Kachako におけるコーパスアノテーションの重ね合わせ

狩野 芳伸（科学技術振興機構さきがけ）†

Layering BCCWJ Corpus Annotations in Kachako

Yoshinobu Kano (PRESTO, Japan Science and Technology Agency)

１．はじめに

テキストコーパスの利用にあたっては、テキスト本文の利用のみならず、そこに付与されたアノテーションの利用が重要である。日本語コーパスについては、国立国語研究所を中心に開発されてきた大規模日本語均衡コーパス BCCWJ(前川, 2009)が現在最も大規模なものである。BCCWJ には、国語研と個々の協力研究グループによって形態素・係り受け(浅原ほか, 2011)・述語項構造(小町ほか, 2011)・フレーム構造(小原ほか, 2011)・動詞項構造(竹内ほか, 2012)など様々なアノテーションが付与されてきており、アノテーションという点でも最大規模と考えられる。BCCWJ のアノテーション部分については現在も開発が進められている。本稿では執筆時点の情報に基づいて記述する。

同一テキストに対して異なる種類のアノテーションが重層的に付与されている場合、利用者としてはこれらのアノテーションを統一的に扱えれば理想的である。もっといえば、アノテーションを利用する労力が少ないほどよい。しかし実際には、アノテーションの開発は個々の担当者によって独立して行われ、フォーマットだけでなく基にするコーパステキスト文字列の差異なども存在する。そのためアノテーションの統一的利用にはかなりの労力を必要とするのが現状である。利用者からすれば、データが手元にあり内容も把握しているのに、利用者にとって本質的でない部分で時間を使わねばならないのは時間の浪費である。

本稿では、これらの問題点を解決すべく、異なる種類のアノテーションの重ね合わせを行った我々の研究について報告する。我々は、自然言語処理ツールの利用を自動化するための統合プラットフォーム Kachako を開発してきた。プラットフォーム上では、各種ツールが互換化されて統合されており、ツールの組み合わせや実行、評価が容易に行える。コーパスアノテーションを読み込むリーダーについてもいくつか統合されている。本研究では、Kachako 上で動作するコーパスリーダーの一種として BCCWJ Reader を実装した。

第２節では本研究の背景となった既存研究について述べる。第３節ではアノテーションの重ね合わせの設計とシステムについて述べ、第４節で Kachako システムを通じた応用の可能性を議論し、第５節で将来の展望や応用に触れつつ締めくくる。

２．背景

２.１ 大規模日本語均衡コーパス BCCWJ

国立国語研究所で開発された BCCWJ は日本語において最大規模の均衡コーパスである。テキストは新聞・小説・ブログなどさまざまなジャンルのものからランダムに選択されたものが収録されている。

BCCWJ の一部データに対ついては、テキストに対しアノテーション情報が付与されている。どのアノテーションがどの部分に付与されているかはアノテーションによって異なり、重複する部分もあればしない部分もある。

BCCWJ の利用には国語研究所との間で契約が必要で、契約者にはデータを収録した DVD が配布されている。現在配布されている DVD には、テキスト情報に加え形態素レベルの情報が収録されている。形態素は短単位（SUW）と長単位（LUW）が別個に付与されている。これらの情報は、基本的に XML 形式で記述されている。本稿では BCCWJ の詳細

† kano@nii.ac.jp

については記述しない。

それ以外のアノテーションは、DVD には現在収録されていない。本研究では国語研が別途配布している係り受けアノテーション DepPara2[1]・奈良科学技術先端大学院大学を中心に開発された述語項構造(小町ほか, 2011)・慶応大学の開発した日本語フレーム構造(小原ほか., 2011)・岡山大学の開発した動詞項構造(竹内ほか, 2012)に対応した。それぞれのアノテーションの詳細は３節で詳述する。

２.２ 国際標準 UIMA フレームワーク

UIMA (Unstructured Information Management Architecture)(Ferrucci et al., 2006)は様々な企業・研究機関で利用されている相互運用性のための枠組みである。UIMA はメタデータとそのための API を提供しており、実装は Apache　UIMA としてオープンソースで提供されている。最近では、テレビ番組のクイズ王に勝利した IBM の Watson システム(Ferrucci, 2012)でも用いられた。

UIMA の実行単位はコンポーネントと呼ばれ、コンポーネントを組み合わせて実行可能な UIMA ワークフローを作成する。おおむね、コンポーネントはいわゆるツールに相当し、ワークフローはアプリケーションに相当する。実行時のデータ構造は CAS と呼ばれる汎用構造に統一されている。XML のようなインライン形式と異なり、CAS ではスタンドオフ形式を前提としている。スタンドオフ形式ではテキストとアノテーションが分離され、アノテーションはテキスト内のオフセット位置を参照することでテキストに紐づけされる。CAS は概念的な構造であり、実行時は Java ヒープ内にオンメモリで格納されるが、XMI(XML Metadata Interchange)などファイル形式で保存することもできる。

UIMA は他に、データ型を階層的に定義する type system を記述する XML 形式やコンポーネントのメタデータ記述の XML 形式などを提供している。UIMA のアノテーションは type system により定義済みのデータ型で型付けされなければならない。UIMA 自体は整数や文字列といった基本的な型しか提供しないため、開発者が必要に応じてアノテーションの型階層を定義する必要がある。型階層は木構造で、型には素性と呼ぶ属性値を定義することができる。また、コンポーネントそのものも基本的に個々の開発者が作成して提供することになっており、様々な研究グループ・企業からコンポーネントが提供されている。

ウェブサービスの形式としては、ウェブサービスコンテナである Apache ActiveMQ 上で展開される UIMA-AS サービスが用意されている。

２.３ 統合自動自然言語処理システム Kachako

Kachako は UIMA 準拠のプラットフォームと互換ツールキット（UIMA コンポーネント群）からなる、統合全自動言語処理システムである(狩野, 2012a)。Kachako プラットフォームは、ツールの選択・組合せ・並列分散展開実行・視覚化・評価までを徹底的に自動化している。ツールキットでは統一 type system を定義した上で、その type system に互換な UIMA コンポーネント群を構築して、プラットフォームと互換ツール群をポータブルな形で配布し、ユーザの指定した任意の計算資源上でインストールから大規模処理まで自動実行することを可能にしている。

Kachako の特徴の一つは、自動ワークフロー生成機能である。Kachako はコンポーネント群から可能なワークフローを自動計算するためにコンポーネントの入出力データ型情報を用いる。そのためデータ型の定義は自動計算が可能なように設計すると同時に、コンポーネントの入出力条件を過不足なく表現できるようにする必要がある。

`Morpheme`（形態素）, `Segment`（文節等）, `Dependency`（係り受け）など本稿の対象とする基本的なデータ型はすでに Kachako で定義されており、これを利用・拡張することで Kachako の既存コンポーネントと互換にできる。なお、本稿において UIMA の型名や属性名は `TypeName` のような `Courier` フォントを用いて記す。データ型定義の詳細は３節で述べる。

---

[1] https://sites.google.com/site/masayua/bccwjdep

３．アノテーションの重ね合わせ

本節では、BCCWJ に付与されている各種のアノテーションを重ね合わせるためのデータ型階層の設計（３．１）、テキストと文字位置の正規化（３．２）、それを用いた UIMA 準拠・Kachako 互換の BCCWJ コーパスリーダーの実装（３．３）について述べる。

３.１ データ型階層の設計

３.１.１ BCCWJ DVD 内の文書構造および形態素アノテーション

BCCWJ では、文書の構造を表すため以下のような階層構造を用いている：

<article articleID="OW6X_00000_V001" isWholeArticle="false">
　　<titleBlock>
　　<title>
日本文化発信による国際文化交流(実際にはテキストアノテーションが付与されている)
　　</title>
　　</titleBlock>
　　<cluster>
　　<titleBlock>
　　<title>
文化庁文化交流使事業(同上)
　　</title>
　　</titleBlock>
　　<paragraph>
（本文テキストおよびアノテーション）
　　:

このような文書構造タグ対応して UIMA のデータ型階層を設計した。データ型は共通して `DocumentClassAnnotation` から派生した型を使用した。`DocumentClassAnnotation` は Kachako 標準のトップレベルの文書構造アノテーション型である。また、各タイプは共通して `begin, end` の素性を持っており、それぞれ BCCWJ のデータ中のタグの表層文字列への開始位置、終了位置を格納している。`begin, end` は UIMA 全体で共有されている汎用アノテーション型の `Annotation` 型で定義された素性である。BCCWJ における文書構造タグの入れ子関係は、これらの位置情報により判断できる。`begin, end` はほとんどのアノテーションに共通のため、以下のデータ型の説明では説明を省略する。

BCCWJ における文書構造は(a)サンプル、(b)階層構造、(c)図表、(d)引用、(e)注記、(f)その他１、(g)その他２、(h)文字・表記、(i)可変長タグセットに無いタグ、の各タグにより表現されている。表 1 に各タグに対応して設計した Kachako 互換の UIMA データ型を示す。BCCWJ

表 1 文章構造タグに対応する BCCWJ のタグ名

BCCWJ の各タグ名に `org.kachako.types.japanese.document.` の接頭辞を付与したものを Kachako のデータ型名とし、対応する属性値も素性値として保持する。

sample, sampling, type, article, blockEnd, cluster, titleBlock, list, paragraph, sentence, figureBlock, figure, caption, table, quotation, citation, source, speech, speaker, noteBody, noteBodyInline
abstract, authorsData, contents, profile, rejectedBlock, verse, verseLine, ruby, correction, missingCharacter, enclosedCharacter, cursive, image, superscript, subscript, fraction, delete
br, info, rejectedSpan, substitution, denominator, div, listItem, noteMarker, mergedSample, numerator, orphanedTitle, superSentence, supplement, title, vinculum, webBr, webLine

表 2　SUW（左）および LUW（右、SUW との差分のみ）に対応するデータ型

SUW（短単位）タグ	LUW（長単位）タグ
BCCWJ 属性値 start, end, pos, cType, cForm, lForm, lemma, pron, pronBase, orthBase, orth, goshu, iType, iForm, fType, fForm Kachako データ型: org.kachako.types.japanese.syntactic.morphCeme.SUW 素性 begin --- "start"の値を 10 で割った値を入れる end --- "end"の値を 10 で割った値を入れる pos --- "pos"の値を'-'で分割した 1 番目の要素 detailedPos --- "pos"の値を'-'で分割した 2 番目の要素 posList[0] --- "pos"の値を'-'で分割した 3 番目の要素 posList[1] --- "pos"の値を'-'で分割した 4 番目の要素 conjugateType --- "cType"の内容を格納する surfaceForm --- "orthBase"の内容を格納する readings[0] --- "lForm"の内容を格納する baseForm --- "formBase"の内容を格納する conjugateForm --- "cForm"の内容を格納する pronunciations[0] --- "pron"の内容を格納する pronunciations[1] --- "pronBase"の内容を格納する semanticInformations --- その他の情報を纏めて格納する elements --- 子要素となるタグ情報を保持する	BCCWJ 属性値 B, SL, l_lemma, l_lForm, l_cType, l_cForm, l_pos, l_orthBase Kachako データ型: org.kachako.types.japanese.syntactic.morpheme.LUW 素性 begin --- documentText の開始位置 end --- documentText の終了位置 B --- "B" 'B'=文節境界、'S'=文境界のいずれか SL --- "SL" baseForm --- "l_lema" readings[0] --- "l_Form" conjugateType --- "l_cType" conjugateForm --- "l_cForm" pos --- "l_pos"の値を'-'で分割した 1 番目の要素 detailedPos --- "l_pos"の値を'-'で分割した 2 番目の要素 posList[0] --- "l_pos"の値を'-'で分割した 3 番目の要素 posList[1] --- "l_pos"の値を'-'で分割した 4 番目の要素 surfaceForm --- "l_orthBase" elements --- 子要素を保持する

の各タグ名に org.kachako.types.japanese.bccwj.document.の接頭辞を付与したものを Kachako のデータ型名とし、対応する属性値も素性として定義した。

短単位形態素情報（SUW タグ）の情報は Morpheme の子として定義した SUW に保持する（表 2 左）。Morpheme は形態素一般のデータ型として定義したため、SUW タグの属性に対応する素性の定義が無いものは、semanticInformation 素性にまとめて格納している。

長単位形態素情報（LUW タグ）の情報は、LUW 型に保持する。構造的には SUW と同様であるが、一部属性の項目が異なっているほか、文やサンプル長の属性が追加されている（表 2 右）。

３.１.２ DepPara2 係り受けアノテーション

DepPara2 は DVD が利用可能であることを前提に、DVD データ相当の拡張 CaboCha 形式データへの差分パッチという形式で追加アノテーションを配布している。DepPara2 では単に係り受けを追加するだけでなく、DVD 内の一部アノテーションに対して修正が行われる内容になっている。また、差分とすることで本文テキストのデータが含まれないため、著作権の問題を回避し DepPara2 データのみを一般に無料配布している。

DepPara2 はアノテーションツール茶器(松本ほか, 2010)等で用いられている拡張 CaboCha 形式に基づいた表記になっている。拡張 CaboCha 形式では形態素および係り受け情報を独自のフォーマットで表現する。DepPara2 では茶器にインポートできるよう、以下の a),b)を実行する Ruby スクリプトを提供している(図 1)。

a)　DVD 内の XML ファイルを拡張 CaboCha 形式に変換。係り受け部分はダミーを挿入

b)　拡張 CaboCha 形式間の差分 patch により、a)のダミー係り受け情報を実データに置換
これに対し、我々は実行効率を考慮して patch ファイルと DVD の XML ファイルから直接 Kachako 互換の UIMA 形式への変換を実装し、BCCWJ Reader 内で実行するようにした。ユーザは BCCWJ Reader のオプションに patch ファイルの位置を指定するだけでよい。

DepPara2 における文境界 Sentence は、BCCWJ DVD のものとは異なる可能性があり、どちらが正しいというものでもないため、双方を残しておくのが妥当である。そこで BCCWJ DepPara2 の処理後の文境界情報は JapaneseSentence に保持するようにした。保持する情報は DepPara2 処理後の開始位置と終了位置のみである。

BCCWJ DVD では sentence タグにより文が区切られ EOS が追加されるが、DepPara2 の処理時は DepPara2 のデータで EOS 情報を付与、削除、移動し、その情報を元に Kachako 標準の日本語文境界を表すデータ型 JapaneseSentence を作成している。

表 3　DepPara2 および拡張 CaboCha 形式に対応するデータ型

	Kachako データ型名	素性の説明
DepPara2 独自文境界	org.kachako.types.japanese.syntactic.JapaneseSentence	begin, end
DepPara2 独自文節境界	org.kachako.types.japanese.syntactic.CabochaSegment	begin, end
文節係り受け関係（ラベル付き）	org.kachako.types.japanese.syntactic.Dependency	begin --- 係り元の文節の開始位置 end --- 係り元の文節の終了位置 source --- 係り元の文節 target --- 係り先の文節 label --- ラベル
セグメント情報	org.kachako.types.japanese.syntactic.CabochaSegment	label --- ラベル begin --- 開始位置 end --- 終了位置
グループ情報	org.kachako.types.japanese.syntactic.CabochaGroup	label --- ラベル begin --- 開始位置 end --- 終了位置
リンク情報	org.kachako.types.japanese.syntactic.CabochaLink	label --- ラベル begin --- 開始位置 end --- 終了位置 source --- 開始セグメントの ID target --- 終了セグメントの ID

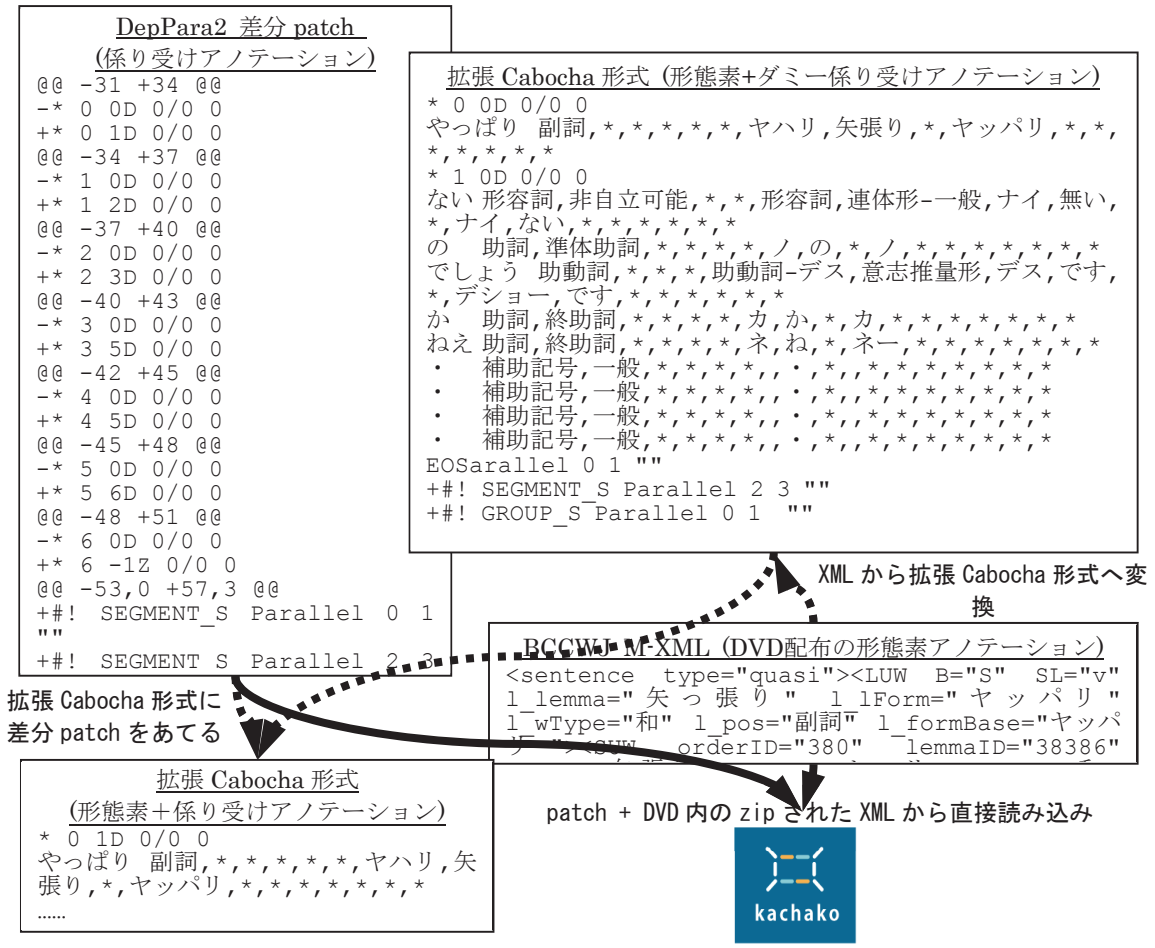

図１　DepPara2 のデータ変換の流れ。点線は DepPara2 提供のスクリプトで拡張 Cabocha 形式を生成する場合、実線は Kachako の BCCWJ Reader 内部での変換の流れを示す。

BCCWJ DepPara2 での文節情報は、CabochaSegment に保持する。BCCWJ DepPara2 では使用していないが、信頼値が入っている場合には Kachako 標準のデータ型である AnnotationMetadata に保存する。

DepPara2 の patch データが無い場合の文節境界は、BCCWJ DVD データの LUW タグの"B"の値に"S"あるいは"B"が入っている場合、そこで文節境界として決定している。

文節係り受け情報は、Kachako の汎用係り受け関係用データ型である Dependency に保持する。セグメント情報(SEGMENT_S)は org.kachako.types.japanese.syntactic.ExtraSegment に保持する。グループの情報(GROUP_S)は、CabochaGroup に保持する。またリンク情報(LINK_S)については、CabochaLink に保持する。これらのデータ型は、茶器等で用いられている（拡張）CaboCha 形式を表現するために定義された汎用データ型である。表 3 に各データ型の一覧を示す。

３.１.３ 述語項構造および並列関係アノテーション

述語項構造アノテーションは、BCCWJ 内のいくつかの文書について付与されている。述語を Pred 型、名詞句を Np 型とし、ガ格・ヲ格・ニ格の項構造のリンクを保持するようにした(図 2)。また、並列関係を Coref 型とし、並列関係のリンクも保持するようにした(図 3)。

３.１.４ 日本語フレームネットアノテーション

日本語フレームネットアノテーションは、BCCWJ 内のいくつかの文書の先頭部分にのみ付与されている。これは旧版の BCCWJ DVD に対し付与されたものを引き継いでいるた

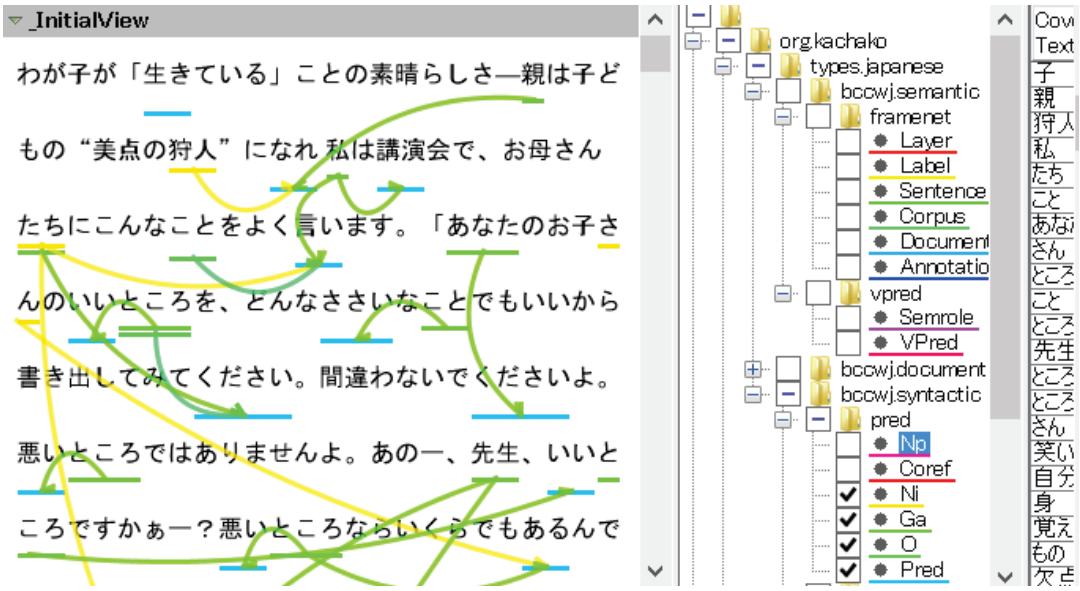

図 2　述語項構造を Kachako の概観ビューで表示した例。水色が述語、黄色がニ格、濃緑がヲ格、緑がガ格に対応している。右側には表示させたいアノテーション種を選択するリストがあり、任意の種類のアノテーションを同時に表示させることができる。

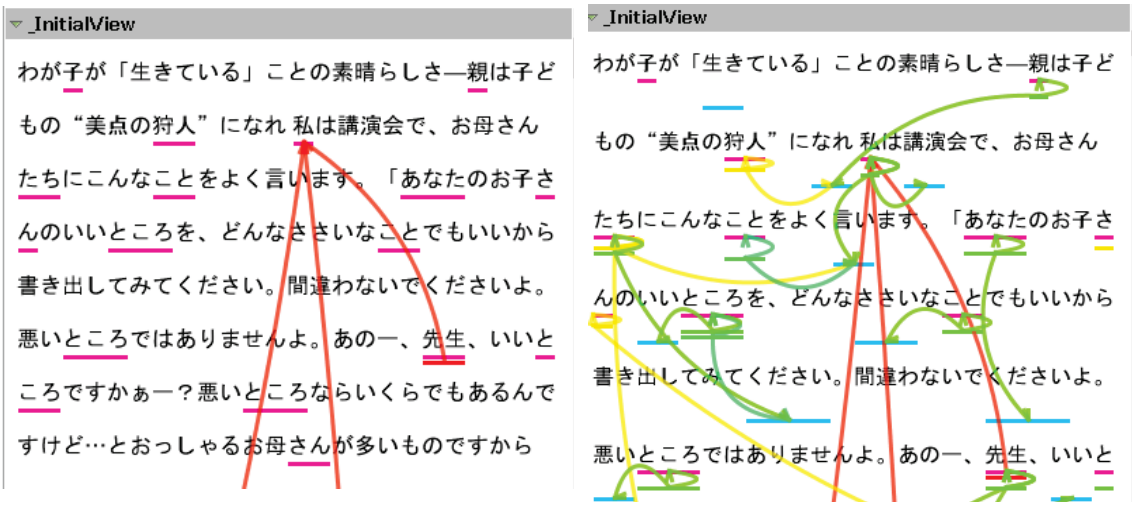

図 3　並列構造を Kachako の概観ビューで表示した例。赤が並列関係を示している。左は並列構造のみ、右は述語項構造を同時に表示した例。

め、そのままではテキスト位置が一致しない。BCCWJ Reader ではテキストが一致しなくても文字数が一致する場合はそのまま読み込んで重ねるようにし、文字数が一致しないデータは読み込まないものとした。

日本語フレームネットでは `Sentence`、`AnnotationSet`、`Layer`、`Label` の順に、親から子に入れ子になっており、親はそれぞれ複数の子を保持しうる。また、属性値としてフレーム名や役割名などが格納されている。属性値の意味については、アノテーション開発者の提供する情報を参照する必要があり、Kachako では単に文字列として扱っている。データ型には `org.kachako.types.japanese.bccwj.document.semantic.framenet` の接頭辞を付与した。

３.１.５ 動詞項構造アノテーション

動詞項構造のアノテーションは、BCCWJ 内のいくつかの文書のうち、一部の動詞について付与されている。アノテーションは改行・タブ区切り形式で保存されており、各行に中心となる動詞と動詞のとる項構造のリストが繰り返し記述され、行内には属性値がタブ区切りで格納されている。動詞を `VPred`、項構造を `Semrole` として定義し、`VPred` から対応する `Semrole` にリンクを張るようにした。これらのデータ型には接頭辞として `org.kachako.types.japanese.bccwj.document.semantic.vpred` を付与した。

３.２ テキスト位置の処理

BCCWJ の形態素単位にはいくつか種類があり、アノテーションの重ね合わせのためには基になるテキスト上でのアノテーションの開始位置、終了位置の計算を適切に行いテキストが重なるようにする必要がある。

SUW 以下のアノテーション位置については、BCCWJ のデータの開始位置・終了位置は、基本的には SUW のタグにある start/end の値を元に計算を行っているが、SUW より下位のタグについては開始位置・終了位置が無いため、XML データ中のテキストの長さから開始位置・終了位置を求めている。SUW の下位タグとしてオフセット計算を行っているものは correction, ruby, subscript, superScript の 4 つのタグである。

BCCWJ では、アラビア数字による数値の表層文字列を漢数字に置き換えたうえで、もともとの表記を NumTrans タグとして追記している。NumTrans タグの情報は `org.kachako.types.japanese.document.NumTrans` に保持する。NumTrans は元のタグ同様 `originalText` 素性を持つ。

BCCWJ の NumTrans タグには現在不具合があり、SUW より上位のタグであっても NumTrans タグのタグ以下では start/end の値に正しくない値が入っているため、タグ開始位置として使用すると不正な値になってしまう。このため NumTrans より下にある SUW は、start/end の値は使用せず、実際の XML 中のテキスト要素の文字列の長さを元に UIMA 側アノテーションの `begin/end` の値を設定している。また、NumTrans のタグより後では、誤った値が SUW の start/end に入っているため、以降のアノテーションに対応する `begin/end` には、この NumTrans タグで誤った値の差分を計算して設定している。

３.３ BCCWJ Reader コンポーネントの設定と処理

コーパスとアノテーションを読み込み、上記のデータ型階層互換かつ UIMA の形式に変換するツールは、BCCWJ Reader という名称の Kachako 互換 UIMA コンポーネントとして実装した。単にデータ型互換というだけでなく、Java 実装とパッケージングにより自動インストール・自動実行に対応したコンポーネントになっており、Java 7 の稼働するマシンであればどこでも自動実行できる。表 4 に BCCWJ データの読み込みと処理にかかる時間を計測した結果を示す。ここで行った処理には、zip ファイルの展開・XML 解析・各種重ね合わせ処理と UIMA 形式への変換が含まれる。

BCCWJ Reader の実行時オプションは以下のとおりである。

まず、`Directory` に BCCWJ DVD あるいはそのコピーの格納されている先頭ディレクトリを指定する。DVD データについては、ユーザ指定が必須のオプションはこれだけである。BCCWJ Reader は実行時に `Directory` 下の M-XML 以下のフォルダに含まれる zip 形式で圧縮されたデータをその場で展開しつつ読み込む。

`InputEncoding` はデフォルトで UTF-8 である。DVD で配布されているデータは UTF-8 形式のため通常は変更の必要がない。

DepPara2 のデータを読み込みたい場合は、以下のオプション群を適宜指定する。`AddDepPara2Data` は DepPara2 で使用する `CabochaSegment` 等の各種データをとりこむか否かを指定する。`DepPara2PatchFile` には DepPara2 サイトで配布されている patch 形式のデータを指定する。`OutputOnlyDepPara2Data` には BCCWJ DepPara2 の patch ファイルに patch 情報が存在するデータのみを対象にするか否かを指定する。この項目がチェックされていない場合、BCCWJ DVD にあるデータで DepPara2 の patch 情報が無い

ものは BCCWJ DVD 情報から生成した CabochaSegment 情報を出力する。

述語項構造・日本語フレームネット・動詞項構造などのアノテーションを読み込む場合も、同様にアノテーションを記述したファイルの指定が必要である。なおこれらのアノテーションは現時点では一般に公開されておらず、今後詳細が変更される可能性もあるため、本稿では詳細には触れない。

以下は Kachako システムで実行するコンポーネントに共通で設定可能なパラメータである。KachakoSystem_Vmargs には実行時に Java VM に渡すパラメータを指定できる。デフォルトは-Xss16m (スタックメモリの使用値を 16MB に指定)である。KachakoSystem_IvyUrls には BCCWJ Reader 実行時にコンポーネント実行に必要なファイルをダウンロードする URL が指定されている。デフォルトは Kachako の配布サイトになっており変更の必要はない。

表 4 実行時間の計測結果

データ	ファイルサイズ	処理時間(分)	平均処理速度(KB/sec)
LB.zip	1.31GB	366	357.9
OB.zip	163MB	47	346.8
OC.zip	546MB	132	413.6
OL.zip	35.9MB	13	276.1
OM.zip	198MB	463	42.7
OP.zip	159MB	109	145.8
OT.zip	39.3MB	12	327.5
OV.zip	10MB	2	500.0
OW.zip	191.1MB	58	329.4
OY.zip	525.7MB	156	336.9
PB.zip	1.22GB	357	341.7
PM.zip	202.7MB	56	361.9
PN.zip	61.8MB	11	561.8
合計	4.66GB	1782	261.5

４．Kachako システムを通じた応用の可能性

Kachako プラットフォームは現在開発を継続しており、視覚化機能以外にも多様な機能を提供する予定である。Kachako の中核的な機能は、ツールの組合せと自動実行である。自動実行では任意のマシンに対しリモート自動インストール・自動実行可能なうえ、Kachako の Hadoop-UIMA ブリッジ機構と Java・Hadoop 全自動インストール機構により大規模並列処理も自動実行できる。待ち受けウェブサービスとして自動展開することも可能である。Kachako プラットフォーム側で機能が追加されれば BCCWJ Reader をはじめとするコンポーネント側では何も変更しなくとも新たな機能が利用できるようになる。これは標準化・互換化を考慮して実装を行ったことの大きな利点といえる。

そうした対応ツール、すなわち Kachako に互換化統合済みで利用可能なコンポーネントは、現在のところ日本語・英語のツールを中心に多数実装済である。応用的なツールとしては、Javelin および Minerva の二つの日本語質問応答システムをコンポーネント分割し互換化実装した(狩野, 2012b)。ほかにも互換コンポーネント化実装が進行中・テスト中の対応ツールとして、音声認識の Sphinx-4(Walker et al., 2004)や NTCIR MedNLP の医療言語情報処理のもの(狩野, 2013)などがある。これらすべてのツールが全自動で組み合わせ・実行・結果の重ね合わせが可能であり、そこに BCCWJ アノテーションが加わったことによるメリットは大きいと考えられる。

５．おわりに

本稿では、BCCWJ に付与されたアノテーションを重ね合わせ、Kachako との標準化・互換化により統一的に扱えるようする設計と実装について述べた。重ね合わせのもっとも直接的な利用例として、Kachako の汎用視覚化機能による表示を紹介した。Kachako の他の機能や、対応ツール群との組み合わせにより、さまざまな応用が期待できる。

将来の課題として、今後各研究グループから公開される予定の、未対応の種類のアノテーションへの対応があげられる。アノテーションは随時更新される可能性があり、配布形態も今後決まるものと思われるので、適宜対応が必要である。また、テキストと重層的なアノテ

ーションを同時に検索できるような検索機能の実装を検討したいと考えている。

謝　辞

本研究の一部は、国立国語研究所共同研究プロジェクト「コーパスアノテーションの基礎研究」（プロジェクトリーダー：前川喜久雄）および科学技術振興機構さきがけ「情報環境と人」領域「解析過程と応用を重視した再利用が容易な言語処理の実現」（研究代表者：狩野芳伸）による補助を得ています。

文　献

Ferrucci, D. (2012). Introduction to “This is Watson”. *IBM Journal of Research and Development*, *56*(3.4), 1:1–1:15. doi:10.1147/JRD.2012.2184356

Ferrucci, D., Lally, A., Gruhl, D., Epstein, E., Schor, M., Murdock, J. W., Frenkiel, A., et al. (2006). Towards an Interoperability Standard for Text and Multi-Modal Analytics. IBM Research Report.

Walker, W., Lamere, P., Kwok, P., Raj, B., Singh, R., Gouvea, E., Wolf, P., et al. (2004). *Sphinx-4: A Flexible Open Source Framework for Speech Recognition*. Mountain View, CA, USA: Sun Microsystems, Inc.

前川喜久雄. (2009). 代表性を有する大規模日本語書き言葉コーパスの構築. 人工知能学会誌, *24*(5), pp.616–622.

小原京子, 加藤淳也, 斎藤博昭. (2011). 日本語フレームネットにおける BCCWJ への意味アノテーション. *日本語コーパス平成 22 年度公開ワークショップ* (pp. 513–518).

小町守, 飯田龍. (2011). BCCWJ に対する述語項構造と照応関係のアノテーション. *日本語コーパス平成 22 年度公開ワークショップ* (pp. 325–330).

松本裕治, 浅原正幸, 岩立将和, 森田敏生. (2010). 形態素・係り受け解析済みコーパス管理・検索ツール「茶器」. 情報処理学会研究報告. 自然言語処理研究会報告, *2010*(18), pp.1–6.

浅原正幸, 岩立将和, 松本裕治. (2011). BCCWJ コアデータへの係り受け・並列構造のアノテーション. *日本語コーパス平成 22 年度公開ワークショップ*. pp. 317–324.

狩野芳伸. (2012a). Kachako: 誰でも使える全自動自然言語処理プラットホーム. *2012 年度人工知能学会全国大会（第 26 回）*.

狩野芳伸. (2012b). 統合研究基盤：質問応答システムの互換コンポーネント化による再利用性向上と開発自動化支援(<特集>ロボットは東大に入れるか?). 人工知能学会誌, *27*(5), pp.492–495.

狩野芳伸. (2013). 医療言語処理ツールとコーパスの互換化: Kachako をベースとした大規模処理に向けて. *言語処理学会第 19 回年次大会* (pp. 81–82).

竹内孔一, 竹内奈央, & 石原靖弘. (2012). 述語の分析に基づく文書解析の考察. *IPSJ SIG Notes* (Vol. 2012, pp. 1–7). Information Processing Society of Japan (IPSJ).