Reconstructing Financial Statements

Frank G. Bennett, Jr.[1]

Nagoya University, Nagoya 464-8601, Japan,
bennett@law.nagoya-u.ac.jp,
WWW home page: http://gsl-nagoya-u.net/faculty/member/gsliF_Bennett.html

Abstract. This paper introduces a tool for the reconstruction and validation of categorized totals embedded in untrusted and unformatted text, such as OCR scans of financial statements. The tool is a spinoff of academic research into the funding of Japanese third-sector organizations, the annual reports of which are frequently published reports in the form of PDF files containing document images. A number of techniques at string- line- and document-level are used to resolve ambiguities and obtain the greatest possible recovery rate for the underlying data, while excluding the content of untrustworthy documents from the final sample. In a preliminary trial "in the wild", the tool has returned validated income totals for 47.9% of the documents in a heterogeous set of 2205 annual reports.

1 The Problem

In common with litigation, empirical research in Japan is at times afflicted by what one might call the "last byte nuisance", whereby information is disclosed in forms that are costly to analyze in bulk. In the last century, it was the common practice, when disclosure was required by administrative law, for government agencies to offer documents for reading at a single location, without provision (sometimes without permission) for copying.[1] Such practices have their modern corollary in the copy-protected, print-restricted PDF wrapper for documents presented in graphical form.

While PDF content restrictions are trivial to handle, and OCR tools can produce an electronic text version of a document, raw OCR output cannot be trusted as a basis for analysis. This is a particularly telling issue in the case of financial data, where small recognition errors can greatly affect the apparent meaning of document content. In this case, means of both reconstructing the essential structure of the document, and of validating the figures themselves can greatly increase the value of such data as a basis for answering research questions. This paper presents a tool designed for this purpose. While the incentives for the drafting of this tool are "entirely academic", the methods outlined here

[1] A particularly odd example of such "managed transparency" is the now defunct practice of restricting notetaking by spectators to court proceedings. *See, e.g.* Supreme Court judgment of Mar. 8, 1989 (Case of note-taking in court without permission), 1299 Harei Jiho 341.

can be applied to any large archive of similar data, and the source code of our implementation is available for download under an open source license. [2]

The paper begins with an introduction of the research context, to provide a sense of the development environment and the objectives of the researcher. This is followed by an outline of the data collection infrastructure into which the tool fits, and brief technical overviews of programming logic at the string, line, and document levels of processing. The paper concludes with comments on the results of initial testing, and the on the focus of future development.

2 The Research Context

Japanese rules on non-profit corporations changed significantly under law reforms taking effect in 1999 and in 2008. Prior to this time, most non-profits depended on specific approval by a national-level ministry as a condition of their creation. The first significant loosening of this restriction came with legislation defining a new legal person (the so-called "NPO law"), suitable for small or volunteer-driven associations, which can be created with no investment of capital, upon the satisfaction of a set of objective requirements. The legislation has been well received, and this year the advance approval requirement was removed from all non-profit entities.

The Japanese third sector is currently very small, but expanding at a much faster rate than the surrounding economy. The first 10 years of the NPO law have seen the founding of 35,000 entities, with an average of about 15 more being added every working day. Following the recent extension of favourable tax treatment to qualifying non-profits, we can expect this trend to gather greater force in future years.

Non-profit organizations can be used for various purposes. In many industrialized societies, they provide a platform for the lobbying of politicians and bureaucrats. In Japan they have heretofor served, on the contrary, as an extension of government administration. Such organizations can also provide an internal support network to their membership, with a degree of independence from government and the surrounding community. The mere fact that the number of these organizations is on the rise therefore tells us little about the impact that they have on government and society. [1] To explore this research question, we need to explore what makes them tick, and the obvious way to do so is through their financial reports.

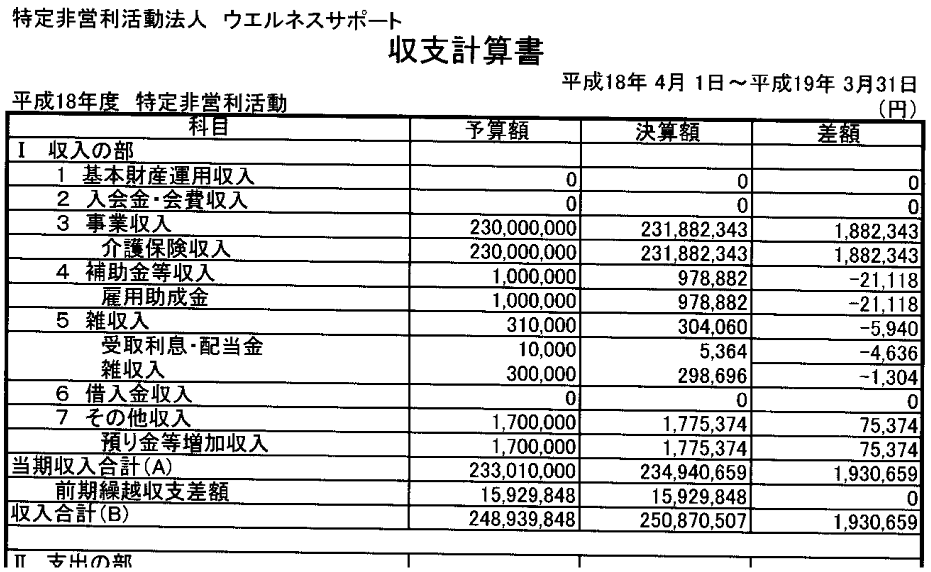
Each non-profit is required by law to file a financial report each year with the government authority (local or national) responsible for its incorporation. Government is required to archive these reports for a period of three years, and to make them available to members of the public on request. Most of the relevant government bodies fulfill this requirement in the traditional way described above, by providing a reading room where concerned citizens can request and view the documents in paper form. [3] But a significant number of local authorities now

[2] The URL for download is http://gsl-nagoya-u.net/appendix/software/renumerate.

[3] See, for example, the disclosure policy of Metropolitan Tokyo. [2]

provide these annual reports via the World Wide Web, in the form of graphical images encased in a PDF wrapper. These latter documents are the primary target of the digitization strategy discussed here.

Fig. 1. Sample double-entry financial statement



特定非営利活動法人　ウエルネスサポート

収支計算書

平成18年 4月 1日～平成19年 3月31日

平成18年度　特定非営利活動

(円)

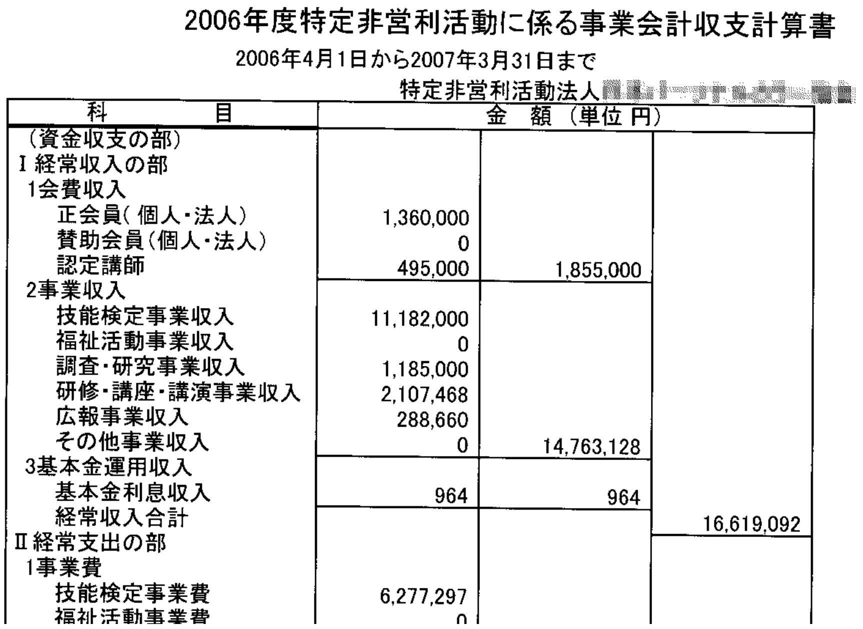
科目	予算額	決算額	差額
I　収入の部			
1　基本財産運用収入	0	0	0
2　入会金・会費収入	0	0	0
3　事業収入	230,000,000	231,882,343	1,882,343
介護保険収入	230,000,000	231,882,343	1,882,343
4　補助金等収入	1,000,000	978,882	-21,118
雇用助成金	1,000,000	978,882	-21,118
5　雑収入	310,000	304,060	-5,940
受取利息・配当金	10,000	5,364	-4,636
雑収入	300,000	298,696	-1,304
6　借入金収入	0	0	0
7　その他収入	1,700,000	1,775,374	75,374
預り金等増加収入	1,700,000	1,775,374	75,374
当期収入合計(A)	233,010,000	234,940,659	1,930,659
前期繰越収支差額	15,929,848	15,929,848	0
収入合計(B)	248,939,848	250,870,507	1,930,659
Ⅱ　支出の部			

3 The Documents

Each document in the target set is in one of two common formats, referred to here as the *double-entry* and the *running-totals* formats. Samples given in Figures 1&2 show the respective structures of these document types. In both types of income statement, totals are embedded in the series of figures, reading the document from left to right and top to bottom. In *double-entry* format, a total occurs at the end of each line. In *running-totals* format, totals occur at irregular intervals down the page, with a grand total at the bottom of the run of numbers.

These patterns are obvious to the human eye, but as Figure 3 illustrates, much of the information on which a human reader relies can be lost in OCR output text. Our aim is to exploit embedded totals as a checksum hint for the repair of OCR damage and the reconstruction of the original document content.

Fig. 2. Sample running-totals financial statement



2006年度特定非営利活動に係る事業会計収支計算書

2006年4月1日から2007年3月31日まで

特定非営利活動法人[redacted]

科　　目	金　額（単位 円）		
（資金収支の部）			
Ⅰ 経常収入の部			
1会費収入			
正会員（個人・法人）	1,360,000		
賛助会員（個人・法人）	0		
認定講師	495,000	1,855,000	
2事業収入			
技能検定事業収入	11,182,000		
福祉活動事業収入	0		
調査・研究事業収入	1,185,000		
研修・講座・講演事業収入	2,107,468		
広報事業収入	288,660		
その他事業収入	0	14,763,128	
3基本金運用収入			
基本金利息収入	964	964	
経常収入合計			16,619,092
Ⅱ 経常支出の部			
1事業費			
技能検定事業費	6,277,297		
福祉活動事業費	0		

4 The Recognition Engine

In our specific project, we have settled on the *Tesseract* OCR engine for use in the first step of the processing chain. This product was developed by Hewelett-Packard between 1985 and 1994, [4] was later released under an open source license in 2005, and is now under active development at Google. [5], [9] Alone among open source systems, *Tesseract* has been subjected to rigorous competitive testing, ranking third overall in a test of eight leading contemporary systems carried out by the Information Science Research Institute in 1995. [3]

Tesseract is fully trainable and is able to achieve a high recognition rate against heterogenous text. The output against a sample document after training is shown in Figure 3. Two features of this target data will be immediately apparent. Most obviously, much of the page layout information has been lost. Furthermore (as Japanese readers will immediately note), the system recognizes only a limited set of Japanese characters, plus digits and symbols found in the numeric data. In fact, there is an upside to both of these limitations.

Page layout information is extremely useful when interpreting documents with a known homogeous structure, but the target documents in this case are in varying formats. The position of a number on the page, cannot be used as a primary hint to the significance of a particular number. Loss of layout information forces us to concentrate on the pattern of sums in the number series, which is a more certain means of validation. The limited scope of recognition is also a beneficial constraint, in that it avoids some of the ambiguous glyphs, such as "one" and "lowercase letter *el*" (`1` and `l`), or "zero" and "capital letter *oh*" (`0` and `O`), that plague OCR systems operating on English/Roman documents.

The Japanese phrases used in our early recognition trials are presented in Table 1. In all, the engine was trained for a total of 157 common characters and symbols, including the digits zero through nine. Because *Tesseract* is aggressive about recognizing individual character blobs even at very low confidence levels, providing additional "noise" characters in the training set reduces the possibility of false positives when post-processing the OCR output, while staying well within the capacity limits of the current version of *Tesseract*. The boxed areas highlight successful recognition of target phrases, and are used to identify the category of individual items of income during processing.

Table 1. Minimal character set for OCR training

Characters	Romanization	Translation
収支	*shūshi*	income and expenses
会費	*kaihi*	dues
事業	*jigyō*	project
寄付	*kifu*	gift
助成	*josei*	grant (private)
補助	*hojo*	grant (public)
利息	*risoku*	interest
経常	*keijō*	ordinary
収入	*shūnyū*	income
合計	*gōkei*	total
支出	*shishutsu*	expenses

5 Post-Processing

To prepare the text for string-level analysis, a series of regular expression substitutions are applied to normalize the text, repairing "impossible" character combinations. In preparation for later line-level analysis, the document is split into individual lines, and lines are fed to line-level resolvers. Line resolvers split the line into the string-level units that form the basis of document resolution.

Processing at the string, line and documents levels is outlined below. While the author has no formal training in computer science, the discussion should be accessible to readers knowledgeable in `C++`, `Java`, and other object oriented languages.

Fig. 3. Result of recognition (boxes and explict space markers added)

1 2006年度特定非営利活動に係る$\boxed{\text{事業}}$会計$\boxed{\text{収支}}$計算書
2 2006年4月␣1␣日から2007年3月␣31␣日まで
3 特定非営利活動5ま人␣日␣事】␣《一・）十ル方5一構会
4 科␣目␣額␣(単位␣円)
5 （>␣収␣の，）
6 1経常$\boxed{\text{収入}}$の部
7 1$\boxed{\text{会費}}$ $\boxed{\text{収入}}$
8 年会員(␣1団人・5ま人）␣1␣360␣000
9 賛助会員(借人・貫五人)␣0
10 書講定講日市␣495,000␣1␣855,000
11 2$\boxed{\text{事業}}$ $\boxed{\text{収入}}$
12 講講構定$\boxed{\text{事業}}$ $\boxed{\text{収入}}$␣11␣182,000
13 補業位活動$\boxed{\text{事業}}$ $\boxed{\text{収入}}$␣0
14 講費・町開$\boxed{\text{事業}}$ $\boxed{\text{収入}}$␣1,185,000
15 町借・講座・講講$\boxed{\text{事業}}$ $\boxed{\text{収入}}$␣2␣107␣468
16 座講$\boxed{\text{事業}}$ $\boxed{\text{収入}}$␣288,660
17 受の地␣$\boxed{\text{収入}}$␣0␣14,763,128
18 3講ま金道開$\boxed{\text{収入}}$
19 講取金$\boxed{\text{利息}}$ $\boxed{\text{収入}}$␣964␣964
20 経常$\boxed{\text{収入}}$ $\boxed{\text{合計}}$␣16,619,092
21 0経常$\boxed{\text{支出}}$の部
22 1$\boxed{\text{事業}}$費
23 位講構定$\boxed{\text{事業}}$費␣6,277,297
24 補業ま活動$\boxed{\text{事業}}$費␣0

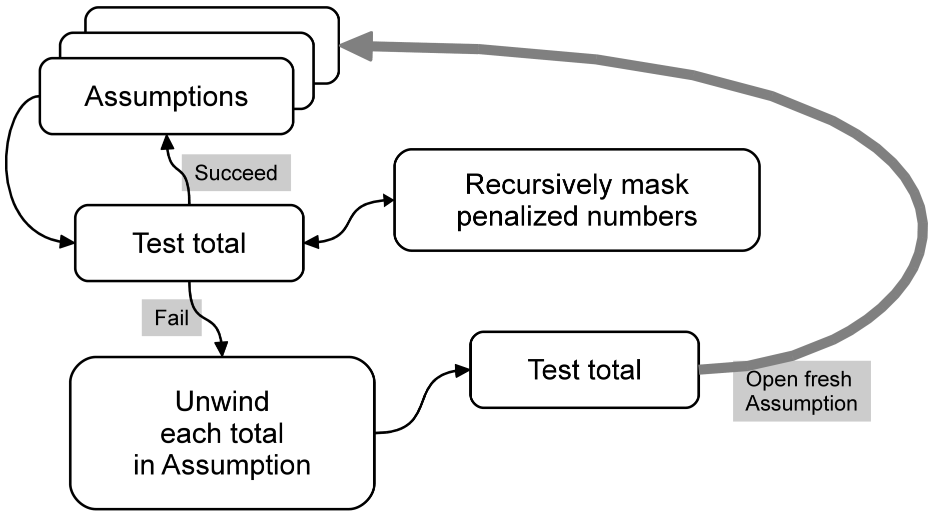
5.1 String Level Resolution

After normalization, each line is scanned for relevant strings with a regular expression parser to extract a list of relevant string elements. These elements (for example, at line `12` of Figure 3, the strings 事業, `11`, and `182,000`) are then identified as either relevant text strings (such as 事業) or numeric strings (`11` and `182,000`). Relevant text strings are associated with a category, and this is recorded for reference.[4] The text string itself is discarded.

Numeric strings come in two flavours, depending on whether they can be interpreted as a single unambiguous unit, or have multiple elements that introduce ambiguity. For simplicity of processing, all numeric strings are first cast as

[4] A record of the current category is maintained in a `categoryHinter` state object that persists across lines in the splitting process.

Fig. 4. Reconstructing totals within data stream


Assumptions
Succeed
Recursively mask penalized numbers
Test total
Fail
Unwind each total in Assumption
Test total
Open fresh Assumption

`ambiguousCluster` objects, and each contiguous run of numbers and commas within them is instantiated as an `ambiguousNumber`. In the example at line `12` of Figure 3, the numeric strings `11` and `182,000` constitute an `ambiguousCluster` composed of two `ambiguousNumber` objects.

The `ambiguousNumber` class is the fundamental unit for subsequent processing, and carries important metadata used in the remainder of the resolution process, and in final reports generated from validated number sets.

5.2 Line Level Resolution

After creating `ambiguousNumber` objects and wrapping them in `ambiguous-Clusters`, an attempt is made at the line level to resolve ambiguities in multi-element clusters. The version of the tool used in our project applies two line-level resolvers in separate attempts at "disambiguation". The first applied is the *double-entry* resolver. As implemented for the first trial, this operates exclusively on `ambiguousCluster` objects containing three or more numbers, based on the hypothesis that three numbers exist in the line, and that the third represents the difference of the first two. This resolver always returns either the second number in this sequence, or an empty line. (There was a difficulty with this assumption, as explained in the conclusion.)

The second resolver, aimed at the *running-totals* format, operates on all numbers in the line, on the hypothesis that any multi-element `ambiguousCluster` could consist of either one or two numbers. In the latter case, the second number would represent the total of the preceding element, plus some unknown number of elements on previous lines of the document. Multi-element clusters are therefore joined, beginning from the last item, such that the value of the joined number is *not greater than* the value of its prececessors joined. This resolver always returns all values in the line; but because the algorithm exits when the not-greater-than condition is reached, the resulting line may still contain unresolved multi-element `ambiguousCluster` objects when it is passed on for document-level resolution.

5.3 Document Level Resolution

Document level processing is the final stage of resolution. At this stage, all potential ambiguities in the text must be excluded, with the return of a single result representing the numbers in the original document and their characteristics. The tool returns this data in the form of a list of `ambiguousNumber` objects, in the same order as in the original document, with the income category and the number's role as an ordinary numeric component, total or grand total stored on the object as metadata. From this final list, spreadsheets and other reports can be generated.

In the case of *running-totals* line resolution, ambiguities may persist in the set of numbers returned for document level resolution. As more refined methods have been exhausted at this point, the tool now resorts to the delicate application of brute force. One complete, unambiguous series of numbers is produced for every possible combination of joins within the multi-element clusters remaining in the document. These number sets are the basis for final resolution of the document.

The preferred method of resolution at the document level is the mathematical pursuit of totals. The tool steps through the numbers in each `Assumption`, maintaining a running total of their values. When a number is encountered with equals this total, that number is treated as a `total`, and its value is added to a running grand total. When a final number is encountered which matches this grand total, its component `ambiguousNumber` objects are composed into a list of `candidates`, and processing of that `Assumption` finishes.

At signficant values, the possibility of a false grand total is so small that it can be ignored for our purposes. However, at very small values, false totals are sometimes returned. After the processing of all `Assumptions` is complete, the tool returns the set of numbers for the `candidate` which results in the largest grand total. Because this recovery method can be foiled by OCR corruption of the numbers constituting the true total, very small values (those less than 5000) are excluded from the result.

Two further potential sources of failure or error remain. As shown in Figure 1, an income statement often contains numbers irrelevant to the finance figures to be extracted. Not all of these can be safely excluded at the initial stage, before `ambiguousNumber` objects are instantiated. Their presence will block identification of the correct total. To address this problem, each `ambiguousNumber` object is affixed with a `penalty` value based on its value and its position within the document.[5] After each failed attempt to achieve a valid total, the number with the lowest negative penalty value is discarded, and the totals process is retried.

[5] The assignment of penalties is controlled by the `penaltyEngine` class, which can be customized to fit the characteristics of a given document set.

A second issue is the possibility of falsely identified totals within the number set. For example, in the sequence `111`, `222`, `333`, `666`, `666`, the grand total of `666` will not be identified if the third number in the sequence is treated as a subtotal (in that case, the last `666` would be recognized only as a subtotal of the single number before it). To obtain the best internally consistent set of totals, each time a total is identified, an additional `Assumption` is generated, in which the same number can only be treated as a non-total value. This alternative `Assumption` is placed in the processing queue, and its later processing will result in identification of the correct grand total and its components.

This extended iterative attempt at checksum validation is ineffective for documents that report zero income. Because there are significant numbers of such documents in our target data set, the tool attempts to identify this special category of documents when the following conservative conditions are satisfied:

– The number is a single zero;
– The number occurs on a line containing the phrase for “Total” (合計);
– The phrase for “Total” has not previously been encountered in the document; and
– There are more single zeros than other values among the preceding numbers in the current `Assumption`, excluding those excluded by the penalty mechanism.

Comparison of the results of this heuristic against original documents has shown this to be a highly reliable indicator that the document does indeed report zero income. The program flow for processing `Assumption` objects is illustrated in Figure 3. For a more complete description of the logic, interested readers may wish to refer to the `PursueTotals` class in the source code of the tool.

6 Conclusion

As an initial trial of the tool during the preparation of this paper, PDF files containing the 2205 most recent financial statements filed by national-level Japanese NPO non-profits were downloaded and processed. This returned 863 validated totals (39.1% of the total document set) with elements of income classified by category. Zero income was identified with confidence in a further 194 documents (8.8% of the set), for a total overall recognition rate of 47.9%. For our statistical purposes, this is more than adequate.

Three important lessons emerged from this initial trial. First, it appears that the logic applied to statements in the *double-entry* format described above failed comprehensively, and must be revisited. The assumption that all lines in such statements consist of three numbers was mistaken. By convention, columns in which a number is zero is often left blank. This gives rise to a further layer of ambiguity that must be addressed when performing this style of line validation.

Many resolution failures resulted from OCR corruption of the digits, an error which no post-processing tool will be able to remedy. The OCR training data used for the trial can and should be improved by extracting character images from the set of failed documents. This can be expected to substantially improve the recognition rate against the specific document set targeted by our research. Such improvements in the OCR layer are, of course, specific to our particular to our particular project and document set.

Because the categorization of income is based on pattern matching of the "hinting" phrases in the OCR output, there are occasional errors in income categorization. Addressing these for ultimate publication of our data sets will require human intervention. However, because the numbers and totals in the validated returns are known to be correct, this final proofreading can be carried out quickly by human operators with minimal training, given a software interface designed for this specific purpose. Preparation of the necessary proofreading infrastructure is planned for the next phase of development.

This paper has discussed an application of open source OCR technology to financial records stored as graphic data. Techniques for recovering numeric structures and essential metadata from heterogenous documents have been outlined and tested. The results of an initial trial indicate that this method is useful as a means of recovering a significant proportion of such records for the purpose of statistical analysis. While the development of this tool has been driven by a specific social science research project, the code is public, the performance of the tool is proven, and the approach should prove useful as a means of recovering such data in many other areas, including discovery proceedings in the context of litigation.

References

1. F. Bennett & J. Whitney, "Activism in the Wild: A Preliminary Study of NPO Incomes," 法政論集 [Hōsei Ronshū] (to appear)
2. 東京都生活文化スポーツ局 [Metropolitan Tokyo, Bureau of Citizens, Culture and Sports], "NPO 法人情報公開 [NPO Information Disclosure]", Mar. 26 2008; http://www.seikatubunka.metro.tokyo.jp/index4files/etsuran.htm
3. S.V. Rice, F.R. Jenkins & T.A. Parker, "Fourth Annual Test of OCR Accuracy," Information Science Research Institute, University of Nevada, Las Vegas, 1995; http://www.isri.unlv.edu/downloads/AT-1995.pdf
4. R. Smith, "ReadMe: Important Information All Tesseract Users Need to Know," Oct. 11, 2007; http://code.google.com/p/tesseract-ocr/wiki/ReadMe
5. N. Willis, "Google's Tesseract OCR Engine is a Quantum Leap Forward," *Linux.com*, Sep. 28, 2006; http://www.linux.com/articles/57222
6. 特定非営利活動促進法 [Act for the Promotion of Non-Profit Activities], art. 1 (Law no. 7, Mar. 25, 1998).
7. 一般社団法人及び一般財団法人に関する法律 [Act Concerning Ordinary Civil Corporations and Ordinary Civil Foundations] (Law no. 48, 2006)
8. 公益社団法人及び公益財団法人の認定等に関する法律 [Act Concerning the Certification of Public Benefit Civil Corporations and Public Benefit Civil Foundations] (Law no. 49, 1998).
9. Post by Ray Smith, 10:53am, Jan. 19, 2008; http://groups.google.com/group/tesseract-ocr/ ("[W]e are actively developing (and using) Tesseract here at Google, and we are committed to putting our improvements back into the open source codebase.")