PDF 中の TEX 記号の復元と ACL Anthology への適用

磯崎 秀樹
岡山県立大学大学 情報工学部

isozaki@cse.oka-pu.ac.jp

1 要旨

今や、多くの論文が電子的に入手可能であり、そのほとんどは PDF ファイルである。したがって論文を対象とした検索などのシステムを作成するためには、PDF ファイルからテキストを抽出しなければならない。そのためのツールはいくつもあるが、著者が試したツールは、いずれも TEX の記号の多くが消えたり、文字化けすることがわかった。現在の自然言語処理は数理的な手法が多く、これらの記号が正しく復元できなければ、多くの情報が失われる。そこで、TEX の記号を復元するツール BackToLATEX を作成し、ACL Anthology のインデキシングに利用したので報告する。

2 はじめに

著者は以前、医療関係の英語論文アブストラクトのデータ PubMed[1] を日本語で検索できる「日英言語横断医療情報アクセスシステム」[2] を構築した。このシステムは、京大ライフサイエンス辞書 (LSD)[2]、日英言語横断検索 [3]、統計的機械翻訳による英日翻訳 [4] などを組み合わせることにより、論文に基づいた医療の最新情報を日本人にわかりやすく提示することを目指したシステムである。

このシステムをデモしたところ、「自分の研究分野にも欲しい」「研究を始めたばかりの大学院生や新入社員の教育に使えるのではないか」という意見を複数聞いた。著者自身、自然言語処理の分野で同じようなシステムがあれば、自分が助かるのではないか、と感じた。

そこで、実際に自然言語処理分野の研究動向調査のための検索と可視化を行うシステム「Academic Surfing」を作ることにした。自然言語処理分野の論文の多くは、ACL Anthology[3] から PDF ファイルを入手可能であり、現在 2 万本以上が登録されている。

PubMed はアブストラクトだけであったが、ACL Anthology の場合は、PDF ファイルからテキストを抽出する必要がある。

ACL Anthology の解析については、ACL Anthology Network (AAN)[4]や ACL Anthology Reference Corpus (ACL ARC)[5]などの解析結果のデータが公開されている。しかし、著者は自分に使いやすいシステムで、最新のデータを手元で利用したいので、これらの解析結果とは独立に解析を進めている。

PDF からテキストを自動抽出する方法には、大きく分けて、文字認識するアプローチと、PDF ファイルを解読するアプローチがある。Adobe Acrobat には文字認識の機能がついていて、文字認識することができる。InftyReader[6] というソフトを使えば、数式の認識も可能である。

しかし、文字認識では、「1」と「l」のように似た文字を読み間違えることが予想される。PDF を解読するアプローチでは、ファイル中に文字コードが残っていれば、この二つを混同することはない。InftyReader が出力する LATEX のファイルは、非常に完成度が高く、実用的であるが、予想通り、この問題は発生した。また、著者が予想してなかった「@」と「§」の読み間違いなどもあった。

本研究では、PDF ファイルを解読するアプローチを採用する。PDF ファイルからテキストを抽出するツールには Apache PDFBox[7]の ExtractText、xpdf[8]の pdftotext、PDFMiner[9] などがある。

しかし、これらのツールを使ってみたところ、TEX の記号の多くが消えたり文字化けしたりすることが判明した。現在の自然言語処理は数理的な原理に基づいていおり、これらの記号が正しく復元できなければ、多くの情報が失われる。

[1] http://www.ncbi.nlm.nih.gov/pubmed
[2] http://lsd.pharm.kyoto-u.ac.jp
[3] http://aclweb.org/anthology-new/
[4] http://clair.eecs.umich.edu/aan/index.php
[5] http://acl-arc.comp.nus.edu.sg
[6] http://www.sciaccess.net/jp/InftyReader/
[7] http://pdfbox.apache.org/
[8] http://www.foolabs.com/xpdf/
[9] http://www.unixuser.org/~euske/python/pdfminer/

表 1: フォントによる文字配列の違い

cmti10	*'0*	*'1*	*'2*	*'3*	*'4*	*'5*	*'6*	*'7*
'00x	*Γ*	*Δ*	*Θ*	*Λ*	*Ξ*	*Π*	*Σ*	*Υ*
'01x	*Φ*	*Ψ*	*Ω*	*ff*	*fi*	*fl*	*ffi*	*ffl*
'02x	*ı*	*ȷ*	*`*	*´*	*ˇ*	*˘*	*¯*	*˚*
'03x	*¸*	*ß*	*æ*	*œ*	*ø*	*Æ*	*Œ*	*Ø*

cmmi10	*'0*	*'1*	*'2*	*'3*	*'4*	*'5*	*'6*	*'7*
'00x	*Γ*	*Δ*	*Θ*	*Λ*	*Ξ*	*Π*	*Σ*	*Υ*
'01x	*Φ*	*Ψ*	*Ω*	*α*	*β*	*γ*	*δ*	*ϵ*
'02x	*ζ*	*η*	*θ*	*ι*	*κ*	*λ*	*μ*	*ν*
'03x	*ξ*	*π*	*ρ*	*σ*	*τ*	*υ*	*ϕ*	*χ*

国立情報学研究所の相澤教授らのグループは、最近、論文中の数式の解析にアクティブに取り組んでいるが、LaTeX のソース [5] あるいは MathML や InftyReader[6] などを前提としており、PDF からの数式の解読そのものは研究対象として扱っていないようである。

そこで、TeX の記号を復元するツール BackToLaTeX を作成したので報告する。

3 テキストの抽出法

PDF はプレインテキストの中にバイナリの混じったファイルであり、抽出したいテキストは、バイナリ部分に含まれている。バイナリの部分は `stream` と `endstream` の間に挟まれていて、`stream` の直前に `/FlateDecode` が書かれていれば、zip の decompress で復元できる。

```
\rm This is Roman. \it This is Italic.
```

と LaTeX で書いて pdfLaTeX で作成した PDF を解読すると、テキストは以下のように `BT` と `ET` ではさまれている。

```
BT
/F18 9.9626 Tf 148.712 707.125 Td
[(This)-250(is)-250(Roman.)]TJ
/F19 7.9701 Tf 63.701 0 Td
[(This)-250(is)-250(Italic.)]TJ
ET
```

`/F18 9.9626 Tf` はフォントの種類とサイズの指定である。`[...]TJ` がテキストを出力するコマンドである。出力する文字列が ( ) で囲まれていて、その間の数字が水平方向の移動量を表す。これらのテキストに関する PDF のコマンドについては、PDF のリファレンス・マニュアルや書籍 [8] を参照されたい。ここから `This is Roman.  This is Italic.` というテキストを抽出できる。こう書くと、PDF からのテキストの抽出は簡単に思える。

4 TeX のフォントの問題

しかし、色んな要因が解読を困難にする。一つは合字である。たとえば "*Difficulty*" は、10 文字あるように見えるが、実際には `ffi` が一体となった "*ffi*" という文字が利用されていて、8 文字しかない。PDF を解読すると、`(Di\016culty)` と出力されている。`\016` が "*ffi*" である。これはテキストイタリック cmti10 の場合である。数式イタリック cmmi10 の場合は "$Dif\,ficulty$" となって、一体どころか、離されている。数式イタリックは、関数や変数など 1 文字で独立しているものを書くフォントであり、くっついては困るのである。

ギリシア文字の場合はどうなるだろうか？$\Sigma\sigma$ は

```
/F8 9.9626 Tf 148.712 707.125 Td [(\006)]TJ
/F11 9.9626 Tf 7.196 0 Td [(\033)]TJ
```

となる。フォント `/F8` の `\006` が Σ で、フォント `/F11` の `\033` が σ である。

どのフォントの何番目にどんな文字が入っているかを示すフォントテーブルは、次のようにして得られる。

```
> pdftex testfont
Name of the font to test = cmti10
Now type a test command (\help for help):)
*\table\end
```

`testfont.tex` は TeX に標準的にインストールされているファイルで、これを plain TeX にかけると、フォント名を尋ねられる。そこで見てみたいフォント名、たとえば `cmti10` を入力して ENTER すると、今度はテストコマンドを聞かれるので、`\table\end` と入力して ENTER する。すると、`testfont.pdf` にフォントテーブルが出力される。表 1 は、その表の上半分をコンパクトに再現したものである。「'」は TeX で 8 進数を表す。'013 から先が大幅に違うことがわかる。

この表から、先ほど出てきた *ffi* は cmti10 の'016 であり、Σ と σ は cmmi10 でそれぞれ'006 と'033 であることがわかる。PDF を解読するソフトの多くは、このような TeX のフォントによる文字の違いに対応していないので文字が消えたり化けたりする。

表示された PDF の対応する部分

We define $V^*_{\mathcal{D}}(f_n)$ as $V^*_{\mathcal{D}}(f_n) = \lceil \delta V'_{\mathcal{D}}(f_n) \rceil$ if $V'_{\mathcal{D}}(f_n) > 0$ and $V^*_{\mathcal{D}}(f_n) = \lfloor \delta V'_{\mathcal{D}}(f_n) \rfloor$ otherwise,

xpdfbin-win-3.03 の `pdftotext -raw`

```
We define V D(fn) as V D(fn) = VD(fn) if VD(fn) > 0 and V D(fn) = VD(fn) otherwise,
```

PDFMiner-20110515 の `pdf2txt.py` （ギリシア文字 δ や合字 `fi` はユニコード）

```
WedefineV*D(fn)asV*D(fn)=dδV0D(fn)eifV0D(fn)>0andV*D(fn)=bδV0D(fn)cotherwise,
```

PDFBox-1.7.1 の ExtractText

```
We define V ?D(f n) as V ? D(fn) = dδV’ D(fn)e if V ’ D(fn) > 0 and V ? D(fn) = bδV’ D(fn)c
otherwise,
```

InftyReader-2.9.4.1

```
We define $V_{D}^{*}(f_{n})$ as $V_{D}^{*}(f_n) = \lceil\delta V_{D}’(f_{n})\rceil$ if
$V_{D}’(f_{n})>0$ and $ V_{D}^{*}(f_{n})= \lfloor\delta V_{D}’(f_{n})\rfloor$ otherwise,
```

BackToLaTeX

```
We define V * calD ( f n) as V * calD ( f n) = \lceil \delta V ’ calD ( f n) \rceil if
 V ’ calD ( f n) > 0 and V * calD ( f n) = \lfloor \delta V ’ calD ( f n) \rfloor otherwise,
```

図 1: BackToLaTeX と他のソフトの出力の比較

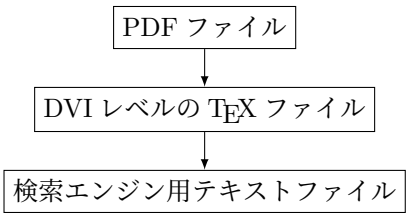

図 2: BackToLaTeX の構成

```
({\cmmi f}{\scriptsize \cmmi n}) as {\cmmi V }{\scriptsize
\cmsy \(*\)\({\cal D}\) }({\cmmi f}{\scriptsize \cmmi n})
= {\cmsy \(\lceil \) }{\cmmi  \(\delta\) V }{\scriptsize
\cmsy \(’\)\({\cal D}\) }({\cmmi f}{\scriptsize \cmmi n})
{\cmsy \(\rceil \)  }if {\cmmi V }{\scriptsize \cmsy \(’\)
\({\cal D}\) }({\cmmi f}{\scriptsize \cmmi n})
{\cmmi \(>\) }0 and {\cmmi V }{\scriptsize \cmsy \(*\)
\({\cal D}\) }({\cmmi f}{\scriptsize \cmmi n}) = {\cmsy
\(\lfloor \) }{\cmmi  \(\delta\) V }{\scriptsize \cmsy
\(’\)\({\cal D}\) }({\cmmi f}{\scriptsize \cmmi n}){\cmsy
\(\rfloor \)  }otherwise,
```

図 3: DVI レベルの TeX コマンドによる解読

そこで、各フォントごとに何番目の文字に対してどういう文字列を出力したいかを表にまとめておけば、その通りに出力してくれるプログラム BackToLaTeX を作成した。鈴木らの論文 [7] に出てくる数式と、この数式を BackToLaTeX や他のソフトで解読と比較したものを図 1 にまとめておく。改行は、見易さと紙面の都合により適宜変更している。

この例からわかるように、数式の文字認識を専門とする InftyReader の出力にはまだ及ばないが、BackToLaTeX の出力は、他の PDF 解読ソフトに比べれば、かなり情報を保存していることがわかる。

現在の BackToLaTeX の構成は図 2 のようになっている。一度、TeX の出力である DVI レベルに対応する、フォント名と文字を羅列した TeX コマンドの羅列を作り、そこから検索エンジン用テキストファイルを抽出している。

当初は、PDF からいきなり LaTeX レベルのコマンドを作ろうとしたのだが、使われている文脈がわからないと、LaTeX レベルのコマンドを出力できないことに気づいた。たとえば、「•」という記号は、箇条書きにも使われれば、演算子としても使われる。箇条書きであれば、`\item` を出力すべきだが、演算子として使われているのであれば、`\bullet` を出力すべきであろう。

しかし、検索エンジンのインデックスに入れるためのテキストを作るだけであれば、当面、このような区別をする必要はない。そこで、まず、PDF をフォントと文字のコマンドの羅列に復元することにした。

しかし、アブストラクトや参考文献を本文とは別フィールドとして登録したいとなると、LaTeX レベルの解析が必要になる。今の実装では、参考文献を削除しただけで、これらのフィールド分けはまだ行ってい

ない。インデキシングするテキストから参考文献を削除したのは、その研究の中身と直接関係のない内容でマッチするのを防ぐためである。

5 ACL Anthology への適用

以上のようにして作成した BackToLaTeX を ACL Anthology のデータに適用して検索インデックスを作っている。2000 年より以前の論文はスキャナーで読み込まれており、OCR で文字認識されたデータが提供されているうえ、本研究の目的は、最新の研究動向を把握するところにあり、これらは対象外としている。また、MT Summit, AMTA, NTCIR などは、ACL Anthology に含まれていないが、著者の研究上、参照することが多いので、これらに `Qyy-xxxx` という形式の独自の ID を割り振って加えている。現在、約 1.6 万件の論文が Lucene のインデックスに入っており、Let's Note 1 台で動く。

出力は、単なる検索結果ではなく、研究動向や論文の間の関係がなるべくわかりやすくなるような可視化を考えて実装中であり、これについても近々報告したい。

なお、PDF から抽出したテキストから、著者やタイトルを抽出しようとすると、所属などが邪魔になるので、これらは ACL Anthology の HTML ファイルから抽出している。

参考文献は引用解析を行い、対応する論文の PDF がパソコン上にある場合は、リンクを張ってある。

2012 年 9 月の「特許文書の機械翻訳結果評価方法検討会」で、翻訳自動評価方法の最近の研究動向について報告したが [1] が、本システムを活用して、効率よく調べることができた。

ACL Anthology の PDF は、さまざまなツールで生成されており、それぞれのツールが、まったく違うパターンの PDF プログラムを出力しているので、それらの差異をカバーするために、BackToLaTeX のプログラムが複雑化してきているのが問題である。

6 おわりに

PDF から数式を復元しようと思うと、文字列の解読だけでは不十分である。たとえば分数式 $\frac{a}{b}$ の横線は文字ではないため、文字列だけからは a/b との区別がつかない。今後は線を含めた解読が課題である。

なお、本研究にあたっては、岡山県立大学独創的研究助成費の支援を受けた。

参考文献

[1] 磯崎秀樹. 最近の自動評価法の研究動向と RIBES, 2012. http://aamtjapio.com/kenkyu/discussion01-01.html.

[2] Hideki Isozaki, Tsutomu Hirao, Katsuhito Sudoh, Jun Suzuki, Akinori Fujino, Hajime Tsukada, and Masaaki Nagata. A patient support system based on crosslingual IR and semi-supervised learning. In *Proceedings of SIGIR-2009 Workshop on Information Access in a Multilingual World*, pp. 59–61, 2009.

[3] Hideki Isozaki, Katsuhito Sudoh, and Hajime Tsukada. NTT's Japanese-English cross-language question answering system. In *Working Notes of the NTCIR Workshop Meeting (NTCIR)*, pp. 186–193, 2005.

[4] Hideki Isozaki, Katsuhito Sudoh, Hajime Tsukada, and Kevin Duh. HPSG-based preprocessing for English-to-Japanese translation. *ACM Transactions on Asian Language Information Processing*, Vol. 11, Issue 3, , 2012.

[5] Giovanni Yoko Kristianto, Minh-Quoc Nghiem, Yuichiroh Matsubayashi, and Akiko Aizawa. Extracting definitions of mathematical expressions in scientific papers. In *International Organized Session, Proceedings of the 26th Annual Conference of the Japanese Society for Arftificial Intelligence*, 2012.

[6] Minh-Quoc Nghiem, Giovanni Yoko, Yuichiroh Matsubayashi, and Akiko Aizawa. Automatic approach to understanding mathematical expressions using MathML parallel markup corpora. In *International Organized Session, Proceedings of the 26th Annual Conference of the Japanese Society for Arftificial Intelligence*, 2012.

[7] Jun Suzuki, Hideki Isozaki, and Masaaki Nagata. Learning condensed feature representations from large unsupervided data sets for supervised learning. In *Proc. of the Annual Meeting of the Association of Computational Linguistics (ACL)*, pp. 636–641, 2011.

[8] John Whitington. PDF 構造解説. オライリー・ジャパン, 2012.