フランス語機械可読辞書とメタ言語

中尾　浩

1：はじめに

デジタル化された言語データを処理するために，機械可読辞書(Machine Readable Dictionary)（以下，MRDと略すこともあり）の開発は不可欠である。言うまでもなく言語の数だけMRDは必要である。さらに，利用目的次第で複数のMRDが必要だろう。構文解析の利用を前提としたMRDが自動翻訳のMRDとしてもふさわしいかどうかは設計次第であったり，使い方次第の面もあるが，一般には別のMRDを設計する方が妥当である。

機械可読辞書を何らかの言語処理に実装するために，どのように設計・作成するかについては，非常に多くのことを考慮に入れる必要がある。どのような目的のMRDなのか，どのような環境での処理を前提とするのか，どのような構造にすればよいか，どの程度の規模のMRDなのか，など，列挙し始めたらキリがないほどである。本論文ではそれらの中でも特に，MRDをコーディングするためのメタ言語の問題について考察したい。MRDをどのメタ言語でどのようにコーディングするかは，そのデータがどの程度共有可能であるかにもかかわる重要な問題である。

2：メタ言語の選定

機械可読辞書を作成するにあたっては，いわゆる文書フォーマットが必要になる。たとえば，

見出し語，品詞，語義1，語義2

といったいわゆるCSV（Comma Separated Value）も文書フォーマット（データフォーマット）の一種である。しかし，こうした原始的なフォーマットは必ずしも言語処理には向かない。たとえば，語義は最大でいくつまで分類するのか，同形異義語はどのように処理するのかなど，さまざまな問題が出てきて，単にフィールドがコンマで区切られていればよいといった単純なものではない。MRDも辞書である以上，構造が必要不可欠である。単に形式上のフォーマットではなく，文書フォーマット≒文書構造が記述できなければならないのである。

では，デジタル文書に構造を付与するためにはどうするかと言うと，マークアップ言語（Markup Language）または文書構造記述言語というメタ言語を利用する。ただし，メタ言語が常にマークアップ言語とは限らない。たとえばコンパイラをコンパイルするための言語はコンパイラにとってはメタ言語だが，コンパイラ・コンパイル言語はマークアップ言語ではない。

マークアップ言語の中にもさまざまなものがある。最も身近なところではインターネットのホームページを作成するためのHTML（Hyper Text Markup Language）だろう（本論で検討するXMLはHTMLと無関係ではない。それどころか大いに関係があるのだが，それは次第に明らかになる）。組み版システムのTeXやLaTeX，清書印刷システムのroff（troffやfroffなど何種類もある）もマークアップ言語の1つである。それらのマークアップ言語はそれぞれが，組み版や文書構造記述といった目的を持っている。組み版システムとしてはLaTeXは非常に優れたもの

であるが，これは印刷用のマークアップ言語なので，MRDにはいささか不向きである。確かにTeXはデータベースとして利用することも可能で，事実，そのような利用方法も存在すると言えば存在するのだが，TeXそのものが元来印刷を目的として設計されているので，データベースに応用しようとすればするほど，使い勝手が悪くなるばかりか，他の言語との整合性も取りにくくなって，再利用しにくくなる[1)]。

HTMLは学生でも使いこなせるほど手軽だが，こちらはWWWのWebページを作成することが目的であって，実は文書の構造についてはほとんど何も記述することができない。一般に（印刷）レイアウト重視型のマークアップ言語は本格的なデータベースに利用するには限界が出てきても仕方ない。

では，MRD専用のマークアップ言語が存在するかと言えば，存在しないと答えざるを得ない。あえてMRD専用のマークアップ言語は必要とされておらず，辞書としての構造が記述できさえすればよいのである。従って，選択肢は最初からSGML（Standard Generalized Markup Language）かXML（eXtensible Markup Language）のどちらかしかない。SGMLは確かにほとんどパーフェクトな文書記述言語で，実際に，SGMLで記述された文書はあちこちで蓄積されている。しかし，SGMLの欠点はまさしくそのパーフェクトさにあり，簡単に言えば使いにくいのである[2)]。現時点では文書構造を記述化するには，事実上XMLの選択肢以外はありえない。

3：XMLの特徴

では，MRDのコーディング用メタ言語としてほとんど唯一の選択肢であるXMLとはどのような言語なのか。まずは全体的な概略を紹介しておこう。XMLはそもそもはSGMLのサブセットである。HTMLも

SGMLのサブセットなので，全てはSGMLから始まったと言える。実はSGMLの前身はIBMによって作られたGML（Generalized Markup Language）だが，さしあたってそこまでさかのぼる必要はない。

SGMLは1986年にISOの規格を受けた（ISO-8879）。SGMLが必要とされたのはネットワークにおいて，お互いに共通の文書フォーマットに基づいてデータを共有するためだった。事実，アメリカ国防総省の公文書フォーマットとして採用されたり，航空産業界でも標準文書フォーマットとして採用された。日本でもいくつかの官公庁が採用している。言語学関係で言えば，TEI（Text Encoding Initiative）が提唱する言語データの標準フォーマットはSGMLに準拠している。このように色々な分野での取り組みはあるが，さほど広範囲に普及しなかったと認めざるを得ない。

同じ頃にCERN（Conseil Européen pour la Recherche Nucléaire。欧州原子核研究機構）の研究者たちが，SGMLを参考にして，ネットワーク上で他の研究者たちと簡単に情報をやり取りするためにHTMLを作った。それに呼応して，アメリカのイリノイ大学のNCSA（National Center for Supercomputing Applications）チームが開発したのが，かの有名なWWWブラウザソフトのMosaic（モザイク）である。NCSAがモザイクを開発した頃には，すでに商用パソコン通信経由でもtelnetやftpなどが少しずつ使えるようになっていたが，一般にはインターネットを利用することがどのようなことなのか今ひとつわかりにくかった。このソフトがインターネットブームに火をつけたのはあまりにも有名な話である。ちなみに，モザイク開発者の一人であるMarc AndressenとJim Clarkが立ち上げたのがNetscape社で，同社が開発したNetscape Navigatorは今でも二大WWWブラウザの一方の雄である。

ところが，歴史の偶然はここから始まる。SGMLは元来ネットワークにおいて共通の文書フォーマットに基づいてデータを共有することを目

的として作成されていた。他方において，インターネットのWWWを利用するためにも，共通のフォーマットに基づいて文書が作成されていなければならなかった。その共通のフォーマットこそ，SGMLのサブセットのHTMLである。あまりに完璧主義なSGMLの普及が遅れるのを尻目に，WWWの利用は爆発的に進み，それに伴ってHTMLも2001年現在でversion 4.01まで進んでいて，HTMLを拡張したXHTMLまでも普及の兆しを見せている。本家本元が普及するより早く，分家の方が津々浦々まで行き届いてしまったのである。

他方，SGMLの普及が遅れ，HTMLの普及が進むにつれて，逆にHTMLの限界もますます明確になってきた。HTMLでは限られたタグしか利用できず，それらはどちらかと言うとレイアウト関係で，文書の構造を記述するものではなかった。一例をあげよう。ボールドとかイタリックというのはレイアウトに関係するが，ボールドで書かれた部分は「著者名」で，イタリックで書かれた部分は「書名」であるというのは，文書の構造にかかわる。HTMLではボールドは<B></B>，イタリックは<I></I>というタグが存在するが，著者名<author></author>は存在しないし，作成してもHTML文書としては意味をなさない。

つまり，デジタル文書の共有を目指すのであれば，ボールドやイタリックの指定よりも，どこが著者名でどこが書名であるかがわからないと困るのである。先ほどのCSVでも，

Leo Wanner, Lexical Functions, 1996, John Benjamins

このような形式では，第1フィールドが著者名，第2フィールドが書名，……という「暗黙の了解」がなければ機能しない。たとえ，第1レコードを，

Author, Title, Year, Publisher

といった具合に定義しておいたところで，第2レコード以降には，フィールド桁数以外に，それが書名であることを意味する手がかりがな

いのである。万一，別のデータベースでは第1フィールドが書名だったりすると，お互いに混乱の元である。それを解決するためにこそ，そもそもSGMLは作られたのである。もし，

<author>Leo Wanner</author>, <title>Lexcial Functions</title>, <year>1996</year>, <publisher>John Benjamins</publisher>

と記述しておけば，実はフィールドを区切るコンマも不要である。もちろん，全てのレコードに対して<author>などのタグが入るとデータの容量をいたずらに増やしてしまう欠点はあるが，フィールドの構造の方が重要な場合が多い。

ところが，SGMLは必ずしも使いやすい言語ではない。HTMLくらい取り組みやすければSGMLが普及しなかったわけはないのだが，専門家でさえてこずるほど，厳密さを要求する使いにくい言語であった。とはいっても，HTML程度ではデータ共有の実務に耐えない，というジレンマが近年ますます大きくなってきた。

XMLの取り組みはそのジレンマを解決しようという試みとも言える。XMLが登場したのは1998年初頭である。ここ10年ほどの間に立て続けにSGML，HTML，XMLと三つの重要なマークアップ言語が登場したことになる。はたして，XMLが本当に実務に耐えうるのかどうかは予断を許さないが，少なくとも二つの先例の良いところを取り入れて，悪いところは取り除こうとしていることは事実である。XMLが今後とも実務に耐えうるかどうかは，SGML（厳密にはGML）に始まるマークアップ方式そのものの根幹にかかわるような仕様変更を迫られるかどうかにかかっているだろう。SGML発足当時は参考文献でさえ見つけるのが大変だったが，今や書店にはXML関連書籍のコーナーまであるほどである。今度こそ離陸が期待できる。

言うまでもなく，XMLはMRDのコーディングのために開発されたのではない。むしろ非常に汎用的なマークアップ言語である。あとはその

汎用性をうまく利用するしかない。少なくとも現時点ではMRDのコーディング用メタ言語としてはXMLが唯一の選択肢と考えられる[3]。

4：XMLによるコーディングの例

次に，簡単にXMLのコーディング例を紹介しておこう。XMLは一般に複数の要素から成り立っている。大まかに言えば，データそのもの（XML文書そのもの），その内容の構造を定義したファイル（DTDやXML Schema），そしてレイアウトに関する情報ファイル（XSLTやCSS）である。Internet ExplorerやNetscapeの最新バージョンはXMLのパーサとしての機能も持っているので（パーサというより，相変わらずブラウザと呼んだほうが正確だが），ついXSLTやCSSが注目されがちだが，XMLにおいて最も重要なのはSGMLから引き継いだDTDとXML Schemaである。DTDは厳密にはXML Schemaの一種である。おそらく今後は厳密に定義可能なXML Schemaをどのように扱うかという方に重点が移行していくものと思われるが[4]，ここではまずデジタル文書における「定義」の重要性を理解していただくために，DTDに絞って紹介することにする。

DTDとはDocument Type Definitionの略で文書型定義と訳される。XMLのeXtensibleさはまさしくDTDのおかげである。XSLやCSSなどは後回しにしても，DTDだけは厳密に定義しておく必要がある。

DTDはXML文書そのものの中に，XML宣言に続いて，内部サブセットとして定義することもできる。小さなデータであれば，内部サブセットでも十分だろう。ある程度大規模なデータになると，外部サブセットとして切り離した方が，後々わかりやすい。

DTDの定義そのものは決して難しくはない。むしろ単純すぎて，XML Schemaの重要性が高まっているとさえ言える。DTDの基本的な考

え方は，何が一まとまりのデータの集まりで，そのデータの中はどのような階層構造になっているか，である。先の書誌データなどの場合は，以下のようなDTDの定義が考えられる。

fig. 1　DTD定義例

```
<!ELEMENT bookdata (author, title, year, publisher) >
<!ELEMENT author (#PCDATA)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT year (#PCDATA)>
<!ELEMENT publisher (#PCDATA)>
```

この程度の定義でも十分である。上のDTDの意味は，まず一行目でXML文書の中の1つのデータのかたまりであるbookdataを定義し，その中にはauthorとtitleとyearとpulisherの四つのデータがあることを宣言している。以下は，それぞれのデータの型を定義していて，(#PCDATA)とは要素がテキストを内容として持つことを意味している。従って，画像でも音声でも定義可能である。そして，実際にコーディングしてみたのが，以下のようなサンプルである。

fig. 2　XMLサンプル

```
<? XML version = "1.0" encoding= "UTF-8" ?>
<bookdata>
<author> Leo Wanner </author>
<title> Lexical Functions </title>
<year> 1996 </year>
<publisher> John Benjamins </publisher>
</bookdata>
```

これなら，どこからどこまでがauthorなのか，はっきりわかることに

なる。

しかし，少し考えてみれば，これでもまだ情報は足りないことがわかる。たとえば書籍の場合はこれでもよいが，雑誌の場合，号巻数はどうするのか，日本語の書籍かフランス語の書籍かといった情報は不要か，もし必要だとして，さまざまな言語をどのようにして入力するのか，などの問題が出てくる。

5：文書型定義をめぐる諸問題

上で取り上げたような例の場合ならわざわざXMLでなくてもよいとも言える。この程度のデータなら（もちろん，本格的な書誌データなら話は別である）CSVで作っておいても，フィールド数が4つくらいなら，単なるテキスト処理でフィールドの入れ替えなどいくらでもできる。もちろん，データさえ正しく作られていれば，という前提条件のもとにおいての話だが。

問題は，その程度の構造では記述しきれない場合である。機械可読ではなくても，一般の印刷物の辞書の項目定義がどれほど難しいかを考えてみよう。

まず，どの単語を見出し語として取り上げるのかを決める必要がある。専門用語はどこまで取り上げるのか，固有名詞を取り上げる基準はどこに置くか，複数の綴り字のある語はどうするか，等を決める必要がある。これはフォーマットさえ決まれば，あとはいくらでも追加可能と考えてもよい。

次に，個々の単語において，何を定義する必要があるのか決めなければならない。発音記号，品詞，意味，用例はどの辞書にもあるとして，類義語や反意語はどうするのか，類義語や反意語がない場合はどうするか，同形異義語や複数の品詞，複数の意味のある場合はどうするか，も

決める必要がある。これも原理的には追加可能だが，ここは可能な限り厳密に定義をしておかないと，泥縄式に追加しなければならなくなる。

項目は厳密に決まっていても，個々の項目の数が厳密に決まらないことが多い。ほとんど一義的にしか使われない語もあれば，多義語もある。しかも意味が3つあるものもあれば，5つでも足りないものもある。DTDやXML Schemaの中では，意味に関しては当然，「1個以上」という可変長の定義しかできない。そして，それぞれの意味に応じて用例も異なるので，用例は意味の数に応じて決めなければならないが，その用例も必ず1つと決まっていればよいが，場合によっては2つ以上必要なこともある。たとえばフランス語では従属節のque以下で接続法が用いられたり直説法が用いられたり条件法が用いられるといった，複雑な用例の動詞が少なくない。しかも接続法の使用は主動詞が否定形または疑問形に置かれたときと決まっていればよいが，現実にはそのようにきれいにはまとまらない。否定形の8割方は接続法が用いられているのに，残り2割は直説法だったり条件法だったり，といったケースもありうる。そうなると，最初は接続法だと思われていた第一群規則動詞のほとんどは，実は判別不可能になってしまう。このような場合は1つの意味に対して，複数の用例が必要になりうる。

ところが，現実の辞書を見ればわかることだが，全ての意味に対して用例があるわけではない。紙幅の都合や重要度に応じて用例は適宜省略される。もちろん，あらゆる語のあらゆる語義に対して用例がつく辞書というのも1つのセールスポイントであろうが，果たしてどれくらい現実的あるいは実用的かわからない[5]。となると，結果的に用例は「0個以上」というさらに曖昧な可変長の定義しかできなくなる。

そして，これらの辞書に関する項目定義は，またもや，辞書の使用目的によって問題点が大きく異なってくる。印刷物の辞書に限っても，それは英和辞典なのか，英英辞典なのかによって，辞書の作りは全く異

なってしまうことは言うまでもない。つまり単一言語でよいのか，それとも複数言語が必要になるのかは，現時点では機械可読辞書にとって極めて大きな問題である。1つのファイルの中で複数の言語を扱うことがまだ完全ではないのだ。

実はXMLはJava言語と同じくUnicodeに準拠しており，UTF-8とUTF-16に対応している。Unicodeの問題については中尾1997，1999でも若干述べたが，筆者の立場は，全面支持もしないが，全面否定もしない，である。全面支持しないのは，Unicodeが完全なコード体系ではないのは明らかだからであり，全面否定しないのは，従来のコード体系では不可能だったことのいくつかがUnicoodeであれば実現可能になってきたからである。

たとえば筆者の関係で言えば，従来は日本語とフランス語を1つのファイルの中で同時に扱おうとすれば，特定メーカの製品によるバイナリファイルを作成するのでなければ，Muleなどのように同一ファイル内で複数の文字コードを切り替える以外には方法がなかった。文字コードの段階で異なった文字に同じコード番号が振られているからである。フランス語やドイツ語の特殊文字は1バイトの8ビット目に割り当てられているので，Shift-JISでもEUCでもどちらも日本語表示に8ビット目を使う以上，混在できないのは明らかであり，8ビット目を使わないJIS-2022を使うと，今度は8ビット目を使ってしまうISO-8859-1との整合性が取れなくなってしまう。これではまともな多言語混在ファイルを作成することすらできなかった。同一のテキストファイル内で日本語のひらがなとハングル文字を混在させるなど，ほとんど不可能だった。

言うまでもなく，日本語とフランス語（やドイツ語など主要な西ヨーロッパ言語）が同時に扱えさえすればそれでよいはずはない。Unicodeで同時に扱える言語には限りがあることも事実である。筆者としては，従来不可能だったことのいくつかが可能になった以上，まだ可能ではな

いことが残っていることを理由に全否定するのは理性的な判断ではないと考えている。さらに改良を加えるために生産的な提案や研究をすればよいだけのことである。

XMLはUnicode準拠とはいえ，現実にはUnicodeを処理できる環境は十分とはまだ言えない。たとえば，XMLファイルをコーディングしている最中には当然，何らかのエディットソフトが必要になり，それがUnicodeに対応していなければ，エディット中は文字参照で書くしかない。一般にエディットソフトはフォントのコントロールが必ずしも簡単ではないので（左から右に英文を書いている途中で，右から左に書くヘブライ語やアラビア語を引用する必要が出てきた場合にどうすればよいかを考えれば，フォントがありさえすればよいとは考えられないことはすぐに理解できるだろう），XMLでは現時点では，以下のような文字参照をする必要がある。

fig. 3　ラテン拡張文字と文字参照例（10進法と16進法）

	10進法		16進法
À	&#192;		&#x00C0;
Â	&#194;		&#x00C2;
Ç	&#199;		&#x00C7;
È	&#200;		&#x00C8;
É	&#201;		&#x00C9;
Ê	&#202;		&#x00CA;
Ë	&#203;		&#x00CB;
Î	&#206;		&#x00CE;
Ï	&#207;		&#x00CF;
Ô	&#212;		&#x00D4;
Ù	&#217;		&#x00D9;
Û	&#219;		&#x00DB;
Ü	&#220;		&#x00DC;

（以下略）

もちろん，DTDかXML Schemaの中で定義さえしておけばHTMLなど

でおなじみのネームエンティティ(éなど)も使えるし，この方がまだ可読度が高い。

しかし，辞書で使う文字はこれだけではない。フランス語と日本語の辞書に限っても，発音記号を使うのであれば，またもや困難が待ち受けている。Unicodeでは88のIPA文字（International Phonetic Alphabet）が定義されているが，これをネームエンティティであろうと文字参照であろうと記憶することは音声学か音韻論の専門家でなければ，ほとんど不可能だろう。文字参照という苦肉の策によって，曲がりなりにも扱えるようになってきたというのが唯一の救いなのである。

文字の話がいささか長くなってしまったが，文字を表示させるというだけなら画像として処理してしまう方法もあるので，決して乗り越えがたい問題ではない。検索においても若干煩雑ではあるが，たとえば発音記号の画像とファイル名が一対一に対応していれば，むしろ処理しやすいとさえ言える。

6：メタ言語の未来

XMLの登場によって，確かに（フランス語）機械可読辞書の作成の外的な問題点はかなり解決されたと言える。たとえばUnicodeの問題にしても，まずはXMLを利用したMRDがいくつも作成されて，それぞれの使用目的に応じて評価されて，その結果をフィードバックしていく方が，現実的で生産的な方法だろう。

しかし，XMLの登場によってかなり改善されてきたのは言語にとって外的な問題である。言語に内在する問題はXMLだろうが，SGMLだろうが，何一つ解決されていないと言ってもよい。たとえばフランス語では過去分詞と形容詞の境界が極めて曖昧な語がいくつもある。つまり，見出し語はいくらでも立てられるが，たとえばそのMRDが構文解

析や自動翻訳に利用されるのであれば，今度はその形容詞か動詞かをどのようにして判定するかの条件分岐を辞書の中に記述しておいてプログラムがその情報を参照するようにアルゴリズムを組むか，あるいは何らかの「機械可読文法」を実装しておかなければ，いくらMRDがあっても実用にはならないだろう。

XMLがその真価を問われるのはまさしくここかもしれない。XMLはその仕様上，基本的にデータベースには向いている。辞書とはまさしくデータベースである。問題は，文法はデータベースか否かである。確かに文法にも若干のデータベース的性質があるが（なぜなら，規則の「集まり」である），文法とは基本的に「規則」の集まりである。辞書の中の個々の項目には，出現頻度順位や利用頻度順は存在しても，優先順位は存在しない。しかし文法の規則の中には優先順位の高い規則もあれば，特定の限られた条件においてしか機能しない規則もある。たとえばフランス語では形容詞は一般に名詞の後ろに置かれ，一部の形容詞は名詞の前に置かれるといった場合，優先順位の高い規則は「形容詞は名詞の後ろ」である。それが出現順位も高いかどうかはわからない。なぜなら，名詞の前に置かれる一部の形容詞とは使用頻度の高いものばかりだからである。あるいは，他動詞の過去分詞が性・数一致するのは「複合時制で直接目的補語が過去分詞より前に置かれた場合」などというのは，条件が3つも重なっている以上（複合時制，直接目的補語，その位置），かなり制限された規則とみなしてよかろう。実際には日常的に目にする現象ではあるが[6]。

また，現実の文が必ずしも規則に従っているわけではない。それらの中には，規範文法からすれば，「誤用」と断定されるものもあれば，「例外」や「稀」といった扱いを受けるものもある。しかし，異文化コミュニケーション的な観点に立てば，現実の言語現象は，誤用，例外，稀，省略，言い淀み，迂遠，重複などの連続である。文法とはそれらのmass

でしかありえない。さらに重要な要素として，文法とは常に，あらかじめ存在するものの最大公約数でしかありえず，原理的には決してup to dateに追いつくことはない。言語の変化は現在も進行中で，文法がその変化を先取りすることはありえない。

問題はこのようなダイナミックな動きにどれだけ対応できるかである。データベースや文法に全てを記述することの不可能性はすでに人工知能の分野で証明済みである。今は，いくつかの規則からいかにして未知のものを推論したり予想できるか，といった方向に向かっている。MRDについても同じことが求められるだろう。

註

1) 事実，1995年ころには情報知識学会などで，TeX文書をどのようにしてHTMLあるいはSGML文書に変換するかといったことがよく議論されたり発表されたりした。そのときに最も問題になったのは，TeXとHTMLでは文書構造の記述の仕方が違う（つまり文書構造の解釈の仕方が異なっている）点であった。
2) 場合によってはタグの方が本来のデータより容量が大きくなってしまうことさえありうるのは，決して誉められた仕様ではない。もっとも，その程度のことは，ハード的なマシンパワーさえ向上すれば，微々たる問題ともいえる。むしろ，情報の中心点と考えられるテキスト部分（文書本体部分）より，文書構造やレイアウトの情報量の方が多くなりがちなのは，言語学的，情報理論的に見て極めて興味深い現象である。その他にハイパーリンクが実装されていないなど，時代にそぐわない欠点も出てきている。
3) 1998年に出版された*De l'écrit au numérique* (Habert et al.)では，SGMLのサブセットという位置付けではあるが，XMLがいち早く言語データのマークアップ言語として紹介されている。
4) 最新の情報では，XML文書から必要な情報だけを抽出して組み版に渡すようなシステムが次々と開発されている。ペーパーレス社会にはほど遠いが，XMLの柔軟さをよく示している。
5) 通常の電子辞書などなら，データベースに対して常時アクセスできるようにハイパーリンクを設定しておけば，意味を参照するそれぞれの場合に最新の用例を引き出すことができる，といった方向を考えた方が将

来性があるだろう。

6) このように，優先順位が高いからといって，使用頻度も高いとは限らないとか，かなり制限された規則なのだが，言語現象としては日常的といった「ねじれ」がフランス語の自動構文解析を難しくしている要因の一つでもある。

参考文献

Claude FREY, Danièle LATIN, *Le corpus lexicographique*, 1997, Duculot.

Benoît HABERT, Cécile FABRE, Fabrice ISSAC, *De l'écrit au numérique, constituer, normaliser et exploiter les corpus électroniques*, 1998, Masson.

Benoît HABERT, Adeline NAZARENKO, André SALEM, *Les Linguistiques de Corpus*, 1997, Arman Colin.

Charles F. GOLDFARB, Paul PRESCOD, *The XML Handbook*, 1998, Prentice Hall（邦訳，安藤慶一，『XML技術大全』，ピアソン・エデュケーション，1999年。）.

Elizabeth CASTRO, *XML for the World Wide Web --- Visual QuickStart Guide*, MdN Corporation, 2001（邦訳，『XMLクイックスタートガイド』，エムディエヌコーポレーション，2001年）.

政瀧浩和，匂坂芳典，「品詞および可変長形態素列の複合N-gramを用いた日本語形態素解析」，『自然言語処理』Vol. 6–2，1999年1月。

春野雅彦，「対訳テキストから辞書を自動生成」，『情報処理』Vol. 40–4，1999年4月。

松本，影山他，『単語と辞書』，岩波講座「言語の科学」3，1997年。

長尾，黒橋他，『言語情報処理』，岩波講座「言語の科学」9，1998年。

中尾浩，赤間啓之，馬場雄二，「マルチリンガル環境の構築：Unicodeの理論と実践」，『人文学と情報処理』第16号，1997年。

中尾浩，「文字コードにおけるマルチリンガル概念について」，『日本フランス語フランス文学会中部支部研究報告書』第23号，1999年。

一二五

＜付記＞

本研究は1999–2000年度愛知大学研究助成（課題番号C-85）による成果の一部である。