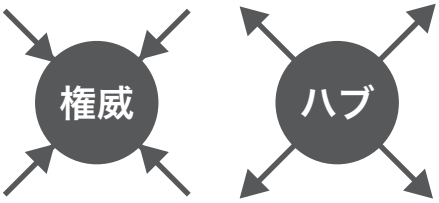

図 2.3: 権威とハブ

HITS では WEB の構造を有向グラフとして捉え、各スコアの計算を行う。権威スコアとハブスコアを計算する上で有向グラフの隣接行列 L を用いて以下のように行列の形で書くことができる。

$$L_(i,j) = \begin{cases} 1 & (\text{ノード } i \text{ からノード } j \text{ へのリンクがあるとき}) \\ 0 & (\text{ノード } i \text{ からノード } j \text{ へのリンクがないとき}) \end{cases} \tag{2.4}$$

この隣接行列 L を用いて、権威スコア x とハブスコア y の計算は以下の式 (2.5)(2.6)(2.7) で定義される。この計算は、ハブスコアの初期値を $y(0) = e$ とし、各スコアが収束するまで計算を繰り返す。ここで e は全て 1 の列ベクトルである。

$$x^{(k)} = L^T y^{(k-1)} \tag{2.5}$$

$$y^{(k)} = L x^{(k)} \tag{2.6}$$

$$k = k + 1 \tag{2.7}$$

HITS の特徴の一つとして、クエリ依存型のリンク解析手法であるという点が挙げれられる。クエリが与えられると、クエリに関連するページの集合である root set の抽出を行う。そこから、root set とリンク関係にある距離 1 のページを含めた集合である base set に拡張する。その後、base set の隣接行列を作成し、base set 内の各ページに対してスコアを割り当て正規化を行い、WEB ランキングを作成する。以下の図 2.4 は root set から base set を作成する処理を視覚化したものである。

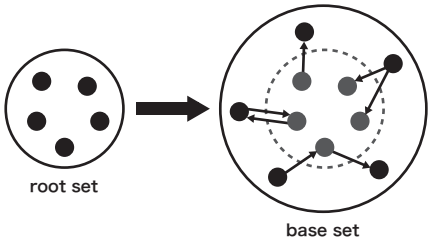

図 2.4: root set と base set

HITS はクエリに適した検索結果を得られるが、クエリが与えられてから base set の作成とスコア計算を行うため、無視できないほどの応答時間がかかってしまう。

さらに、ランキングの対象となる base set を WEB のリンク構造から作成するため、クエリと関係の小さいページがランキング上位に表示されてしまうことがある。この現象を話題の横滑り (topic draft) と呼ぶ。

2.4 SALSA

SALSA は、2000 年に Ronny Lampel と Shlomo Moran によって発明された。クエリ依存型のリンク解析手法の一つで、PageRank と SALSA の長所が組み込まれている。HITS と同じように、SALSA は WEB ページの権威得点とハブ得点の両方を作り、PageRank のように、それ

らはマルコフ連鎖から作られる。そのため、近年の研究においてSALSAはPageRankやHITSよりも、さらにクエリに適した検索結果が得られるということがわかっている。

SALSAでは、HITSで用いた隣接行列Lに対し、各ノードの出リンク数および入リンク数によって行と列の両方の重み付けを用いている。これは、HITSにおける話題の横滑り現象を回避するためである。Lの各非ゼロ行列をその行の総和で割ったものをL_r、Lの各非ゼロ行列をその列の総和で割ったものをL_cとし、これらは以下のように定義される。

$$L_{r(i,j)} = \begin{cases} 1/out(i) & (out(i)\text{はノード}i\text{の出リンク数}) \\ 0 & (\text{ノード}i\text{からノード}j\text{へのリンクがないとき}) \end{cases} \tag{2.8}$$

$$L_{c(i,j)} = \begin{cases} 1/in(i) & (in(i)\text{はノード}i\text{の入リンク数}) \\ 0 & (\text{ノード}i\text{からノード}j\text{へのリンクがないとき}) \end{cases} \tag{2.9}$$

権威スコアxとハブスコアyの計算は以下の式(2.10)(2.11)(2.12)で定義される。HITSと同様に、ハブスコアの初期値を$y^(0) = e$とし、各スコアが収束するまで計算を繰り返す。

$$x^{(k)} = L_c^T y^{(k-1)} \tag{2.10}$$

$$y^{(k)} = L_r x^{(k)} \tag{2.11}$$

$$k = k + 1 \tag{2.12}$$

SALSAもHITSと同様に、クエリが与えられてからbase setの作成とスコア計算を行うため、無視できないほどの応答時間がかかってしまう。

第3章　提案手法

本章では、クエリ依存型リンク解析手法である SALSA アルゴリズムを高速化する手法について解説する。

3.1　クエリ依存型リンク解析手法高速化の概要

以下の図 3.1 は PageRank に代表されるクエリ独立型リンク解析手法における処理の流れを視覚化したものである。

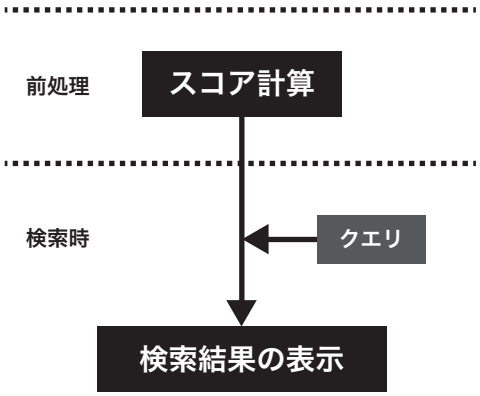

図 3.1: クエリ独立型リンク解析手法

クエリ独立型リンク解析手法では、あらかじめ全 WEB ページのスコアを計算し、クエリに依存することなく WEB ランキングを作成する。クエリが与えられてからは適切なフィルタリングを施しランキングを表示するだけなので、応答時間は極めて短い。

以下の図 3.2 はクエリ依存型リンク解析手法における処理の流れを視覚化したものである。クエリ依存型リンク解析手法では、WEB ページの集合からクエリに関連する部分を base set として抽出し、抽出された WEB ページの権威スコア、ハブスコアを計算し、ランキングを表示する。クエリが与えられてから base set の作成、各スコアの計算を行うため長い応答時間がかかってしまう。

そこで、先行研究では、WEB グラフの抽出と各スコアの計算を前処理化することでクエリが与えられてからのランキング作成を高速化する手法が提案された。この手法では、あらかじめ WEB グラフをクラスタリングしておき、各クラスタ内での WEB ページに対して権威スコアとハブスコアを計算しておく。その後、クエリが与えられるとクエリに関連する WEB ページを含むクラスタを抽出し、そこからクラスタ内での各 WEB ページのスコアを元に最終的なスコアを近似する。

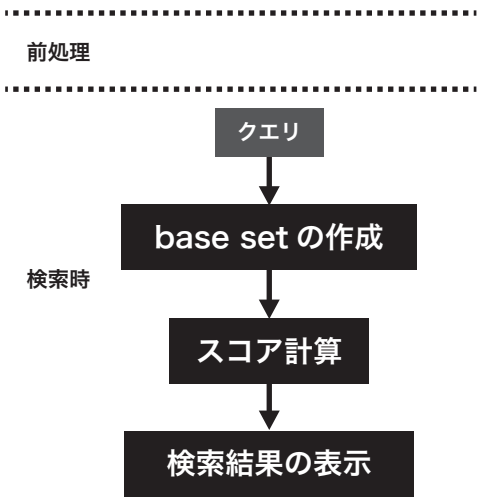

図 3.2: クエリ依存型リンク解析手法

以下の図 3.3 は高速化したクエリ依存型リンク解析手法における処理の流れを視覚化したものである。図の左側は従来のアルゴリズムによる処理を示し、右側は高速化したアルゴリズムによる処理を示している。

全ての WEB ページをクラスタリングし、各スコアの計算をあらかじめ処理しておくことで、クエリが与えられてからの応答時間を大幅に短くすることができる。しかし、高速化した手法で求めた WEB ランキングと、従来の方法で求めた WEB ランキングには大きな差ができてしまった。これは、適切なクラスタリングが行われず、最終的なスコア近似の精度が低くなり、クエリと関連の少ないページがランキングに入ってしまうためであると考えられる。

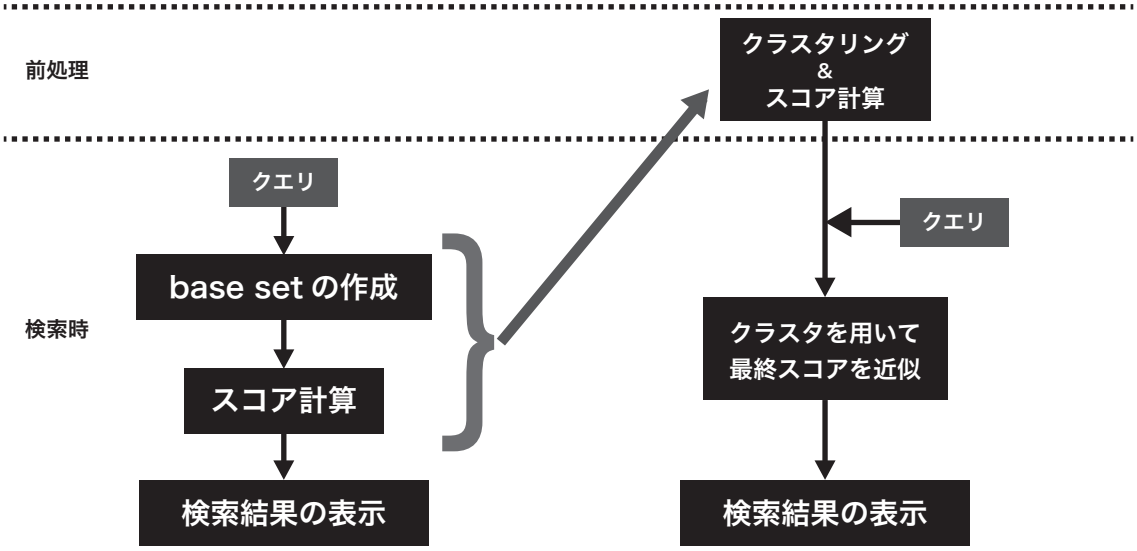

図 3.3: クエリ依存型リンク解析手法の高速化

3.2 提案手法の概要

この節では、まずはじめに提案手法の概要について解説する。高速化した SALSA アルゴリズムでは、従来の SALSA アルゴリズムより高速な処理が可能だが、従来の SALSA アルゴリズムで作成したランキングと比べてランキングの精度が落ちてしまうという問題があった。そこで本研究では、従来の SALSA アルゴリズムより高速で、かつ従来の SALSA アルゴリズムに近い WEB ランキングを作成する手法を提案する。具体的には、先行研究と同様に WEB ページのクラスタリングと各スコアの計算を前処理として行うが、そのクラスタリング部分と最終スコアの近似部分において改良を加える。

クラスタリングでは、まず少数のページで初期セットを作成し、そこにリンク関係のあるページを追加していくことでクラスタを大きくしていく。先行研究による高速化アルゴリズムでは、初期セットのサイズを固定し、全てのページが 1 つ以上のクラスタに所属するまでクラスタリングを行う。この場合、最終スコアの近似を行う際に、多くのクラスタに所属しているページほどスコアが大きくなり、適切なランキングを得ることができない。

そこで今回提案する手法では、全てのページはそれぞれ 1 つのクラスタにのみ所属するものとする。したがって、1 度クラスタに追加されたページが再び別のクラスタに所属することはないので、クラスタリングにおける処理の高速化も期待することができる。これに伴い、最終スコアの近似式に改良を加える。

詳しいクラスタリングアルゴリズム、スコアの近似については、次節以降で解説する。

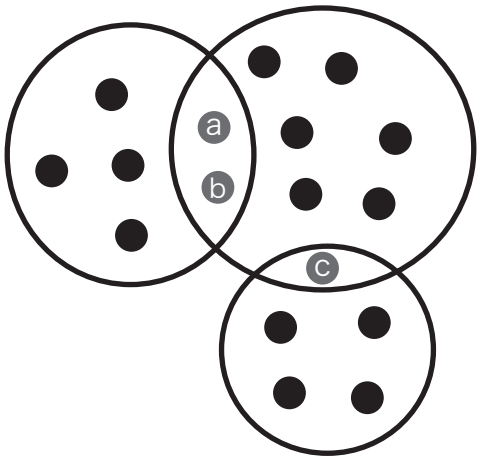

図 3.4: 全てのページが 1 つ以上のクラスタに所属

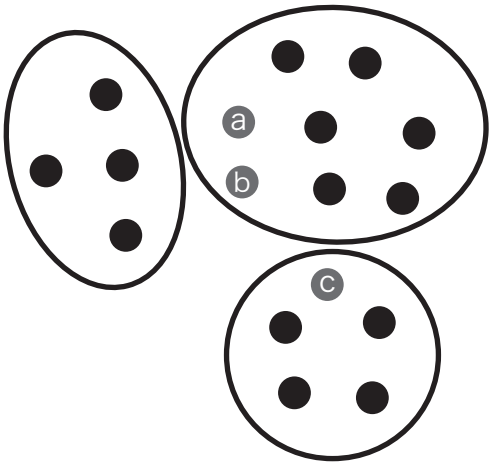

図 3.5: 全てのページが 1 つのクラスタに所属

3.3 クラスタリングアルゴリズム

高速化のための前処理として、全ての WEB ページをクラスタと呼ばれる集合に分けて、それぞれのクラスタ内で各スコア計算を行う。ここでは、クラスタリングアルゴリズムについて解説する。

まず、クラスタの seed page を選択する。seed page は、まだどのクラスタにも属していないページのうち、出リンク数、入リンク数の多いものを選択する。

その後、seed page と相互リンクしているページ、出リンク先や入リンク元となるページを順にクラスタへ追加し、初期セットを作成する。以下の図 3.6 から 3.8 で初期セット作成までの流れを視覚化した。

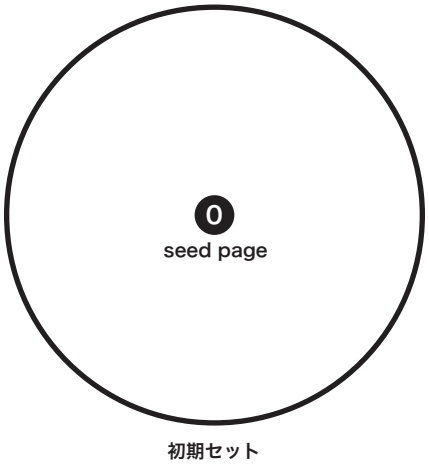

図 3.6: クラスタへ seed page の追加

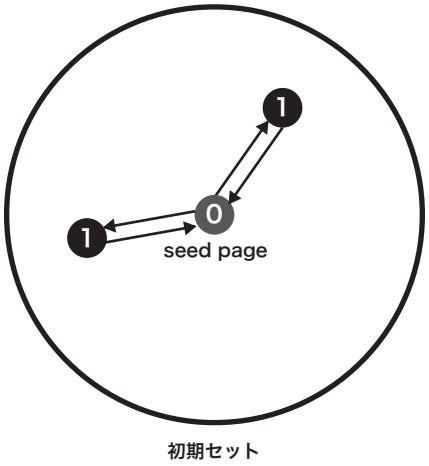

図 3.7: seed page と相互リンクしているページの追加

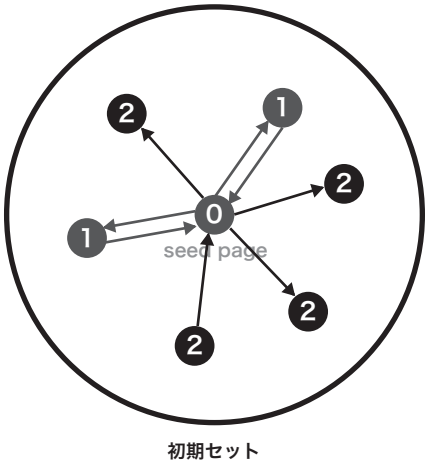

図 3.8: seed page の出リンク先、入リンク元であるページの追加

先行研究における手法では、初期セットの最大サイズを全ページ数の 1/100 としていた。これは、クラスタの最大サイズを全ページ数の約 1/10 と見積もり、初期セットの最大サイズがさらにその 1/10 となるように見積もった数値である。この場合、初期セット候補が最大サイズを越えた時、本来 seed page とリンク関係にあるページを適切に抽出することができない。

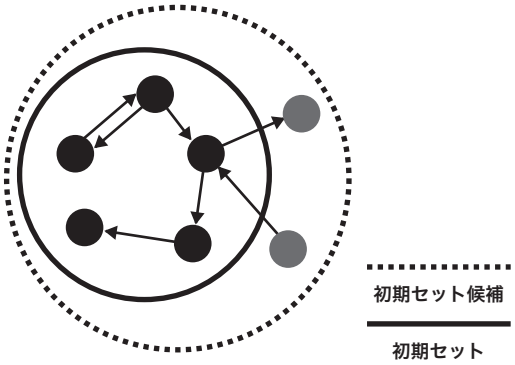

図 3.9: 先行研究における手法での初期セット定義 (最大サイズ 5 とした時)

そこで今回提案する手法では、最終スコアの近似により適したクラスタを作成するため、初期セットのサイズを固定にするのではなく、seed page とリンク関係にある距離 1 のページを全て初期セットとする。

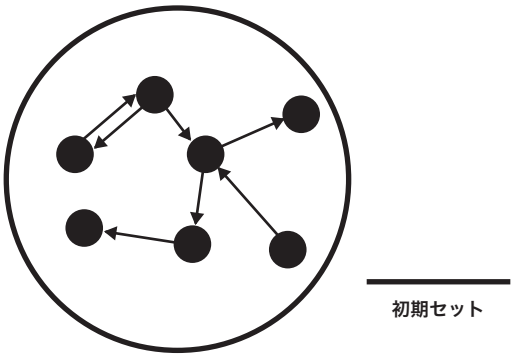

図 3.10: 本研究における手法での初期セット定義

初期セットを作成した後は、初期セット内の各ページに対して SALSA スコアを計算し、スコアの高いページとリンクしているページから順にクラスタへ追加していく。

この時、WEB ランキングの元となる SALSA スコアの計算上、注意しなければならない点がある。SALSA アルゴリズムでは、各ページの出リンク数と入リンク数によってスコアに重み付けを行う。例えば、ハブスコアの高いページ i からリンクされているページ j があるとする。このとき、ページ i の出リンク数が多い場合、必ずしもページ j の権威スコアが高くなるとは限らない。このことから、単純にハブスコアの高いページの出リンク先や、権威スコアの高いページの入リンク元をクラスタに追加することは適切ではない。

そこで、本研究における手法では、クラスタ内での各ページに対して「ハブスコアを出リンク数で割った値」と「権威スコアを入リンク数で割った値」を計算し、それぞれ新たなスコアとする。そこから、新たなハブスコアが高いページの出リンク先と、新たな権威スコアが高いページの入リンク元を、交互にクラスタへ追加することにする。以降、「新たなスコア (元の SALSA スコアを出入リンク数で割った値) が高いページ」のことを「スコアが高いページ」と表現する。

以下の図 3.9 と 3.10 でクラスタにページを追加する様子を視覚化した。

その後、クラスタ内にあるページの各スコアを基準としてクラスタへのページの追加を終了する。クラスタにページを追加する際は、前に述べたように、クラスタ内でのスコアが高いページから順にリンクを辿る。ここで、リンクを辿ろうとしているページのスコアがあらかじめ設定した閾値より低くなった時、そのページ以降リンクは辿らずにページの追加を終了する。つまり、新たに追加するページは、まだどのクラスタにも所属せず、かつクラスタ内で閾値より高いスコアを持つページとリンクしている必要がある。この条件を満たすページが存在しなくなった時点でページの追加を終了し、次のクラスタの作成に移る。以下の図

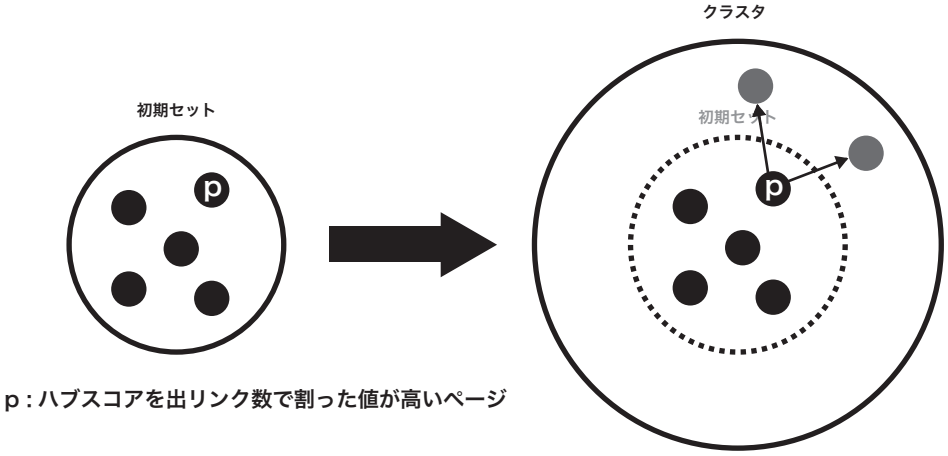

図 3.11: ハブスコアによるページの追加

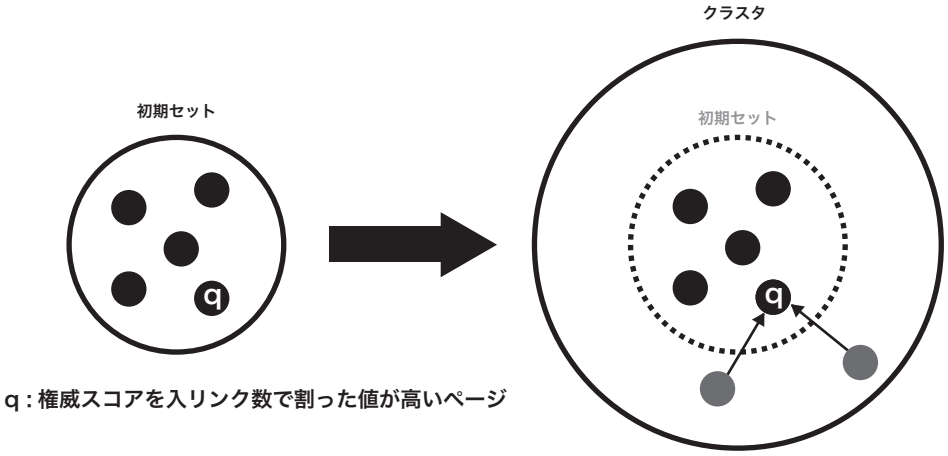

図 3.12: 権威スコアによるページの追加

3.13 でページの追加を終了する時点でのクラスタの様子を視覚化した。

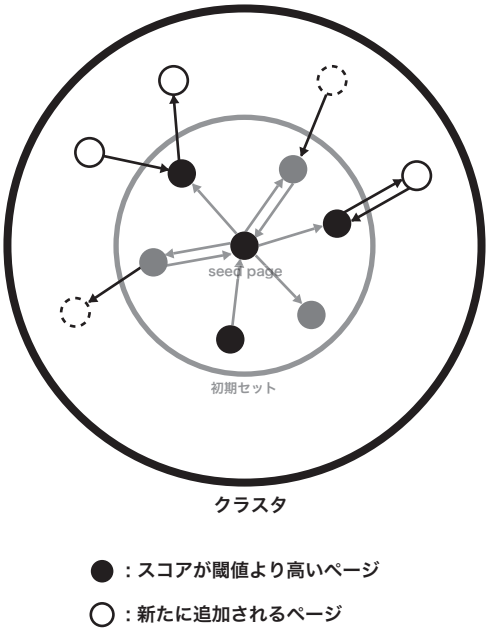

図 3.13: ページの追加を終了する時点でのクラスタ

このようにしてクラスタを作成していき、全てのページが 1 つのクラスタに所属した時点でクラスタリングを全て終了する。

以上のことを踏まえ、クラスタリングアルゴリズムを以下に示す。

Algorithm 1 クラスタリングアルゴリズム

```
 1: PageList = AllPages
 2: while PageList ≠ empty do
 3:    Cluster << FindSeed
 4:    Cluster << MutualPages(Seed)
 5:    Cluster << InLinkPages(Seed)
 6:    Cluster << OutLinkPages(Seed)
 7:    CalcScore(Cluster)
 8:    loop
 9:       if TopAuthScorePage.score > 閾値 then
10:          Cluster << InLinkPages(TopAuthScorePage)
11:       end if
12:       if TopHubScorePage.score > 閾値 then
13:          Cluster << OutLinkPages(TopHubScorePage)
14:       end if
15:       if TopAuthScorePage.score < 閾値 ∩ TopHubScorePage.score < 閾値 then
16:          break
17:       end if
18:    end loop
19:    ClusterList << Cluster
20:    PageList = PageList - Cluster
21: end while
```

1-2 行目は、全てのページがクラスタに所属するまでクラスタリングを続けることを示している。まず 1 行目で全てのページを記憶した PageList を作成する。1 つのクラスタを作成し終わるごとに、20 行目でそのクラスタに含まれるページが PageList から削除される。そして、2 行目が示しているように、PageList が空になるまでクラスタリングを続ける。

3-6 行目ではクラスタの初期セットを作成している。3 行目の FindSeed 関数でシードとなるページをクラスタへ追加する。ここでシードとなるページは、まだどのクラスタにも所属していない、かつ PageList の中で出リンク数や入リンク数の高いものを優先して選ぶ。その後 4-5 行目で、seed page と相互リンクしているページを MutualLinkPages 関数、seed page の入リンク元を InLinkPage 関数、出リンク先を OutLinkPage 関数により取得し、クラスタに追加する。初期セットの作成が完了すると、7 行目の CalcScore 関数でクラスタ内での各 SALSA スコアを計算する。

その後 8-18 行目を繰り返し、クラスタへページを追加していく。15-17 行目が示すように、追加すべきページが存在しなくなるまでクラスタの拡張を続ける。

追加するページを選択するのに、7 行目で計算した SALSA スコアを用いる。9-11 行目では、権威スコアが閾値より高いページのみ、その入リンク元となるページを追加している。TopAuthSocrePage というのは、初期セット内において権威スコアを入リンク数で割った値が

最も高いページであり、.score 関数によりその値を取得する。同様に 12-14 行目では、ハブスコアを出リンク数で割った値が閾値より高いページのみ、その出リンク先となるページを追加している。

これらを繰り返してクラスタの拡張を行う。19 行目で ClusterList へ保存し、20 行目でクラスタに追加したページを PageList から削除している。

3.4 最終スコアの近似

最後に、最終スコアの近似方法について解説する。この時点でクラスタリングは終了し、クラスタ内での各ページのスコア計算は終わっているものとする。

クエリが与えられた後、まずクエリと関連するページ (以降クエリページと呼ぶ) が含まれるクラスタを抽出する。その後、それぞれのクラスタに対し、そのクラスタの大きさ、そのクラスタがどれだけクエリページを含んでいるかによって重み付けを行う。これは、クエリとより関連のあるクラスタでのスコアを多く採点するためである。

先行研究におけるクラスタ c に対する重み w_c を以下の式 (3.1) で定義する。

$$w_c = \frac{q}{QN_c} \tag{3.1}$$

ここで、q はクラスタ c に含まれているクエリページの数であり、Q はクエリページの総数である。また、N_c はクラスタ c の総ページ数である。

本研究における手法では、全てのページは 1 つのクラスタにのみ所属する。これは先に述べたように、多くのクラスタに所属しているページほどスコアが大きくなることを防ぐためである。それに伴い、最終スコアの近似式にも改良を加える。先行研究では分母に持ってきていたクラスタの総ページ数 N_c を分子に持ってくる。これによって、クラスタサイズの大きいものほど大きい重みを持つことになる。

本研究におけるクラスタ c に対する重み w_c を以下の式 (3.2) で定義する。

$$w_c = \frac{qN_c}{Q} \tag{3.2}$$

抽出した全てのクラスタに対する重みを計算した後、さらにクエリと関連の高いクラスタの抽出を行う。そして、クエリとの関連が低いクラスタは最終スコアの近似に使用しないこととする。これは、最終スコアの近似計算をするクラスタを減らすことで応答時間を減らすとともに、余計なページをランキングに追加しないためである。

先行研究では、クラスタに対するクエリとの関連度に重みの平均値を用いていた。しかし、平均値を用いる手法では重みに大きくばらつきがあった際に、重みの大きなクラスタのページみがランキングに追加され、重みの小さなクラスタのページはランキングに追加されず、適切なランキングを得ることができなかった。

そこで本研究では、クエリと関連の高いクラスタの抽出に中央値を用いる。中央値とは、測定値を小さい順に並べたとき，ちょうど真ん中にくる値であり、重みの両端に大きな値や小さな値があっても影響されない。また、重みがゼロのものは中央値の計算に考慮しないものとする。重みがゼロのものを除いたクラスタにおける重みの中央値 M_e は以下の式 (3.3) で定義する。

$$M_e = \begin{cases} X_m & n \text{が奇数のとき}, m = (n+1)/2 \\ (X_m + X_{m+1})/2 & n \text{が偶数のとき}, m = n/2 \end{cases} \tag{3.3}$$

ここで、X は小さい順から並べた各クラスタの重みであり、n は重みがゼロのクラスタを除いたクラスタの数である。

次に、抽出したクラスタ内での全てのページに対して近似後の最終スコアを計算する。各ページにおけるスコアは以下の式 (3.4) で定義する。

$$score(p, Q) = w_c s(p, c) \tag{3.4}$$

ここで、Q はクエリページ集合を表し、$score(p, Q)$ はクエリページ集合 Q に対するページ p のスコアを表している。$s(p, c)$ はクラスタ c におけるページ p のスコアである。ここで、c は重みが中央値より高いものとする。

以下の図 3.14 と 3.15 で最終スコアを近似する流れを視覚化した。図中の円はそれぞれクラスタを表している。

まず、クエリページを含むクラスタを抽出する。図 3.14 の例ではクラスタ a,c,d が該当する。次にそれぞれのクラスタに対する重みをクラスタサイズ、クエリページの総数、クラスタに含まれるクエリページ数から計算する。その結果、重みが中央値以上だったクラスタを a,d とする。ここから、クラスタ a と d 内の全てのページに対し、近似後の最終スコアを計算する。最後にクエリページを含む全てのクラスタ内の全ページをソートし、ランキングを作成する。

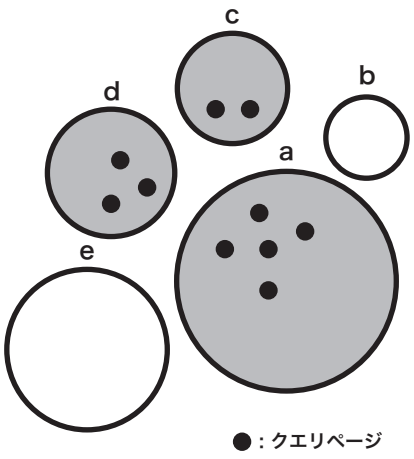

図 3.14: クエリページが含まれるクラスタの抽出

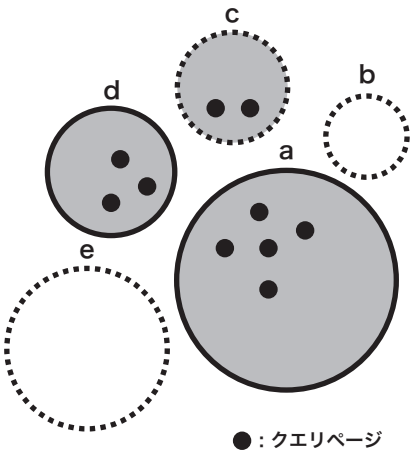

図 3.15: 重みが中央値以上であるクラスタの抽出

以上のことを踏まえ、スコアの近似アルゴリズムを以下に示す。

Algorithm 2 最終スコアの近似アルゴリズム

```
1:  ClusterList = getQueryCluseter(Query)
2:  for all Cluster such that Cluster ∈ ClusterList do
3:      Wight << (HaveQueryNum(Cluster) ∗ Cluster.size) / Query.size
4:  end for
5:  for all Cluster such that Cluster ∈ ClusterList do
6:      if Weight[ClusterID] ≥ Weight.median then
7:          ApproxAuthScore[PageID] = Weight[ClusterID] ∗ Cluster.AuthScore[PageID]
8:          ApproxHubScore[PageID] = Weight[ClusterID] ∗ Cluster.HubScore[PageID]
9:      end if
10: end for
```

変数 Query は、クエリページの集合である。1 行目では、少なくとも 1 つ以上のクエリページを含むクラスタを getQueryCluster 関数により抽出し、ClusterList を作成している。

2-4 行目では、ClusterList の全てのクラスタに対して重みを計算している。3 行目の HaveQueryNum 関数により、Cluster に所属しているクエリページの数を取得し、クラスタのサイズとかけあわせ、クラスタの総ページ数で割る。計算した重みは中央値の算出と最終スコアの近似に使用するため、配列を使い全て記憶しておく。中央値は.median 関数により取得する。

5-9 行目で最終スコアの近似を行う。5-6 行目が示すように、ClusterList のうち重みが中央値以上であるクラスタのみを用いて計算を行う。7 行目では、それぞれのページに対し権威スコアに重みをかけている。8 行目では同様にハブスコアに重みをかけている。Cluster.AuthScore

および Cluster.HubScore は、それぞれ書くページにおけるクラスタ内での権威スコアとハブスコアである。ApproxAuthScore および ApproxHubScore は、それぞれ各ページにおける最終的な権威スコアとハブスコアである。

最後に、近似された最終スコアをソートすることでランキングを作成し、掲示する。

第4章　実験と考察

4.1　評価指標

従来の SALSA アルゴリズムと提案手法を用いて、同一の WEB ページの集合から 2 つの WEB ランキングを作成し比較する。従来の SALSA アルゴリズムで作成された WEB ランキングと、提案手法によって作成された WEB ランキング、どれだけ一致しているかについての評価指標として適合率を用いた。

適合率は以下の式 (4.1) で表すことができる。

$$presicion = \frac{R}{N} \tag{4.1}$$

ここで R は、一方の WEB ランキングに含まれるページの内、もう一方の WEB ランキングにも含まれているページの数である。N はランキングのサイズである。適合率は、2 つのランキングがどれほどランキング内に同じページを含んでいるかを表し、0 から 1 までの値を取る。2 つのランキングが全て同じページの場合で最大の 1 を、同じページが全く含まれていない場合で最小の 0 を返す。

また、従来の SALSA アルゴリズムと提案手法の応答速度の比較は以下の式 (4.2) で定義する。

$$response = \frac{q}{p} \times 100 \tag{4.2}$$

ここで p は従来の SALSA アルゴリズムにおける応答時間であり、q は前処理を行った高速化 SALSA アルゴリズムの応答時間である。従来の SALSA アルゴリズムの応答時間を 100% とした時、提案手法の応答時間がどれだけ短くなっているかを表している。

4.2　実データでの実験

4.3　大規模データでの実験

第5章　まとめ

てすてす

第6章　形式

ここでは、論文の表紙および本体の記述方法について述べる。

6.1　表紙

表紙は、\maketitle によって作成するため、以下の項目に相当する文字列をそれぞれ記述する。

題目: 題目は \title に記述する。行替えを行う場合は \ \ を入力する。ただし、題目の最後に \ \ を入力するとコンパイルが通らなくなるので注意する。なお、4 行以上の題目の場合、表紙ページがあふれるためスタイルファイル “mast-jp-sjis.sty” を変更する必要がある。

著者名: 著者名は \author に記述する。

指導教員名: 指導教教員は \advisor に記述する。

年月: 年月は \yearandmonth に記述する。年月は提出時のものを記述すること。

6.2　本体

本体は 1 段組で記述する。

図表には番号と説明 (caption) を付け、文章中で参照する。表 6.1 と図 3.15 はそれぞれ表と図の例である。表の説明は上に、図の説明は下に書くことが多い。図の挿入に用いるパッケージについては使用環境に合わせて自由に選択してほしい。

表 6.1: 表の例

年 度	1 年次	2 年次	3 年次	4 年次
1995	85	92	86	88
1996	83	89	90	102
1997	88	87	91	112
1998	144	93	90	115

詳しくは参考書 [1, 2] などを参照のこと。また、奥村晴彦氏の「TEXWiki」http://oku.edu.mie-u.ac.jp/~okumura/texwiki/ は、日本語の TEX に関する情報が充実している。また、具体的な論文としての文献参照例として (本の例)[3]、(雑誌論文の例)[4]、(予稿集の例)[5] を挙げておく。

謝辞

感謝

参考文献

[1] 奥村晴彦, LaTeX2e 美文書作成入門 改訂第 5 版, 技術評論社, 2010 年.

[2] 吉永徹美, LaTeX2 ε 辞典, 翔泳社, 2009 年.

[3] Colin Ware, Information Visualization — Perception for Design, Second Edition, Morgan Kaufmann Publishers, 486 p., 2004.

[4] Miriah Meyer and Tamara Munzner, MizBee: A Multiscale Synteny Browser, *IEEE Transactions on Visualization and Computer Graphics*, Vol. 15, No. 6, pp. 897–904, 2009.

[5] Emerson Murphy-Hill and Andrew P. Plack, An Interactive Ambient Visualization for Code Smells, *in Proceedings of the 2010 International Symposium on Software Visualization (SOFTVIS ' 10)*, pp. 5–14, 2010.