修 士 論 文

先読み代理サーバを用いたWWW情報探索支援

指導教官　白井清昭 助教授

北陸先端科学技術大学院大学
情報科学研究科情報処理学専攻

新井 孝之

2002年3月

概 要

現在 Web 上には膨大な情報が存在するため，Web から有益な情報を探し出す作業に多くの時間がかかる場合も多い．本研究では，WWW 情報探索支援のため，ユーザが参照する前にリンク先の Web ページの要約をユーザに提示することによって，どのリンク先が重要かを判断できるシステムを提案する．システムはプロキシサーバとして実装され，ユーザがリンクにマウスカーソルを置いたときにリンク先の Web ページの要約を表示する．リンク先の要約は，ユーザから要求がくる前にあらかじめ先読みして作成しておく．通常 Web ページにはリンクが複数あるため，効率や実時間性の制約から先読みする順序を工夫する必要がある．本研究では先読みする順序をアンカーテキストとユーザプロファイルから決定する．

目 次

図 目 次

表 目 次

第1章　はじめに

1.1　研究の背景

近年，World Wide Web(以下 WWW，Web と略す) の普及により，世界中に分散された文書に容易にアクセスすることが可能になった．しかし，その情報空間は膨大で，ユーザが知りたい情報をすぐに得られないことも多い．そのため，WWW における情報探索を支援する技術を開発することは重要な課題である．

しかし，WWW では，それぞれの情報をどこに置くか決まっているわけではない．また，それぞれの情報の価値や内容について評価がなされていない．現状では，情報探索を支援するための検索エンジンやディレクトリサービスがいくつも存在するが，それらを利用してさえも，目的の情報にたどり着くためにはブラウジング作業に多大な時間がかかる．

通常，Web ページには多数のリンクが付けられており，ユーザはリンク先にジャンプしたり，また戻ってきたりして Web ブラウジングを行う．この動作を行うことは非常に煩わしい．途中に興味のあるページが多数あると，ディスプレイがウィンドウでいっぱいになったり，自分が現在どこにいるのかわからなくなることさえある．リンク先へジャンプする動作を減らすことができれば，より快適にブラウジング作業を行うことができると考えられる．

1.2　研究の目的

本研究では，WWW 情報探索において，ユーザがどのリンク先をたどればよいかの手がかりとして，リンク先の要約を提示することによって，どのリンク先が重要かをユーザが実際に参照する前に判断できるシステムの構築を目指す．

近年，Web のように，電子化されたテキストが大量に利用可能になっていることから，テキスト自動要約は脚光を浴びてきている [奥村 00] [奥村 99]．本研究では，要約生成技術を WWW 情報探索支援に用いることを考える．本研究で提案するシステムは，ユーザが Web ページ上のリンクにマウスカーソルを置くと，リンク先のページの要約が見えるようにすることでブラウジングの支援を行う．システムは，先読み代理サーバとして実装する．要約は，ユーザのブラウザに Web ページを表示する際にリンクを解析し，ユーザから要約の要求がくる前に作成しておく．しかし，通常 Web ページ上にはリンクが複数あるため，効率や迅速性の観点から先読みする順序を工夫する必要がある．本研究で

は，どのリンク先から要約を作成するかの順序付けを Web ページ中のアンカーテキストとユーザプロファイルから決定する．これによって，ユーザの要約の要求に迅速に応えることができる．

1.3 論文の構成

第 2 章では，本研究と関連する研究を概観し，本研究と比較する．

第 3 章では，本研究で提案する WWW 情報探索支援システムに必要とする条件を考察する．また，それらの条件をどのように実現するのかといったシステムの設計方針について述べる．

第 4 章では，第 3 章で述べた設計方針に基づくプロトタイプシステムの実装について説明する．

第 5 章では，WWW 情報探索のタスクを設定し，プロトタイプシステムの評価実験を行う，また，その結果について考察する．

第 6 章では，結びとして本研究のまとめを行い，今後の課題について述べる．

第2章　関連研究

2.1　情報ナビゲーション

WWW での情報探索におけるブラウジング作業を支援する研究は，情報ナビゲーションの分野でなされている．Greenberg らは，ユーザの検索履歴を分析して，全体の 58 %はすでに一度見た Web ページであり，前の画面に戻るという操作はすべてのナビゲーションイベントの 30 %を占めることを明らかにした [Greenberg 97]．この前の画面に戻るという操作を減らすこと，すなわち，リンク先に頻繁にジャンプするイベントを減らすことは，Web ブラウジングの効率を上げるのに役立つと考える．本章では，このような考えに基づいてブラウジング作業を支援する既存の研究を概観し，本研究と比較する．

2.1.1　検索結果の Web ページを自動表示

林らは,「単位時間当りに多くの Web ページを表示すること」を目的として，Web ページを紙芝居のように自動的に次々と切り替えて表示させるシステムを提案している [林 99]．このシステムは既存のロボット型検索サービスをそのまま利用し，単位時間当たりに閲覧する Web ページ数を増加させることにより，必要な情報を含む Web ページを探し出すまでの時間を短縮させることができる．この研究では，検索サービスの支援のみを行っているのに対し，本研究では，検索結果として出力された Web ページだけでなく，通常の Web ページのブラウジング支援も行う．本研究では，ユーザが Web ページ上のリンクにマウスカーソルを置くと，リンク先のページの要約が見えるようにすることでブラウジングの支援を行う．

2.1.2　情報空間を表示

武藤らは，Web 情報空間を 3 次元画像として総覧できる InfoLead ブラウザを提案している [武藤 99]．このシステムは，キーワード検索などで Web ページ情報を絞り込んだ結果を一枚一枚個別に閲覧することなく，一度に総覧することが可能である．マウスの操作により 3 次元空間内で視点を自由自在にコントロールすることもできる．また，Web ページを何らかの方法で評価して，サービスや利用目的に合わせて 3 次元空間に配置することを提案している．しかし，どのような評価方法を利用するかに関しては考えられていない．

このシステムでは，Web 情報空間を 3 次元画像として作成する際に，コンテンツを入手する時間的問題，トラフィックの問題は全く考慮されていない．本研究では，先読みする情報はリンク先の HTML テキストとプレインテキストに限定し，キャッシュに格納しておくことで他のユーザも再利用できる．また，先読みして要約を作成する順序をユーザの興味から決定することで，ユーザの要約の要求に迅速に応えることができると考える．

2.1.3 複数のウィンドウを制御

Kandogan らは，ウィンドウを制御し，ユーザが Web ページ上のリンクをクリックすると，リンク先のページを，現在のウィンドウに表示せずに階層構造がわかるように並べて表示する [Kandogan 97]．しかし，ディスプレイの広さに限りがあることから，多数のウィンドウを並べて表示することはかえってユーザにとってわかりにくくなる．本研究では，リンク先の情報を要約することによって，ユーザに提示する情報量を適度に減らし，リンク先にジャンプする必要があるのかどうかユーザが判断できるようにする．

2.1.4 リンク先の情報を表示

Harald らは，Web ページ上のリンクにマウスカーソルを置くと，リンク先の様々な情報を表示するシステムを提案している [Harald 00]．表示する情報はリンク先のタイトル，言語，最終訪問日，サーバの応答速度，サイズなどである．これに対し，本研究では，このようなリンク先の情報ではなく，コンテンツの要約を表示する点が異なる．また，Harald らは，リンク先の情報が実際に Web 探索に有用であるかを実験的に確認したわけではない．本研究では，プロトタイプシステムを作成して評価実験を行い，要約を事前に表示させることが Web 探索の支援に有効であるかどうかを実験的に確認する．

Kopetzky らは，Web ページ上のリンクにマウスカーソルを置くと，リンク先のサムネイル画像を表示するシステムを提案している [Kopetzky 99]．このシステムは Proxy サーバで実装され，ユーザからリクエストのあった Web ページを解析し，リンク先のサムネイル画像を表示できるように Web ページを改変する．しかし，サムネイル画像を作成し表示する時間的な問題に関しては全く考慮されていない．これに対し，本研究では，サムネイル画像ではなく要約を表示する，要約を先読みすることにより，要約を表示する時間を短縮させている．

2.2 要約

テキスト自動要約はすでに４０年以上にわたって研究されてきた歴史がある．また，近年，電子化されたテキストがあふれ，情報洪水という言葉が日常的に使われていることから，テキスト自動要約は再び脚光を浴びてきている [奥村 99] [奥村 00]．本節では，このような現状を鑑み，これまでのテキスト自動要約手法を概観する．

2.2.1 重要文抽出

テキスト自動要約研究のこれまでの多くのものは，テキスト中の文 (あるいは，形式段落) を 1 つの単位とし，それらに何らかの情報を基に重要度を付与し，その重要度で順序付け，重要な文または形式段落を選択し，それらを寄せ集めることで，要約を作成する．本節では，この重要度評価の際に用いられている，テキスト中の (主に表層的な) 情報について述べる．

TF・IDF

テキスト中によく出現する内容語はテキストの主題を示す傾向があるとの仮定が情報検索分野などではしばしば用いられる．この仮定に基づき，テキスト中で出現頻度の高い名詞をキーワードと考える TF 法やさらにキーワードが現れるテキストの数も考慮する TF・IDF 法など，情報検索分野では，さまざまな単語の重み付け技法が用いられている [Salton 89]．単語の重要度から文の重要度を計算する手法はさまざま提案されているが，その一例としては，文中に出現する単語の重要度の総和を文の重要度とするものがある．

位置情報

テキストの構造から，テキスト中での重要な箇所の位置はある程度予測可能であると仮定して，テキスト中での文の位置情報をその文の重要度計算に利用する手法がいくつか考えられている．論説文の場合に，テキスト全体のまとめは書き出しや結び近くにあると仮定するものや，重要な文はテキストの先頭，最後，段落の先頭，最後，節の見出しの直後にあると考える [Edmundson 69] ものなどがある．また，新聞記事を対象とした重要文抽出では，本文の先頭数文を抽出するのが良いとされる [Brandow 95]．この手法は lead 手法と呼ばれることが多い．

タイトル等の情報

ジャンルにより決まったテキストの構造から得られるもう一つの情報として，本文の他に，テキスト中に付与されたタイトル，見出しの情報が挙げられる．たとえば，学術論文

の場合は，テキスト自体がタイトルを持つ場合もあり，また，各章，節にもタイトルが付与されることが多い．また，新聞には，見出し (headings)，小見出しが本文とは別に付与されることもある．

このタイトル，見出しは，テキスト本文の非常に簡潔な要約とも考えられる．そのため，タイトル，見出しに現れる内容語を含む文が重要であると考え，タイトル，見出し中の単語を重要文抽出に利用する手法がいくつか提案されている． 最近では，見出しに含まれる名詞を多く含む文を重要として抽出する [仲尾 97] などもその一例と考えられる．

手がかり表現

テキスト中の重要箇所を指示すると考えられる手がかり表現がいくつか存在する．たとえば，学術論文などでは，‘this report’, ‘in conclusion’, ‘our work’ など の表現は，論文の主題を表す文中に出現すると考えられる．このような手がかり表現を利用して，テキスト中の重要文を抽出する研究も存在する [Edmundson 69].

2.2.2 言い換えによる要約

重要文抽出による要約の他に，言い換えたり，合成したりすることで，原文の内容を表現し直し，要約 (abstract) として生成する試みが近年いくつか見られるようになってきた．この abstract の生成のためには，extract(テキスト中の重要概念の抽出) 以外に，抽出した概念の統合，生成の過程が必要である．概念の統合は，抽出された複数の重要概念を，何らかの知識を用いて，より高い階層の概念にまとめることである．これにより，テキスト中の重要概念は，より少ない数の概念で表されることになる．概念の統合には，概念階層やスクリプトといった知識が必要となる．Hovy らの SUMMARIST[Hovy 97] システムは，WordNet を概念階層として利用し，このような概念統合を実現している．Hovy らは，概念階層を用いた概念統合の例として，

John bought some vegetables, fruit, bread, and milk.

のような文を，概念階層を用いて，

John bought some groceries.

のように言い換える処理を示している．

しかし，このような概念階層やスクリプトといった知識を利用した要約は複雑で，時間もかかる．本研究の目的は，Web ページ閲覧中，リンク先が重要かどうかをユーザが判断する手助けをすることであり，要約を早く生成する必要がある．従って，言い換えによる要約は，実時間の制約がある本研究では不向きである．

第3章　システムの設計

本研究では，WWW 情報探索において，ユーザがどのリンク先をたどればよいかの手がかりとして，リンク先の要約を提示することによって，どのリンク先が重要かをユーザが実際に参照する前に判断できるようなシステムの構築を目指す．本章では，この目的を実現するためのシステムに必要とする条件について考察する．また，それらの条件をどのように実現するのかといったシステムの設計方針について述べる．

3.1　概要

まず，提案する WWW 情報探索支援システムの大まかな処理の流れについて説明する．

1. ユーザは通常の Web ブラウジングと同様に好きな Web ページを閲覧する．システムは，ユーザが要求した Web ページを解析し，要約が表示できるように JavaScript を追加する．また，そのページに存在するすべてのハイパーリンクについて，リンク先のページを先読みし，要約を作成してキャッシュに保存する．

2. ユーザが好きなアンカーの上にマウスポインタを置くと，JavaScript はリンク先の要約をツールチップまたはサブウィンドウに表示させる．要約を表示することにより，ユーザはリンク先の情報が有用であるかどうかを判断することができる．これにより，ユーザがあまり関心のないページを表示させてから元のページに戻る操作を少なくすることが期待できる．

3. ユーザがアンカーからマウスポインタを離すと，JavaScript はツールチップを消す．

4. ユーザが実際にリンクをクリックするなどして Web ページの要求をしたら，キャッシュに保存されている情報を取り出すことによってリンク先のページを高速に表示できる．キャッシュに保存されていなければ，インターネット上から入手する．

ユーザが WWW 情報探索を行っている間，上記の操作を繰り返す．本研究では，要約の表示方法として，ツールチップとサブウィンドウの 2 つを用いる．これらについては 3.2 節で詳しく説明する．

ツールチップによる要約の表示例を図 3.1 に示す．また，サブウィンドウによる要約の表示例を図 3.2 に示す．これらの例では,「インターネット入試」というアンカーにマウスポインタを置いたとき，そのリンク先の要約を表示している．ユーザは，要約を読むことによって，リンク先の概要を事前に把握することができる．

図 3.1: ツールチップによるリンク先の要約の表示

図 3.2: サブウィンドウによるリンク先の要約の表示

3.2 ユーザインタフェース

ユーザが Web ページ閲覧中，リンク先の要約を提示するためのユーザインタフェース設計方針を以下に述べる．

- 参照元のページのレイアウトは極力変更しない
  リンク先の要約を参照元のページ中に直接貼り付けてしまう方法や，リンク先の要約を表示させるためのボタンを参照元のページ中に貼り付ける方法などが考えられるが，これでは，Web ページの作者の意図に反してレイアウトが変更されてしまう．

- 参照元のページとリンク先の要約を分割し，別ページとして表示する
  リンク先の要約作成に時間がかかった場合でも，参照元のページは先に表示されるようにするため，参照元のページとリンク先の要約は別ページとする．

- なるべく簡単な操作方法でリンク先の要約を表示する
  リンク先の要約を表示させるのに複雑な操作が必要であれば，ユーザは面倒に感じて，リンク先に直接ジャンプしてしまうかもしれない．

上記の条件を満たすユーザインタフェースとして，本研究では以下の 2 つを考えた．

- ツールチップ表示
  ユーザが Web ページ閲覧中，参照元のページのリンクの上にマウスポインタを置くと，図.3.1 に示すように，リンク先の要約をツールチップとして表示する．ユーザがマウスポインタをリンクから離すと，ツールチップは消える．

- サブウィンドウ表示
  ユーザが Web ページ閲覧中，参照元のページのリンクの上にマウスポインタを置くと，図.3.2 に示すように，リンク先の要約をサブウィンドウに表示する．ユーザがマウスポインタをリンクから離しても，サブウィンドウは残ったままである．

上記 2 つとも，JavaScript で実現できる．また，ほぼ同じ手法で実装が可能であり，参照元のページに貼り付ける JavaScript のコード（関数）を置き換えるだけで表示方法を変更できる．本システムは，ユーザが要求した Web ページを解析し，リンク先が HTML テキストやプレインテキストのアンカーに JavaScript を追加し，上記 2 つのような表示が出来るように Web ページを改変する．

3.3 要約生成

本研究では，要約生成技術をWWW情報探索に用いることを考える．要約の対象はWebページなので，それに特化した要約手法を用いることも考えられる．Webページに特化した自動要約としては，HTMLのタグを利用した要約が考えられる．HTMLのタグは本来，ドキュメントの構造（タイトル，見出し，表など）を記述する言語であるため，要約の作成に利用できるはずである．しかし，実際にインターネット上にあるWebページの多くは，HTMLのタグをレイアウトに利用しており，必ずしもドキュメントの構造をうまく記述していない．従って，HTMLのタグを利用した要約は考えにくい．そこで，本研究では，Webページに特化した自動要約を考えるのではなく，既存のテキストを対象とした要約技術を利用する．

既存のテキスト自動要約技術でまず考えられるのは，テキスト中での位置情報を利用する方法である．しかし，特にポータルサイトなどでは，広告や決まり文句（例えば，ニュースサイトでは，地方サイトへのリンクなど）が，サイトによって，テキスト中の様々な位置に出現し，位置情報を手がかりとした有効な要約手法も見出せそうにない．したがって，テキストの位置情報を手がかりに要約を作成するのはさけるべきである．

また，本研究の目的は，Webページ閲覧中，リンク先が重要かどうかをユーザが判断する手助けをすることである．要約を作成するのに時間がかかっては，リンク先を直接見た方が早くなって，要約を作成すること自体が無駄になる．そのため，要約作成アルゴリズムはあまり複雑でないほうがよい．言い換えなどの高度な処理は行わず，処理の軽い重要文抽出による要約を行う．

3.4 要約対象の先読み

本研究では，WWW情報探索において，ユーザがどのリンク先をたどればよいかの手がかりとして，リンク先の要約を提示する．しかし，要約を表示するのに時間がかかってしまっては，ユーザを支援する効果も小さい．リンク先の要約は，ユーザから要求が起こる前にあらかじめ作成しておくことが望ましい．そこで，本システムでは，ユーザが要求したWebページを解析し，要約対象を決定した後，要約を先読みして作成する．

また，Webページには通常多数のアンカーが付けられているが，先読みはネットワークに対して多大なトラフィックを発生させるため，Webページ中のすべてのアンカーについて同時に先読みを行うことは効率が悪い．従って，同時に先読みする数を制限する必要がある．そこで，先読みする順序を決定し，その順序に従って逐次的に先読みを行う．本研究では，先読みの順序を決定する手法としては，ユーザの興味（ユーザプロファイル）とアンカーテキストから決定する．ユーザにとって興味のある単語がアンカーテキストに多く含まれていれば，ユーザは，そのアンカーを他と比較して見たいであろうと考える．システムは，ユーザにとって興味があると思われる単語をユーザプロファイルとして保存しておき，アンカーテキストの単語とマッチングさせて先読みの順序を決定する．

ユーザプロファイルをユーザに登録してもらう方法は，ユーザにとって煩わしいと思われる．従って，ユーザプロファイルは，システムが自動で登録することとする．

3.5 先読み代理サーバ

本研究では，プラットフォームに依存しない実装方法として，代理サーバ（プロキシサーバ）を採用する．プロキシサーバは，ネットワーク間でデータを中継するものであり，ブラウザからの要求を代行する役割を果たす．また，一度ユーザに要求された情報は「キャッシュ」に蓄えておくことが出来る．先読み代理サーバ [知念 96] は，プロキシサーバに先読み機能を追加したものであり，ユーザから要求のあった Web ページを解析し，リンク先を先読みして「キャッシュ」に蓄えることが出来る．本研究では，この先読み代理サーバに，リンク先の要約を生成，保存する機能を追加する．

第4章　プロトタイプシステムの実装

本章では，前章で述べた設計方針に基づくプロトタイプシステムの実装について説明する．4.1 節では，システムの構成について述べる．4.2 節では，処理の大まかな流れを説明する．4.3 節以降では，各モジュールの詳細について述べる．

4.1　システムの構成

図.4.1 にシステム構成の概略図を示す．システムは，以下のモジュールで構成される．

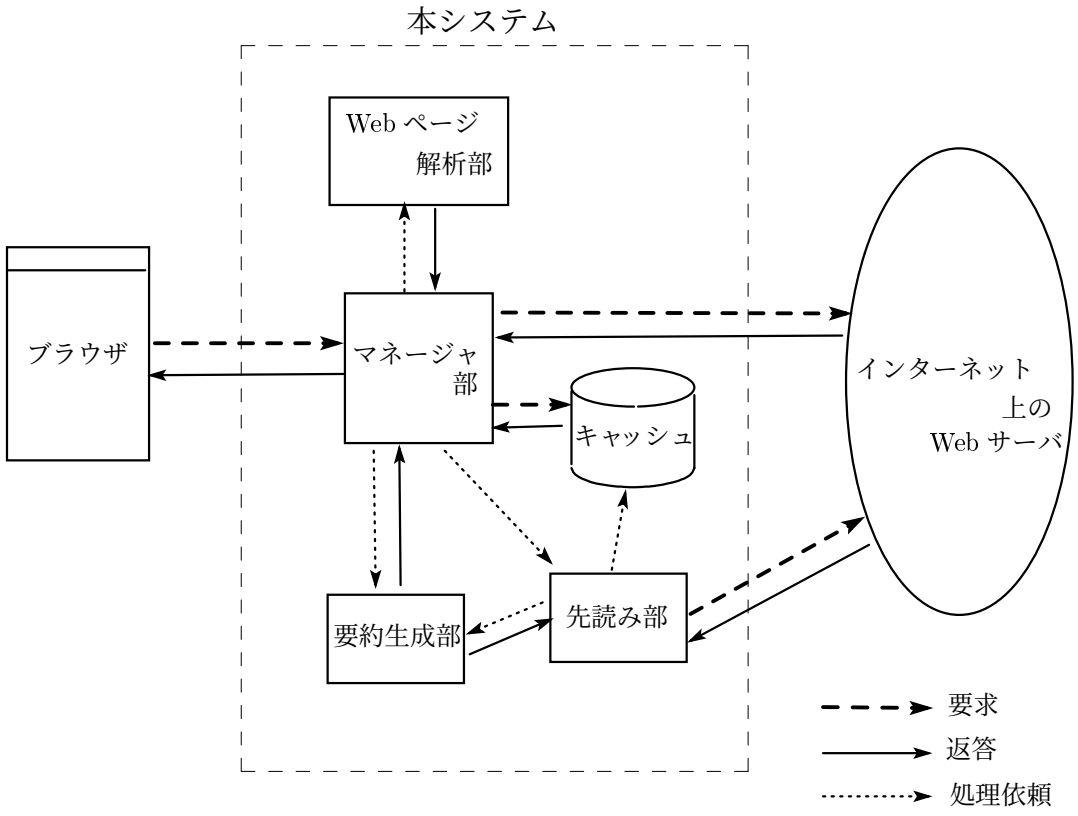

図 4.1: システム構成の概略図

- Web ページ解析部
  ユーザから要求のあった Web ページを解析し，どのリンク先の要約を生成するのかを決定する．また，その要約を提示出来るように Web ページを改変する．要約の生成に時間がかかる可能性があるため，参照元のページとリンク先の要約は別ページとして扱う．そして，先読み要約対象の URL リストをマネージャ部に渡す．

- 先読み部
  マネージャ部から，先読み要約対象の URL リストを受け取り，これらの URL のリソースをインターネット上から入手する．また，そのページの要約を生成するように要約生成部に依頼する．先読みはネットワークに対して多大なトラフィックを発生させるため，同時に先読みする数を制限する必要がある．そこで，先読みする順序を決定し，その順序に従って逐次的に先読みを行う．

- 要約生成部
  要約を生成する．要約は，TF に基づく重要文抽出法によって作成する．

- マネージャ部
  ユーザから，Web ページや要約の要求を受け付ける．通常の Web ページの要求であれば，Web ページ解析部に処理を依頼し，要約の要求であれば，要約生成部に処理を依頼する．また，Web ページ解析部から受け取った先読み要約対象の URL リストを先読み部に渡し，先読み処理を依頼する．

4.2 処理の流れ

システムが行う処理の流れは，大きく分けて「通常の Web ページの表示処理」,「先読み要約作成処理」,「要約の表示処理」の三つに分けられる．以下にそれぞれの処理の流れを説明する．

4.2.1 通常の Web ページの表示処理

ブラウザから通常の Web ページの要求があったとき，そのページのリンク先の要約を表示できるように改変してブラウザに表示させる．処理手順は，以下の通りである．各行の先頭にある番号は，図 4.2 中の番号に対応している．

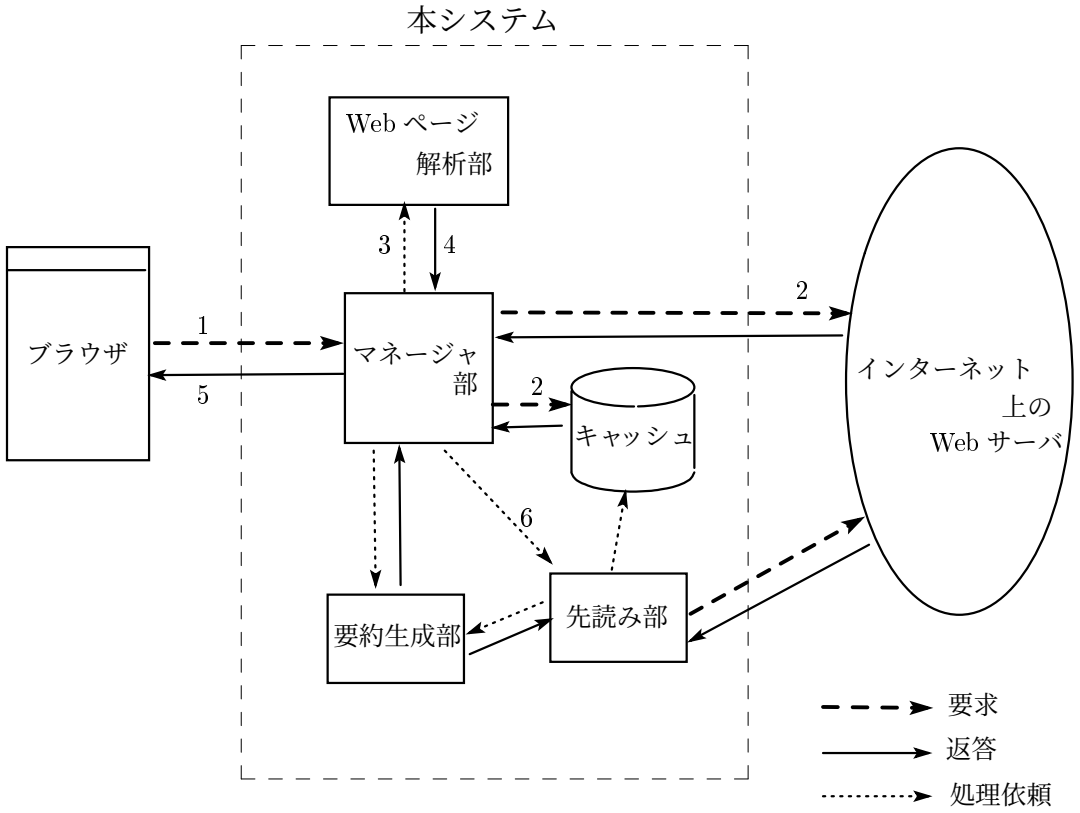

図 4.2: システムの処理の流れ（Web ページ表示）

1. 本システムの「マネージャ部」は,「ブラウザ」からの要求をあらかじめ待ち受けている.「ブラウザ」からの要求を「マネージャ部」が受け取り，4.3 節に示すように,通常の Web ページの要求か，要約の要求かを URL から判断する.

2. 「マネージャ部」は「ブラウザ」から要求のあった URL が通常の Web ページの要求であった場合，その URL のリソースをインターネット上か，または「キャッシュ」から取得する．ただし，現在の実装では，ディスク容量の制約から，Web ページ自体は「キャッシュ」に保存していない.

3. 「マネージャ部」は，取得したリソースを「Web ページ解析部」に渡し，Web ページを解析するように処理を依頼する.

4. 「Web ページ解析部」は Web ページを解析して，リンク先の要約を表示できるように改変し,「マネージャ部」に渡す．また，要約対象のアンカーテキストと URL を抽出し,「マネージャ部」に渡す．要約対象の決定方法は，4.3 節に示す.

5. 「マネージャ部」は「ブラウザ」に処理済みの Web ページを表示させ,「ブラウザ」との接続を切る.

6. 「マネージャ部」は要約対象のアンカーテキストと URL のリストを「先読み部」に渡し，要約を順次作成するように依頼する．

4.2.2 先読み要約処理

先読み要約対象を次々と先読みしていき，要約を作成して「キャッシュ」に格納する．処理手順は，以下の通りである．各行の先頭にある番号は，図 4.3 中の番号に対応している．

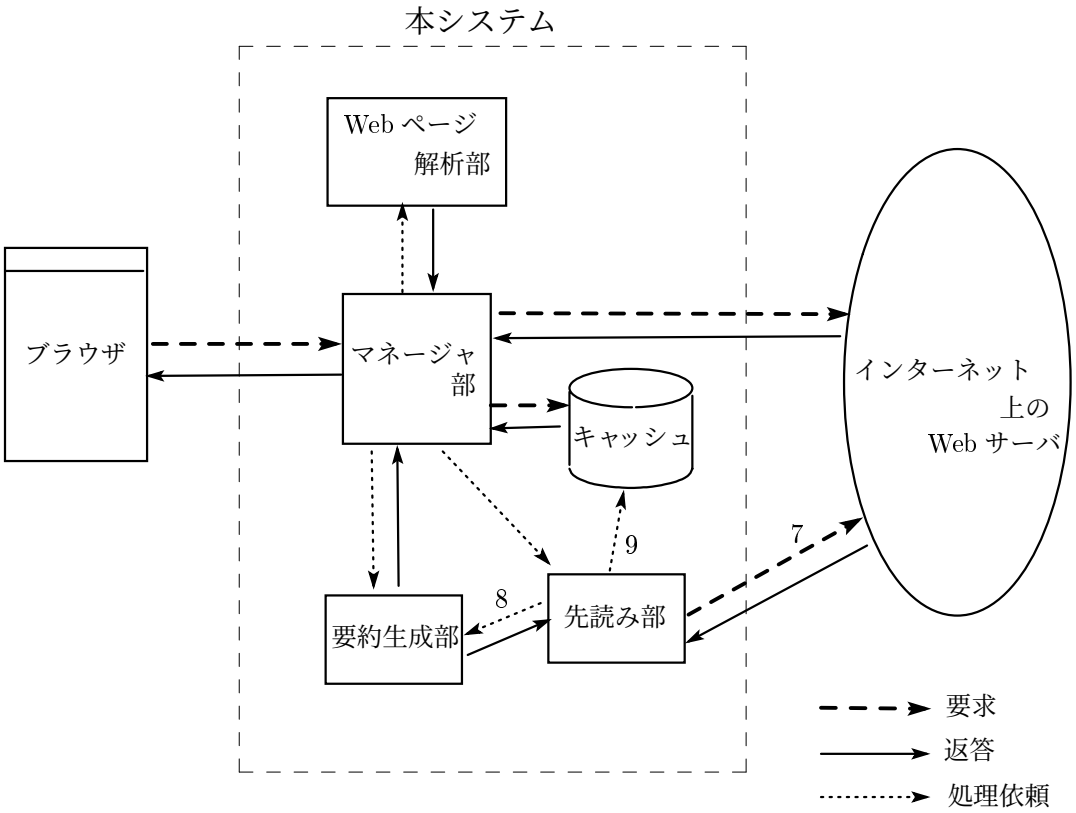

図 4.3: システムの処理の流れ（要約生成）

7. 「先読み部」は，4.4 節に示すように，要約対象のアンカーテキストとユーザプロファイルとを比較して先読み順序を決定し，順次リソースをインターネット上から取得する．

8. 「先読み部」は，リソースを「要約作成部」に渡して，要約を作成するように処理を依頼する.「要約作成部」は要約を作成し,「先読み部」に渡す.

9. 「先読み部」は要約を「キャッシュ」に格納する．

4.2.3 要約表示

「ブラウザ」からの要約の要求に応える．処理手順は，以下の通りである．各行の先頭にある番号は，図 4.4 中の番号に対応している．

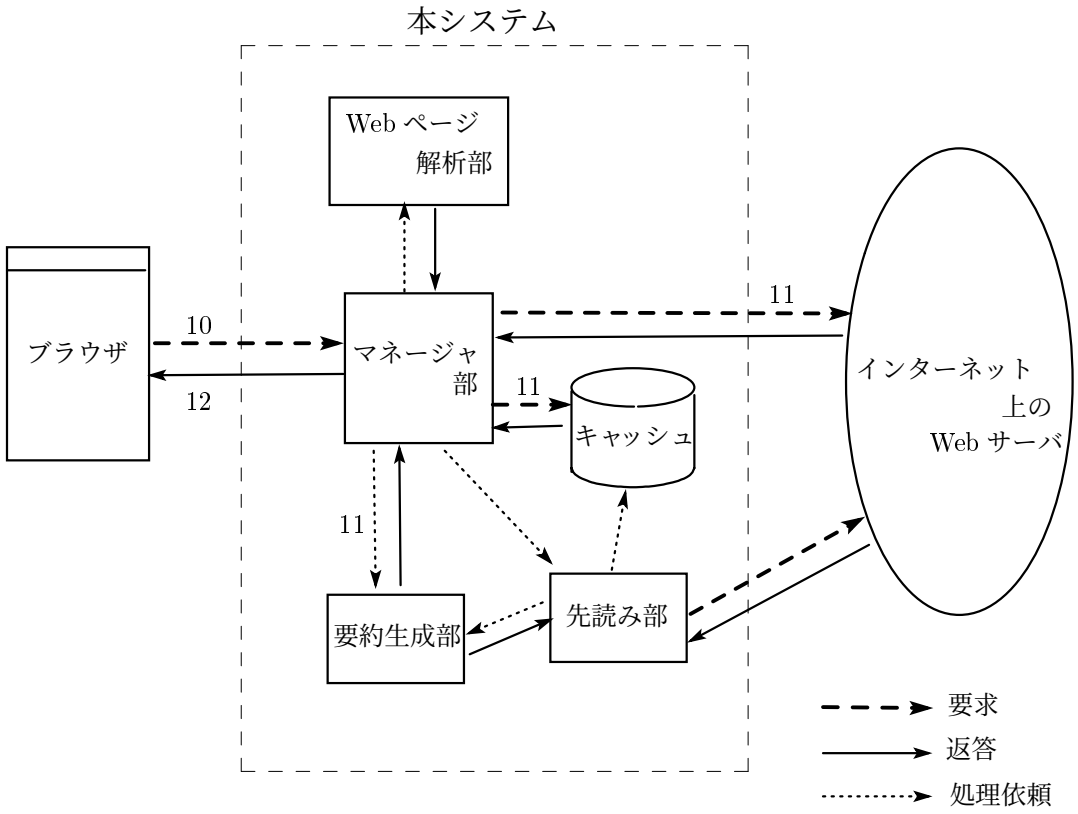

図 4.4: システムの処理の流れ（要約表示）

10. ユーザがリンクの上にマウスポインタを置くと，ツールチップまたはサブウィンドウから本システムの「マネージャ部」に要約が要求される．

11. 「マネージャ部」は「キャッシュ」に要約が格納されているか調べ，格納されていれば，要約を取り出す.「キャッシュ」に要約が格納されていなかった場合，インターネット上からリソースを取得し,「要約生成部」に要約を作成するように依頼する．「要約生成部」は要約を「マネージャ部」に渡す．そして，要約を「キャッシュ」に格納しておく．

12. 「マネージャ部」は要約を「ブラウザ」に渡し，接続を切る．

4.3 Webページ解析部

このモジュールは，まず，ユーザから要求のあったWebページを解析し，そのページに含まれるすべてのリンク先ページの中から要約を生成するページを決定する．その方法は，4.3.1項で述べる．また，要約を表示できるようにWebページを改変する．さらに，Webページの形態素解析を行い，ユーザプロファイルを更新する．

4.3.1 要約対象決定方法

本研究では，リンク先ページがテキストファイルの場合のみ要約対象とし，要約を生成する．リンク先のページがテキストファイルであるかを判定するためには，URLを手がかりとする方法がある．例えば，URLが.htmlや/などで終わるページのみをテキストファイルとみなす方法がある．しかし，URLだけを手がかりにWebページがテキストファイルかどうか判定するだけでは不十分である．例えば，Yahooニュースでは，ニュースのリンク先が

---

http://headlines.yahoo.co.jp/hl?a=20011205-00000101-yom-soci

のようになっており，拡張子だけではテキストファイルかどうか判定できない．したがって，次の手続きにしたがって，リンク先ページがテキストファイルか否かを判定する．

- URLが.jpgや.zipなどで終わるようなものは，明らかにテキストファイルではないとして，要約対象とはしない．
- それ以外は実際にページを取得してテキストファイルか否かを判定する．インターネット上からWebページを取得した際，レスポンスヘッダー内のContent-typeフィールドのMIMEタイプがtext/htmlまたはtext/plainなら，そのページはテキストファイルであるとみなす．

上記の手続きで，テキストファイルではないと判定された場合は，MIMEタイプがtext/htmlまたはtext/plainでなければ，リンク先ページがテキストファイルではないことを伝えるメッセージを要約の代わりに表示する．テキストファイルのときは，要約を生成して表示する．

4.3.2 要約のURLの取り扱い

3.1節で述べたように，要約の生成に時間がかかる可能性があるため，参照元のページとリンク先の要約は別ページとして取り扱う．具体的には，要約対象のWebページのURL

が

---

http://www.jaist.ac.jp/top.html

である場合，要約の URL は

---

http://foo.ac.jp/www.jaist.ac.jp/top.html

とする．ここで，ホスト名の「foo.ac.jp」の部分は，本システムが動作しているサーバとする．システムは，ホスト名がこのサーバ名の URL を受け取った場合，それは要約の要求であるとみなす．それ以外の場合，通常の Web ページの要求とみなす．

4.3.3 Web ページ解析部の処理手順

以下に，Web ページ解析部の処理手順を示す．

1. ユーザが要求した Web ページを解析し，リンクにマウスポインタを置くとツールチップか，サブウィンドウに要約を提示するように，参照元のページを改変する．具体的には，要約対象のアンカーに onmouseover と onmouseout を追加して，要約を表示するための JavaScript が実行されるようにする．すべての要約対象のアンカータグについて，アンカーテキストと URL を記憶し，以下のような処理を行う．例えば，

```
<a href="http://www.jaist.ac.jp/top.html">トップページ</a>
```

のようなタグがあった場合,

```
<a href="http://www.jaist.ac.jp/top.html"
 onMouseOver="_ppsSummary(event,'
                        http://foo.ac.jp/www.jaist.ac.jp/top.html');"
 onMouseOut="_ppsSummaryOut();">トップページ</a>
```

と書き換える.

また，アンカーのURLが相対アドレスだった場合，システムは，参照元のWebページのURL（例えば，http://www.jaist.ac.jp/）をあらかじめ記憶しておき，このアドレスから絶対アドレスを作成してから，以下のように書き換える.

```
<a href="top.html">トップページ</a>
```

のようなタグがあった場合,

```
<a href="top.html"
 onMouseOver="_ppsSummary(event,'
                        http://foo.ac.jp/www.jaist.ac.jp/top.html');"
 onMouseOut="_ppsSummaryOut();">トップページ</a>
```

と書き換える.

この結果，リンクの上にマウスポインタが置かれたときには，JavaScript の_ppsSummary() が起動され，リンクの上からマウスポインタが離れたら，JavaScript の_ppsSummaryOut() が起動されるようになる．

2. Web ページの最後に JavaScript の_ppsSummary() と_ppsSummaryOut() を書き加える．この JavaScript コードによって，要約を表示するためのツールチップやサブウィンドウを表示する．書き換えるコードはツールチップ表示の場合とサブウィンドウ表示の場合とで二種類ある．ツールチップ表示をする JavaScript のコードを付録 A に示す．また，サブウィンドウ表示をする JavaScript のコードを付録 B に示す．そして，その書き換えた Web ページと，要約対象のアンカーテキストと URL のリストをマネージャ部に渡す．

3. ユーザプロファイルを作成する．
ユーザが過去に閲覧した Web ページに含まれている単語は，ユーザの興味を表していると見なし，ユーザプロファイルに自動的に登録する．Web ページのすべての JavaScript コード，HTML タグを取り除き，形態素解析ソフトウェア茶筅 (ChaSen)[松本 00] で処理して，名詞と未定義語を抽出する．抽出した名詞，未定義語をユーザプロファイルに登録する．また，それらの出現頻度も記録する．図 4.5 に，ユーザプロファイルの例を示す．

単語	頻度
緒方	58
貞子	40
さだこ	3
国連	62
朝日新聞社	38
アフガン	87
⋮	⋮

図 4.5: ユーザプロファイルの例

4.4 先読み部

このモジュールでは，マネージャ部から要約対象の URL とアンカーテキストのリストを受け取る．そして，これらの URL の要約を生成するように要約生成部に依頼し，生成された要約を「キャッシュ」に格納する．3.4 節で述べたように，ユーザプロファイルとアンカーテキストを手がかりに，ユーザが興味を持つと思われるアンカーから順に先読みを行う．

先読み順序は，以下のようにして決定する．

1. アンカーテキストを形態素解析して，名詞と未定義語を取り出す．
2. 取り出した単語のユーザプロファイルにおける出現頻度の合計をそのアンカーのスコアとする．
3. すべてのアンカーについてスコアを計算し，スコアの高い順に並べて先読みの順序とする．

4.5 要約生成部

このモジュールでは，TF に基づく重要文抽出法によって要約を作成する．本研究では，Web ページに特化した自動要約を考えるのではなく，テキストを対象とした既存の要約技術を利用する．具体的には，テキスト簡易要約器 Posum[1] [望月 02] を利用し，TF に基づく重要文抽出法によって要約を作成する．要約対象のテキストの HTML タグをすべて取り除き，テキスト簡易要約器 Posum を利用して重要文 300 字の要約を作成する．

Posum は，テキスト中に出現するすべての固有名詞と未定義語を索引語とし，それぞれの索引語 t_i のテキスト中での出現頻度 tf_i を計算する．そして，tf_i の合計値が高い文から順に取り出して 300 字の要約を作成する．

4.6 マネージャ部

このモジュールでは，Web ブラウザや他のモジュールとの間の情報の受け渡しを行う．まず，Web ブラウザから，Web ページや要約の要求を受け付ける．4.3.2 節で述べたように，URL のホスト名がシステムが動いているサーバなら要約の要求とみなし，それ以外は通常の Web ページの要求とみなす．通常の Web ページの要求であれば，Web ページ解析部に処理を依頼する．要約の要求であれば，要約を「キャッシュ」から取り出す．要約がまだ作成されていなければ，要約生成部に処理を依頼する．そして，結果のコンテンツをユーザのブラウザに渡す．また，Web ページ解析部から受け取った先読み要約対象

[1] http://nlp-www.jaist.ac.jp:8000/~motizuki/software/posumcl/

の URL とアンカーテキストのリストを先読み部に渡し，先読み処理を依頼する役割も果たす.

第5章　評価と考察

5.1　目的

本研究で提案するシステムの有効性を確認するために，作成したプロトタイプシステムを用いて評価実験を行った．評価の目標は，以下の３点を確認することであった．

- ２つの要約の表示方法（ツールチップ表示とサブウィンドウ表示）は，それぞれ使いやすいか．
- リンク先の要約を見ることが WWW 情報探索にとって役に立つのか．
- 本システムの使い心地は良いか．

本研究は，WWW 情報探索支援を目的としているため，評価実験の被験者は，すでに WWW をある程度知っていて，本システムを使った WWW 情報探索と通常の WWW 情報探索とを比較し，どちらが優れているかを判断できることが望ましい．そこで，本実験の被験者としては，本大学院の学生，卒業生８人を採用した．

5.2　実験環境

本研究で提案するシステムをプロキシサーバとして作成した．4.5 節の要約生成部と形態素解析ソフトウェア茶筅 (ChaSen) 以外のモジュールをすべてプログラミング言語 JAVA で記述し，これら２つのモジュールは TCP/IP で通信することによって利用可能とした．表 5.1 に本研究で作成したプロトタイプシステムを動作させた PC の詳細を示す．要約生成部は，テキスト簡易要約器 Posum をサーバ化させて，本学のワークステーションの端末で動作させた．また，形態素解析ソフトウェア茶筅 (ChaSen) も同様に本学のワークステーションの端末で動作させた．

表 5.1: プロトタイプシステムを動作させた PC の詳細

CPU	Celeron 500MHz
メモリ	384MB
OS	Windows98
ネットワーク	10Base-T
プログラミング言語	JAVA Version 1.3.1

5.3 実験内容

本研究で提案するシステムの有効性を確認するための WWW 情報探索のタスクとして，以下の２つを被験者に行ってもらい，システムの使いやすさに関するアンケートに答えてもらった．

- ニュースサイトを一通り閲覧して，被験者にとっての重大ニュース３つを選んでもらう．ニュースサイトは,「毎日新聞」[1] と「ZDnet」[2] を利用した．両サイトとも，本システムで問題なくブラウジングできることを確認済みである．

- 検索サイトを利用して調べ物をしてもらう．検索サイトは,「LYCOS Japan」[3] を利用した．本システムで「LYCOS Japan」の検索サービスが問題なく利用できることは確認済みである．

現在のシステムの実装は十分に効率化されていないため，先読みが完了する前にユーザがリンク先のページを表示させることも多い．このため，先読みを行う効果に対する正当な評価が得られない可能性がある．そこで，上記２つの WWW 情報探索のタスクを，それぞれ次のシステムで被験者に行ってもらった．

- システム A
  リンク先の要約が初期状態では「キャッシュ」に格納されておらず，ユーザが Web ページを閲覧してから，Web ページを解析して要約対象の先読みを行うシステム．要約はツールチップで表示する．

- システム B
  リンク先のすべての要約がキャッシュに格納されている理想的な状態を擬似的に実現したシステム．要約はツールチップで表示する．

システム A は現在の実装のままである．一方，システム B は，要約作成が十分に早く，ユーザの要求の前に常に要約を作成しキャッシュに保存できるような理想的なシステムで

[1] http://www.mainichi.co.jp/
[2] http://www.zdnet.co.jp/news/
[3] http://www.lycos.co.jp/

ある．ただし，被験者には，２つのシステムの違いを説明せずに実験を行った．また，実験する順番も被験者の半数はシステム A，B の順で行い，もう半数はシステム B，A の順で行った．

さらに２つの要約の表示方法（ツールチップ表示とサブウィンドウ表示）を比較するために，次のシステム C を用意した．

- システム C
  リンク先のすべての要約がキャッシュに格納されている理想的な状態を擬似的に実現したシステム．要約はサブウィンドウに表示する．

システム A，B での実験の後，システム C を用いてニュースサイトを閲覧してもらい，どちらの要約表示が使いやすいかを質問した．

5.4 実験手順

実験は，以下の手順で行われた．

I. システム A(被験者の半数はシステム B) を利用．

   (1) 「毎日新聞」のニュースサイトを一通り，要約を見るか直接閲覧し，被験者にとっての重大ニュース３つを選んでもらう．

   (2) 「LYCOS Japan」の検索サービスを利用し，以下の２つの問題を解いてもらう．
   - 夏目漱石が小説「こころ」を書いたのは何年か?
   - 狂牛病と似た症状を示すスクレイピーを発症する動物は?

   (3) システムの表示速度に関してアンケートに答えてもらう．

II. システム B(被験者の半数はシステム A) を利用．

   (1) 「ZDnet」のニュースサイトを一通り，要約を見るか直接閲覧し，被験者にとっての重大ニュース３つを選んでもらう．

   (2) 「LYCOS Japan」の検索サービスを利用し，以下の２つの問題を解いてもらう．
   - 安土城の発掘調査は何年から何カ年計画で行われているか?
   - 1999 年 3 月 9 日以降，ブルガリアへビザなしで旅行できる最大の期間は何日か?

   (3) システムの表示速度に関してアンケートに答えてもらう．

III. ツールチップによる要約の表示方法についてのアンケートに答えてもらう．

IV. システム C を利用して，毎日新聞のニュースサイトを閲覧してもらい，アンケートに答えてもらう．

アンケート項目と結果については，次節に示す．

5.5 実験結果

アンケートの項目と結果を以下に示す.

1. このシステムは使いやすいと思いますか.

使いやすいと思う	1 人
改良されれば使いやすいと思う	7 人
どちらともいえない	0 人
使いやすくないと思う	0 人

2. 質問項目 1 について，理由や，改善策等があれば自由にお答えください.

   - 操作が面倒くさくないのがよかった.
   - 表示速度が一番の問題だと思う.
   - 速度，表示するデータに改善の余地があると思う.
   - システム A が少し遅いのでそれが改善されればいいと思います.
   - 具体的にどうと言えないのですが，少し要約が読みにくい印象を受けたのでその辺が直れば良いと思います.
   - もう少し速度が必要であると感じた．日記等の長い文の要約をもう少しまとめた方がよいと思う.

3. 表示方法は別にして，リンク先の要約を見ることは役に立つと思いますか.

役に立つと思う	4 人
要約の質が良ければ役に立つと思う	4 人
あまり役に立たないと思う	0 人
ない方が良い	0 人

4. 質問項目 3 について，理由や，改善策等があれば自由にお答えください.

   - 時として不要な情報が表示されすぎる.
   - より的確にデータの量を減らしてほしい.
   - 一見するだけで，自分の求めるサイトがわかるので有用だ.
   - 次のページに行かずに必要かどうかわかるため役に立つと思う.
   - 自分のほしい情報があるかの確認にはなると思うので役に立つと思う.
   - 検索結果の表示は見にくい時があり，そのようなときの要約は見やすいと感じた.

5. 要約の表示方法（ツールチップ表示）は良いと思いますか.

良いと思う	3 人
表示の大きさ等が改善されれば良いと思う	4 人
どちらともいえない	1 人
良くないと思う	0 人

6. 質問項目 5 について，理由や，改善策等があれば自由にお答えください.

- どちらともいえない．記事の長さが短ければ充分使えた．記事が長いページはリンクを見た方がよかった．表示の大きさ等はよかった.
- 表示する文字を減らしたほうがぱっと見に行ける気がする.
- おおむねよかったと思います．少し読みにくい感じを受けました.
- 要約終了までの残り時間が表示されれば待つかもしれない.（ページへのアクセスが成功して処理するのに時間がかかっているのか，ページ自体へアクセスするのに時間がかかっているのかわからない.）
- システム A の方は表示の速度が遅く感じがち．しかし，システム B は快適であった.

7. 要約の表示方法（サブウィンドウ表示）は良いと思いますか.

良いと思う	4 人
表示の大きさ等が改善されれば良いと思う	3 人
どちらともいえない	1 人
良くないと思う	0 人

8. 質問項目 7 について，理由や，改善策等があれば自由にお答えください.

- ちらちらしなくていい.
- どちらともいえない．早く表示された上で，常に一番手前に表示されたら良いかもしれない.
- 遅いとリンクを見てしまうので表示の大きさ等が改善されれば良いと思う.
- 別のウィンドウで見やすい.
- ツールチップより邪魔にならなくて良い.
- 新しいウィンドウが開かれた方が，枠がはっきりしていて，固定されているので目を動かさず楽に感じた.

9. 要約の表示方法は，ツールチップ表示とサブウィンドウ表示のどちらが良いと思いますか.

ツールチップ表示の方がいいと思う	2 人
どちらともいえない	1 人
サブウィンドウ表示の方がいいと思う	5 人
どちらも良くないと思う	0 人

10. システム A の場合，要約の表示はどう感じましたか.

遅くは感じなかった	0 人
少し遅いが許容範囲だ	1 人
時々遅いときがあり，そのときはリンク先を直接見に行く方がましだ.	6 人
遅いときの方が多く，そのときはリンク先を直接見に行く方がましだ.	1 人
すべて遅て，リンク先を直接見に行く方がましだ.	0 人

11. システム B の場合，要約の表示はどう感じましたか.

遅くは感じなかった	5 人
少し遅いが許容範囲だ	2 人
時々遅いときがあり，そのときはリンク先を直接見に行く方がましだ.	1 人
遅いときの方が多く，そのときはリンク先を直接見に行く方がましだ.	0 人
すべて遅て，リンク先を直接見に行く方がましだ.	0 人

12. このシステムの要約の質は良いと思いますか.

良いと思う	4 人
どちらともいえない	4 人
良くないと思う	0 人

13. 質問項目 12 について，理由等があれば自由にお答えください.

- すべての必要な情報が要約から得られるとは限らなかったのでどちらともいえない.
- 良いと思う．ただし，自分が知りたいと思う具体的な場合は時間短縮できてよい.
- 必要とされるところがあり，良いと思います.
- おおむね良かったのですが，たまにちがう部分まで（主にニュース記事で）表示されている場合があったので，どちらともいえない.

- リンクをクリックしなくても先にあるものが想像できるのは便利だと思った．文章としてはちょっと変なものであっても，これはこれでいいかと思う．

14. 最後に，感想等があれば自由にお答えください．

- ツールチップ表示の方が直感的に使えるので使いやすい気もするが，目がちらつくのでサブウィンドウ表示も捨てがたいところです．
- 全体的に使いやすかった．マウス等で ON/OFF できれば実用的だと思いました．しかし，全体表示が重くなりそうだ．
- サブウィンドウ表示はかなり使いやすかったです．
- ページを開く前に要約を見られるのは非常に判断材料の一つになりうると実感しました．表示までの時間や見やすさをさらに追求すれば，非常に実用性があると思います．
- このような要約は，必要ではあると思う．後は，実際に使うときにユーザがその場で必要としているか，いないかに対応することが必要ではないかと思う．
- 表示がもっと早くなれば，使いやすくなると思います．

5.6 考察

質問項目 1 では，システム全体の使いやすさに関して,「改良すれば使いやすい」という回答が多かった．したがって，プロトタイプシステムには改良すべき点が多いと考えられる．

質問項目 3 では，リンク先の要約を見ることは,「質が良ければ」という回答も含めて，全員から役に立つという意見が得られた．このことから，要約の表示は WWW 情報探索支援に有効であると言える．

ツールチップ表示は先行研究でも多く使われているため，本研究では，要約の表示方法としてツールチップを用いることがサブウィンドウ表示よりも有効であると考えた．しかし，質問項目 5〜9 によれば，ツールチップ表示は「目がちらつく」といった意見もあり，どちらかというとサブウィンドウ表示の方が良いという結果が得られた．しかし，質問項目 5 の回答を見ると，ツールチップ表示に対する好意的な意見もあることがわかる．したがって，要約の教示方法は，ユーザに選ばせるのが良いだろう．

質問項目 10 と 11 の回答を見ると，2 つのシステムの表示速度には明確な差が見られる．このことは，本システムが効率化という面で十分に洗練されていないことを示唆している．先読みの効果を定量的に評価するために，システム A について，以下のような項目をログとして記録した．

- 被験者が Web ページを閲覧した時のタイムスタンプと URL

- 被験者が要約を要求したときのタイムスタンプと URL およびキャッシュヒットしたかどうか
- システムが先読みして要約を作成したタイムスタンプと URL

これらのログの例を図 5.1 に示す．このログの見方を以下に示す．

- 1 列目は，被験者およびシステムが起こした動作を表す．
  - 「browsing」は，ユーザが通常の Web ページの要求を行ったことを示す．
  - 「hit」は，ユーザが要約の要求を行い，その要約がすでにキャッシュの中に格納されていたことを示す．
  - 「nocache」は，ユーザが要約の要求を行い，その要約がまだキャッシュの中に格納されていなかったことを示す．
  - 「prefetch」は，システムが要約を先読みして作成したことを示す．
- 2 列目は，ページ番号を表す．これは，ユーザが Web ページを閲覧する度に，つまり， 1 列目が「browsing」である行が追加される度に，ユニークな番号を Web ページに与える．また， 1 列目がその他の場合は，その Web ページの URL が付けられている参照元の Web ページの番号を表す．
- 3 〜 6 列目は，タイムスタンプを表す．
- 1 列目が「prefetch」の場合， 7 〜 8 列目は先読みの順位と，要約を作成した Web ページの URL を表す．その他の場合， 7 列目は，Web ページの URL を表す．

これらのログから，システム A について，リンク先の要約を表示する前に先読みが完了していた割合を各被験者毎に計算した結果を，表 5.2 に示す． 8 人の被験者の平均は 27.6 %であり，最低で 3.6 %，最高で 51.0 %であった．したがって，この割合を 100 %に近づけるようにシステムを高速化する必要がある．しかし，システム A についても，質問項目 10 に対して「時々遅いときがある」と答えた人が 6 人と最も多く，先読みの効果はある程度認められる．

```
browsing  0    Thu Jan 31 07:40:44    http://www.mainichi.co.jp/
browsing  1    Thu Jan 31 07:40:46    http://www.mainichi.co.jp/ad/ad.html
prefetch  0    Thu Jan 31 07:40:46    1    http://www.mainichi.co.jp/universalon/
prefetch  1    Thu Jan 31 07:40:47    1    http://www.mainichi.co.jp/event.ng/Type=click&ProfileID=176&
prefetch  1    Thu Jan 31 07:40:48    2    http://www.mainichi.co.jp/event.ng/Type=click&ProfileID=178&
browsing  2    Thu Jan 31 07:40:50    http://www.mainichi.co.jp/main.html
prefetch  2    Thu Jan 31 07:40:53    1    http://www.mainichi.co.jp/eye/gong/
prefetch  0    Thu Jan 31 07:40:54    2    http://www.mainichi.co.jp/main.html
hit       0    Thu Jan 31 07:40:54    http://www.mainichi.co.jp/news/selection/20020131k0000m030178000c
nocache   0    Thu Jan 31 07:40:55    http://www.mainichi.co.jp/news/journal/photojournal/index.html
prefetch  2    Thu Jan 31 07:40:55    2    http://www.mainichi.co.jp/news/selection/20020131k0000m03017
prefetch  2    Thu Jan 31 07:40:57    3    http://www.mainichi.co.jp/news/flash/seiji/20020131k0000m010
nocache   0    Thu Jan 31 07:40:58    http://www.mainichi.co.jp/life/hobby/ousyou02/2
prefetch  2    Thu Jan 31 07:40:58    4    http://www.mainichi.co.jp/news/flash/keizai/20020131k0000m02
prefetch  2    Thu Jan 31 07:40:58    5    http://www.mainichi.co.jp/life/hobby/igo02/
nocache   0    Thu Jan 31 07:40:58    http://www.mainichi.co.jp/life/fashion/
prefetch  2    Thu Jan 31 07:41:00    6    http://www.mainichi.co.jp/news/selection/20020131k0000m01011
prefetch  2    Thu Jan 31 07:41:00    7    http://www.mainichi.co.jp/eye/feature/disease/
prefetch  2    Thu Jan 31 07:41:02    8    http://www.mainichi.co.jp/life/hobby/senryuu/
prefetch  2    Thu Jan 31 07:41:03    9    http://www.mainichi.co.jp/info/job/
prefetch  2    Thu Jan 31 07:41:03    10   http://www.mainichi.co.jp/life/family/syuppan/hemingway/
prefetch  2    Thu Jan 31 07:41:05    11   http://www.mainichi.co.jp/news/flash/sports/20020131k0000m05
nocache   0    Thu Jan 31 07:41:05    http://www.mainichi.co.jp/eye/opinion/
nocache   0    Thu Jan 31 07:41:08    http://www.mainichi.co.jp/eye/feature/article/kimitsu/
hit       0    Thu Jan 31 07:41:09    http://www.mainichi.co.jp/eye/feature/disease/
  :
```

図 5.1: システム A のログの例

表 5.2: 先読みのヒット率

被験者	キャッシュヒット数	全要約要求数	ヒット率 (%)
被験者 1	123	241	51.0
被験者 2	32	146	21.9
被験者 3	108	254	42.5
被験者 4	27	146	18.5
被験者 5	11	77	14.3
被験者 6	33	105	31.4
被験者 7	29	78	37.2
被験者 8	2	56	3.6
平均			27.6

第6章　おわりに

6.1　まとめ

本研究では，WWW 情報探索支援のために，リンク先のページを先読みして要約を作成し，ユーザに提示するシステムを作成した．そのシステムの設計方針は，参照元のページのレイアウトは極力変更せずに，なるべく簡単な操作方法で，しかも先読みすることによって迅速にリンク先の要約をユーザに提示することである．そして，この設計方針に基づき，プロトタイプシステムの実装を行い，評価実験を行った．

本研究の成果を以下にまとめる．

- 以下の特徴を持つシステムを設計し，プロトタイプシステムの実装を行った．
  - ユーザの WWW 情報探索支援として，リンク先の要約を提示することができる．
  - リンク先の要約は，参照元のページとは別のページとして扱うので，要約の作成に時間がかかっても参照元のページは先に表示される．
  - リンク先の要約を先読みして作成し，キャッシュに蓄えておくことで，ユーザの要求に迅速に答えることができる．
  - 先読みの順序をアンカーテキストとユーザプロファイルから決定することによって，ユーザにとってより興味があると思われるリンク先から先読みをする．
  - プロキシサーバとして実装することによって，ユーザのプラットフォームに依存しない実装が可能である．また，要約のキャッシュを複数のユーザで共有することもできる．
- 評価実験により，リンク先の要約を見ることが WWW 情報探索支援に有効であることが示された．
- 評価実験により，本研究で作成したプロトタイプシステムでも，先読みの効果は認められた．

一方，評価実験により，以下の問題点が明らかになった．

- 本研究で作成したプロトタイプシステムでは，表示速度が遅いといった問題点がある．
- リンク先の要約を表示するのに時間がかかってしまう場合，ユーザは，何が原因で遅いのかがわからない．

6.2 今後の課題

評価実験によって明らかになった問題点に対処し，以下の改良を進めることで，本システムによる情報探索の支援がよりいっそう有効なものになると考えられる．

- システムの効率化
  本研究で作成したプロトタイプシステムは，JAVA 言語で作成されており，C++などのコンパイラ言語で作成した場合よりも処理速度が遅いと考えられるまた，要約を作成したり形態素解析をしたりする場合に，それぞれのモジュールと TCP/IP で通信しており，これらのモジュールをシステム本体に組み込めれば処理速度がより速くなると考えられる．

- 先読み順序決定戦略の工夫
  本研究で作成したプロトタイプシステムでは，先読みのヒット率が平均 27.6 %と低い．先読み順序決定戦略を改良して先読みのヒット率を上げることが必要と考えられる．

- 要約作成手法の工夫
  現時点では，要約の作成に要する時間も無視できない．要約の質を向上させることも重要だが，高速に要約を生成する手法を開発することも重要な課題である．

謝辞

本研究を進めるにあたり，研究の機会を与えられ，御指導頂きました東京工業大学の奥村学助教授に心から感謝いたします．また，数多くの御指導を頂きました白井清昭助教授に深く感謝いたします．さらに，御指導，御討論を頂きました島津明教授に熱く御礼申し上げます．

本研究全般にわたって熱心に御指導頂き，多くの助言を頂きました望月源助手に深く感謝いたします．また，本研究を行うにあたり貴重な御意見，御討論をしていただき，評価実験の被験者として貴重な時間を割いて頂きました自然言語処理学講座の学生，および，卒業生の皆様に心から感謝いたします．

付 録A　ツールチップ表示をするJavaScriptのコード

```
<script language="JavaScript">

<!--

function getMouseXY(e,flag) {
  if (document.all) {
    if (flag) return event.clientX;
    else return event.clientY;
  }
  if (document.layers) {
    if (flag) return e.pageX;
    else return e.pageY;
  }
}

function setStXY(name,x,y) {
  if (document.all) {
    if (document.body.clientWidth < x + 350) {
      x = x - 410;
    }
    if (document.body.clientHeight < y + 350) {
      y = y - 350;
    }
  } else if (document.layers) {
    if (window.innerWidth < x + 350) {
      x = x - 410;
    }
    if (window.innerHeight < y + 350) {
      y = y - 350;
```

```
    }
  }
  if (document.all) {
    document.all(name).style.posLeft=x+20;
    document.all(name).style.posTop=y+20;
  }
  if (document.layers) document.layers[name].moveTo(x+20,y+20);
}

function writeSt(name,summary_url) {
  if (document.all) document.all(name).innerHTML='<iframe src="'+
                        summary_url+'" width=320 height=350></iframe>';
  if (document.layers) {
    document.layers[name].bgColor = "#faf0e6";
    document.layers[name].load(summary_url,350);
  }
}

function setVisiSt(name,flag) {
  if (document.all) {
    if (flag) document.all(name).style.visibility="visible";
    else document.all(name).style.visibility="hidden";
  }
  if (document.layers) {
    if (flag) document.layers[name].visibility="show";
    else document.layers[name].visibility="hide";
  }
}

function _ppsSummary(e,summary_url) {
  if (document.all || document.layers) {
    var x=getMouseXY(e,true);
    var y=getMouseXY(e,false);
    setStXY("_ppsSummary",x,y);
    writeSt("_ppsSummary",summary_url);
    setVisiSt("_ppsSummary",true);
  }
}
```

```
function _ppsSummaryOut() {
  if (document.all || document.layers) setVisiSt("_ppsSummary",false);
}

// -->

</script>

<SPAN id=_ppsSummary style="LEFT: 0px; POSITION: absolute; TOP: 0px">
</SPAN>
```

付 録B　サブウィンドウ表示をするJavaScriptのコード

```
<script language="JavaScript">

<!--

var _popupsummaryfocus = false;
var _popupURL;
function _ppsSummary2() {
  if(_popupsummaryfocus == true){
    var popup = window.open(_popupURL,"summary",'toolbar=0,location=0,
                    directories=0,status=0,menubar=0,scrollbars=1,
                    resizable=0,left=50,top=50,width=300,height=600');
  }
}

function _ppsSummary(e,_popupURL) {
  _popupsummaryfocus = true;
  this._popupURL = _popupURL;
  _ppsSummary2();
}
function _ppsSummaryOut() {
  _popupsummaryfocus = false
}

// -->

</script>
```

参考文献

[Greenberg 97] Linda Tauscher and Saul Greenberg. How People Revisit Web Pages: Empirical Findings and Implications for the Design of History Systems. International Journal of Human-Computer Studies, 47(1):97 - 138, 1997.

[Kandogan 97] Eser Kandogan and Ben Shneiderman. Elastic Windows: A Hierarchical Multi-Window World-Wide Web Browser. In Proceedings of UIST'97, pages 169 - 177, 1997.

[武藤 99] 武藤哲幸，新井克也. 情報探索におけるブラウジング作業を支援する WWW ブラウザの提案. 情報処理学会報告，99-DPS-92,Feb.，1999.

[林 99] 林憲亨，新井克也. ネット情報自動ガイド技術を用いた情報検索支援システムの開発. 情報処理学会報告，99-DPS-92,Feb.，1999.

[Harald 00] Harald W. R. Weinreich and Winfried Lamersdorf. Concepts for Improved Visualization of Web Link Attributes. In Proceedings of the WWW9, 2000.

[Kopetzky 99] Kopetzky, T. and Muhlhauser, M. Visual Preview for Link Traversal on the WWW. In Proc. 8th Intl. WWW Conf., pp.447 - 454,May 1999.

[奥村 99] 奥村 学，難波英嗣,「テキスト自動要約に関する研究動向」, 自然言語処理, Vol.6, No.6, pp.1-26, 1999.

[奥村 00] 奥村学，望月源,「テキストを自動的に要約する技術 -第 1 回- テキスト中の重要な文を抜き出す」, コンピュータサイエンス誌 bit 2 月号, 共立出版， pp.37-42,2000.2.

[Salton 89] Salton, G. Automatic Text Processing. Addison-Wesley,1989.

[Edmundson 69] Edmundson, H. New methods in automatic abstracting, Journal of ACM, Vol. 16, No.2, pp.264-285. 1969.

[Brandow 95] Brandow, R., Mitze, K., Rau, L. Automatic Condensation of Electronic Publications by Sentence Selection Information Processing and Management, Vol. 31, No.5, pp.675-685. 1995.

[Hovy 97] Hovy, E. Lin, C. Automated Text Summarization in SUMMARIST In Proc. of the ACL Workshop on Intelligent Scalable Text Summarization, pp.18-24, 1997.

[仲尾 97] 仲尾由雄. 見出しを利用した新聞・レポートからのダイジェスト情報 の抽出. 情報処理学会自然言語処理研究会報告, pp.121-128. 1997.

[望月 02] 望月源,「テキスト簡易要約器 Posum version1.50.2 マニュアル」, JAIST Technical Memorandum, IS-TM-2002-002.

[知念 96] 知念賢一，山口英. WWW 先読み代理サーバにおける先読み対象決定戦略. 情報処理学会 研究報告，マルチメディア通信と分散処理研究会 96-DPS-77，第 96 巻, July 1996.

[松本 00] 松本裕治，北内啓，山下達雄，平野善隆，松田寛, 高岡一馬, 浅原 正幸 , "日本語形態素解析システム『茶筌』 version 2.2.1 使用説明書" , Dec, 2000 .