学習者コーパス入門

学習者コーパスのあるべき条件とその作成の具体的方法は?

英語学習者の発話や作文などの performance data を
コンピュータ処理し，教育や研究に役立てるために

第3回 「学習者データの作成と管理」

元東京学芸大学講師／ランカスター大学言語学科博士課程在籍 投野 由紀夫

はじめに

今回は，いよいよデータを一応取れた段階で次にどうするかという話をしよう。大きく分けて transcription の実際的な方法と，ファイル管理について説明したい。ここからは，多少専門的な内容も多くなってくる。できるだけわかりやすく解説するのでデータ作成・管理の基本を身に付けてほしい。

書き起こし(transcription)の実際

前回はデータ入力の際の労力を省く工夫についてお話したが，実は書き起こしの段階にはデータの本質的な価値を決める重要な判断が必要である。たとえば，ある同一テープのtranscription を2名の学生に頼んだところ，次のような transcription が戻ってきたとしよう。

Transcription A :

His son,washed his socks, but he is angry to his son. His father is angry to him. Sorry, his son washed his socks, but ...

Transcription B :

えー，えーと，... Hi ..., his son is, えー, washed, his, his son is, his son washed his socks, but he is angry to his son ...His father is angry to, to he ? えーとすいません his, his son washed his socks,but ...

なんの打ち合わせもなく書き起こしをさせた場合，程度の差はあるが A のようになるべく英文の意味が通るように書き起こそうとするタイプの人と，B のようにできるだけ聞こえたままを書き写すようにするタイプの人がいる。「できるだけしゃべったとおりに書き起こしてください」と言えばこのようなずれはかなり回避できるのだが，実際のところ問題はそんなに単純ではない。

B の場合，「えー」などと filled pause の部分を気を利かせて日本語にしているが，これも統一しないと人によっては英語風に uh... などと書くかもしれない。Hi..., his son という文頭の部分は [hi:] という発音だけから推測すると，He ..., his son かもしれない。His father is angry to, to he? の部分は him なのに he と言ったことに確信がなかったから ? を付けたのか，それとも実際は上がり調子で質問しているのかどうか，第三者が見てもよくわからない。A に比べると，B のほうが書き起こしとして忠実ではあるが，語のダブりや文の区切り目などについてコンピュータに認識させるのはよりむずかしい。

この例のように，案外気楽に考えていた transcription も，いろいろな問題を含んでいることがわかる。書き起こしの際には対象が英作文か会話データかによって注意すべき点も異なってくる。いずれにせよ，書き起こしの精度を高めるために「複数の人が書き起こしをして同じような結果になる」ための基準を設定する必要がある。現場の

先生の場合にはどのくらい時間的な余裕があるかという現実的な問題があるわけだが，ここではデータをきちんと作るという観点から多少専門的になるが，どのような方法が必要かポイントをあげておきたい。

POINT 1 パイロットの書き起こしで問題点を洗い出す

まずデータが集まったら，自分も含めた数人の異なるメンバーで実際に2，3本の英作文（または発話データ）の書き起こしをしてみるとよい。そうすると，やり始めたとたんにいろいろな疑問点が出てくるはずである。Punctuation はどうするか，書きなおし，言い間違えはどうするか，などなど個人的な判断ではまずい点がわかってくる。これらをメモにしておたがいに問題点のリストを作成するようにする。

POINT 2 書き起こしガイドラインを作る

問題点を洗い出したあとは，自分たちの研究目的に合ったガイドラインを設定する。大まかなポイントは以下の点だ。

① 音声レベルを扱うかどうか？

平たく言えば，会話データで発音の誤りをカウントするかしないか，という大きな判断になる。専門的には会話データだと acoustic phonetic / narrow phonetic / broad phonetic / citation phonemic などいろいろなレベルに分けるのだが，ここでは大きく分けて，phonetic level と orthographic level に分けておこう。

結論から言えば，会話データの transcription では，特別明確な研究目的がないかぎり，phonetic transcription からスタートするのは避けたほうがよい。1つにはもし発音の誤りを考慮するとなると，IPA や代替文字の使用などテキスト処理上ややこしい問題が出てくるし，transcription をする人も専門的な音声学の心得がないといけない。2つにはそうやって膨大なエネルギーと時間をかけて途中で挫折してもその失われた時間は帰ってこないからだ。（一般的に自発的な会話データではネイティブが普通に orthographic transcription をする時間は会話時間の約10倍と言われている。）Gibbon, Moore & Winski (1997) でも，最初は orthographical な書き起こしからスタートし，量をこなしてから，データの一部を余裕があれば音声面の表記をかぶせることを薦めている。本稿でも文法・語彙関係が中心事例になるので，orthographic level に留めておくことにしたい。

② 書き起こしの共通記号を決める

Orthographic なレベルでもさまざまなことを考慮しておかねばならない。専門的には基本フォーマット（たとえば CHILDES ならCHAT，あるいは汎用の SGML/XML）などによっても表記の仕方や詳しさが異なるのだが，一般的な事項を表1にまとめてみたので参考にしてほしい。

表1：書き起こしで共通記号が必要になるケース

A. 発話データの場合

(1) Speaker ID：とくに対話データの場合に必要
【対処法】適宜わかりやすいコードを付ければいい（<$A>，<$B> など）

(2) Word fragments:単語の一部だけ言っている場合
(例) Fri, Friday
【対処法】<f> Fri</f> としたり Fri-Friday などと書く。

(3) Verbal deletions or corrections：誤った単語を言い直している場合
(例) the price, I mean the place ... (explicit correction)
the price, place ... (implicit correction)
【対処法】<r> ... </r> タグで囲む

(4) Unintelligible words: 聞き取り不可能な単語
【対処法】コーパスによるが，【Unintelligible】と入れたり，"•••" としたりする。

(5) hesitations and filled pauses：「あー，えーと」などに当たるもの
【対処法】日本語の filled pauses との区別はしないほうがよい。通常，母音系の "uh" と鼻音系の "mm" の2タイプを用意して書き分けることが多い。

(6) Simultaneous speech:発話が重なる場合
【対処法】ダイアログやインタビューなどでは重なる部分の表記を【 ... 】でくくったり，tier（段）を分けてダブりを表現したりなどいろいろなシステムがある。

B. 作文データの場合

(1) Spelling mistake:綴りの間違い
【対処法】普通はそのまま表記してから正しいスペリングを< >や tier を分けて別の注釈行に充てておいたりする。

(2) Punctuations: ピリオドやカンマなどの認識
【対処法】ピリオドがないと品詞タグ付与などが自動でできないので，通常は書き起こしをする人のほうで適宜判断して文を区切ることが多い。その際，自分が入れた句読点であることを明示的に示す示さないは目的によって異なってくる。

(3) Rewriting: 書き間違い
【対処法】最初は自分で訂正して書き直している作文は訂正したものを入力しておけばよい。目的に応じて，あとで自己訂正個所をチェックしたりできる。

(4) Foreign words: 英語の場合には日本語が挿入されている場合など
【対処法】単語リストを作るときなどに必要ならば除去できるように <JP> ... </JP> といったタグで該当語を囲んでおく。

POINT 3 ガイドラインに沿って再度一定量のデータの書き起こしを試行してみる

これによって，ガイドラインが明確か，書き起こす人によって極端に個人差が出ないか，どの程度の労力が実際にかかるか，といったことの目安が得られる。大きなプロジェクトではこの段階で書き起こしのための謝金や研究補助の要員を試算することになる。自分の教室内での小さな規模の研究ではとてもこのような余裕はないかもしれない。しかし，ある意味で大は小を兼ねるので，このくらい注意を払ってやるということを肝に銘じておいて，あとは適宜実情に合わせて方法を考える，ということになろう。

データ・フォーマット

Transcription はデータの中身の精度の問題だったが，もう1つ忘れてならないのはコーパスデータをコンピュータに理解させる際の情報の書き方の問題だ。いわゆるフォーマットである。この部分だけでもコーパスの専門書が1冊書けてしまうほどであるが，ここではごく初歩的なフォーマットの方法を解説しよう。

<いろいろなフォーマット>

コーパスデータを扱うと目的に応じていろいろなフォーマットがあることがわかる。代表的なものに (1) Fixed Format References，(2) COCOA 形式，(3) CHAT 形式，(4) SGML / XML 形式がある。サンプルを見ながら大まかな特徴を押さえておこう。

★Fixed Format References

この形式はテキストの各行の左端にテキスト情報と行番号が来るという形式。テキストそのものの書誌情報は別ファイルに記されている。1960年～70年代にかけて Brown / LOB Corpus（注：百万語のバランスを考えた米語・英語コーパス）やその他の電子テキストに広範に利用された。現在でも，日本の TXTANA，海外の WordSmith などの主要コンコーダンサーにはこの形式に対応した設定をできるようになっているものが多い。一時代前ではあるが，Brown / LOB という二大コーパスの標準書式だったので今でも影響力がある。しかしこの方式は，最近編纂されているコーパスではあまり使われていない。

```
R01 0010  It was among these that Hinkle
          identified a photograph of Barco!
R01 0020  For it seems that Barco, fancying
          himself a ladies' man (and why not,
R01 0030  after seven marriages?), had listed
          himself for Mormon Beard roles
R01 0040  at the instigation of his fourth
          murder victim who had said:"With
R01 0050  your beard, dear, you ought to be
          in movies"! Mills
```

Fixed Format References (Brown Corpus からの引用)

★COCOA

80年代に OUP から発売されていた Micro-OCP というコンコーダンサーはじめ広範囲に利用されていた形式。最初のアルファベットが変数名 (T = title ; A = act ; S = scene ; C = character など) で，それに続いて変数の値が文字列で記入され，全体を < > で囲むというようになっている。これらの変数名は自分で任意に付けることができ，柔軟性が高かった。このころからテキスト部分とヘッダ部分を区別して記述するようになった。ただし，後の SGML などと比べると，テキスト自体が構造化はされておらず，変数の定義ファイルなどもないので，電子文書としては過渡的なフォーマットだったと言えよう。

```
<T Merchant of Venice>
<A 1>
<S 1>
((Enter Antonio, Salerio, and Solanio))
<C Antonio>
In sooth, I know not why I am so sad.
It wearies me, you say it wearies you,
But how I caught it, found it, or came
by it,
What stuff 'tis made of, whereof it is
born,
I am to learn;
```

COCOA 形式 (Merchant of Venice の冒頭)

★CHAT

幼児の言語習得のデータベースとして有名な CHILDES のフォーマット形式。現在，言語習得関係の標準フォーマットの1つ。サンプルを見ていただくとわかるが，開始は @Begin，終了は @End でそれぞれ改行する。@Begin の直後に @Participants という変数行で発話中に出てくる speaker を定義して，発話部分 (speaker tier) はそこで定義した3文字の speaker code に * を付けて発話

の先頭に置くという決まりがある。記号と発話を区切るコロンのあとは必ずタブ区切りにする。% で始まる tier はいろいろな注釈行になる。

CHILDES は CHAT 形式をサポートする大量のツール群（CLAN という）を開発しているので、とくに対話データの分析に興味がある方はぜひマスターしてみるといいだろう。本稿では、英作文の処理や品詞・構文解析のようなツールの紹介まで考慮して、CHAT 形式をメインには採用しないが、どちらもテキストデータであるので、フォーマットを変換して利用したりする方法もあとで紹介したい。

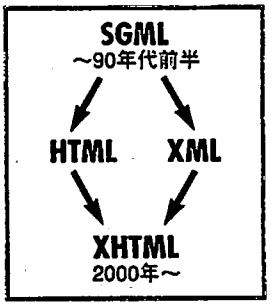
```
@Begin
@Participants: CHI Adam Target_Child,
MOT Mother, URS Ursula_Bellugi,
Investigator, DIA Diaperman Adult
@ID: brown.ad.adam20.0300=CHI
@Sex of CHI: Male
@Age of CHI: 3;0.10
@Date: 15-JUL-1963
@Time Duration: 11:00-12:00
@Situation: Adam has a cough and runny
nose and had it for a week. Only Mrs
Smith, Ursula and baby Paul are present.
Paul's crib has been moved into the
living room with tape recorder
*CHI:   why dis got holes?
%act:   looking at holes in Ursula's pad
*URS:   so you can put it in a notebook
# if you like.
*CHI:   0.
%act:   falls from bike
*URS:   what happened?
*CHI:   I fall # broke my head.
%spa:   $RES
*URS:   you didn't.
*CHI:   tell me story.
*CHI:   tell me story.
*URS:   shall we look at these first?
%act:   gives Adam bag of toys
```

CHAT 形式で書かれた Roger Brown の Adam corpus

★XML / SGML

世界的に電子テキストの標準となっているフォーマット。これはコーパスデータ専用のフォーマットというわけではなく、広く電子テキストを共有化するための一般的な規格である。大量の文書の構造化やそのチェックをするためのツール群がほとんど無償で公開されているので、書式に慣れてしまえば非常に柔軟性が高い電子テキストを構築できる。80年代は SGML（Standard Generalized Markup Language）が主流だったが、最近は WWW の技術が進んで、SGML よりもハイパーテキストを扱える XML という書式のほうが優勢になってきている。ホームページの書式の HTML はこれらの親類みたいなものだ。このへんはむずかしいのであまり真面目に覚えなくてもよいが、ちなみに有名な British National Corpus は SGML 形式だし、現在主要辞書出版社の持っている電子データのかなりのものがSGML 形式になっている。次のセクションで XML を詳しく扱うので例は省略する。

＜ニーズに合ったフォーマットを選ぼう＞



さて、いよいよ我々のデータのフォーマットを決めよう。ニーズによって CHAT でもよいし、COCOA でもよいが、今回は XML を採用しておこう。XML は“ eXtensive Markup Language”の略だ。なぜこの形式を利用するかというと、(1) SGMLのように構造化テキストを扱える、(2) HTMLのようにハイパーテキストを扱える、(3) 将来的に web ベースのツール類やデータベースなどと連携する際に威力を発揮する、といった点が挙げられる。一般の中高の英語の先生にはなんのことやらさっぱりわからないかもしれないが、要するに今後もっとも利用される可能性の高いフォーマットだということだ。現在、Internet Explorer の最新バージョン（5.0以上）では XHTML という形式が採用されていて、これは XMLでフォーマットされた内容を XSL と CSS という2種類のスタイルシートを通してブラウザに表示させる仕組みになっている。一般の電子テキストの世界でも基礎的な構造部分のフォーマットはどんどん XML になってきているのだ。CES（Corpus Encoding Standard）という大規模コーパスの標準フォーマットを決めようというプロジェクトでも、昨年辺りから SGML から XML へとフォーマットを移行して、ハイパー文書的にコーパスを扱うという試みが研究され始めている。

＜XML による記述の基礎の基礎＞

(1) テキストのデータ構造を定める

それでは具体的な例として生徒の英作文データをフォー

マットしてみよう。まず肝心なことは，作文データとして管理したい内容をできるだけきちんとリストすることだ。その際に作文データの「**テキスト外情報** (extra-textual information)」と「**テキスト情報** (textual information)」を大きく分けて考えるとよい。テキスト外情報というのは，例えば「学校」「学年」「クラス」「氏名」「実施時期」「課題内容」「制限時間」「辞書使用」「添削」などなどのデータ管理上必要な情報だ。これはどれだけデータを一般的に共有するかにもよるので，必要な項目を各自で選定すればいい。

テキスト情報は実際に書かれた作文や録音した発話データの中身に関する情報になる。発話データならば，話者ID，作文ならば段落区切りなどの情報がまず考えられる。文区切りに関しても途中で切れたりして境界がはっきりしないものが多いので，あったほうがよいだろう。また prosodic な情報や，error tag などの実際のデータに付ける annotation（注釈）部分は今回は扱わない。

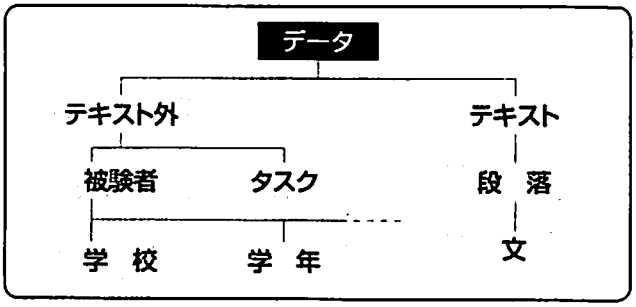
▶テキストの構造を分析，洗い出す◀

テキスト外情報　学校，学年，クラス，氏名，実施時期，課題内容，制限時間，辞書使用，添削有無，などなど…

＋

テキスト情報　話者ID，段落区切り，文区切り，などなど…

次にこれらの中で階層構造を持たせたほうがいいものを特定する。たとえば，テキスト外情報とテキスト情報は並列の関係，テキスト外情報のうち「学校」「学年」「実施学期」「クラス」「氏名」「性別」に関するものは被験者情報，「課題内容」「制限時間」「辞書使用」「添削有無」などは課題情報として分類する。テキスト情報のほうも話者ID（作文の場合は不要）―段落―文といった階層を持たせる。



いよいよこれを XML で記述してみよう。

(2) 簡単な XML フォーマット

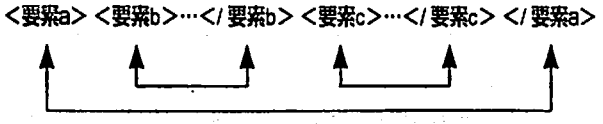
非常に簡単なXML の例として，英作文データをもとにほぼ (1) でとりあげたような変数を考慮しながら書式を書いてみたい。XML は通例，構造に関する定義を DTD (Document Type Definition) と呼ばれる部分に書き込んで参照し，それに対応する文書 (XML instance という) をペアにして持つ。しかし，今回のスペースではDTDの書き方を十分解説しきれないので，DTDを使わずにXML instanceのみを作成して利用しておこう。こうしておけば，今後，大量のXML ファイルを扱うようになった際に，DTD をしっかり作ってそれにそったチェックを再び行ったりできるので，別に心配はない。DTD でチェックをしなくても XML の書き方を守っていれば well-formed XML instance ということができる。

まずはXML文書だということを宣言する。これには <?xml version="1.0" ?> というおまじないの言葉を文書の先頭につける。「XML の version1.0 を用いなさい」という命令文だと思えばいい。次からが構造を示すタグ (記号) が来る。XML は必ず文書の要素はタグで表すことに決まっていて，**内容** (content) を**開始タグ** (< >) で始めて，**終了タグ** (</ >) で終る。そこで，コーパスデータ全体は大きくは，<data> ... </data> というくくりを設けてみよう。その中が2つに別れる。<head>部分と <text> 部分だ。そして <head> 部分に大別して <subject>関連の要素 (ここではそれぞれ <school>, <year>, <term>, <class>, <sex>, <name> としてみた)，<task>関連の要素 (<topic>, <time> など) をそれぞれ記述する。ここで記述の際のいくつかの注意点を書いておこう。

① タグの中の要素名は小文字にする

② 必ず終了タグをつける

③ タグが交差しない (下図参照)



以上の書き方に注意して，書き起こしデータをXML形式でフォーマットしたものが次頁のサンプルだ。データそのものはわかりやすくするために割愛してあるが，大きく <head>部分の記述と <text> 中に書き起こしたデータを置くという形式を理解していただけると思う。

```
<?xml version="1.0" encoding="Shift_JIS"?>
<data>
<head>
<subject>
<school>Lancaster High School</school>
<year>9</year>
<term>summer</term>
<class>4</class>
<sex>f</sex>
<name>Yuki Tono</name>
</subject>
<task>
<topic>breakfast</topic>
<time>20 min</time>
<dicuse>no</dicuse>
<revision>no</revision>
</task>
</head>
<text>
<s>I like rice better than bread. </s>
...
</text>
</data>
```

XML 形式で書かれた英作文データの例

面倒くさい!という人に

ここまで解説を読んで,「おもしろそう!」と思っている方は案外こういう細かい作業に快感を覚える人だろう。大多数の読者はそろそろ「こんなことやっている時間がないよ, 別の記事を読もうかな」と思っているに違いない。実は私も面倒くさがり屋なので, この手の作業をできるだけ省エネ化できる方法をいつも考える。いくつか雑談風にヒントをあげておこう。

1つは同じ種類のデータであれば, ほとんどヘッダ部分をいじる必要がないので, 本文を書き起こしたファイルの先頭にコピーして貼り付けてしまえばいい。私はもっぱら今は UNIX のコマンドでなんでもやってしまうが, Windows だけ使っていたころはよく Win Easy (http://www.fluid3d.com) というフリーウェアを使って, 1クラスのファイル全部に一気にヘッダ部を挿入するような作業をしていた。あとは, 個々のファイルの変数部分をちょこっといじればよい。

作業効率を考える人は, MS-Word のようなワープロソフトで作業するならばノーマルモード (WYSWYGでないモード) で作業するようにして, できるならば重たいワープロソフトを使わず普通のテキスト・エディタを用いるといい。テキスト・エディタのほうが軽くて断然速い。「秀丸」,「VZ Editor」,「MM Editor」などの優秀なエディタが安価で入手できる。とくに MM Editor はHTMLマクロが非常によくできており私は愛用していた。(していた, と過去形なのは最近 Emacs に慣れて Meadow ばかりを使っているからだ。) タグの挿入も自分がカスタマイズしたタグをなん個でも登録できるので非常に便利だ。ぜひ試してみるといいだろう。

＊　　＊　　＊

ちょっと消化不良は否めないが, 紙数が尽きたので今回はこのへんで切り上げる。上記のような方法でファイルをとりあえず作ってみて "test.xml" (引用符は実際はなし) というファイル名で保存しておこう。可能ならば手持ちのデータをこのような形で整理しておくといい。次回は, XML にしたことのありがたみがもう少しわかるように DTD の基礎的なことに触れたり, XMLの作業の効率を高めるツール類を紹介し, いよいよこのデータをWordSmith というコンコーダンサーで検索してみることにしたい。もしXMLに興味がある方はパソコンショップの書籍セクションに行って関連する本を眺めてみるといい。今はXML, XHTMLに関する本もたくさん出ているはずだ。他のフォーマット形式に関することは参考文献をご覧いただきたい。それではお疲れさま!

参考文献

Garside, R., G. Leech and T. McEnery (eds.). *Corpus Annotation : Linguistic Information from Computer Text Corpora*. London, Longman, 1997.

Gibbon, D., R. Moore and R. Winski (eds.). *Handbook of Standards and Resources of Spoken Language Systems*. 4vols. Berlin: Moutonde Gruyter, 1998.

『標準XML完全解説』. (XML / SGMLサロン著). 技術評論社. 1998.

有益な web ページ

Studies in Humanity Computing by Susan Hockey (http://www.humanities.ualberta.ca/english402-98/default.htm)

問い合わせ：y.tono@lancaster.ac.uk