テキストアノーテーション環境「他の盾（たてのたて）」の設計と実装

— SVG とフリーソフトウェアを用いたグラフィカルな環境 —

*NHK 放送技術研究所
**(株) 漢字情報サービス / KIS

小早川 健（こばやかわ たけし）*
kobayakawa.t-ko@nhk.or.jp

小鶴 友樹（こづる ともき）**
kozuru@strlstaff.strl.nhk.or.jp

宮崎 勝（みやざき まさる）*
miyazaki.m-fk@nhk.or.jp

山田 一郎（やまだ いちろう）*
yamada.i-hy@nhk.or.jp

浦谷 則好（うらたに のりよし）*
uratani.n-fc@nhk.or.jp

1 はじめに

文章中の主張部分を検出する研究を始めるにあたり、統計モデルの学習データを整備する環境 (ツール群) を開発したので、その設計と実装を報告する。

目指している研究は、句間の係り受けを分類したものを用いることを予定している。それには、木構造である係り受け解析結果に、係り受けの種類に応じて人間がアノーテーション (タグ付け) した学習データの整備が必要である。

本報告の動機は、係り受け解析器 cabocha[1] の出力結果である木構造データのアノーテーションや構造修正をする場面において、自然言語の構造をグラフィカルに訴えるツールが見付からなかったことによるものである。

そこで、ある程度汎用的なテキストアノーテーションツールを構築できないかという観点から XML[2] に着目した。

1.1 自然言語へのアノーテーションと XML の表現能力

自然言語の解析によって得られる構造には、1) 形態素間の接続確率 (n-gram) のような線形のもの、2) 句構造文法による構文解析や係り受け解析のような木構造のもの、3) より一般的なグラフ構造のものなどがある。

XML は、要素 (element) を用いて木構造を表現できる。また、ノード[*1] の属性 (attribute) も表現できる。しかし、ノードを結ぶ辺の属性は、XML で直接に指定する方法はない。

辺の属性を XML で表現するひとつの方法[*2] は、その辺の子ノードにその辺の属性を表す属性を追加することである。この場合、追加された属性は文法的にはノードの属性であるので、本当は辺の属性を表すという**意味**を覚えておく必要がある。

同様に、グラフ構造を XML で表現する場合にも**意味**を覚えておく必要が生じる。グラフ構造を表現する方法のひとつは、各ノードが複数の親を持ち得るというものであるが、この表現を用いる場合、いくつかの親は子ノードの属性で表すことになる。

> 例えば、係り受けの種類を分類するアノーテーションの場合、係り受けを表現する木の辺に種類があり、XML 構文木で直接に表現する事はできない。このように、自然言語へアノーテーションをする場合、XML 構文木で直接に表現される範囲を超える必要は、しばしば生じる。この超えた部分の文法は XML で表現できるが、意味を扱うのは XML アプリケーション固有の問題とされる。

以下では、意味に関知しない XML を汎用 XML、意味を扱う XML をアプリケーション専用 XML と呼ぶことにする。

1.2 汎用 XML エディタと問題点

XML の普及により、多種の汎用 XML エディタが存在している。汎用 XML エディタは XML の意味に関知しないため、XML 構文木とノードの属性を超える内容は表現しないのが通常である。

> 例えば、amaya[3] では、汎用な部分のグラフィカルな表現は XML 構文木に留まっている。もう少し正確に述べると、amaya は汎用 XML の他に XHTML や CSS や SVG なども扱う事もできるため、構文木以上の WYSIWYG レイアウト表示を行う事もできるが、自然言語処理で用いる XML を XML 構文木以上にグラフィカルに表示する事は

[*1] 要素から子要素を除いたものをノードと呼ぶことにする。これは、木構造表現で用いるノードと辺という用語と一致する。

[*2] 辺の属性を XML で表現するのに、本文での記述とは別の方法を好む筆者もいて、親ノードと子ノードの間に、辺の属性を表す新たなノードを挿入する方法が自然だと考える。この方法は、ノードの移動操作に関して扱いが容易な場合がある。しかし、やはり、辺の属性を表すノードの**意味**を覚えておく必要が生じる。

できない。

このため、汎用 XML エディタをテキストアノーテーションに用いるには、1) XML の知識のない人は、XML を勉強しなくてはならない、2) XML の知識があっても、XML 構文木と自然言語構造との対応関係に隠された (1.1節で述べた) **意味**を把握しにくい、という欠点がある。

1.3 アプリケーション専用 XML エディタと問題点

アプリケーション固有のデータを XML で記述することも盛んであり、自然言語処理研究の学習データを XML で記述すること (例えば、[4, 5]) も一般的になってきている。

自然言語を記述する XML の**意味**に応じて、専用のエディタを作成すれば、**意味**を把握しやすいテキストアノーテーションが行える。しかし、一度決めたアノーテーション規則を変更することはよくあることであり、その場合、テキストアノーテーション作業のやり直しだけではなくて、専用エディタの作り直しの必要にも迫られる。

2 設計

汎用 XML エディタとアプリケーション専用 XML エディタの中間的なものを目指し、設計では、次の点に留意した。1) 自然言語処理のアノーテーションに生じる共通の部分は再構築する事なく、個別の処理に特化した部分だけを再構築すればよい。2) 各ツール入手の利便性と経済性からフリーソフトウェアを用いる事を優先する。3) XML の予備知識のない方にもテキストアノーテーションが可能なくらいのグラフィカルな編集を行う。その結果、複数のツールからなる一連の環境が構築できた。

2.1 XML の拡張性

XML[2] は、既存の文書定義 (DTD、または、XML Schema) を拡張できるように設計されている。特に、ある特定のアプリケーション用の XML 文書定義に対して、拡張したい部分だけを記述することができる。本環境は、この性質に着目して、既存の汎用グラフィック表現 (SVG[6]) を基盤として、それに、自然言語処理固有のアノーテーションを拡張する手法を用いた。

ここで、XML 構文木と自然言語の構造をなるべく一致させようということは一旦放棄[*3] し、かわりに、自然言語の構造を SVG でグラフィカルに表現する事にした。

2.2 SVG

SVG[6] は、2 次元のベクトル、及びベクトルとラスターが混在するグラフィックスを XML で記述する為の言語である。前小節に述べたように、XML で表現されているために、グラフィック表現を基盤としてアプリケーション固有のデータを付加する事ができる。

> 係り受けの分類を例に用いると、係り受けの木構造表現には SVG を用い、係り受けの種類を表現するには、辺を表す SVG の拡張属性に値を設定することで係り受けの分類の表現が実現できる。

SVG 編集アプリケーションには、`inkscape`[7] というフリーソフトウェアが存在している。`inkscape` は、拡張された XML タグに対しても属性の設定が可能であり、本環境が生成する拡張された SVG の編集に用いることができる。そこで、`inkscape` を本環境のアノーテーションプログラムとして採用した。[*4]

2.3 構造の描画 — 木とグラフの描画

自然言語の構造は大部分が木やグラフで表現される。その描画アプリケーション [8, 9] はいくつか存在するが、ここでは、AT&T Research で開発された完成度の高い `graphviz`[8] を用いる。`graphviz` は SVG 形式を直接出力できる。

> グラフ表現へのアノーテーションは構文構造の表現には過度な機能に感じるかも知れないが、木の制限は容易に超えることがある。例えば、係り受け解析器の出力の不正解箇所を人手で訂正し、正解と不正解を両方保持しようとする場合である。こういった事情もあって、本環境は、かなり汎用的なグラフ構造も対象にしている。

2.4 アノーテーション処理の流れと非リアルタイム性

本環境で、アノーテーション規則を用いて、自然言語にアノーテーションしていく処理の流れを図 1に示す。1) 自然言語が入力され、解析されて構造を持った表現になる。2) アノーテーション規則に従って拡張 SVG 属性が付与可能な編集対象ファイルが生成される。3) `inkscape` を用いて編集を行う。4) アノーテーション済みの構造を持った解析結果が得られる。人手で付与されるデータは、編集中に XML 属性として確認することができる。

しかしながら、構造の再描画を要する確認は、編集中にリアルタイムに行うことはできない。構造の再描画の確認には、図 2に示すように、編集と保存と再描画を繰り返す「編集サイクル」を経る必要がある。

> 例えば、係り受け関係をグラフの辺で表し、辺の種類を分類するタグを付与し、その種類をグラフィカルに辺の周辺に描き入れる場合、一旦編集作業を保存し、再度 SVG を生成することによって確認する。

`inkscape` は専用 XML エディタではないため、入力された XML 属性が構造の再描画を必要とすることは理解できない。そのため、入力されたデータを一旦ファイルに保存し、構造を再描画することによって、ユーザーに

[*3] 最後に、XML 構文木と自然言語の構造がなるべく一致するようにデータ構造を復元している。(3.3節参照)

[*4] 拡張された SVG に対して XML 属性の値を設定できるアプリケーションがあれば、商用/非商用を問わず、アノーテーションに用いることが可能である。

フィードバックさせている。これは、明らかに専用 XML エディタに劣る部分である。この点については、5.2節で考察する。

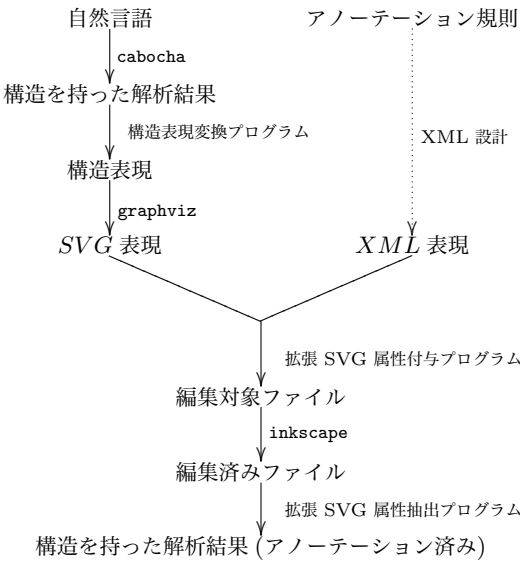

図 1　中間ファイルと各ツールの関係

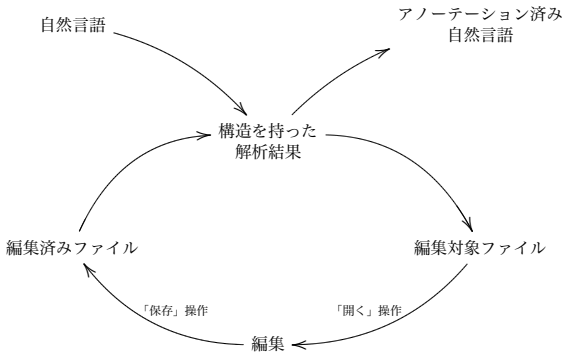

図 2　一つのファイルの編集サイクル

3　実装と Screenshot

以上の設計のもと、Debian GNU/Linux (3.1)[10]OS 上で実装を行った。本環境を構成する下記プログラムは、スクリプト言語 perl[11] と同言語を用いた XML パーサ (XML::LibXML[12]) で記述されている。cabocha の出力結果にアノーテーションする場合を例に、図 1に現れる処理の詳細を以下の小節で順に述べる。

図 2の編集サイクルにしたがって、一旦編集したファイルを処理し、inkscape から再度開いて編集している様子を図 3に示す。

3.1　構造表現変換プログラム

構造の描画に用いる graphviz は入力として dot という言語を用いている。構造表現変換プログラムは、cabocha の XML 形式の出力結果を dot に変換する。cabocha の出力する句は、グラフのノードになるように、句の係り受け関係は、グラフの辺になるように変換される。後述する拡張 SVG 属性抽出プログラムで用いるため、ノードには固有の ID が割り振られる。

3.2　拡張 SVG 属性付与プログラム

アノーテーション規則を XML で表現する方法が定めてある。拡張 SVG 属性付与プログラムは、SVG の言語定義を拡張し、空の値を持つ拡張 SVG 属性を付与する。編集作業では、空の値を適切な値で埋めていく。

3.3　拡張 SVG 属性抽出プログラム

3.1節で述べた ID と、編集前の構造を持った解析結果から、アノーテーション済みの構造を持った解析結果を抽出する。SVG 構文木の親子関係と SVG で表現されているグラフィックの親子関係は対応していないので、このプログラムで本来の XML 構造に近くなるように復元している。

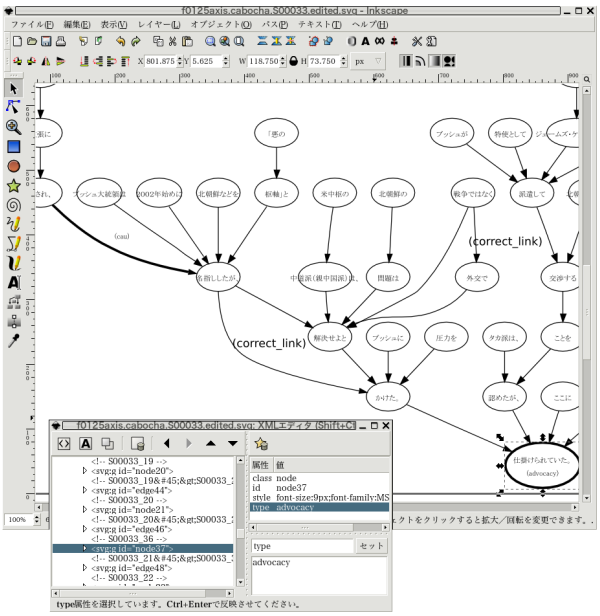

図 3　編集サイクルを通じてアノーテーションしている様子

4　他のアノーテーションに用いる場合

cabocha の出力する係り受け分類作業以外のアノーテーションに本環境を用いる場合について述べる。個別のアノーテーション作業環境に固有な部分のみを作成すればアノーテーションができるように設計したため、変更の必要のない共通部分は再利用できる。

まず、どういうアノーテーションを行うのかを決定する。それは、どういう構造にどういう情報を付与するのかを決定する作業である。

4.1 構造の決定

構造が決定されれば、構造を描画するプログラムに任せて描画すればすればよい。自然言語処理ツールの出力は、大部分が木、グラフ構造であるので、graphviz に任せて十分である。

> 単純なグラフ構造を超える一例として、句の係り受けと同時に、個々の指示代名詞などの形態素の照応をアノーテーションする場合がある。この構造は、句を表すグラフのノードの中に、句を構成する個々の形態素を表す別のグラフがサブグラフとして入れ子になっている。筆者らは、このアノーテーションに本環境を用いることを検討しているが、この場合も構造の描画は graphviz で対応できる。さらに複雑な構造を扱いたい時は、graphviz に変わる構造描画プログラムを作成する必要が生じるかも知れない。

4.2 アノーテーション規則の決定

どういう情報を付与するかという観点から、アノーテーション規則を決め、3節で述べた構造表現変換/拡張 SVG 属性付与/拡張 SVG 属性抽出の各プログラムを作成する。

5 課題と今後

以上、「他(た)の盾(たて)」(英語表記: tannotate、Text ANNOTATE の略) の設計と実装を述べてきた。現在、この環境でコーパスを作成している。この節では、実用的な観点から、この環境にさらに必要な機能と他の標準化との関係を述べる。

5.1 アノーテーション規則の記述

cabocha と graphviz の組合せでアノーテーションする場合でも、違うアノーテーションの観点から違う描画をしたいことがあり得る。よって、構造表現変換プログラムは、自然言語処理の出力形式、構造の描画プログラムの入力形式だけでなく、アノーテーション規則とも密接に結び付いていることがわかる。

現状では、表現変換/拡張 SVG 属性付与/拡張 SVG 属性抽出の各プログラムは、アノーテーション規則と一体となっている。将来的には、個別のアノーテーション規則を記述する部分と、各プログラムを分離していきたい。

このようにアノーテーション規則が別個に記述されると、規則からの自動的な文法チェックが可能になる。XML Schema では XML の文法チェックが可能であるが、それよりも詳細な、アノーテーション規則違反の入力がないかを自動的にチェックできるようになる。また、拡張 SVG 属性付与/拡張 SVG 属性抽出の各プログラムの自動生成も可能となってくる。テキストアノーテーションを行うには、研究者はアノーテーション規則の整備だけをすればよい環境の構築を最終的に目指している。

5.2 計算機科学用データ構造の XML 表現と編集のリアルタイム性

本環境は、汎用 SVG エディタを用いているため、データへのアノーテーションをグラフィカルに確認するには、一度ファイルを保存してから、グラフの再生成を行う必要があった。グラフを用いたモデリングを意図した XML データ構造 [13, 14] が提案されているが、これらの編集用フリーソフトウェアを筆者らは発見することができなかった。GraphML[14] には商用の編集ソフトウェアが存在するので、そのソフトウェアを用いて、本ツールの意図しているグラフへのアノーテーションが、リアルタイムに行える可能性がある。これは今後検討したい。また、グラフ構造は、計算機科学に用いられる基本的なデータ構造なので、編集用のフリーソフトウェアの出現も期待する。

謝辞

様々な面から支援を頂いた NHK 技研主任研究員 田中 英輝博士、NHK 技研人間情報科学副部長 比留間 伸行 博士に感謝します。NHK 技研研究員 熊野 正さんには、本環境作成のきっかけとなった議論を頂いたこと、および、様々な面からの支援を頂いた事に感謝します。

付録 A SVG エディタ

本環境の一部として利用できる可能性を探ったアプリケーションのうちフリーソフトウェアであるもの紹介する。それぞれ、わずかな問題を含んでいたために、本環境の一部として採択しなかった。広範囲な動作環境や利用可能性を網羅した調査ではなく、あくまで参考としての記述である。

amaya[3] w3c の xml ページ [2] からリンクが張られているフリーソフトウェア。SVG を編集できるものの、グルーピングされた SVG オブジェクトに対して、テキストの表示位置がずれてしまう。

sodipodi[15] inkscape と根源を同じにしているフリーソフトウェア。SVG データのグルーピングを解除した後に、日本語の表示が化けてしまう。

参考文献

[1] “ CaboCha/南瓜: Yet Another Japanese Dependency Structure Analyzer 0.52” (2004). http://chasen.org/~taku/software/cabocha/.
[2] “Extensible Markup Language (XML) 1.0 (Third Edition)” (2004). http://www.w3.org/XML/.
[3] “ amaya 9.2.2 ”. http://www.w3.org/Amaya/.
[4] 前川, 菊池 : “大規模音声データベース「日本語話し言葉コーパス」の概要とその利用法”, 日本音声学会第 11 回音声学セミナー (2004).
[5] 菊池 : “XML 文書とその利用法”, 日本音声学会第 11 回音声学セミナー (2004).
[6] “Scalable Vector Graphics (SVG)”. http://www.w3.org/Graphics/SVG/.
[7] “ inkscape 0.43 ”. http://www.inkscape.org/.
[8] “ graphviz 2.6 ”. http://www.graphviz.org/.
[9] http://openjgraph.sourceforge.net/.
[10] http://www.debian.org/.
[11] “ Perl 5.8.7 ”. http://www.perl.org.
[12] “ XML::LibXML 1.58 ”. http://search.cpan.org/dist/XML-LibXML/.
[13] “ XGMML (eXtensible Graph Markup and Modeling Language) ”. http://www.cs.rpi.edu/~puninj/XGMML/.
[14] “ graphML (graph Markup Language)”. http://graphml.graphdrawing.org/.
[15] “ Sodipodi 0.34 ”. http://sourceforge.net/projects/sodipodi.