DEWS2006 1A-i5

アノテーション処理のための基盤ライブラリの構築

阿部　裕行† 　伊藤　一成† 　Martin J.Dürst†

† 青山学院大学 理工学部
〒 229–8558 神奈川県相模原市淵野辺 5–10–1
E-mail: hiroyuki@sw.it.aoyama.ac.jp, {kaz, duerst}@it.aoyama.ac.jp

あらまし　情報化社会の今，膨大な量の Web ページやマルチメディアコンテンツが散在している．その中から専門知識を持っていない一般のユーザが，自ら目的のコンテンツを獲得することはとても困難である．この問題を解決し，より容易に有益な情報を獲得する仕組みとして，メタデータ·アノテーション技術が注目をあびている．そこでアノテーションを活用するための基盤ライブラリを提案する．このライブラリの特徴として RDF 形式としてデータを生成するだけではなく，自然言語処理をする際に時間のかかる言語的情報の抽出をデータ生成時あるいは，計算機の遊休時に行う．その結果を言語情報記述に関する XML タグセットである GDA でタグ付けするので，言語情報に基づく処理が必要な際，高速に処理できる．自然言語処理と構造マイニングの複合処理により，応用として考えられるサービスを支える基盤ライブラリを目標とする．

キーワード　アノテーション，自然言語処理，構造マイニング

Construction of an Infrastructure Library for Annotation Processing

Hiroyuki ABE†, Kazunari ITO†, and Martin J. DÜRST†

† College of Science and Engineering, Aoyama Gakuin University
Fuchinobe 5–10–1, Sagamihara, Kanagawa, 229–8558 Japan
E-mail: hiroyuki@sw.it.aoyama.ac.jp, {kaz, duerst}@it.aoyama.ac.jp

Abstract　It is very hard for general users to obtain information from the huge amount of Web pages and multimedia content available today. To solve this problem, there is metadata and annotation technology as a method to get high precision information easily. This paper proposes an infrastructure library for generating annotations which can process based on natural language processing and structure mining. This library not only creates RDF data but also attaches GDA tags, an XML tag set concerning description of language information simultaneously or machine idling. Therefore, while it usually takes time to produce language information, the library can respond very quickly even to request needing natural language processing. Combining the handling of natural language processing and structure mining combining, this library will be infrastrucure which supports a wide range of services.

Key words　annotation, natural language processing, structure mining

1. はじめに

情報化社会の今，膨大な量の Web ページやマルチメディアコンテンツが散在している．その中からユーザの要求にあったコンテンツの獲得は困難である．さらに計算機が自動的に，より高度にコンテンツを扱ってくれるような仕組みが必要である．近年，コンテンツの意味や内容に関する特徴を情報として付与し，付与したデータを計算機処理対象とすることで，元のコンテンツ情報を効率よく検索，要約などを行い，高度に扱うアノテーションの研究 [1] が注目されている．

例えば次世代の Web として提案され，メタデータとオントロジーにより計算機が自動的に Web 上の情報の意味理解の実現を目指す Semantic Web では，Web 上のリソースのためのメタデータ記述方法として RDF (Resource Description Framework) [2] が策定されている．RDF はリソース（Subject），プロパティ（Predicate）及びプロパティ値（Object）の RDF トリプルの集まりとして定義され，さらに RDFS (RDF Schema) によってプロパティが定義できる．そしてオントロジーの記述言語として OWL (Web Ontology Language) が策定されている．しかしながら，だれがメタデータやオントロジーを作るのかという問題がある．メタデータやオントロジーを作るためには，ある程度の知識が必要となり，ユーザにとってその知識の習得は大きな負担となる．

アノテーションは作成の際にユーザにかかる負担を少なくし，新たに知識を習得しなくても作成可能な簡単な記述方法である必要がある．つまりタグなどによるデータ構造化は最小限にし，自然言語主体のデータとして考えるのが望ましい．自然言語主体であるがゆえにアノテーションに対して自然言語処理を行うのは有用である．ただし，自然言語処理単独ではなく，データマイニング処理と複合した新しい処理形態を検討しなくてはならない．これらの処理は研究分野が異なるために今までは別々に行われている感が否めないが，お互いを融合するのが重要であると考える．

自然言語処理と構造マイニングの両方を結びつけるようなインタフェース設計のアノテーション処理ライブラリを提案する．複合的な処理により応用として考えられるサービスを支える基盤ライブラリの構築を目標とする．

2. アノテーション

アノテーションの定義と，アノテーションの内部データ構造を説明する．

2.1 定　　義

アノテーションは人間にとって記述及び理解しやすいものであるべきだと考え，自然言語表現であり，なんらかのコンテンツに対して情報を付与することをアノテーションと定義する．さらに，すべてのコンテンツは結局なんらかのアノテーションであるという概念を用いる．

2.2 アノテーションの内部データ構造

アノテーションデータを RDF で表現する．内容情報を Dublin Core [3] と Dublin Core の拡張語彙を用いて記述し，さらにそのテキストに言語情報記述に関する XML (Extensible Markup Language) タグセットである GDA (Global Document Annotation) [4] の付与を行う．これを一つのアノテーションとする．

2.2.1 内 容 情 報

内容情報記述のため，RDF のプロパティに Dublin Core を使っている．Dublin Core では任意の Web ページや文書の書誌データに付与可能な情報として，基本 15 項目を設定している．その中からリソースの内容に主たる責任を持つ人や組織記述する `dc:creator`，リソースのライフサイクルにおける主要な出来事に関連する日，一般には文書の作成日もしくは公開日を記述する `dc:date`，リソースに与えられた名前を記述する `dc:title`，リソースの内容の説明．要約，目次，文書の内容を表現した画像への参照などを自由な記述方法で記述する `dc:description` を使用する．しかし，アノテーションのグラフ構造を考えるときにこれらのプロパティだけでは子から親に対してのリンクは参照できるが，その逆は参照できない．これを解決するために，`dc:relation` の拡張語彙の一つであり，要素が示すリソースの参照，引用を記述する `dcterms:references` を使用する．`dcterms:references` により子から親だけでなく親から子に対してもリンクを参照できる．`dcterms:references` の目的語はリソースとなる．今回は上記のプロパティのみを使用するが，他のプロパティに関しては検討中であり適宜追加，もしくは独自のプロパティを定義する．

2.2.2 言語情報の付与

`dc:description` のプロパティ値として設定しようとしているテキストに対し，本ライブラリではアノテーションの生成時または，計算機の遊休時にテキストを解析器にかけ，GDA タギングしている．通常，テキストの言語的情報の抽出には，形態素あるいは構文解析器の処理を伴うため，一定の処理時間が必要となる．あらかじめ GDA タギングすることにより，自然言語処理が必要な場合，高速な処理が実現でき，自然言語処理が XML の処理に帰着される．人手によるタギングは専門知識を必要

とし，かなりのコストがかかるが，機械の自動的なタギングにより，コストを削減している．

GDA とは，産業技術総合研究所の橋田が提案する，多言語間に共通の統語・意味などに関する XML タグの標準を作って普及させようというプロジェクトである．GDA では，文法機能（主語，目的語，間接目的語），主題役割（動作主，非動作主，受益者など），修辞関係（理由，結果など）や照応関係を表せる．既に検索，要約，翻訳，対話処理，質問応答システムをはじめとした自然言語処理の分野で GDA を活用した研究報告がなされている．

本ライブラリでは Support Vector Machines に基づく日本語係り受け解析器 Cabocha [5] を用いて得られる単語の品詞や，文節間の大雑把な係り受けについて出力された情報から，機械処理によるタグ付けが比較的容易な，文章の形態素情報，構文情報，単語の読み仮名，動詞の基本形及び助詞の種別に関してタギングする．その際使用する GDA タグは，文書の全体を表す `<gda>`，文を表す `<su>`，名詞および名詞を主辞とする語句を表す `<n>`, `<np>`．なお `<n>` は他の語句の係り受け対象となることができ, `<np>` はならない．他についても同様である．形容詞，形容動詞語幹，またはそれらを主辞とする語句を表す `<aj>`，`<ajp>`，終助詞以外の助詞，副詞，連体詞，接続詞，およびこれらの（最大）投射を表す `<ad>`，`<adp>`，動詞（サ変動詞も含む），助動詞，終助詞，またはそれらを主辞とする語句を表す `<v>`, `<vp>`，固有名詞を表す `<name>`，`<namep>`，人名を表す `<persname>`，`<persnamep>`，及び地名を表す `<placename>`，`<placenamep>` である．

他に自然言語処理の前処理として，その文章内の名詞である単語の出現頻度を表す属性として `tf` を追加する．出現頻度は単語とその単語の GDA タグの両方が一致しているものを同じ単語とみなして算出している．

なお，データ自体をプロパティ値とするので，`rdf:parseType="Literal"` 属性を使用して対応する．

2.3 アノテーションのグラフ構造

すべてのアノテーションはその対象が存在し，さらにアノテーションに対するアノテーションを考えると，アノテーションを子とし，その対象を親とする親子関係が定義できる．すると，図 1 のようにアノテーション群はグラフ構造となる．例えば，Annotation1 の親は Content で，子は Annotation4 となる．Annotation1，Annotation2 及び Annotation3 は全て同一の Content に対するアノテーションなので，これらのアノテーションは兄弟アノテーションとなる．Annotation5 と Annotation6 も同様である．

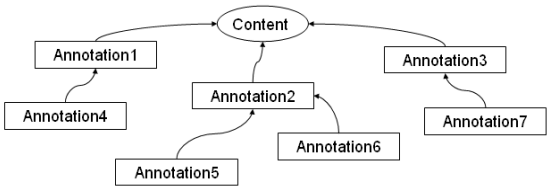

図 1　アノテーショングラフ構造

2.4 アノテーション記述例

アノテーション記述例を図 2 に示す．この例は http://www.sw.it.aoyama.ac.jp/2005/hiroyuki/hiroyuki.html に対してのアノテーションである．さらに，このアノテーション自体の URI が http://www.sw.it.aoyama.ac.jp/2005/hiroyuki/hiroyuki.rdf であり，http://www.sw.it.aoyama.ac.jp/2005/hiroyuki/data.rdf からアノテートされていることを示している．

```
<rdf:RDF
  xmlns:dcterms="http://purl.org/dc/terms/"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:gda="http://i-content.org/gda/"
  xmlns:dc="http://purl.org/dc/elements/1.1/"
 <rdf:Description
  "rdf:about="http://www.sw.it.aoyama.ac.jp/2005/hiroyuki/
  data.rdf"/>
  <dcterms:references rdf:resource=
   "http://www.sw.it.aoyama.ac.jp/2005/hiroyuki/hiroyuki.rdf">
 </rdf:Description>
 <rdf:Description
  rdf:about="http://www.sw.it.aoyama.ac.jp/2005/hiroyuki/
  hiroyuki.html">
  <dc:creator>阿部 裕行</dc:creator>
  <dc:date>2005-11-24T18:22:50+0900</dc:date>
  <dc:title>Martin 研究室 阿部 裕行 (Hiroyuki Abe)</dc:title>
  <dc:description rdf:parseType="Literal">
   <gda:gda>
    <gda:su>
     <gda:adp>
      <gda:adp>
       <gda:np bfm="構造" prn="コウゾウ" tf="1">構造</gda:np>
       <gda:ad bfm="と" prn="ト" sem="並立助詞">と</gda:ad>
      </gda:adp>
      <gda:ad>
       <gda:np bfm="自然" prn="シゼン" tf="1">自然</gda:np>
       <gda:n bfm="言語" prn="ゲンゴ" tf="1">言語</gda:n>
       <gda:ad bfm="の" prn="ノ" sem="連体化">の</gda:ad>
           :(省略)
       <gda:np bfm="実行" prn="ジッコウ" tf="1">実行</gda:np>
       <gda:v bfm="できる" prn="デキル">できる</gda:v>
      </gda:v>
      <gda:ad>
       <gda:np bfm="環境" prn="カンキョウ" tf="1">環境</gda:np>
       <gda:ad bfm="を" prn="ヲ" sem="格助詞">を</gda:ad>
      </gda:ad>
     </gda:adp>
     <gda:v>
      <gda:np bfm="構築" prn="コウチク" tf="1">構築</gda:np>
      <gda:v bfm="する" prn="スル">する</gda:v>
     </gda:v>
    </gda:su>
   </gda:gda>
  </dc:description>
 </rdf:Description>
</rdf:RDF>
```

図 2　アノテーションの例（一部省略）

3. アノテーション処理ライブラリのインタフェース設計

前章で定義したアノテーション仕様に基づく処理ライブラリを実装する．

3.1 ライブラリの構成

実装は Java で行っている．ライブラリの構成図を図 3 に示す．

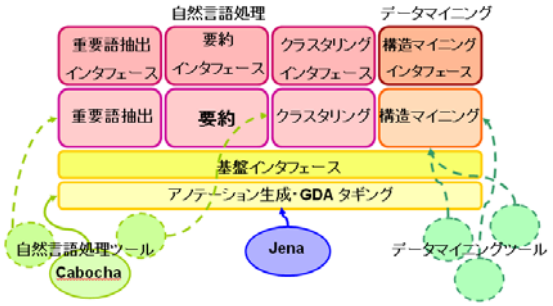

図 3　ライブラリ構成図

このライブラリでは大きく分けて 2 つのインタフェースを設計している．アノテーション自体の生成，GDA タギングを行う基盤インタフェースとアノテーションに対してそれぞれ特化された処理を行うインタフェースである．

基盤インタフェースの実装の際には，RDF の生成や解析のために Jena [6] と，GDA タギングのために Cabocha を利用している．Jena とは Semantic Web アプリケーション構築のための Java プログラミングツールキットである．

このライブラリにより，例えば，利用者が自分の手法で自然言語処理の実装を行う際に，基盤インタフェースとその処理のインタフェースに基づいて実装を行ってもらう．その際に，構造マイニングの手法を部分的に自然言語処理に取り入れることができる．構造マイニングの手法に自然言語処理の手法を取り入れるのも同様で，これにより両分野を融合することができる．さらに，それぞれツールを用いて実装を行っているということも考えられるが，このライブラリによりそれらツール群も融合させる仕組みを簡単に構築できる．

3.2 インタフェース

それぞれのインタフェースの詳細について説明する．

3.2.1 基盤インタフェース

アノテーションインタフェース (IAnnotation) は，アノテーション自体をデータオブジェクト化するためのインタフェースである．ここで定義されたメソッド群を使うことにより，アノテーションに関するさまざまな情報を取得できる．RDF の各属性値のゲッタ，セッタメソッドに加え，子アノテーションや兄弟アノテーション集合を取得するためのゲッタメソッドを定義している．図 4 の IAnnotation に定義したメソッドを示す．

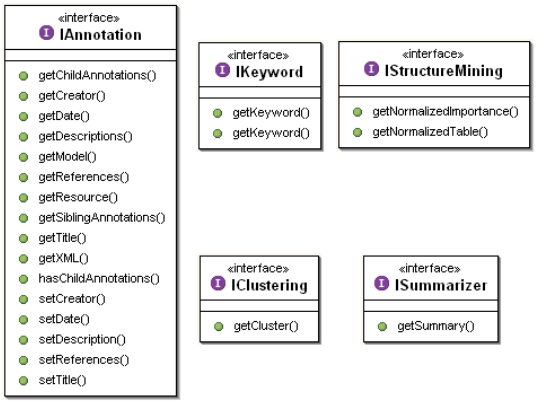

図 4　インタフェース

3.2.2 重要語抽出インタフェース

重要語抽出を行うための IKeyword インタフェースを定義している．`getKeyword()` メソッドで単体，または複数のアノテーションを引数として渡し，単体，または複数のアノテーション内の重要語を得られる．

3.2.3 要約インタフェース

要約を行う ISummarizer インタフェースを定義している．`getSummary()` メソッドで単体のアノテーションを引数として渡し，アノテーションを要約した結果を得られる．アノテーションに対する要約は，その概要だけ知りたい様な場合に有効である．

3.2.4 クラスタリング処理インタフェース

クラスタリング処理を行う IClustering インタフェースを定義している．`getCluster()` メソッドで複数のアノテーションを引数として渡し，アノテーションをクラスタリング処理した結果を得られる．アノテーションが増加すると，アノテーションをグループ分けする処理が必要だと考えられる．

3.2.5 構造マイニングインタフェース

アノテーショングラフ構造に対して構造マイニングを行う IStructureMining インタフェースを定義する．グラフ構造内から重要なアノテーションを判定できることは有用である．`getNormalizedImportance()` メソッドでアノテーションを引数として渡し，そのアノテーションの正規化された重要度を得られる．そして `getNormalizedTable()` メソッドで複数のアノテーションを引数として渡し，アノテーションと正規化された重要度からなるテーブルが `java.util.Hashtable` クラスのオブジェクトとして得ら

れる．

4. 自然言語処理

前章でも示したように，GDA タギングされたアノテーションに対して，重要語抽出，要約，クラスタリング処理の 3 種類の自然言語処理をインタフェースを定義している．しかしながらライブラリの利用を考えた場合，インタフェースだけ定義してあっても実際の処理はできないので，今回サンプル的に実装を行っている．

4.1 重要語抽出

IKeyword インタフェースに基づいて Keyword クラスを実装した．アノテーションを渡すとその中の重要語を判定し，それを抽出できる．

4.1.1 TF–IDF

今回の実装ではアノテーション内の重要語を取り出すのに，TF–IDF [7] という指標を用いている．TF–IDF は，文章から重要単語を選択する手法で，TF–IDF を使うと，ある単語の，その文章における相対的な重要性を算出できる．TF (Term Frequency) はある文書 d における単語 t の頻度を，IDF (Inverted Document Frequency) は単語が現れる相対文書頻度の逆数の対数を表している．この 2 つの積が TF–IDF となる．TF–IDF を求める式を以下に示す．

$$TF - IDF(w) = TF(w) \times IDF(w)$$

$$IDF(w) = \log \frac{N}{DF(w)}$$

ここで，単語 w の TF が $TF(w)$，IDF が $IDF(w)$．N は全体の文書数，$DF(w)$ は単語 w が出現する文書数とする．

各名詞要素の属性 `tf` に単語出現頻度を表す TF 値を算出してある．あらかじめ算出してある数値の使用で素早く重要語を抽出できる．なお，引数として単体のアノテーションを渡す場合はそのアノテーションからリンクをたどれるアノテーション全てを全体の文書とし，引数として複数のアノテーションを渡す場合は渡したアノテーション全てを全体の文書とする．次節以降の処理でも重要語抽出処理を内部的に用いている．

4.1.2 コーディング例

図 5 にコーディング例を示す．

```
Annotation data = new Annotation( "読み込む URI" );
Keyword k = new Keyword();
Element keyword = k.getKeyword( data );
```

図 5 重要語抽出のためのサンプルコード

4.2 要　　約

Isummarizer インタフェースに基づいて Summarizer クラスを実装した．Summarizer クラスでは，アノテーションを渡すとそれを要約する．以前我々が提案した重要文抽出と文内要約を併用した手法 [8] を用いて，を任意に設定した要約率で要約する．2 つの処理を同時または別々に行い要約の精度を上げる．

4.2.1 重要文抽出

重要文抽出による要約では，`<su>` 要素を一つの単位とし，それらに何らかの情報をもとに重要度を付与する．その重要度で文の順序付けを行い，重要な文要素を選択し，それらを寄せ集めて要約を作成する．本稿では重要文抽出の手法として吉見らの手法 [9] を採用している．吉見らの手法では，表題 (`dc:title`) が一番重要な文であり，重要な文とのつながりが強ければ強いほど，その文は重要であるという仮定に基づいている．

まず GDA タギングした表題内の単語の重み付けを行う．さらに，テキストを構成する文 $S_1, S_2, \cdots, S_n$ の間に冒頭文 S_1 はどの文にもつながらないで，S_1 以外の各文 S_j について，S_j が直接つながる先行文 $S_i (i < j)$ が唯一存在すると仮定する．この仮定に基づいて，表題内の単語と各文内の単語を比較し，各文の重要度，文間の関連度を求める．

以上の処理で求められたそれぞれの文の重要度から，設定された要約率をもとに重要度の高い文を抽出する．実際には重要度の低い文要素を削除している．重要度を表す式を以下に示す．

文 S_j の重要度

$$= \text{関連文 } S_i \text{の重要度} \times \text{文 } S_i \text{と文 } S_j \text{の関連度}$$

4.2.2 文 内 要 約

文内要約では野村らの手法 [10] の基本概念をもとに，GDA の情報を利用して得られるテキスト構造から各要素の非重要度を求め，要素を削除する．

各要素の非重要度の決定には，GDA テキスト構造から得られる係り受け情報を用いる．この場合まず自動生成された GDA テキストの `<su>` 要素を抽出して，同じ親要素を持つ同じ階層の要素の集合を一つの処理対象とする．非重要度を付与するには，最後の要素の非重要度を 0 とし，次にその他の要素の非重要度を決定する．その他の要素の非重要度については非重要度が 0 の要素に係るまでの距離をその要素の非重要度とする．基本的には非重要度が大きい要素の削除で決定するが，その他にいくつかの経験則を適用している．

4.2.3 コーディング例

図 6 にコーディング例を示す.

```
Annotation data = new Annotation( "読み込む URI" );
Summarizer s = new Summarizer();
Annotation summary = s.getSummary( data );
```

図 6　要約のためのサンプルコード

4.3 クラスタリング処理

IClustering インタフェースに基づいて Clustering クラスを実装した．アノテーションの集合を渡すとクラスリングされた結果を得られる．以前我々が提案した手法 [11] を用いて，アノテーション集合に対してクラスリング処理を行う．各アノテーション内に出現する単語から単語ベクトルを生成し，ベクトルの比較でクラスタの統合を繰り返し最終的なクラスタを選別する．

4.3.1 単語ベクトルの生成

クラスタリング対象のアノテーション集合からアノテーション内に出現する名詞要素（<n>，<np> など）の単語を抽出し，各アノテーションをクラスタと見なした単語ベクトルを生成する．アノテーション a_i の単語ベクトル $\vec{v}(a_i)$ は次のように定義する．

$$\vec{v}(a_i) = (w_1, w_2, \cdots, w_m) \tag{1}$$

$$w_k = TF(t_k, a_i) \tag{2}$$

ここで w_k は a_i における語 t_k の重み，$TF(t_k, a_i)$ は a_i における語 t_k の出現頻度とする．

4.3.2 クラスタの生成

各アノテーションをそれぞれ一つのクラスタとみなし，初期クラスタとして生成する，クラスタ C_i，アノテーションを a_i，アノテーションの総数を n とすると，初期クラスタは次のようになる．

$$C_1 = \{a_1\}, C_2 = \{a_2\}, \cdots, C_n = \{a_n\} \tag{3}$$

クラスタ C_i の単語ベクトル $\vec{v}(C_i)$ を，そのクラスタに含まれるアノテーションの各単語ベクトルの重心として次のよう定義する．

$$\vec{v}(C_i) = \frac{\sum_{d \in C_i} \vec{v}(a_i)}{|C_i|} \tag{4}$$

クラスタ間の類似度をコサイン類似度を用いて以下のように定義する．

$$sim(C_i, C_i) = \frac{\vec{v}(C_i) \cdot \vec{v}(C_j)}{|\vec{v}(C_i)||\vec{v}(C_j)|} \tag{5}$$

以下，クラスタの統合手順を示す．クラスタ間の類似度を比較し，最も類似度の高い 2 つのクラスタを一つのクラスタに統合する．統合するクラスタを C_i，C_j とすると，統合後のクラスタ C_{new} は以下のように定義される．

$$(i, j) = \arg_{i,j} \max sim(C_i, C_i) \tag{6}$$

クラスタの統合は，クラスタ間の最大類似度が閾値 th_{sim} より小さくなるまで繰り返し行う．

クラスタに属している各アノテーションの単語ベクトルの和を求め，求めた和ベクトルの要素の中から重みが最も大きい上位 2 つの単語を，そのクラスタのラベルとして抽出する．なお，クラスタ C_{other} のラベルは「その他」とする．

4.3.3 コーディング例

図 7 にコーディング例を示す.

```
Annotation[] graph;
        :
Clustering c = new Clustering();
ArrayList cluster = c.getCluster( graph );
```

図 7　クラスタリングのためのサンプルコード

5. 構造マイニング

アノテーショングラフ構造に対して構造マイニングを行うインタフェースも定義している．前章と同様，インタフェース定義だけでは実際の処理ができないので，活性拡散という手法 [12] を用いてサンプル実装を行う．

IStructureMining インタフェースに基づいて実装した StructureMining クラスでは，活性拡散で構造マイニングを行い，グラフ構造内の各アノテーションの重要度を得る．

5.1 活 性 拡 散

活性拡散とは，グラフ構造上のノードの重要度がわからないときにリンクの情報からノードの重要度を算出する方法である．あるグラフ構造上で，あるノード，または，全てのノードから活性拡散を行うとする．活性が拡散するとは，ある活性化したノードから，リンクで結ばれている隣のノードに活性を伝播させることを指す．この際，活性元のノードに入ってきた値を活性先のノードに加算する．n 個のノードからなるグラフに対して、どのノードからどのノードにリンクがあるかは $n \times n$ の行列（接続行列）で表現できる．この接続行列 A の各要素がリンクを表わし，その値がリンクの重みとなる．各ノードが状態を表わし，ノード x からノード y へのリンクが状態 x から状態 y への遷移確率であるとする．グラフは状態遷移の可能性を表現し，グラフが連結ならば，A^{∞} の

列はすべて同じ有限なベクトルに収束し，そのベクトルはグラフ上のランダムウォークにおける状態空間の定常確率分布を表す．ここでアノテーションがノード，ノードの存在確率がアノテーションの重要度と対応している．

t 回目の活性値を $x(t)$ とすると，活性拡散は以下の式で表される．

$$
\begin{aligned}
x(0) &= C \\
x(t+1) &= A \times x(t) \quad (t = 1, 2, \ldots) \\
x(t) &= {}^t\left[\begin{array}{ccc} x_1(t) & \cdots & x_n(t) \end{array} \right] \\
A &= \left[\begin{array}{ccc} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{array} \right]
\end{aligned}
$$

ここで，C は各ノードの初期存在確率（各アノテーションの初期重要度）を表し，各要素の和は 1 となる．$x_n(t)$ は t 回活性拡散後のノード n の活性値，a_{xy} はノード x からノード y へのリンク重みを表す．故に，活性拡散を無限回繰り返したときの収束値 X は以下のように表される．存在確率は C に依存せず収束する．

$$X = A^{\infty}C$$

最初にノードの初期値があり，リンクによって補正したい場合はそのノードに対して初期値に応じた仮想的なリンクをはることでノードに遷移する確率を補正する．仮想的なリンクのリンク重みは ノードの重要度 $\times \alpha$ で表される．この場合もノードからでているリンク重みをそれぞれ 接続されているリンク数 + 自ノード以外のノードの重要度の和 $\times \alpha$ で割り正規化する．ここで α は構造と初期値のどちらを重視するかを決めるためのパラメータである．図 8 に簡単な例を示す．

5.2 コーディング例

図 9 にコーディング例を示す．

6. ま と め

アノテーションはあくまでも人手によって作られるものであり，アノテーションの増加により高度なシステムが実現できると考えている．そのためのアノテーションを生成，処理するための基盤ライブラリを構築した．インタフェース実装だけではなくサンプルとして本体実装も行ったが，専門家にライブラリのインタフェースに沿って，実装し公開してもらうことにより，より高精度の処理性能が享受されていくことを期待している．今回提示し

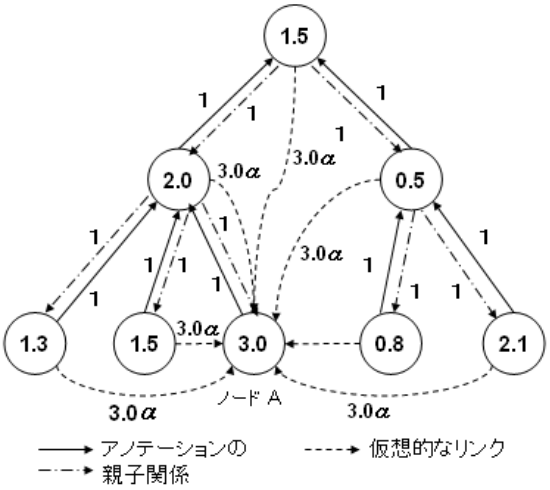

図 8　ノード A に対する仮想的なリンクの例

```
Annotation data = new Annotation( "読み込む URI" );
StructureMining sm = new StructureMining();
double value = sm.getNormalizedImportance( data );

Annotation[] graph;
          :
StructureMining sm = new StructureMining();
Hashtable table = sm.getNormalizedTable( graph );
```

図 9　構造マイニングを行うためのサンプルソース

たアノテーションライブラリを利用すれば，自然言語処理やデータマイニング等の工学的知見がなくとも，Java の基本文法さえ習得すれば，アノテーションを応用した様々なシステムの機軸となる処理部分が構築できる．これにより多くのユーザがアノテーションを自ら日常的に生成し，それによって実現される，より高度なサービスを利用するという正帰還の仕組みが確立されることを目指している．さらには，このライブラリがデータマイニングと自然言語処理分野の研究者に対しての橋渡しの役割になれば幸いである．なお，このライブラリは近日中に http://www.sw.it.aoyama.ac.jp/2005/hiroyuki/上で公開する予定である．

最後に今後の課題を述べる．今回は GDA タギングされたテキストを XML として処理をした．そもそも RDF のつくり出すモデルは XML のような階層的なものではなく，一般的な有向グラフであり，自由度の高いものとなっている．GDA タギングされた文章を RDF で表現すると同時に，現在 Semantic Web 分野で積極的に構築が進められているオントロジーと有機的に結びつけることで，処理や応用の幅が広がるであろう．同様に，今は設定できない GDA タグ，属性の付与，日本語だけではな

く他の言語へのタギングの実現も必要となるだろう.

文　　献

[1] 伊藤一成, 斎藤博昭：汎用アノテーション記述言語 MAML の提案とその生成処理プロセス，情報処理学会論文誌 TOD，Vol.45 No.SIG7(TOD22)，pp. 137–150，2004.

[2] Dave Beckett：RDF/XML Syntax Specification (Revised)，W3C Recommendation 10 February 2004.
http://www.w3.org/TR/rdf-syntax-grammar/

[3] DCMI Usage Board：DCMI Metadata Terms，2005-06-13.
http://dublincore.org/documents/dcmi-terms/

[4] 橋田浩一：Global Document Annotation (GDA)，草稿 第 0.74 版 (2005 年 10 月 17 日).
http://i-content.org/gda/tagman.html

[5] CaboCha/南瓜: Yet Another Japanese Dependency Structure Analyzer
http://chasen.org/~taku/software/cabocha/

[6] Jena Semantic Web Framework
http://jena.sourceforge.net/

[7] Salton，G.：Automatic Text Processing，Addison-Wesley，MA，1989.

[8] 伊藤一成，酒井康旭，斎藤博昭：メタデータ解析と自然言語処理を併用した動画要約，情報処理学会研究報告，DBS-132，pp. 41-48，2004.

[9] 吉見毅彦，奥西稔幸，山路孝浩，福持陽士：表題へのつながりに基づく文の重要度評価，自然言語処理，Vol.6，No.1，pp. 43–57，1999.

[10] 野村雄司，伊藤一成，斎藤博昭：GDA タグを用いたテキスト自動要約，言語処理学会第 9 回年次大会，pp. 206–209，2003.

[11] 滝本湖，伊藤一成，斎藤博昭：汎用アノテーションシステム (MAML System) を利用した Web 検索結果のグラフ表示，データベースワークショップ，DBWS2005.

[12] J. R. Anderson：A Spreading activation theory of memory，Journal of Verbal Learning and Verbal Behavior，pp. 261–295，1983.