学士学位論文

RSS フィード作成のための
ニュース記事タイトルの抽出手法

茨城大学工学部

システム工学科

執筆者：藤村元彦

指導教官：新納浩幸

平成17年3月1日

目 次

図目次

表目次

第1章　はじめに

1.1　研究概要

Web 上には膨大な情報が存在し、それらの情報には誰もが容易にアクセスすることができる。しかしその情報の膨大さから、すべてを把握することできない。また検索エンジンなどを利用しても、所望の情報を見つけ出せないことも多い。

RSS とは RDF Site Summary の略であり、サイトの概要を記述するフォーマットである。RSS で記述された文書は RSS フィードと呼ばれる。RSS フィードはサイトのインデックス的な役割を担っており、RSS フィードを読むだけで、そのニュースサイトを訪れることなしに、そのサイトの新着情報や概要などが把握できる。そのため RSS を利用することで、効率的な情報収集が可能となる。

現在、RSS は blog のコンテンツ配信として広まっているが、本来はニュースサイトのヘッドラインを配信する目的で規格化されたものである。RSS を利用することで、Web 上のニュースサイトを訪れなくとも、そのニュースサイトの RSS フィードを読むだけで、どのような記事が発信されたかを、ほぼリアルタイムで得ることが可能になる。

RSS フィードの発信は利用者には有益であるが、発信者側から見ると、その作成の負荷が大きく、現在、RSS フィードを発信しているニュースサイトは一部のサイトに限られている。このためニュースサイトの記事を独自に解析し、RSS を作成して提供する組織もいくつか存在する。

本論文でもニュースサイトの記事を解析し、自動で RSS フィードを作成することを試みる。これによって多くの人の情報収集が効率化されるはずである。また記事タイトルだけではなく、HTML ファイルから本文の要約を作成し、その要約を含めた RSS フィードを作成する。

この場合、対象とするニュースサイトを固定すれば、そのサイト内の記事は基本的に同じフォーマットで記述されるので、RSS フィードの作成に必要な記事タイ

トルや記事の本文を、記事のページから取り出すことは困難ではない。しかしサイトが変更された場合には、抽出規則を新たに作成する必要があり、手間がかかる。このため、ある程度汎用的に利用できる抽出規則が望まれる。

本論文で提案する手法は、概略、タイトルと本文の候補をニュース記事から取りだし、次に本文と思われる部分を取りだす。そして、その本文で使われている単語の様子からタイトルを決定するアプローチをとる。この手法は比較的頑健であり、様々なニュースサイトで適用可能である。しかも RSS の性格を考えれば、正確にタイトルを抽出する必要はなく、記事本文の内容を想像できる文章であればRSS の目的は達せられるので、本手法が誤って選択したタイトルであっても、記事本文の様子から選ばれた単語を使っている文が選ばれるため、選択が誤ったとしても大きな損失はない。

1.2 本論文の構成

本論文では、まず RSS ついて、どのようなもので、どのように構成されているかを説明し (第 2 章)、その本論文で扱われている RSS に利用されている RDF についての説明 (第 3 章)、XML 名前空間による機能拡張の仕組みの説明を行い (第 4 章)、データの抽出手法 (第 5 章) について説明する。そして、その抽出手法でデータ抽出を行い RSS ファイルを出力するプログラムファイルを使用した実験とその結果 (第 6 章)、考察 (第 7 章)、終わりに (第 8 章) へと続く。

また、付録として実験に使用したプログラムファイルのソースリストを巻末に添付する。

第2章　RSS

2.1　RSSとは

RSSとは、サイトにある情報をより簡単に把握できるようどこにどんな情報があるかをテキスト形式によって記述するフォーマット、形式のことである。RSSは一部のニュースサイトやBlog(日記)で使われており、読み取ることでサイトや記事の見出しの一覧やアドレス、更新時間などを取得することができる。複数のサイトの更新情報をあつめたサイトや、サイトの見出しを集めて、それらを一括して読めるRSSリーダー等に利用されている。

RSSの(正確にはRSSバージョン1.0の)正式名称はRDF Site Summaryであり、RDFを利用してサイトの概略を記述する、というものである。このRDFというのはXMLをベースに企画されており、構文はXML名前空間を用いて定義することができる。RSSはXMLで構成されているわけでもあり、XML対応ブラウザでRSSの中身を見ることも可能である。

2.2　XMLとは

XMLというのは文書やデータの意味や構造を記述するためのマークアップ言語の一つである。マークアップ言語というのは、タグという例えばHTMLのような「<」と「>」の間に特定の文字列が記述されたものを用いて、文書にさまざまな情報を記述することができる言語のことである。またXMLはユーザが自分で定義したタグを自由に使えることから、今回のRSSのように他のマークアップ言語のベースとして利用されることが多い。

2.3 RSSの歴史とバージョン分類

2.3.1 RSS の互換性

単純にRSSといっても、0.9、0.91、1.0、2.0等々、いくつかのバージョンがある。バージョンを見る限りでは、徐々に機能拡張して行ったようにも見えるが、バージョンによっては仕様が異なっており、異なる仕様についてはほとんど互換性がない。また、バージョンによって、RSSの正式名称も異なる。RSSは歴史的な経緯によって複数のバージョンに仕様が分かれてしまい混乱の元となっている。

2.3.2 RSS 0.9

最初のRSSは、バージョン0.9と呼ばれるもので、Netscape社が1999年3月に公開したものである。正式名称はRDF Site Summaryで、これは基本的にサイトの見出しの一覧を配信するためのものである。Netscapeのポータルサイト My.Netscape.Comで、ヘッドラインの提供などに利用されたがしかし、間もなくNetscapeはRSSから手を引き、その使用を米UserLand Softwareに譲渡する。

2.3.3 RSS 0.91

同年7月には、blogツールの開発元として有名な、UserLand Softwareからのリクエストを受け、要約、更新日時、著作権、格付けなどさまざまな情報を提供できるRSS 0.91が、同じくNetscape社から公開された。RSS 0.91では、RDFの仕様が複雑であるとして、RDFは採用せずによりシンプルな仕様となっている。RSS 0.9を拡張し要約などを記述出来るようにしたことから、名称もRich Site Summaryに変更された。

2.3.4 RSS 1.0

Netscape社はRSSから手を引くが、RSSへの要望はむしろ高まり、「基本的な要約提供機能を核とするRSSとして定義し、より高度な機能は、モジュールとして追加する。」という新しい規格が、RDFに基づくものとしてRSS-DEVワーキンググループで検討される。そして2000年12月には、再び正式名称がRDF Site

Summary である RSS 1.0 が提案された。RSS 1.0 は、正式名称のとおり RDF をベースとして、その拡張には XML 名前空間を用いるなど、仕様が RSS 0.91 などに比べより Web Standard に準拠している点や、比較的仕様が厳密に定義されていることが支持されている。

2.3.5 RSS 2.0

UserLand Software では RSS 0.91 との互換性を持たせつつ機能拡張した非 RDF 系の RSS 0.92 を 2000 年にリリースする。この RSS 0.92 をベースに、さらに機能拡張し一部名前空間による仕様拡張などをサポートしたものを 2002 年に UserLand Software が RSS 2.0 として提案した。正式名称は Really Simple Syndication となり、名前空間機能はサポートするものの非 RDF 系の RSS となる。これらの RSS のバージョンは、図 2.1 のように RDF を採用している RDF 系と、RDF を採用していない非 RDF 系の大きく二つに分類することができる。

本論文では RSS は現在日本で最も普及していると思われる RSS 1.0 を RSS として扱っている [1]。

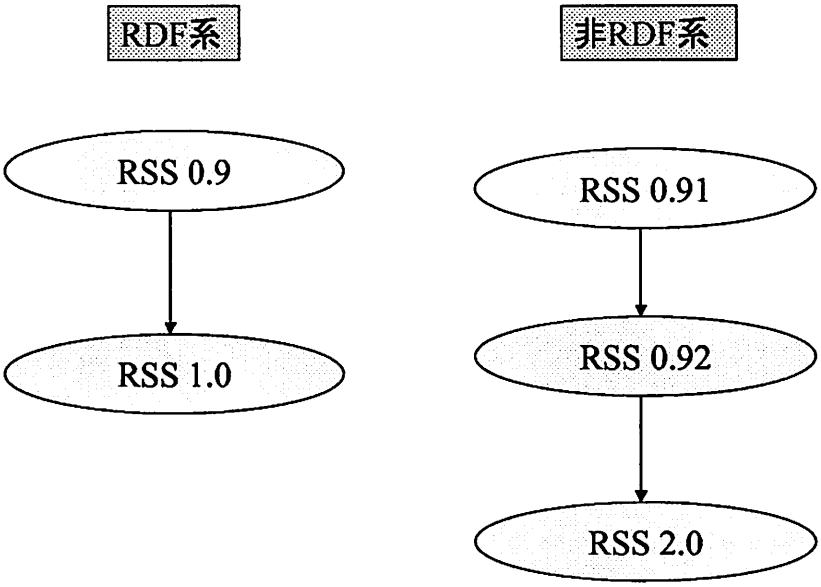

図 2.1: RSS のバージョン分類.

2.4 RSS の基本的な構成要素

RSS は、HTML と同じようにタグを使うことによって情報を記録している。必須な要素としては、まず大きくサイトの情報を示す channel 要素と、サイトにある記事 (リソース) の情報を示す item 要素がある。rdf:about 属性で channel 要素はチャンネルのリンク（通常この RSS 自身の URI)、item 要素には記事のリンク (URI) が記述される。

さらに channel 要素の中には、title、link、description、items という要素があり、title 要素にはサイトのタイトル情報を、link にはサイトのリンクを、description にはサイトの概要を記述する。items 要素は、item 要素に記録されている記事情報の目次に相当し、items 要素の中に、rdf:li 要素の rdf:resource 属性で記事のリンクを示し、item 要素と対応させる。

item 要素の中には、title、link、そしてオプションとして description 要素があり、

それぞれ記事のタイトル、記事のリンク、記事の要約を記述する。なお、description要素については古いバージョンとの互換性から、出来れば500バイト以内であることが好ましい。

RSSの基本的な構成要素による全体の構成を図2.2に示す[2]。

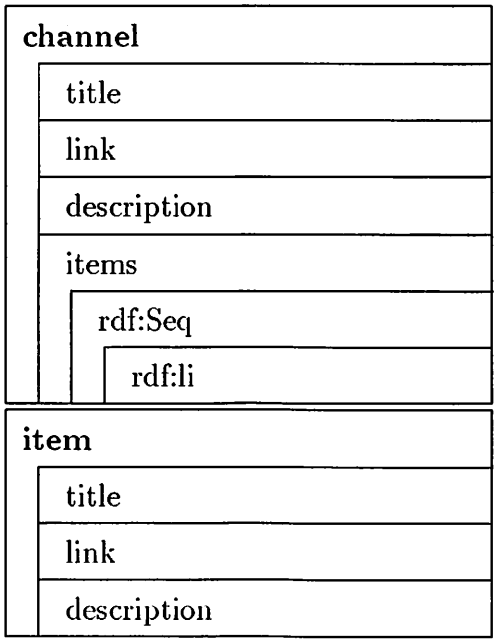

図2.2: RSSの基本的な構成要素による全体の構成.

図2.3は、食事のことを題材にしたhttp://www.syokuji.com/という仮のサイトに、おいしいサラダ、おいしいコロッケという記事があった場合のRSSフィードの記述例である。

```
<?xml version="1.0" encoding="Shift_JIS" ?>
<rdf:RDF
 xmlns="http://purl.org/rss/1.0/"
 xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
 xml:lang="ja">
<channel rdf:about="http://www.syokuji.com/syokuji.rss">
 <title>syokuji.com</title>
 <link>http://www.syokuji.com/</link>
 <description>おいしい食事</description>
 <items>
 <rdf:Seq>
  <rdf:li rdf:resource="http://www.syokuji.com/002.html"/>
  <rdf:li rdf:resource="http://www.syokuji.com/001.html"/>
 </rdf:Seq>
 </items>
</channel>
<item rdf:about="http://www.syokuji.com/002.html">
 <title>おいしいサラダ</title>
 <link>http://www.syokuji.com/002.html</link>
 <description>今日はおいしいポテトサラダを食べました。</description>
</item>
<item rdf:about="http://www.syokuji.com/001.html">
 <title>おいしいコロッケ</title>
 <link>http://www.syokuji.com/001.html</link>
 <description>おいしいコロッケが簡単に作れます。</description>
</item>
</rdf:RDF>
```

図 2.3: RSS の例.

第3章　RDF

3.1　RDFとは

RDFというのはResource Description Framework の略で、メタデータなどのWeb上の情報を表現するための枠組みである。この章では、本論文で扱っているRSS 1.0の仕様の元になっているRDFについての詳しい説明をする。

HTML文書などでは、文による内容以外にも、タグによってつけられた付加情報によってさまざまな情報を得て、ブラウザなどにグラフィカルな表示をするなどが可能である。しかしそれは、HTMLの構文を解読するブラウザ等が、きちんとHTMLを把握していなければならなく、HTMLで定義されていない未知の情報に対しては意味を推論する方法はなく、その未知の情報を処理させることはできない。

RDFは、異なる種類のメタデータ間で相互運用を行うための包括的なフレームワークとして設計され、その未知の定義されていない情報（リソース）について、特定の知識領域などを前提とせずに、リソースを説明するための標準的なメカニズムを提供する。

3.2　RDFのデータモデル

RDFは、ステートメントと呼ばれる文のリストでリソースを、主語（Subject)、述語（Predicate)、目的語（Object)、の3つの要素で説明する。あらゆるステートメントは、主語・述語・目的語のシンプルな構造に従う必要がある。このことから、RDFの主語・目的語間関係のステートメントはトリプルとも呼ばれる。例えば、仮に図3.1のような日本語文があるとする。

Ex. 1

http://www.yunesuko.com/には藤村竜彦という作者がいます。

図 3.1: RDF ステートメントの日本語文例.

図3.1の日本語文をRDFの文として捉えると、表3.1のような構成になる。

表 3.1: RDF ステートメントの構成要素.

主語（Subject）	リソース	http://www.yunesuko.com/
述語（Predicate）	プロパティ	作者
目的語（Object）	プロパティの値	藤村竜彦

表3.1のようにステートメントでは、主語部分でリソースを表し、その属性を述語部分で示し、目的語部分でどのような属性内容なのかを記してリソースを説明することができる。

RDFでは、これらの関係を有向ラベル付きグラフ (directed labeled graph) を用いて表現する。ここで、プロパティはDublin Coreの語彙を使って作者という単語を、dc:creatorという文字列で表現することにする[1]。

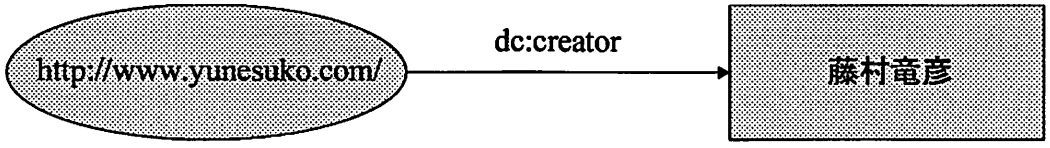

図 3.2: RDF のデータモデル.

図3.2のRDFのグラフでは、主語であるリソースを楕円、目的語のリテラル情報を長方形で表現し、その二つを、述語のプロパティ情報を表した矢印（アーク）で結ぶ。矢印は主語から始まり、目的語に向かった方向となる。

[1]dc:は Dublin Core の要素タイプを示す名前空間接頭辞とする。

3.3 URI参照による名前付け

RDFでは、意味や定義の曖昧さをなくすために、URI参照による名前付けを行う。先の例では、主語はホームページであり、URIを参照できることはすぐわかるが、述語も作者という単語ではなく、dc:creatorというURI[2]となっている。語彙を表すために、人によって使い方や捉える意味の異なってしまう可能性のある単語ではなく、URI参照を用いることで、確実な意味交換を可能にしているわけある。

例で目的語は文字列(リテラル)になっているが、目的語も主語と同様に、URI参照で名前付けされるリソースとすることができる。すなわち、主語、述語、目的語のすべてを、URI参照によって表現できるように設計されている。

3.4 基本データモデルを表現するXML構文

図3.2のようなデータモデルをXMLとして表現するために、RDF/XML Syntax仕様は図3.3のような構文を定めている[3]。

[2]RDFでは、dc:creatorのような修飾名は、名前空間接頭辞とローカル名を直接連結して、http://purl.org/dc/elements/1.1/creatorというひとつのURI参照を構成することになっている。

Ex. 2

```
<?xml version="1.0" encoding="Shift_JIS" ?>
<rdf:RDF
 xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
 xmlns:dc="http://purl.org/dc/elements/1.1/"
 xml:lang="ja">
 <rdf:Description rdf:about="http://www.yunesuko.com/">
  <dc:creator>藤村竜彦</dc:creator>
 </rdf:Description>
</rdf:RDF>
```

図 3.3: XML として表現した RDF の例.

図3.3の構文では、rdf:Description要素が「文」を示し、rdf:about属性が「主語」となるリソースのURIを、要素の内容が「述語」にあたるプロパティ（dc:creator要素）と「目的語」となるリテラル値（藤村竜彦）を記述している。

一行目の部分でXML宣言でXMLの始まりを示しており、xmlsn:rdf="..."などの部分は名前空間接頭辞の宣言をしている。ここで宣言を行うことによって、dc:などの接頭辞を用いた構文を使用することが可能となる（rdf:はRDF Model and Syntaxの名前空間を示す)。

全く同じモデルを、Description要素の内容を使わずに属性として記述することも可能である。こちらは短縮構文（Abbreviated Syntax）と呼ばれる。

Ex. 3

```
<?xml version="1.0" encoding="Shift_JIS" ?>
<rdf:RDF
 xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
 xmlns:dc="http://purl.org/dc/elements/1.1/"
 xml:lang="ja">
 <rdf:Description
 rdf:about="http://www.yunesuko.com/"
 dc:creator="藤村竜彦" />
</rdf:RDF>
```

図 3.4: RDF の短縮構文.

図 3.4 の構文は情報が複雑になると読みにくくなるが、コンパクトに記述でき、レンダリングされてしまう可能性のある要素内容を持たないため、他の言語の中に RDF を埋め込むときには便利である。

3.5 構造化モデル

RDF の文（ステートメント）の目的語にはリソースを指定できる。一方、リソースはそれを主語とした文とすることも可能である。すなわち、ある文の目的語を、別の切り口では主語と見立てて新たな文を作成することができる。この連鎖により、構造化されたモデルで詳細にリソースを記述していくことが可能になる。

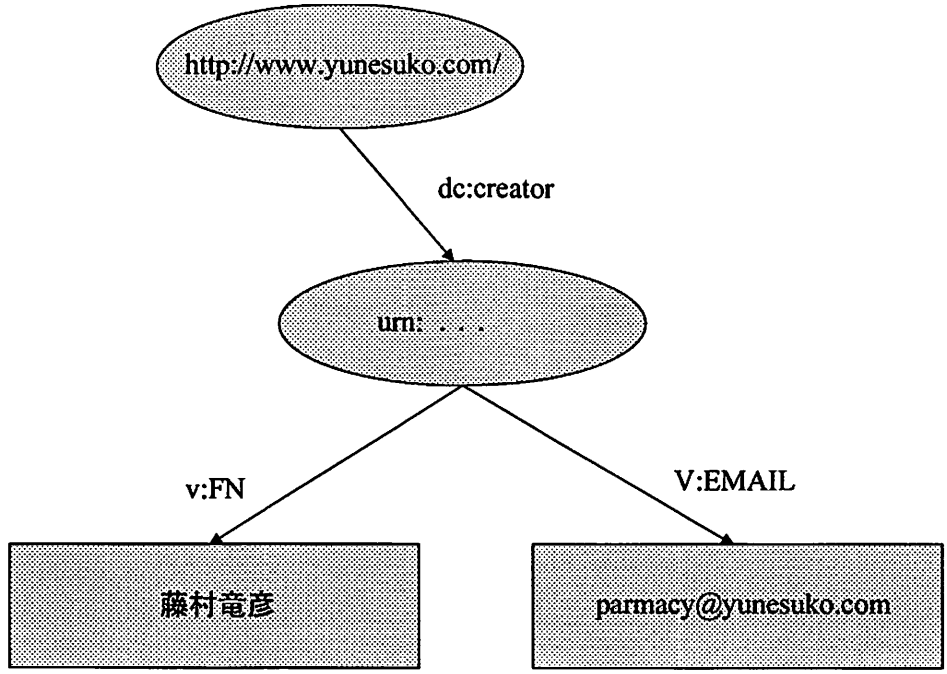

図 3.5: 連鎖による構造化モデル.

図3.5ではvCard[4]の語彙を用いて名前とメールアドレスというプロパティを表現している[3]。このモデルを文章で表現すると、図3.6のようになる。

Ex. 4

http://www.yunesuko.com/の作者はurn: . . . というURIで参照される人物です。彼(彼女)の名前は藤村竜彦、メールアドレスはparmacy@yunesuko.comです。

図 3.6: 連鎖による構造化モデルの日本語文例.

図3.6の文章で示されるように、このモデルは2つの文から構成されるので、XML構文による表現も、それに対応して2つのDescription要素を持つ。

[3]接頭辞 v:は vCard の名前空間を表すものとする

Ex. 5

```
<?xml version="1.0" encoding="Shift_JIS" ?>
<rdf:RDF
 xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
 xmlns:dc="http://purl.org/dc/elements/1.1/"
 xmlns:v="http://www.w3.org/2001/vcard-rdf/3.0#"
 xml:lang="ja">

<rdf:Description rdf:about="http://www.yunesuko.com/">
 <dc:creator rdf:resource="urn: . . . "/>
</rdf:Description>

<rdf:Description rdf:about="urn: . . . ">
 <v:FN>藤村竜彦</v:FN>
 <v:EMAIL>parmacy@yunesuko.com</v:EMAIL>
</rdf:Description>

</rdf:RDF>
```

図 3.7: 連鎖による構造化モデルの RDF 文.

図3.7ではdc:creatorプロパティの値（目的語）をリテラルではなくリソースとするために、rdf:resourceという属性を用いる所に注意する必要がある。このリソースが次のrdf:Description要素（文）ではrdf:about属性によって主語となり、それに関するプロパティが要素内容として表現されている。

このモデルは、rdf:Description 要素を入れ子にして図3.8のように表現することもできる。コンピュータには、どちらの表現でも同じであるが、図3.5の構造化モデルの形からすれば、こちらの方が分かり易いかも知れない。

Ex. 6

```
<?xml version="1.0" encoding="Shift_JIS" ?>
<rdf:RDF
 xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
 xmlns:dc="http://purl.org/dc/elements/1.1/"
 xmlns:v="http://www.w3.org/2001/vcard-rdf/3.0#"
 xml:lang="ja">

<rdf:Description rdf:about="http://www.yunesuko.com/">
 <dc:creator>
  <rdf:Description rdf:about="urn: . . . ">
   <v:FN>藤村竜彦</v:FN>
   <v:EMAIL>parmacy@yunesuko.com</v:EMAIL>
  </rdf:Description>
 </dc:creator>
</rdf:Description>

</rdf:RDF>
```

図 3.8: rdf:Description 要素を入れ子にした場合の例.

さらに、これは短縮構文で図3.9のように記述することもできる。

Ex. 7

```
<?xml version="1.0" encoding="Shift_JIS" ?>
<rdf:RDF
 xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
 xmlns:dc="http://purl.org/dc/elements/1.1/"
 xmlns:v="http://www.w3.org/2001/vcard-rdf/3.0#"
 xml:lang="ja">

<rdf:Description rdf:about="http://www.yunesuko.com/">
 <dc:creator rdf:resource="urn: . . . "
  v:FN="藤村竜彦"
  v:EMAIL="parmacy@yunesuko.com"/>
</rdf:Description>

</rdf:RDF>
```

図 3.9: 入れ子にしてさらに短縮した場合の例.

同じリソースでも、主語として扱う場合は rdf:about 属性に記述し、プロパティ要素の中で、その値（目的語）として示す場合は、rdf:resource 属性の値として URI を記述することに注意しなければならない。

3.6 コンテナモデル

リソースを記述するときに、複数のリソースのまとまりを表現しなければならない場合がある。例えば、著者が複数いたり、言語によって異なる表記があるな

どの場合である。これらを表現するために、RDFではコンテナというモデルを用いる。コンテナには、表3.2のように3つのタイプがある。

表 3.2: コンテナのタイプ.

Bag	順序を特定しないリソース、あるいはリテラルのリスト。
Seq	順番のある（Sequence）リスト。
Alt	同じプロパティ値の代替表現（Alternative）のリスト。異なる言語によるタイトルの翻訳、ミラーサイトなどを示すことができる。RDF処理アプリケーションは、Altリストの中からどれか一つを選んで処理することができる。

たとえば、ある記事を二人の筆者が分担して執筆した場合にはBagを使うことができる。XML構文の場合は、li要素を使って項目を列挙する。

Ex. 8

```
<?xml version="1.0" encoding="Shift_JIS" ?>
<rdf:RDF
 xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
 xmlns:dc="http://purl.org/dc/elements/1.1/"
 xmlns:ex="http://example.org/ex/"
 xml:lang="ja">
 <rdf:Description
  rdf:about="http://www.yunesuko.com/if_str.htm">
 <dc:title>お店紹介</dc:title>
 <ex:authors>
  <rdf:Bag>
   <rdf:li>藤村由里子</rdf:li>
   <rdf:li>藤村竜彦</rdf:li>
  </rdf:Bag>
 </ex:authors>
</rdf:Description>
</rdf:RDF>
```

図 3.10: 順序のない Bag タイプの例.

図 3.10 では述語に ex:autohrs という、「作者のグループ」を表すプロパティを用いている[4]。これに対し、(通常は 1 人の) 作者を表す dc:creator などの目的語がコンテナリソースになっていると、アプリケーションが混乱する可能性があるので、

[4] ex:という接頭辞は、架空の名前空間を示すことにする。

使い方を注意する必要がある。

ダウンロードサイトが複数あるような場合、Altタイプを用いて、アプリケーションがその中からひとつを選択できるような記述が可能である。

Ex. 9

```
<?xml version="1.0" encoding="Shift_JIS" ?>
<rdf:RDF
 xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
 xmlns:dc="http://purl.org/dc/elements/1.1/"
 xmlns:ex="http://example.org/ex/"
 xml:lang="ja">
<rdf:Description
 rdf:about="http://example.org/apps/killerApp">
<ex:downloadSites>
 <rdf:Alt>
  <rdf:li
   rdf:resource="http://dl1.example.org/apps/killerApp"/>
  <rdf:li
   rdf:resource="http://lib.example.com/download/killerApp"/>
 </rdf:Alt>
</ex:downloadSites>
</rdf:Description>
</rdf:RDF>
```

図 3.11: 代替表現 Alt タイプの例.

図 3.11 では ex:downloadSites の値は直接示されていないので、目的語は匿名リソースになり、グラフならば空白の楕円が描かれる。コンテナは、リソースのタイプを示す rdf:type というプロパティによって「rdf:Alt タイプのリソース」という形で表現され、コンテナの目的語としてリストが示される。リストのそれぞれの項目には、rdf:_1、rdf:_2 といったメンバシップ名プロパティが内部的に順番に与えられ、それぞれの値としてリストの内容が表現されるという形になる。ただし、XML 構文を使って記述するときには、これらのプロパティは気にしなくても構わない。

3.7 具体化:文についての文

RDF は文（statement）によってあるリソースについての叙述を行うが、その文そのものについての叙述を行うというケースがある。

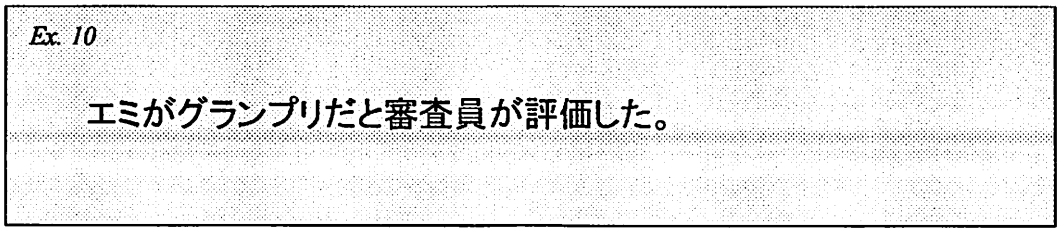

図 3.12: 文をリソースとする場合の日本語文例.

例えば図 3.12 は、「エミがグランプリだ」という文に関する叙述とみることができる。文そのものはリソースではないので、これを主語として RDF で表現するためには、文をリソースとして具体化（reification）する手続きが必要である。

RDF では、文全体を表す rdf:Statement というタイプのリソースをつくり、そのプロパティとして rdf:subject、rdf:predicate、rdf:object の 3 つを与えることで、この文自体を具体化する。

Ex. 11

```
<rdf:Statement>
 <rdf:subject rdf:resource="#Emi"/>
 <rdf:predicate rdf:resource="#deserves"/>
 <rdf:object rdf:resource="#GrandPrix"/>
</rdf:Statement>
```

図 3.13: 新たなタイプ rdf:Statement の例.

図 3.13 のように、元の文の主語、述語、目的語が、それぞれ新しいリソース（rdf:Statement）の 3 プロパティの目的語となる[5]。リソースは文の主語になることができるので、これを使って図 3.12 の文を RDF で表現してみると、図 3.14 のようになる。

[5]#Emi など#で始まる rdf:resource 属性値は、同じ文書内で rdf:ID によって定義されているリソースを参照していることを示す。またここでも ex:という接頭辞は、架空の名前空間を示す。

Ex. 12

```
<?xml version="1.0" encoding="Shift_JIS" ?>
<rdf:RDF
 xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
 xmlns:dc="http://purl.org/dc/elements/1.1/"
 xmlns:ex="http://example.org/ex/"
 xml:lang="ja">
 <rdf:Statement rdf:ID="award2005">
  <rdf:subject rdf:resource="#Emi"/>
  <rdf:predicate rdf:resource="#deserves"/>
  <rdf:object rdf:resource="#GrandPrix"/>
 </rdf:Statement>

 <rdf:Description rdf:about="#award2005">
  <ex:ratedBy rdf:resource="#Jrdge"/>
 </rdf:Description>
</rdf:RDF>
```

図 3.14: 文をリソースとした RDF の例.

図3.14のように、新しいリソース (rdf:Statement) を別の文の主語として rdf:about で参照するためには URI が必要なので、ID 属性によって award2005 という名前を与えている。これは、結局 ex:ratedBy というプロパティが新しいリソースの述語であるので、図 3.15 のように記述しても等価となる。

Ex. 13

```
<?xml version="1.0" encoding="Shift_JIS" ?>
<rdf:RDF
 xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
 xmlns:dc="http://purl.org/dc/elements/1.1/"
 xmlns:ex="http://example.org/ex/"
 xml:lang="ja">
 <rdf:Statement>
 <rdf:subject rdf:resource="#Emi"/>
 <rdf:predicate rdf:resource="#deserves"/>
 <rdf:object rdf:resource="#GrandPrix"/>
 <ex:ratedBy rdf:resource="#Jrdge"/>
 </rdf:Statement>
</rdf:RDF>
```

図 3.15: 全て rdf:Statement の属性にした場合の例.

この場合、文の具体化リソースを URI 参照する必要がないので、ID 属性は加えていない (匿名リソースとなる)。このモデルをグラフで表現すると図 3.16 のようになる。

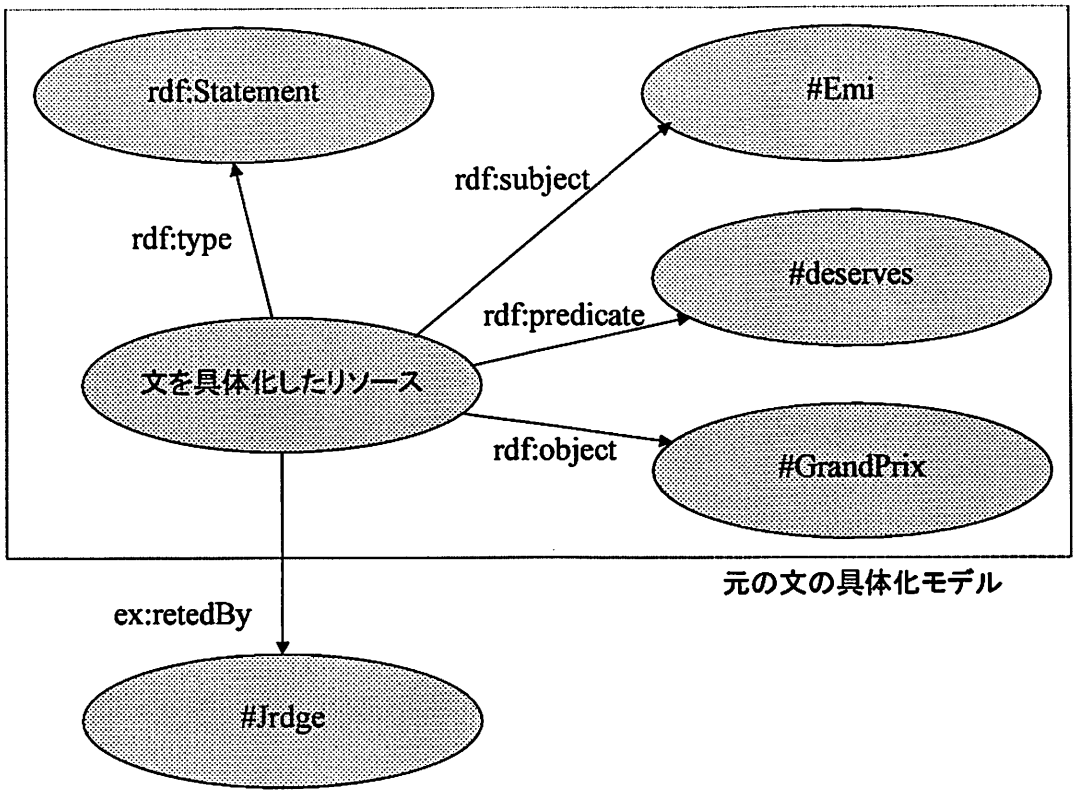

図 3.16: 具体化のモデル.

この具体化のモデルは、1つの文が4つの文に変換されるため、叙述が複雑になると扱いが大変になり、現在も改良の可能性が検討されている。しかし、文を具体化して別の文の主語や述語にするという考え方は、推論やロジックのレベルで重要になってくるので、そのような概念だけは押さえておく必要がある。

第4章　XML名前空間

4.1　名前空間の導入

この章では、RDFの機能拡張として使用されるXML名前空間について説明する。

XMLでは独自のマークアップ言語(語彙)を自由に設計できるが、多くの人が利用する語彙は、標準的な語彙を共有する方が効率が良く、相互運用性も高まる。そこで、ひとつのXML文書を作成するのに、さまざまなマークアップ言語から語彙を拝借し、組み合わせて利用できるような仕組みが必要になる。たとえば論文を公開するのに、全体の構造や本文はXHTMLで記述しつつ、数式部分にはMathMLの語彙を、グラフにはSVGの語彙を利用するという方法である。

このように複数の言語をモジュールとして組み合わせるためには、それぞれの語彙をきちんと区別できなければならない。しばしば取り上げられる例としては、titleという名前を持つ要素が、ある時は書物のタイトルを表したり、別の時は作者の肩書きを表したりすることがあり得るからである。

この問題を解決するためには、それぞれの語彙がユニバーサルな名前、つまり世界で一意に定まる名前を持つための方法が必要である。これを実現するのが、1999年にW3Cから勧告されたXML名前空間[5]である。XML名前空間は、語彙(要素タイプ名、属性名)をURIと組み合わせる(修飾する)ことで、複数の語彙を混在させるメカニズムである。

名前空間は、タグセットを組み合わせて利用できるようにするだけでなく、ウェブ上で共有交換される情報の意味を正確に示し、コンピュータに理解可能にするためにも非常に重要な役割を果たす。

4.2　名前空間URIによる修飾と接頭辞

名前空間では、複数のタグセットを区別するために、語彙をURIで修飾する。ここで、James Clark氏による説明用の表記法[6]を借用し、修飾部をURIと表記し

てその仕組みを説明する[1]。

たとえば、(1) 書物の情報を表現するタグセットを http://eample.com/ns/book/ というURIで、(2)作者の情報を表現するタグセットを http://example.com/ns/profile/ というURIで識別することにすると、書物のタイトル、作者の肩書きを示す要素タイプ名は、それぞれ図4.1のような形で表記できる。

Ex. 1

{http://example.com/ns/book/}title (1)
{http://example.com/ns/profile/}title (2)

図 4.1: URIによる識別例.

URIは世界で重複することのない識別子であるから、語彙ごとにURIを割り当てれば、この方法で必ずタグセットを区別できる。

しかし、全てのタグにURIを記述していては非常に煩雑になるので、XML名前空間では、これらのURIに別名を割り当て、それを接頭辞として用いるよう定めている。URIと接頭辞はxmlnsという特別な属性を使って結びつける。

[1]ここで借用したURIによる修飾は、説明用の便宜的なもので、文法的に正確な表記ではない。

Ex. 2

```
<body xmlns:book="http://example.com/ns/book/"
      xmlns:prof="http://example.com/ns/profile/">
...
<book:title>ユニバーサルHTML/XHTML</book:title>   (1)
...
<prof:title>コントラバス奏者</prof:title>          (2)
...
```

図 4.2: 接頭辞を使用した識別例.

図 4.2 のように接頭辞を加えた名前は QName(qualified name=修飾された名前) と呼ばれる。QName では、title のような各語彙に属する名前の部分はローカル部 (local part) あるいはローカル名 (local name) と呼ばれる。

この例の (1)(2) は、book:、prof:という接頭辞が URI にマップされているので、それぞれ URI で修飾した図 4.2 の (1)(2) と同等の識別情報を持ち、同じようにユニバーサルに区別できる名前になっている。

xmlns 属性を使って名前空間を宣言すると、その接頭辞と URI のマッピングは、宣言を行った要素およびその子孫要素にわたって有効になる。

4.3 デフォルト名前空間

xmlns 属性による名前空間宣言で、接頭辞を省略すると、その URI が宣言のある要素以下でデフォルトとなり、接頭辞なしのタグ名はこのデフォルト名前空間に属することになる。例えば、XHTML の場合はルート要素の html タグで図 4.3 のような名前空間の宣言を行う。

Ex. 3

```
<html xmlns="http://www.w3.org/1999/xhtml">
```

図 4.3: 接頭辞を省略した宣言.

この宣言を最初に行うことで、html 要素内に記述するタグは、接頭辞を持たなくても XHTML の名前空間に属することになるわけである。

デフォルト名前空間を含め、名前空間の接頭辞は宣言している要素とその子孫で有効であるが、図 4.4 のように疎損要素で同じ接頭辞 (あるいはデフォルト) の名前空間が宣言されると、その内部では新しい宣言によって結びつきが上書きされる。

Ex. 4

```
<!-- 最初のデフォルト名前空間はXHTML -->
<html xmlns="http://www.w3.org/1999/xhtml">
...
<body>
...
<!-- ここからデフォルト名前空間はMathML -->
<math xmlns="http://www.w3.org/1998/Math/MathML">
 <mrow>
 ...
</math>
<!-- デフォルト名前空間はXHTMLに戻る -->
...
```

図 4.4: 入れ子宣言によるデフォルト名前空間の有効範囲.

4.4 名前空間URIの意味するところ

名前空間において今一つ分かりにくいのが、そのURIにはどんな意味があるのかというところである。

Ex. 5

```
<html xmlns="http://www.w3.org/1999/xhtml">
```

図 4.5: 名前空間 URI の意味.

単純に、図4.5のように記述されていれば、このURIをたどることにより、XHTMLの仕様書等を取り出せると考えたくなる。

From SPEC

> The namespace name, to serve its intended purpose, should have the characteristics of uniqueness and persistence. It is not a goal that it be directly usable for retrieval of a schema (if any exists).

図 4.6: XML 名前空間の仕様.

しかし、XML名前空間の仕様書では、図4.6のようになっており、このURIは仕様やスキーマを取り出すためのものではなく、ユニークで永続的な識別が目的であるとされている[2]。多くの場合、名前空間を示すURIにhttp:スキームが使われているため、何かを取り出せるという印象があるが、これは第一義的にはその名のとおり、IDとしての役割を果たすもの、というわけである[3]。

[2] namespace name とは URI のこと

[3] もちろん、そこにスキーマ自身や、スキーマにリンクする文書があっても構わないし、それはある意味で推奨されている。

第5章　データの抽出

5.1　抽出の対象

自動的に記事のタイトルと要約の情報を RSS として記録するには、記事のタイトルと本文を HTML ファイルから抽出する必要がある。HTML には、明確にタイトルと本文を示す規則はないので、抽出の手法を考える必要がある[1]。

抽出対象とする HTML ファイルは、一般的なニュースサイトの記事にあるような、一つのページに一つの記事 (本文) と一つのタイトルがあるものとしている。これは抽出精度の問題が最も大きくかかわってくるが、RSS の形式が一つの link に一つの title のようになっていることもある。

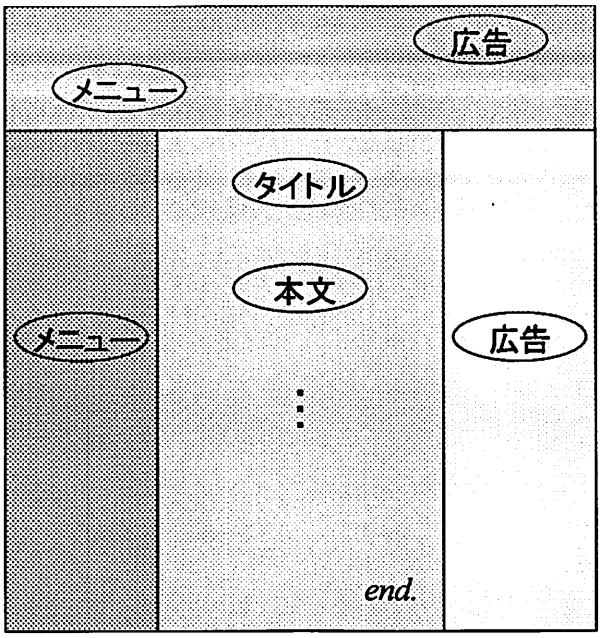

図 5.1: 抽出対象記事の一般的な構成.

[1]特にニュースサイトなどでは、title タグには記事のタイトルではなく、カテゴリ的な内容となっている場合が多い。

5.2 本文の抽出

本論文で行った実験では、まず、本文部分とタイトル部分を文字列として抽出するため、table タグ等の、タイトルや本文の区切り部分に使われている可能性が高いいくつかの HTML タグで区切りを判断し、その区切りごとに文字を抽出して、タイトル・本文の候補となる文字列を抽出している。

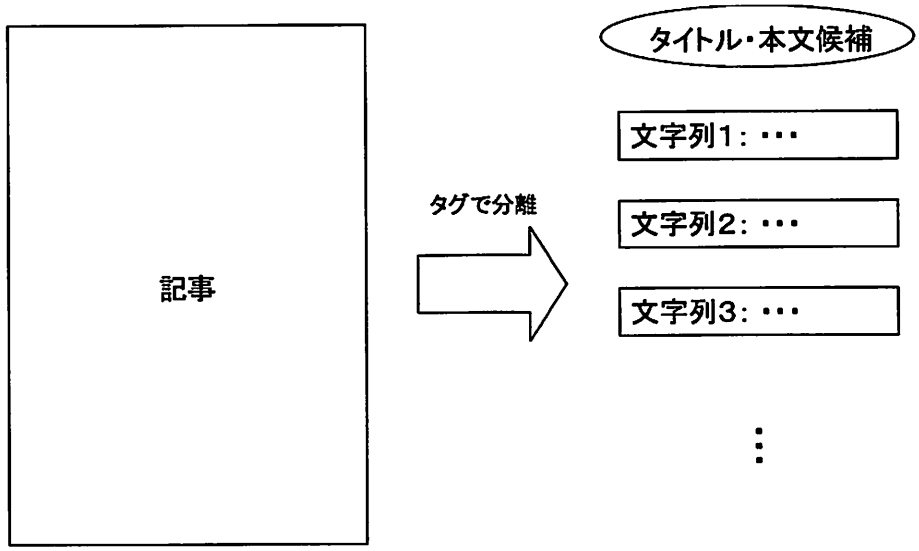

図 5.2: タイトル・本文の候補の抽出.

その抽出された候補の中から、本文と見なされる文字列を抽出するため、その一つの条件として、句点を含むかどうか調べる。本文は文章であるため、単語単位で使われる可能性のあるタグで分離しない限りは、候補の文字列に句点が含まれるはずである。

また、候補の文字列の中に、どれだけの割合でリンクの文字が存在しているのかも調べる。これは特にニュースサイトでは、本文中にリンクの文字はそれほど多くはなく、逆に本文でないものは、他ページへのリンク文字である場合が多いためである。

本文中にある写真の説明などの文章には、リンク文字を含まない場合が多い。したがって、句点やリンク文字の割合だけでは、本文かどうかの区別をすることができない。この写真と文章は、ほとんどの場合テーブルによって配置されている。したがって、この文章を取り除くために、本文部分の中身にテーブルがある場合

には、そのテーブルの中身を参照しないようにしていることで対処している。しかし、この方法では、本文中にあるテーブルの中に、さらに本文である文章があるような構成の場合、その文章は無視されてしまう、といった欠点が生じる。

また、こういった本文以外の文は、大抵文字列長は短く、逆に本文から抽出された候補は、他に比べて長い文字列となる傾向がある。そのため本文として判断するための、最低文字列長を定めている。

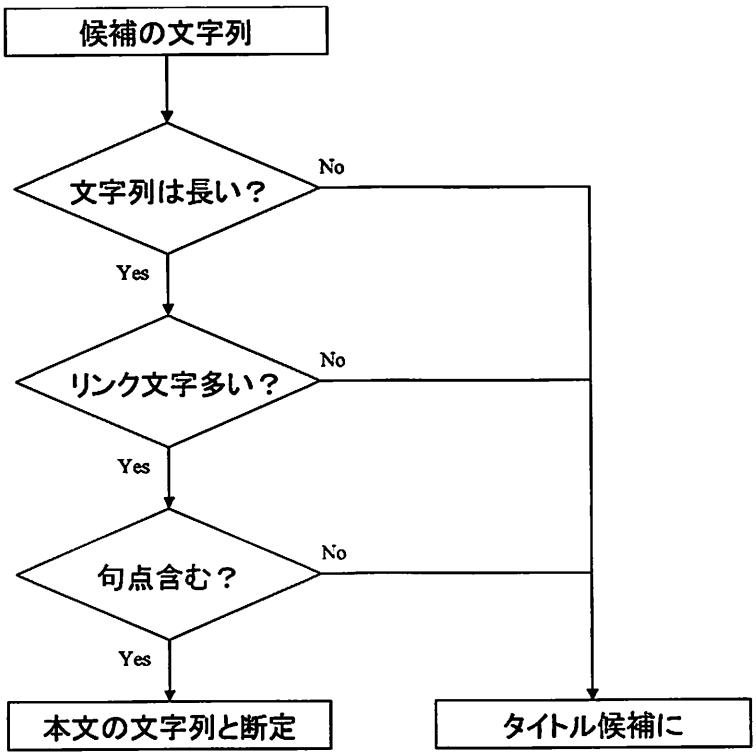

図 5.3: 本文の抽出.

5.3 タイトルの抽出

5.3.1 名詞による類似度判定

本文の抽出を終えた後、そのデータを基にタイトル抽出行う。タイトルの抽出は、まず本文を形態素解析し名詞部分を取り出す。そして取り出した名詞が、タイトル候補の文字の長さに対して、どれだけの割合で存在してるかを計算し、その割合を基に判断するという方法で行っている。タイトルとは、本文の内容が一

目で分かるように書かれた文字列であるため、本文の言葉が使われる可能性の高い文字列であるタイトルに対して、高いヒット率を得る事が出来ると予測できる。

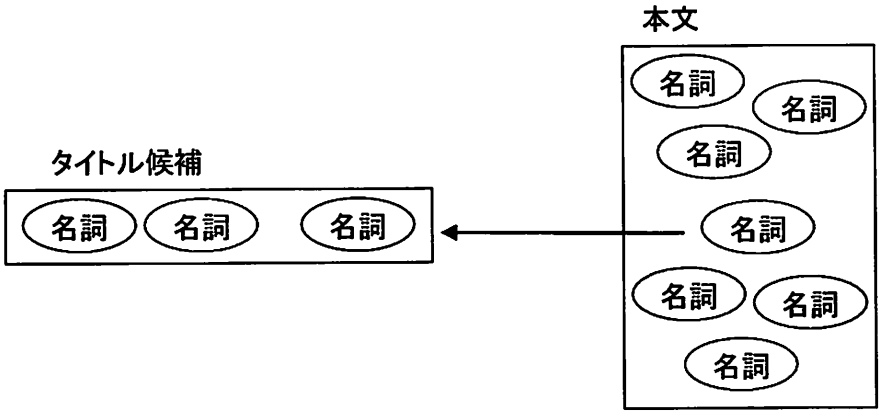

図 5.4: 名詞による類似度判定.

しかし、この方法のみでは、タイトル候補である文字列が短い時、偶然のヒットで高いヒット率となり、タイトルとして誤認されてしまう可能性がある。さらに、タイトル候補が長いほど名詞以外の語句が増えるので、必然的にヒット率が下がってしまう傾向になってしまう。タイトルの長さとは、長すぎず短すぎずのある程度の長さであるのが一般的である。そこで、そのある程度の基準を決め、その長さから離れているもの程、タイトルの候補としての優先度を下げる、というような方法で対処している。

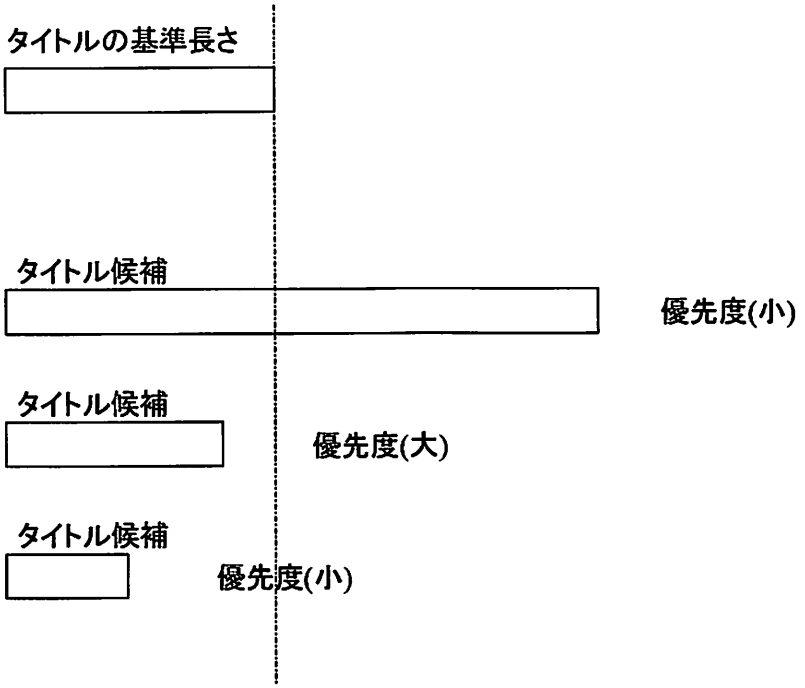

図 5.5: タイトル長による優先度変化.

5.3.2 記事の構成による判断情報

その他にも実験で利用したプログラムでは、精度を上げるために HTML のタグや構成状態から得られる情報を基に、優先度が変化するようになっている。

まず、大きな見出し文字として扱われた文字列はタイトルである可能性がかなり高い。よって見出しタグが使われた候補を優先させることで精度を上げることができる。本論文で行った実験では、h1、h2、h3 のタグが使用されていた候補に対して大きく優先させた。

title タグにある候補に対しても、ある程度優先させている。高いヒット率を得たタイトル候補が得られなかった場合でも、title タグにある候補がタイトルに来やすくなり、まったくおかしな候補がタイトルとなる可能性が低くなる。

このほかに、タイトルは太字で強調されている場合も多いので、b タグが使われていた場合にも、強調されていた文字量の割合に応じて優先度が上がるようにした。

また、タイトルは位置的に本文が始まる少し前にかかれる、ということが多い。

実験で使用したプログラムでは、タイトル候補が本文の文頭から離れている程、優先度が下がるようにした。

本文抽出時に、リンク文字の量を抽出の判断材料に利用したが、タイトル判断にもリンク文字の量を調べている。基本的に、タイトルにはリンク文字がないことが多いが、本文は一部のみ載っており、本文の全文は別ページにある場合などは、タイトルが全文へのリンクになっている場合がある。本文の全文が載っていない記事を対象としなければ、リンク文字の多い候補を除外することで大きく精度が上がることが期待できる。実験プログラムではリンク文字の割合が多いほど多少優先度が下がるようにした。

5.4 問題点と特徴

このように色々と条件をつけていく事で精度を上げているが、基本的には同じ名詞があるかないかで判断している。本文ではあまり利用していない言葉・言いまわしでタイトルが設定されている時には、間違ったタイトル候補がタイトルと判断される事が発生する場合がある。しかし、間違ったタイトル候補が選ばれたとしても、本文の内容に近く、タイトルとしてもおかしくないような言葉が選ばれやすいという特徴がある。

第6章　実験

6.1　プログラムファイルの仕様

実験で使用したプログラムファイルの仕様は、プログラムの引数に、RSS を作成したい記事の URL を指定し、実行することで、RSS ファイルを出力するというものである。出力する RSS ファイル名は-R をつけることで指定することができるが、このとき同じ名前の RSS ファイルが存在すれば、古い RSS ファイルの内容をそのままに、新しい記事の情報を付加できる仕様になっている。

記事の URL だけでは、そのサイトの情報を得ることはできないので、新しい RSS ファイルを出力するときは、channel 要素の items 属性の内容以外は全て空欄となる。channel 要素の内容を埋めるためには、テキストエディタなどで、一度自分で埋める必要がある。すでにある RSS ファイルに channel 要素の内容があれば、その内容は保存され、新しい記事情報だけが記録される。

また、要約情報も RSS に出力するわけであるが、ニュース記事は、読んですぐ何がニュースなのかがわかるように、記事の題材にしている事を始めに書く事が大抵である。よって実験で使用したプログラムファイルでは、本文の先頭 2 文か、もしくは 500 バイトまでを要約情報として出力するような仕様になっている。

例として、図 6.1 のような Web 上にあるニュース記事の URL を、実験プログラムファイルで指定して処理すると、図 6.2 に示す内容の RSS ファイルが出力される。

図 6.1: ニュース記事の例.

```
<?xml version="1.0" encoding="Shift_JIS" ?>
<rdf:RDF
 xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
 xmlns:dc="http://purl.org/dc/elements/1.1/"
 xmlns="http://purl.org/rss/1.0/"
 xml:lang="ja">

<channel rdf:about="">
 <title></title>
 <link></link>
 <description></description>
 <items>
  <rdf:Seq>
   <rdf:li rdf:resource="http://www.sankei.co.jp/news/050202/bun021.htm"/>
  </rdf:Seq>
 </items>
</channel>

<item rdf:about="http://www.sankei.co.jp/news/050202/bun021.htm">
 <title>読んで幸せになった記事は？　日本新聞協会が募集</title>
 <link>http://www.sankei.co.jp/news/050202/bun021.htm</link>
 <description>　日本新聞協会は、平成１６年度の新聞に掲載されたさまざまなニュースの中から、読者が読んで最も幸せな気分になったニュースと、その理由を説明したコメントを募集している。審査の上、「ＨＡＰＰＹ　ＮＥＷＳ　大賞」１点と「ＨＡＰＰＹ　ＮＥＷＳ　２００４」を１０点程度選出する。</description>
 <dc:date>2005-02-04T07:57:04+09:00</dc:date>
</item>

</rdf:RDF>
```

図 6.2: 実験プログラムの RSS ファイル出力例.

6.2 実験と結果

本論文で提案した抽出方法を用いたプログラムファイルにより、タイトルについての抽出精度の実験を行った。今回の実験に使用したサイトは、表6.1に示されている12のサイトである。タイトルがきちんと出力されているものは○、タイトルのようなものがRSSに出力されていれば△、それ以外の出力結果の内容がまったく出来ていないものは×として、各サイトから任意に6記事ずつ、抽出精度についての実験を行った結果を表6.1に示す。

表 6.1: タイトルの抽出精度実験の結果.

	○	△	×
毎日新聞	6	0	0
読売新聞	4	2	0
産経新聞	5	1	0
Gendai.net	6	0	0
四国新聞社	6	0	0
東京新聞	5	1	0
河北新報	6	0	0
SANSPO.COM	5	1	0
CNN.co.jp	0	6	0
ロイター	6	0	0
PC Watch	6	0	0
Japan.internet.com	3	0	3

第7章　考察

7.1　実験の考察

いくつか記事ではサブタイトルや小見出しなどが、タイトルとして誤認される事があった。図7.1が東京新聞サイトのその例で、図7.2がその結果のRSSである。タイトルよりもサブタイトルのほうが、本文の言葉を用いていたり、文頭により近い位置にあったりする事があるためと考えられる。

H2A7号機　打ち上げ成功

ひまわり後継衛星軌道に投入

日本の主力ロケット「H2A」7号機が26日午後6時25分、宇宙航空研究開発機構種子島宇宙センター（鹿児島県南種子町）から打ち上げられた。搭載した運輸多目的衛星（MTSAT）は約40分後に分離。ロケット打ち上げは成功、衛星は予定の円軌道に投入された。H2Aの成功は2003年8月の6号機以来。失敗した6号機から1年3カ月ぶりの名誉回復となった。

種子島地方はこの日、曇り、風も夕方までに弱まった。打ち上げは当初、午後五時九分に予定されていたが、直前の四時ごろ、ロケットの機体から地上設備に送るデータ通信に不具合がみつかり、一時間以上遅れた。

午後六時二十五分、一段目の主エンジン「LE7A」、続いて大型補助ロケットに点火された。轟音の中、機体は赤い炎と白煙を出しながら、高度を上げ、すぐに雲の中に姿を消した。約二分後、6号機の失敗原因となった補助ロケット分離にも成功。計画通りに飛行し、約四十分後、高度約二千キロで衛星を分離した。

MTSATは、これからエンジンを噴射して軌道を変えながら、七日目に太陽電池パネルを展開し、十日目に静止軌道に入る。無事に進めば気象衛星「ひまわり5号」の後継として、米国の老朽化した衛星を借りている観測り状態を解消し、天気予報の精度を向上させることが可能になる。

ロケットの費用は百二十億円、衛星と合わせた費用は二百八十三億円となっている。

■ひまわり監視も　新衛星愛称

打ち上げに成功した運輸多目的衛星新1号の愛称について、気象庁の長坂昇一長官は二十六日夜の会見で「安定運用に入ったら各方面と相談して考えたい。公募は考えていない」とし、「『ひまわり』が国民に広く親しまれていたことも踏まえたい」と、「ひまわり」を踏襲する可能性もあることを示唆した。

■解説　難しい国際市場参入

H2A7号機の打ち上げが成功し、連続失敗という最悪の事態は免れた。H2Aは日本で初めて、世界の商業衛星打ち上げ市場への参入を目指している。6号機の失敗でゼロからの出発となった今回、どうにか復活への一歩を踏み出した。

とはいえ、世界の商業衛星打ち上げ市場は大きくない。二〇〇五年以降、多くて年間四十機、おおむね三十機前後で推移するとの需要予測もあり、欧米などのライバル相手に受注の獲得は容易でない。今後、H2Aは、宇宙航空研究開発機構が「最大の課題」と目標に掲げる信頼性の向上に取り組むことになる。

同機構の実用衛星用ロケットの打ち上げ実績が三十七機なのに対し、欧州のアリアンは百六十一機、米国のアトラス、デルタはともに三百機以上。中国の長征も、日本の二倍の経験と実績を持つ

26日午後6時25分、鹿児島県・種子島宇宙センターから、晴間の中、打ち上げられるH2Aロケット7号機＝本社ヘリ「わかづる」から（鎌詞一徳撮影）

図 7.1: 誤認された東京新聞サイトの記事の例.

```
<?xml version="1.0" encoding="Shift_JIS" ?>
<rdf:RDF
 xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
 xmlns:dc="http://purl.org/dc/elements/1.1/"
 xmlns="http://purl.org/rss/1.0/"
 xml:lang="ja">

<channel rdf:about="">
 <title></title>
 <link></link>
 <description></description>
 <items>
  <rdf:Seq>
   <rdf:li rdf:resource="http://www.tokyo-
np.co.jp/00/sya/20050227/mng_____sya_____005.shtml"/>
  </rdf:Seq>
 </items>
</channel>

<item rdf:about="http://www.tokyo-
np.co.jp/00/sya/20050227/mng_____sya_____005.shtml">
 <title>ひまわり後継衛星軌道に投入</title>
 <link>http://www.tokyo-
np.co.jp/00/sya/20050227/mng_____sya_____005.shtml</link>
 <description>　日本の主力ロケット「Ｈ２Ａ」７号機が２６日午後６時２
５分、宇宙航空研究開発機構種子島宇宙センター（鹿児島県南種子町）か
ら打ち上げられた。搭載した運輸多目的衛星（ＭＴＳＡＴ）は約４０分後
に分離。</description>
 <dc:date>2005-02-28T07:59:33+09:00</dc:date>
</item>

</rdf:RDF>
```

図 7.2: 誤認された東京新聞サイトの記事の RSS.

CNN.CO.JP のサイトの記事からは、○に該当する結果を一つも得ることは出来なかった。図 7.3 の CNN.CO.JP サイトの記事を処理した結果が図 7.4 であるが、図 7.4 のように、どの記事も記事のタイトル部分の文字が要約の要素に出力され、タイトルには title タグの中身が出力される、という結果になった。これはタイトル部分と本文部分の区切り方として、BR タグや P タグで行われていたため、区切り部分が判断することが出来ず、タイトルと本文が一つの候補として扱われてしまったためである。このような問題が生じる可能性は始めから予測していた。しかし、BR タグや P タグを区切りとして判断させてしまうと、本文候補の文字列が短くなり、特に BR タグは文の途中でも使われる可能性があり、句点がない本文の文字列が生じ句点の判定で引っかかる可能性も出る。したがって、部分部分で本文の文章が抜け落ちる可能性があるため、今まで区切りとして判断させてはいなかった。

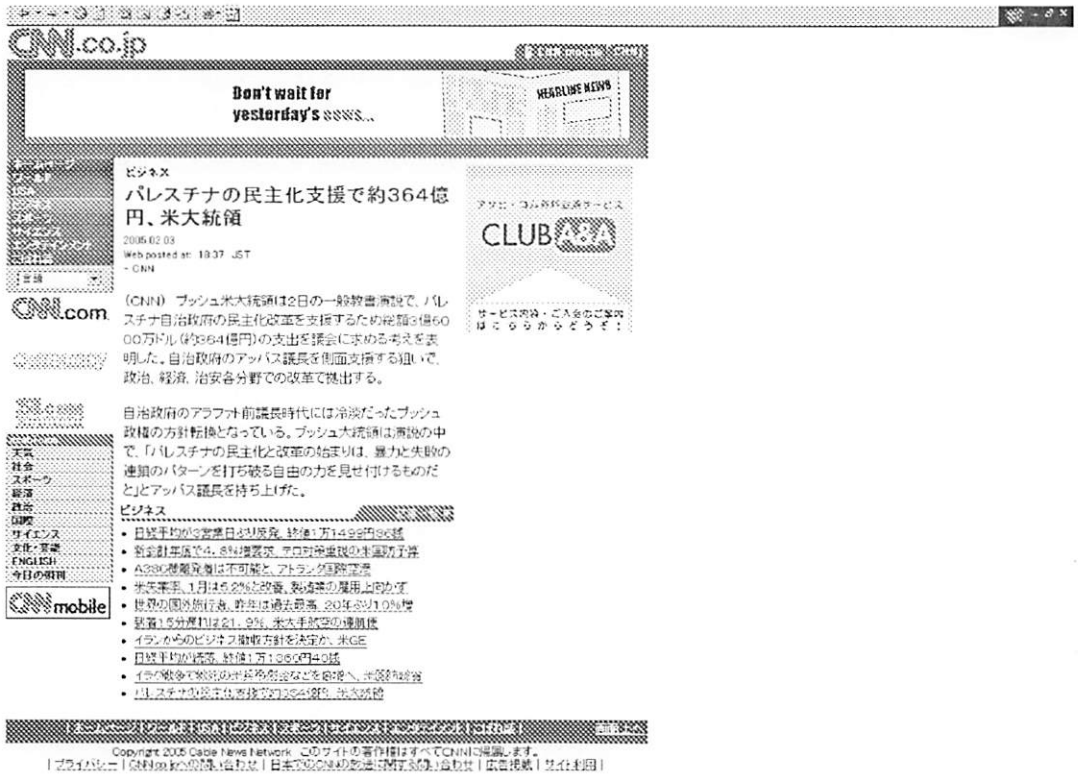

図 7.3: CNN.CO.JP サイトの記事の例.

```
<?xml version="1.0" encoding="Shift_JIS" ?>
<rdf:RDF
 xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
 xmlns:dc="http://purl.org/dc/elements/1.1/"
 xmlns="http://purl.org/rss/1.0/"
 xml:lang="ja">

<channel rdf:about="">
 <title></title>
 <link></link>
 <description></description>
 <items>
  <rdf:Seq>
   <rdf:li rdf:resource="http://www.cnn.co.jp/business/CNN200502030018.html"/>
  </rdf:Seq>
 </items>
</channel>

<item rdf:about="http://www.cnn.co.jp/business/CNN200502030018.html">
 <title>CNN.co.jp:パレスチナの民主化支援で約３６４億円、米大統領 -ビジネス
</title>
 <link>http://www.cnn.co.jp/business/CNN200502030018.html</link>
 <description>パレスチナの民主化支援で約３６４億円、米大統領2005.02.03Web
posted at: 18:37 JST-CNN（ＣＮＮ）　ブッシュ米大統領は２日の一般教書演説で、
パレスチナ自治政府の民主化改革を支援するため総額３億５０００万ドル（約３
６４億円）の支出を議会に求める考えを表明した。自治政府のアッバス議長を側
面支援する狙いで、政治、経済、治安各分野での改革で拠出する。</description>
 <dc:date>2005-02-28T09:43:31+09:00</dc:date>
</item>

</rdf:RDF>
```

図 7.4: CNN.CO.JP サイトの記事の RSS.

japan.internet.com のサイトでは、本文と関連記事のリンクとの区切れが判断されずまとめて本文と判断されてしまった。関連記事はほとんどリンク文字であるため、本文の長さが短い場合には、リンク文字の割合が大きくなり本文として認識することが出来ず、処理が中断される結果となった。この場合も本文と関連記事の区切れとして使われたタグは BR タグであった。

読売新聞サイトでは、社説にある記事のタイトルが、日付の後に読売社説という言葉が付いているだけの、カテゴリ的な内容になっている。そのため、高いヒット率を得る事が出来ず、title タグにある候補がタイトルとして選ばれる結果となった。

また、タイトルはきちんと出力されていたが、写真の説明が本文の前に入ってしまい、要約部分が写真の説明になってしまった記事が一つあった。これは写真の説明が HTML ソースの位置的に本文の前にあったため、写真説明のあるテーブルを判断できなかったためであると考えられる。

7.2 まとめ

今回の実験から、タイトルや本文、関連リンクの境目にあるタグが BR タグや P タグのみ、という記事形式のニュースサイトはいくつかあり、これらの記事については大きく抽出精度が下がるという結果を得る事ができた。特にタイトルについては、この分離が行う事ができない限り、完全なタイトルを得ることはできない。この BR タグや P タグについても区切りとして判定する必要があるだろう。ただ、本文判定が現状のまま判断を行うと、特に句点が付かない本文中の小見出し部分や、短い文などが抽出されなくなったり等、本文の細かい抜け落ちの可能性があると考えられる。

第8章　おわりに

本論文のタイトルを抽出する手法は、ある程度広範囲のニュースサイトに汎用的に使うことができる。また抽出した本文から最初の2文を要約として、RSS フィードへの出力も行っている。しかし現状においては、まだニュースサイトの記事形式によって大きく精度を損なう場合が発生する。

より汎用性を高め、多くのサイトで抽出精度の高い RSS フィードの出力を行えるための問題点の改良と、サイト単位での、RSS フィード自動更新処理の機能を追加が今後の課題である。

謝辞

本研究の遂行及び論文の作成において多大な御助言及び指導を賜った新納 浩幸教官 (茨城大学工学部システム工学科) に深い感謝の意を表します。また、本研究を進めるにあたり助言、協力を頂きました岩崎 唯史 教官、同研究室の紺野 憲一氏 (茨城大学院理工学研究科システム工学専攻)、藤井 丈昭 氏 (茨城大学院理工学研究科システム工学専攻)、正木 祐一 氏 (茨城大学院理工学研究科システム工学専攻)、谷津 哲平 氏 (茨城大学院理工学研究科システム工学専攻)、大北 高広 氏 (茨城大学工学部システム工学科 4 回生)、木本 俊 氏 (茨城大学工学部システム工学科 4 回生)、高橋 宏直 氏 (茨城大学工学部システム工学科 4 回生)、茂木 啓吾 氏 (茨城大学工学部システム工学科 4 回生) に深く感謝致します。

付録 プログラムソースリスト

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include <time.h>

#include "fmdll.c"


#define H_NAGASA 5
#define ANDBUFSIZE 5
#define BUFSIZE 256
#define MSIZE 32
#define LUMP_MAX 5000
#define HONBUN_MIN 200


typedef struct
{
unsigned char *mojis;
int msuu, mhit, b_count, a_count;
int honbun, title, para;
fpos_t fpos;
} LUMPS;
```

```
int main(int argc, char *argv[])
{
FILE *fp, *fp2, *fp3;
LUMPS lp[LUMP_MAX];
     int lump_end, lump_max, title_num, honbun_max;
int i, j, k, a_tag, a_count, b_tag, tb_tag, tmoji;
char *rssname="rss.rdf";
char c, linknum, buf[ANDBUFSIZE+BUFSIZE], *itembuf, *str_src, *str_p;
char *str;
fpos_t  fp_end;
time_t  tp;
double hitav, hittmp, npt, nk;

struct tm *u_tp;

for (i=0; i < argc; i++)
{
  if (strncmp(argv[i], "http://", 7) == 0) {
    linknum = i;
    if ( (str_src = (char *)malloc(2000000)) == NULL) {
      printf("メモリが確保できません\n");
      exit(1);
    }

    if (getfile_http(str_src, argv[i]) == 0) {
      printf("getfile_http error!!(%s)\n", argv[i]);
      exit(1);
```

```
    }
    if ((fp = fopen("$temp.html", "w+")) == NULL) {
      printf("file open error!!(%s)\n", "#temp.html");
      exit(1);
    }
    fprintf(fp, "%s", str_src);
    fseek(fp, 0L, SEEK_SET);
    free(str_src);
    break;
  }
  else if (strncmp(argv[i], "file://", 7) == 0) {
    if ((fp = fopen(&argv[i][7], "r")) == NULL) {
      printf("file open error!!(%s)\n", &argv[i][7]);
      exit(1);
    }
    break;
  }
}
if (i == argc) {
  linknum = 1;
  if ((fp = fopen(argv[1], "r")) == NULL) {
    printf("file open error!!(%s)\n", argv[1]);
    exit(1);
  }
}
if ((fp2 = fopen("#kekka.c", "w")) == NULL) {
  printf("file open error!!(%s)\n", "#kekka.c");
  exit(1);
}
```

```
// デバッグファイル
if ((fp3 = fopen("$debug.c", "w")) == NULL) {
  printf("file open error!!\(%s)n", "$debug.c");
  exit(1);
}

if ( (str = (char *)malloc(200000)) == NULL) {
  printf("メモリが確保できません(str)\n");
  exit(1);
}

if ( (itembuf = (char *)malloc(150*2000)) == NULL) {
  printf("メモリが確保できません\n");
  exit(1);
}



for (i=0; i < argc; i++)
{
  if (strncmp(argv[i], "-O", 2) == 0) {
    tmoji=atoi(&argv[i][2]);  // 文字の塊の最低文字数
  }
}
if (i == argc) tmoji=H_NAGASA;

lump_end=0;
str_p=str; // 文字の長さ
lump_max=0; // 塊の最大数
honbun_max=0; // 本文である塊の最大数
```

```
a_tag=0; // aタグの判別
a_count=0; // aタグ内の文字列長
b_tag=0;
tb_tag=0;
title_num=0;

lp[0].para=0;
lp[0].title=0;
lp[0].honbun=0;
lp[0].b_count=0;



fseek(fp, 0L, SEEK_END);
fgetpos(fp, &fp_end);
fprintf(fp3, "ファイル位置 = %ld .. %ld\n", ftell(fp), fp_end);
fseek(fp, 0L, SEEK_SET);
fprintf(fp3, "ファイル位置 = %ld\n\n", ftell(fp));


while ((c=fgetc(fp)) != EOF) // htmlタグの後から処理の開始
{
  if (c == '<') {
    if ((c=fmjseek(fp, "html", &i)) == 0) {
      while (c != '>') c=fgetc(fp);
      break;
    }
    while (c != '>') c=fgetc(fp);
  }
}
```

```
while ((c=fgetc(fp)) != EOF)
{

// タグ判別

  if (c == '<') {
    c=tolower(fgetc(fp));
    if (c == '!') {
      if (fmjseek(fp, "--", &i) == 0) {
        while ( (c=fmjseek(fp, "-->", &i)) != 0)
        {
          while (c == '-')
          {
            if ((c=fgetc(fp)) == '>') {
              goto bk;
            }
          }
        }
bk:         lump_end = 1;
        c='>';
      }
    }

    else if (c == 's') {
      if ( (c=tolower(fgetc(fp))) == 'c') {
        if (fmjseek(fp, "ript", &i) == 0) {
          while (c != '<') c=fgetc(fp);
          while ( (c=fmjseek(fp, "/script", &i)) != 0)
          {
```

```
        while (c != '<') c=fgetc(fp);
      }
      lump_end = 1;
    }
  }
  else if (c == 't') {
    if (fmjseek(fp, "yle", &i) == 0) {
      while (c != '<') c=fgetc(fp);
      while ( (c=fmjseek(fp, "/style", &i)) != 0)
      {
        while (c != '<') c=fgetc(fp);
      }
      lump_end = 1;
    }
  }
  else if (c == 'e') {
    if (fmjseek(fp, "lect", &i) == 0) {
      while (c != '<') c=fgetc(fp);
      while ( (c=fmjseek(fp, "/select", &i)) != 0)
      {
        while (c != '<') c=fgetc(fp);
      }
      lump_end = 1;
    }
  }
}

else if (c == 'a') {
  if ( (c=tolower(fgetc(fp))) == ' '  ||  c == '>') {
    a_tag=1;
```

```
  }
}

else if (c == '/') {
  if ( (c=tolower(fgetc(fp))) == 'a') {
    if ( (c=tolower(fgetc(fp))) == ' '  ||  c == '>') {
      a_tag=0;
    }
  }
  else if (c == 'b') {
    if ( (c=tolower(fgetc(fp))) == ' '  ||  c == '>') {
      b_tag=0;
    }
  }
  else if (c == 'c') {
    if ((c=fmjseek(fp, "enter", &i)) == 0) {
      lump_end = 1;
    }
  }
  else if (c == 'd') {
    if ((c=fmjseek(fp, "iv", &i)) == 0) {
      lump_end = 1;
    }
  }
  else if (c == 't') {
    if ( (c=tolower(fgetc(fp))) == 'a') {
      if ((c=fmjseek(fp, "ble", &i)) == 0) {
        lump_end = 1;

        if ( HONBUN_MIN < str_p-str ||
```

```
            (honbun_max  && lp[lump_max-1].honbun)
          ) {
          tb_tag--;
        }
      }
    }
    else if (c == 'i') {
      if ((c=fmjseek(fp, "tle", &i)) == 0) {
        lump_end = 1;
        lp[lump_max].para = 1;
      }
    }
    else if (c == 'd') {
      b_tag=0;
    }
  }
  else if (c == 'h') {
    if ( (c=tolower(fgetc(fp))) == '1'  ||  c == '2'  ||  c == '3'
         ||  c == '4'  ||  c == '5'  ||  c == '6') {
    }
  }

}

else if (c == 'h') {
  if ( (c=tolower(fgetc(fp))) == '1'  ||  c == '2') {
    lp[lump_max].para = 2;
  }
  else if (c == '3') {
    lp[lump_max].para = 3;
```

```
  }
  else if (c == '4'  ||  c == '5'  ||  c == '6') {
    lp[lump_max].para = 4;
  }
}

else if (c == 'b') {
  if ( (c=tolower(fgetc(fp))) == 'r') {
    *(str_p++) = '\n';

    if (a_tag) a_count++;
    if (b_tag) lp[lump_max].b_count++;
  }
  else if(c == ' '  ||  c == '>') {
    b_tag=1;
  }
}

else if (c == 'p') {
  if ( (c=tolower(fgetc(fp))) == ' '  ||  c == '>') {
    if (str_p != str) {
      for (i=0; i<2; i++)
      {
        *(str_p++) = '\n';
      }
      if (a_tag) a_count += i;
      if (b_tag) lp[lump_max].b_count++;
    }
  }
}
```

```
else if (c == 't') {
  if ( (c=tolower(fgetc(fp))) == 'd') {
    lump_end = 1;
  }
  else if (c == 'i') {
    if ((c=fmjseek(fp, "tle", &i)) == 0) {
      lump_end = 1;
    }
  }
  else if (c == 'a') {
    if ((c=fmjseek(fp, "ble", &i)) == 0) {
      if ( honbun_max  &&
           (HONBUN_MIN < str_p-str || lp[lump_max-1].honbun)
         ) {
        tb_tag++;
        printf("t%d\n" , tb_tag);
      }
      while (0 < tb_tag)
      {
        if (fgetc(fp) == '<') {
          if ((c=tolower(fgetc(fp))) == 't') {
            if (fmjseek(fp, "able", &i) == 0) tb_tag++;
            printf("A");
          }
          if (c == '/') {
            if (fmjseek(fp, "table", &i) == 0) tb_tag--;
            printf("B");
          }
        }
```

```
        }
        printf("%dc ", str_p-str);
        lump_end = 1;
      }
    }
  }

  else if (c == 'c') {
    if ((c=fmjseek(fp, "enter", &i)) == 0) {
      lump_end = 1;
    }
  }
  else if (c == 'i') {
    if ((c=fmjseek(fp, "mg", &i)) == 0) {
      lump_end = 1;
    }
  }
  else if (c == 'd') {
    if ((c=fmjseek(fp, "iv", &i)) == 0) {
      lump_end = 1;
    }
  }
  else if (c == 'u') {
    if ((c=fmjseek(fp, "l", &i)) == 0) {
      lump_end = 1;
    }
  }

  while (c != '>') c=fgetc(fp);
}
```

```
else if (c == ' ') continue;
else if (c == '\n') continue;
else if (c == 0x09) continue;
else {
  if (c == '&') { // &による文字の変換
    if ((c=fgetc(fp)) == 'l') {
      if ((c=fmjseek(fp, "t;", &i)) == 0) {
        *(str_p++) = '<';
      }
      else {
        strncpy(str_p, "&lt;", i+2);
        str_p += i+2;
        *(str_p++) = c;
      }
    }
    else if (c == 'g') {
      if ((c=fmjseek(fp, "t;", &i)) == 0) {
        *(str_p++) = '>';
      }
      else {
        strncpy(str_p, "&gt;", i+2);
        str_p += i+2;
        *(str_p++) = c;
      }
    }
    else if (c == 'q') {
      if ((c=fmjseek(fp, "uot;", &i)) == 0) {
        *(str_p++) = '"';
      }
      else {
```

```
            strncpy(str_p, "&quot;", i+2);
            str_p += i+2;
            *(str_p++) = c;
          }
        }
        else if (c == 'n') {
          if ((c=fmjseek(fp, "bsp;", &i)) == 0) {
            *(str_p++) = ' ';
          }
          else {
            strncpy(str_p, " ", i+2);
            str_p += i+2;
            *(str_p++) = c;
          }
        }
        else if (c == 'a') {
          if ((c=fmjseek(fp, "mp;", &i)) == 0) {
            *(str_p++) = '&';
          }
          else {
            strncpy(str_p, "&amp;", i+2);
            str_p += i+2;
            *(str_p++) = c;
          }
        }
        else {
          *(str_p++) = '&';
          *(str_p++) = c;
        }
      }
```

```
    else *(str_p++) = c;

    if (a_tag) a_count++;
    if (b_tag) lp[lump_max].b_count++;
  }


  if (lump_end) {  // 区切りタグがあった時
    if (tmoji < str_p-str) {
      *(str_p) = '\0';
      lp[lump_max].msuu = str_p-str;


// デバッグ用
      fprintf(fp3, "\n\n --- aaa%d - str%d ---\n", a_count, lp[lump_max].msuu);
      fprintf(fp3, "%s", str);
      fprintf(fp3, "\n           <-- lump%d -->\n", lump_max);
      if (fgetpos(fp, &lp[lump_max].fpos)) {
        printf("fgetpos error\n");
      }
      lp[lump_max].fpos -= lp[lump_max].msuu;
      fprintf(fp3, "%3dt       " , tb_tag);
      fprintf(fp3, "/-- fpos%d --/\n", lp[lump_max].fpos);


      if (4*a_count < lp[lump_max].msuu) { // リンク全体の 1/4 以下
        if (strstr(str, "。") != NULL  &&  HONBUN_MIN < lp[lump_max].msuu) {
          for (i=0; i<lump_max; i++)
          {
```

```
            if (lp[i].honbun == honbun_max) break;
          }

// 本文の最後と離れている場合
          if (1.5*(lp[lump_max].fpos -lp[i].fpos -lp[i].msuu)
              <  fp_end  ||  !honbun_max) {
            lp[lump_max].honbun = ++honbun_max;
            fprintf(fp2, "%2d:\n%s\n\n", honbun_max, str);
          }
          else {
            fprintf(fp3, "damefpos:%d\n", lp[lump_max].fpos -lp[i].fpos -lp[i].m
          }
        }

      }
      lp[lump_max].a_count=a_count;
      if ( (lp[lump_max].mojis = (char *)malloc(lp[lump_max].msuu+1)) == NULL) {
        printf("メモリが確保できません\n");
        exit(1);
      }
      strcpy(lp[lump_max].mojis, str);
      lump_max++;

      lp[lump_max].para=0;
      lp[lump_max].title=0;
      lp[lump_max].honbun=0;
    }
    str_p=str;
    lump_end=0;
    a_count=0;
```

```
    lp[lump_max].b_count=0;
  }


}


fclose(fp);
fclose(fp2);


if (honbun_max == 0) {
  fprintf(fp3, "指定されたファイルから本文を取り出せませんでした。\n");
  printf("指定されたファイルから本文を取り出せませんでした。\n");
  exit(1);
}


system("\"C:\\Program Files\\ChaSen\\chasen\" #kekka.c > $chasen.c");
// system("del #kekka.c");
```

```
if ((fp = fopen("$chasen.c", "r")) == NULL) {
  printf("file open error!!\n(%s)", "$chasen.c");
  exit(1);
}


// 本文の名詞をとりだしてタイトルをさがす

j=0;
for (i=0; i<lump_max; i++)
{
  lp[i].mhit=0;
}
while (fgets(buf, ANDBUFSIZE+BUFSIZE, fp) != NULL)
{
  if (strstr(buf, "名詞") != NULL && strstr(buf, "名詞-数") == NULL) {
    for (i=0; buf[i]!=0x09; i++)
    {
      *(str+j*MSIZE+i) = buf[i];
    }
    *(str+j*MSIZE+i) = buf[i] = '\0';


// buf の中身が今までに出てきていないか判別
    for (i=0; i<j; i++)
    {
      if (strcmp(buf, str+i*MSIZE) == 0) {
        break;
      }
```

```
    }

// 出て来ていなければタイトル候補に語句があるか判定する
    if (i == j) {
      fprintf(fp3, "*-- 名詞:%s ---\n", buf);
      for (i=0; i<lump_max; i++)
      {
        if (lp[i].honbun) continue;

        if (strstr(lp[i].mojis, buf) != NULL) {
          lp[i].mhit += strlen(buf);
            // hit した文字の長さだけ増やす
        }
      }
      j++;
    }
  }
}
```


```
hitav=j=0;
/* for (j=0; j<honbun_max; j++)
{
*/   fprintf(fp3, "\n\n\n\n\n\n\n\n\n\n------- hon:%d -------\n", j);
  for (i=0; i<lump_max; i++)
  {
    if (lp[i].honbun) continue;
```

```
// 文字列長による優先度合の決定
    if (lp[i].msuu < 75) {
      npt = 1200000000 / (double)(pow((int)(lp[i].msuu-50),
        6) +1200000000 );
    }
    else  npt = 0.85;

// 本文との距離による優先度
    for (k=0; k<lump_max; k++) if (lp[k].honbun) break;
    if (abs(lp[k].fpos - lp[i].fpos) < 150*80) {
      nk = 99000000 / (double)(pow(lp[k].fpos -lp[i].fpos,
        2) +99000000);
    }
    else  nk = 0.3;

// 文字列全体の長さに対するヒット率
    hittmp = lp[i].mhit / (double)lp[i].msuu;

// title タグなら
    if (lp[i].para == 1) {
      hittmp += 0.25;
      if (nk < 0.75) nk = 0.75;
//       if (npt < 0.75) npt = 0.75;
    }
// h1-h2 タグなら
    else if (lp[i].para == 2) {
      hittmp += 0.50;
      if (nk < 0.8) nk = 0.8;
//       if (npt < 0.55) npt = 0.55;
    }
```

```
// h3タグなら
    else if (lp[i].para == 3) {
      hittmp += 0.40;
      if (nk < 0.75) nk = 0.75;
//       if (npt < 0.55) npt = 0.55;
    }
// h4-h6タグなら
    else if (lp[i].para == 4) {
      hittmp += 0.1;
      if (nk < 0.7) nk = 0.7;
    }

// bタグなら
    if (lp[i].b_count) {
      hittmp += 0.1*lp[i].b_count / (double)lp[i].msuu;
      fprintf(fp3, "\n--- b_c%d --", lp[i].b_count);
    }

// aタグなら
    if (lp[i].a_count) {
      hittmp *= 1 - 0.3*lp[i].a_count / (double)lp[i].msuu;
      fprintf(fp3, "\n--- a_c%lf -- ", 1 - 0.3*lp[i].a_count / (double)lp[i].msu
    }


    if (hitav < hittmp * npt * nk) {
      hitav = hittmp * npt * nk;
      title_num = i;
    }
```

```
    if (hittmp * npt * nk) {
      fprintf(fp3, "\n  /-- hit%d - naga%d - pa%d ", lp[i].mhit,
        lp[i].msuu, lp[i].para);
      fprintf(fp3, "--- hav%lf - npt%lf - nk %lf -\n", hittmp*npt*nk,
        npt, nk);
      fprintf(fp3, "%s", lp[i].mojis);
      fprintf(fp3, "\n              --- i:%d ---/", i);
      fprintf(fp3, "/-- fpos%d --/\n\n", lp[i].fpos);
    }
  }
  lp[title_num].title = j+1;


/* }
*/



// 出力処理

*str = *itembuf = '\0';
for (i=0; i < argc; i++)
{
  if (strncmp(argv[i], "-R", 2) == 0) {
    rssname = &argv[i][2];
    break;
  }
}
```

```
if ((fp2 = fopen(rssname, "r+")) != NULL) { // すでにRSSがあるなら
  while (fgets(buf, ANDBUFSIZE+BUFSIZE, fp2) != NULL)
  {
    if (*itembuf == '\0') {
      if (strstr(buf, "<channel") != NULL) {
        if (strstr(buf, "rdf:about=\"\"") != NULL) {
          *itembuf = 1;
        } // 同じサイトのRSSなら
      }
    }
    if (*itembuf) {
      if (strstr(buf, "<rdf:li rdf:resource=") != NULL) {
        strcat(str, buf);
      } // items要素の保存

      if (strstr(buf, "<item ") != NULL) {
        break;
      }
    }
  }
  if (*itembuf == '\0') {
    printf("他サイト用の%sファイルがあります。", argv[2]);
    exit(1);
  }
  *itembuf = '\0';

// item要素の保存
  strcat(itembuf, buf);
  while (fgets(buf, ANDBUFSIZE+BUFSIZE, fp2) != NULL)
  {
```

```
    if (strstr(buf, "</rdf:RDF>") != NULL) {
      break;
    }
    strcat(itembuf, buf);
  }
  fclose(fp2);
}

if ((fp2 = fopen(rssname, "w")) == NULL) {
  printf("file open error!!\n(%s)", rssname);
  exit(1);
}



// RSS &サイト情報
fprintf(fp2, "<?xml version=\"1.0\" encoding=\"Shift_JIS\" ?>\n");
fprintf(fp2, "<rdf:RDF\n");
fprintf(fp2, "  xmlns:rdf=\"http://www.w3.org/1999/02/22-rdf-syntax-ns#\"\n");
fprintf(fp2, "  xmlns:dc=\"http://purl.org/dc/elements/1.1/\"\n");
fprintf(fp2, "  xmlns=\"http://purl.org/rss/1.0/\"\n");
fprintf(fp2, "  xml:lang=\"ja\">\n\n");

fprintf(fp2, " <channel rdf:about=\"\">\n");
fprintf(fp2, "  <title></title>\n");
fprintf(fp2, "  <link></link>\n");
fprintf(fp2, "  <description></description>\n");
fprintf(fp2, "  <items>\n");
fprintf(fp2, "   <rdf:Seq>\n");
```

```
fprintf(fp2, "    <rdf:li rdf:resource=\"");
fprintf(fp2, "%s\"/>\n", argv[linknum]);

fprintf(fp2, "%s   </rdf:Seq>\n", str);
fprintf(fp2, "  </items>\n");
fprintf(fp2, " </channel>\n\n");



// item要素
fprintf(fp2, " <item rdf:about=\"%s\">\n", argv[linknum]);

for (i=0; i<lump_max; i++)
{
  if (lp[i].title) {
    fprintf(fp2, "  <title>%s</title>\n", mojikesi(lp[i].mojis, "\n"));
    fprintf(fp3, "\n--- title:%d ---\n", i);
    for (j=0; j<lump_max; j++)
    {
      if (lp[j].honbun) {
        fprintf(fp3, "--- hon:%d ---\n", j);
      }
    }
  }
}
fprintf(fp2, "  <link>%s</link>\n", argv[linknum]);

// 要約
for (k=0; k<lump_max; k++) if (lp[k].honbun) break;
youyaku(str, lp[k].mojis, lp[k].msuu);
```

```
fprintf(fp2, "  <description>%s</description>\n", mojikesi(str, "\n"));

// 更新時刻
tp = time(NULL);
u_tp = localtime(&tp);
fprintf(fp2, "  <dc:date>%d-%s-",
  u_tp->tm_year + 1900,
  ketawase(str, u_tp->tm_mon + 1, 2, 10)
);
fprintf(fp2, "%sT", ketawase(str, u_tp->tm_mday, 2, 10) );
fprintf(fp2, "%s:", ketawase(str, u_tp->tm_hour, 2, 10) );
fprintf(fp2, "%s:", ketawase(str, u_tp->tm_min, 2, 10) );
fprintf(fp2, "%s", ketawase(str, u_tp->tm_sec, 2, 10) );
fprintf(fp2, "+09:00</dc:date>\n");
fprintf(fp2, " </item>\n\n");

fprintf(fp2, "%s", itembuf);

fprintf(fp2, "</rdf:RDF>\n");


fclose(fp);
fclose(fp2);
fclose(fp3);
free(str);
free(itembuf);
for (i=0; i<lump_max; i++) free(lp[i].mojis);

return  0;
}
```

```
// fmdll.c

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <math.h>
#include <time.h>

#include <winsock2.h>
```

```
int fmjseek(FILE *, unsigned char [], int *);
// 現在ファイル位置からの文字列比較
// 引数：*fp、比較文字列，何文字合っていたかを返還
// 戻り値：等しい−＞0、違う−＞違った文字
// 違いがわかる時点まで fp 進む

char* mojikesi(char *, char *);
// 指定した文字列の消去
  // 引数：出力文字、消去文字列

int youyaku(unsigned char *, unsigned char *, int);
// 要約 （頭2文の抽出、または先頭500バイト）

unsigned char* ketawase(unsigned char *, int, int, int);
// 桁合わせ （足りない桁の分だけ0で文字出力）
// 引数：出力文字、入力値、桁数、進数

char* getfile_http(char *, char *);
// http によるファイル取得
// 引数：出力文字、接続 URL



// ファイルからの文字列比較
int fmjseek(FILE *fp, unsigned char cbuf[], int *i)
{
```

```
int clen, c;
fpos_t  fpos;

*i=0;
clen = strlen(cbuf);
/* if (fgetpos(fp, &fpos) != 0) {
  fprintf(stderr, "fgetpos 異常\n");
}
*/ while (*i < clen)
{
if ((c=tolower(fgetc(fp))) == cbuf[*i]) {
  (*i)++;
}
else {
/*           if (fsetpos(fp, &fpos) == 0) {
            fprintf(stderr, "fsetpos 異常");
            fprintf(stderr, " 位置 = %ld .. %ld\n", ftell(fp), fpos);
          }
*/   return c;
}
}

return 0;
}



// 指定した文字列の消去
char* mojikesi(char *str_out, char *kesimoji)
```

```
{
int kmlen;
char *p, *o_end;


kmlen=strlen(kesimoji);
o_end=strlen(str_out)+str_out-kmlen;

for (p=str_out; *p != '\0'; p++)
{
  if (memcmp(p, kesimoji, kmlen) == 0) {
    memmove(p, p+kmlen, o_end-p+1); // 消す文字列を詰める
    p--;
  }
}


return str_out;
}



// 要約 (頭2文の抜き出し)
int youyaku(unsigned char *str_out, unsigned char *strg, int m_len)
{
int bun_y=0, i;
```

```
if (500 < m_len) m_len = 500;

for (i=0; i<m_len; i++)
{
  *(str_out+i) = *(strg+i);

// Shift_JISの第1バイトか判別
  if (0x80 < *(strg+i)  &&  *(strg+i) < 0xa0  ||
      0xdf < *(strg+i)  &&  *(strg+i) < 0xf0      ) {
    if (!strncmp("。", strg+i, 2)) {
      if (++bun_y == 2) {
        i++;
        *(str_out+i) = *(strg+i);
        i++;
        break;
      }
    }
    i++;
    *(str_out+i) = *(strg+i);
  }

}
*(str_out+i) = '\0';


return i;
}
```

```
// 桁合わせ （足りない桁の分だけ0で文字出力）
unsigned char* ketawase(unsigned char *str_out, int suu_in, int keta, int sinsuu
{
*(str_out+keta) = '\0';
while (keta--)
{
  *(str_out+keta) = suu_in%sinsuu;
  if (10 < sinsuu) *(str_out+keta) += 0x30+7;
  else *(str_out+keta) += 0x30;

  suu_in /= sinsuu;
}

return str_out;
}
```

```
#define BUF_LEN 256 /* バッファのサイズ */

char* getfile_http(char *str_out, char *str_in)
{
int s; /* ソケットのためのファイルディスクリプタ */
struct hostent *servhost; /* ホスト名と IP アドレスを扱うための構造体 */
struct sockaddr_in server; /* ソケットを扱うための構造体 */
struct servent *service; /* サービス（http など）を扱うための構造体 */
```

```
char send_buf[BUF_LEN]; /* サーバに送る HTTP プロトコル用バッファ */
char host[BUF_LEN] = "localhost"; /* 接続するホスト名 */
char path[BUF_LEN] = "/"; /* 要求するパス */
unsigned short port = 0; /* 接続するポート番号 */

FILE *fp;
WSADATA wsadata;


if (WSAStartup(MAKEWORD(1, 1), &wsadata)) {
  printf("WSAStartup関数失敗です\n");
  return 0;
}


if ( str_in != NULL ){ /* URLが指定されていたら */
  char host_path[BUF_LEN];

  if ( strlen(str_in) > BUF_LEN-1 ){
    fprintf(stderr, "URL が長すぎます。\n");
    return 0;
  }
/* http:// から始まる文字列で */
/* sscanf が成功して */
/* http:// の後に何か文字列が存在するなら */
  if (
       strstr(str_in, "http://") &&
       sscanf(str_in, "http://%s", host_path) &&
       strcmp(str_in, "http://")
     ) {
```

```
    char *p;

    p = strchr(host_path, '/'); /* ホストとパスの区切り "/" を調べる */
    if (p != NULL){
      strcpy(path, p); /* "/"以降の文字列を path にコピー */
      *p = '\0';
      strcpy(host, host_path); /* "/"より前の文字列を host にコピー */
    } else { /* "/"がないなら＝http://host という引数なら */
      strcpy(host, host_path); /* 文字列全体を host にコピー */
    }

          p = strchr(host, ':'); /* ホスト名の部分に ":" が含まれて
いたら */
    if (p != NULL){
      port = atoi(p+1); /* ポート番号を取得 */
      if (port <= 0){ /* 数字でない (atoi が失敗) か、0 だったら */
        port = 80; /* ポート番号は 80 に決め打ち */
      }
      *p = '\0';
    }
  } else {
    fprintf(stderr, "URL は http://host/path の形式で指定してください。
\n");
    return 0;
  }
}

printf("\n\nhttp://%s%s を取得します。\n\n", host, path);

/* ホストの情報 (IP アドレスなど) を取得 */
```

```
servhost = gethostbyname(host);
if (servhost == NULL){
  fprintf(stderr, "[%s] から IP アドレスへの変換に失敗しました。\n", host);
  return 0;
}


memset(&server, 0, sizeof(server)); /* 構造体をゼロクリア */
server.sin_family = AF_INET;
/* IP アドレスを示す構造体をコピー */
memcpy(&server.sin_addr, servhost->h_addr, servhost->h_length);


if (port != 0) { /* 引数でポート番号が指定されていたら */
  server.sin_port = htons(port);
} else { /* そうでないなら getservbyname でポート番号を取得 */
  service = getservbyname("http", "tcp");
  if (service != NULL) { /* 成功したらポート番号をコピー */
    server.sin_port = service->s_port;
printf("aaa");
  } else { /* 失敗したら 80 番に決め打ち */
    server.sin_port = htons(80);
printf("bbb");
          }
}
/* ソケット生成 */
if ( (s = socket(AF_INET, SOCK_STREAM, 0)) < 0 ){
  fprintf(stderr, "ソケットの生成に失敗しました。\n");
  return 0;
}
```

```
/* サーバに接続 */
if ( connect(s, (struct sockaddr *)&server, sizeof(server)) == -1 ){
  fprintf(stderr, "connect に失敗しました。\n");
  return 0;
}


   /* HTTP プロトコル生成 & サーバに送信 */
sprintf(send_buf, "GET %s HTTP/1.0\n", path);
// write(s, send_buf, strlen(send_buf));
send(s, send_buf, strlen(send_buf), 0);
printf("%s", send_buf);

if (port) {
  sprintf(send_buf, "Host: %s:%d\n", host, port);
}
else {
  sprintf(send_buf, "Host: %s\n", host);
}
// write(s, send_buf, strlen(send_buf));
send(s, send_buf, strlen(send_buf), 0);
printf("%s", send_buf);

sprintf(send_buf, "\n");
// write(s, send_buf, strlen(send_buf));
send(s, send_buf, strlen(send_buf), 0);
printf("\\n\n");

/* あとは受信して、表示するだけ */
```

```
while (1)
{
  char buf[BUF_LEN];
  int read_size;

//   read_size = read(s, buf, BUF_LEN);
  read_size = recv(s, buf, BUF_LEN, 0);
  if (read_size > 0){
//     write(1, buf, read_size);
    memcpy(str_out, buf, read_size);
    str_out += read_size;
  }
  else {
    break;
  }
}

*str_out = '\0';



/* 後始末 */
// close(s);
fclose(fp);
WSACleanup();


return str_out;
}
```

参考文献

[1] 伊藤直也 , “RSS の技術的概要と最新動向”,
UNIX USER 2004 年 4 月号, pp.80-90.

[2] 神崎正英 , “RSS -- サイト情報の要約と公開”,
http://www.kanzaki.com/docs/sw/rss.html

[3] Dave Beckett ed. , “RDF/XML Syntax Specification (Revised)”,
http://www.w3.org/TR/rdf-syntax-grammar/

[4] Renato Iannella , “Representing vCard Objects in RDF/XML”,
http://www.w3.org/TR/vcard-rdf

[5] Tim Bray et al. , “Namespaces in XML”,
http://www.w3.org/TR/REC-xml-names

[6] James Clark , “XML Namespaces77”,
http://www.jclark.com/xml/xmlns.htm