AUTOMATIC CODING OF PRINTED MATERIALS

JOHANN PETER MURMANN, ERNST HOMBURG,
RUUD GEVEN, Y. SEKOU BERMISS and ALFONZO FORGIONE

I. INTRODUCTION

Traditionally most researchers working with printed data sources have entered data by hand to convert it into electronic format. If a research project involves large amounts of data from similarly formatted sources – for example, when one tries to create a longitudinal database of directory information spanning many years – entering this data by hand is a very labour intensive and tedious task. We wanted to automate the coding of printed directory information in order to cut down the time it takes to transfer this information into electronic data. Once the data is in electronic format, it can be further analysed with a plethora of software packages ranging from Microsoft Excel, FileMaker, SAS and SPSS, depending on the needs of the particular researcher. The purpose of this technical paper is to share with other scholars in a clear and practical way the methods we developed for automating the coding of printed information.

II. SCHEMATIC OVERVIEW OF PROCESS

No automatic coding system of printed material can achieve 100 per cent accuracy. The goal is to achieve the highest accuracy possible (e.g. above 90 per cent) to reduce the time required to make manual corrections of the automatically coded data. Achieving high quality results using automatic coding procedures involves four or five steps, depending on the quality and complexity of printed source material.

1. Creating an electronic picture of each page of the printed document (described in section III of this paper).
2. Converting the picture into a text document using optical character recognition (OCR) software (described in section IV).

International Journal of Humanities and Arts Computing 1 (2) 2007, 151–185
DOI: 10.3366/E1753854808000244

3. Manual correction of the optical character recognition (if necessary) (described in section V).
4. Coding the raw electronic text to produce structured data that generates individual records with multiple variables into which data has been categorized (described in section VI).
5. Manual correction of imperfect coding results and other flaws (described in section VII).

III. CREATING AN ELECTRONIC PICTURE OF EACH PAGE OF THE PRINTED DOCUMENT

There are two ways to make electronic pictures of printed pages: using a scanner or using a camera. Initially we thought about buying a high quality scanner. However, after experimenting with a digital camera, we realised that a camera would be the better option for us.

a. Advantages of using a camera over a scanner

The main two reasons for choosing a camera over a scanner are *increased control* in the way the picture is taken and *higher speed*. If one uses a scanner, one cannot immediately see if a page or book is placed on the glass plate correctly. This often leads to a skewed picture and requires a second scan, making the process very time consuming when one needs to digitise large batches of text. Using a camera increases control because one can instantly see on the LCD screen what the camera will record.

The preview of the resulting image on the LCD screen also gives the user information on its quality. This is not possible with a present-day scanner because light easily gets between the page and the glass plate, especially when scanning books. This creates pages that are not uniformly light but contain darkened areas that will render text useless for OCR software.

The higher speed obtained in getting pictures of text by using a camera rather than a scanner can be illustrated comparing both technologies in use. After two seconds (taking and storing the picture), the camera will be ready to continue. The photographer will only have to turn a page, allow a few seconds to let the device focus, and then take the next picture. A normal consumer scanner, which would provide comparable quality, requires about one minute to record and store a page, not to mention the effort of turning the book around and adjusting its position before the next scan can be started.

Taking photographs of the pages of books instead of scanning them is a much better method to preserve valuable manuscripts. Placing books on a flat-bed scanner creates considerably more wear and tear, although this is less the case when using an admittedly expensive book-page scanner. For this reason, many libraries will not allow researchers to scan their carefully preserved books.

However, many will allow a researcher to take photographs of a book that is only lying in book-cradle.

We used the ***Fuji Finepix f20*** (£170).[1] This camera was the first camera we tried out and it sufficed for our purposes, making it unnecessary to experiment with others. One specific advantage this camera has over other cameras is the possibility of photographing at high ISO values, up to 1600, which are necessary if the lighting is dim. (Also, see in part III b: *How to take pictures of text with camera* for an elaboration on the practice of photographing text.)

b. How to take pictures of text with camera

OmniPage Professional, the OCR software we used to convert jpeg images to editable text, demands highly detailed images. The manual that accompanies OmniPage Professional 15 assumes that their clients use scanners for obtaining digital images and therefore advises the quality setting of 300 dots per inch (dpi) (p. 86). DPI is a quality measure for scanners, not for photographic images. We prefer the term 'pixel dimensions' to describe the detail of a picture taken by a digital camera.

A picture consists of small parts or pixels. The number of pixels that make up an individual character of text on the picture is a measure of the quality of the photograph. We tested the pixel dimensions of a scan, an A4 page, scanned at 300 dpi, which resulted in 2544*3440 pixels. The same picture taken with our camera resulted in 2136*2848 pixels (For the maximum our camera delivered, see below). The pixel number of such a photograph is thus less than a 300 dpi scan. But photographing a smaller page, for example A5 format, would also yield a 2136*2848 pixel picture, the same amount of pixels. This is not the case with a scanner, where we would have to cut the A4 picture in half to A5 format, resulting in 2544*1720 pixels.

To summarise, photographing an A4 page will not reach a pixel count corresponding to 300 dpi, but photographing A5 will go beyond 300 dpi. Keep in mind that the format you are photographing influences the detail of your images. In our particular case, we photographed A5 format pages and thereby reached a sufficiently high picture quality for the OCR process.

A Warning: OmniPage also states in its manual that when pictures have too high a resolution, that is, far beyond 300 dpi (2544*3440 pixels for A4), OmniPage Professional 15 will deliver worse OCR results (p. 83). We think such high values are not likely to be reached with a mid-priced consumer camera soon. Nevertheless, it is important to note that when too many pixels are involved in forming a text character, it will have the same effect as too few. Photographing a smaller area, for example, increases the pixels involved in making up a character. OmniPage will have trouble processing highly detailed characters. Zooming in too much will increase errors in the character recognition, making it later

necessary to spend significantly more time to catch these errors. A balance must be found by testing a few pictures. An alternative would be to adapt the camera-captured image through image processing software to reduce its dots per inch to a level that suited OmniPage.

We made photographs with the Fuji Finepix f20 at its maximum resolution of 2848*2136 pixels per picture. More specifically, we used the following settings with the camera. These settings mostly will apply for other cameras as well.

- Photography Mode TEXT ON. This function heightens the contrast, includes the MACRO ON setting, and disables flash.
  - Heightened contrast improves the quality of conversion from image to text.
  - Macro photography enables the camera to focus close to the page. Out of focus jpeg images are useless for the OCR process.
  - Flash worsens contrast because ink tends to reflect the strong light to the lens when one holds a camera up close to the page. The reflected light will render the character very difficult to process correctly. This mostly is a problem at the centre of the image, where the light is the brightest. Always disable flash.
- ISO settings AUTO.
  - Set the camera on automatic but do not rely on the capabilities of your camera to photograph in poor lighting conditions. A higher ISO value does correct for dimmer light, but it increases shutter time, which could lead to out of focus images. Even slightly out of focus images are useless for the OCR process. Always take pictures with natural light, preferably with the light source (a window) directly in front of what is being photographed.
- Black and White ON. Again the contrast is heightened.
- Digital Zoom OFF. A digital zoom brings the object closer at the expense of resolution.
- Quality HIGHEST. Quality stands for the way an image is saved on the memory card of the camera. The highest quality uses up the most space, so make sure to buy a 'large' memory card. We used a Fujifilm xD-Picture Card, 512 MB. Using the highest quality setting, you can make 170 pictures with the FinePix f20 camera before you run out of storage space.

The aforementioned settings don't automatically result in good pictures. The following guidelines should also be heeded when trying to photograph large numbers of pages.

- Light
  - Find a spot where a table can be placed in front of a window. This assures a light source in front of the camera. This way the shadow of

the photographer or that of the camera, will not interfere with the picture that is being taken.
  - Natural light is important for the quality of the focus in the picture. Shutter times decrease with more light, which diminishes any influence the movements of the photographer might have on the picture.
  - Every effort to get lighting conditions right is earned back later in the proofreading of the processed images.
- The page
  - In the case of flat-bed scanners, the weight of the book pushes the page you want to scan flat on the glass plate. This doesn't happen with photography[2]. When books lay open to the page being photographed, especially tightly bound or thick books, their pages display curvature. This results in skewed pictures. Clasping the pages and tightening them with weight can solve this. A primitive method is simply to weigh the book down with other books. In the case of fragile books, this will not be possible. Book cradles can position fragile books as needed here, without damaging them. Book cradles specifically designed for this purpose can be found online:
    http://new.academicimaging.com/bookcradles.html[3]
  - When you choose to make a clasping-like contraption with weights, first photograph the even (or uneven) pages, then change to the uneven ones (or the other way round) so you do not have to move the book and the contraption around it.
  - Keep track of what id numbers the camera assigns to the images. This information will be vital when you are filing your collection.
  - We photographed one page (similar to A5 format) per picture. It's important to fill the image with as much information as possible. Every inch left white in the margins results in fewer pixels making up the characters. To achieve best results, hold the camera so that the length of the book is parallel to the bottom of the camera.
- Zoom.
  - Always use optical zoom (disable digital zoom). With macro setting on and the camera in the most zoomed out position and up close to the page, the image produced will be somewhat rounded, typically at the upper and lower parts of the picture. This makes the OCR process slightly more difficult. Zoom in a little and hold the camera some distance (approximately 50 centimetres) from the page. Some curvature will always remain, but minimizing it is important. Using too much zoom renders the image more sensitive to the movement of the photographer's hand and can bring the picture out of focus.

- Camera Placement
  - When taking a large number of photographs it is advisable to use a camera tripod to hold the camera in place. This provides two benefits. First, it relieves the researcher of having to hold the camera for hours at a time while photographing, which after extended periods may lead to blurry or unfocused pictures. Second, once you have taken the necessary precautions regarding lighting and the set up of the source text the use of a tripod guarantees better photographic consistency of the text. You will need a tripod in which the centre column can swivel to a horizontal position, enabling the researcher to point the camera down towards the book. In addition, you will need a 3-way head for the tripod. A 3-way head allows optimal rotational movement of the camera to get the best angle for a photograph.

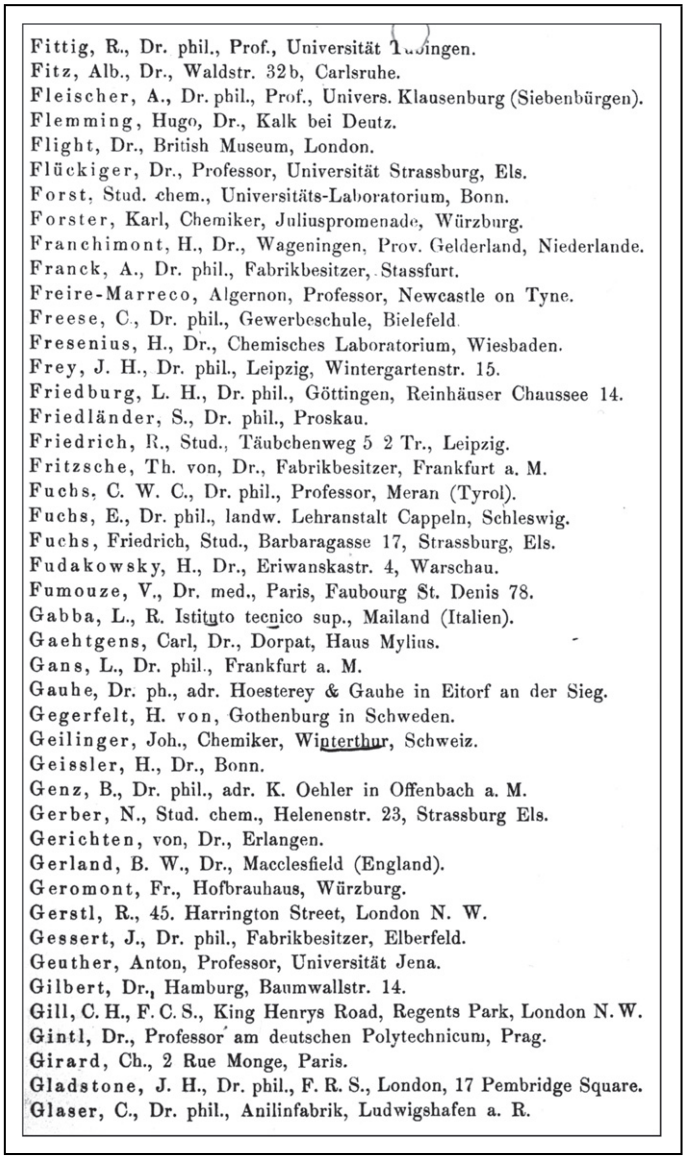
IV. CONVERSION OF PICTURE TO A TEXT DOCUMENT USING OCR SOFTWARE

Usually the best OCR software available will dramatically improve the accuracy of the text generated from the printed page. We used *OmniPage Professional* V15 (retailing for about £190).[4] The program allows one to process multiples pages all with one start command. You can tell the program to take all picture files in a folder of your hard disc and generate text files from them.

a. Example of the German directory pages

Before we explain how OmniPage converts jpeg images to editable text files, we will describe the type of projects for which we developed this procedure. In particular, we will explain what kind of problems had to be solved in order to process the membership lists of the German Chemical Society between 1868 and 1914.

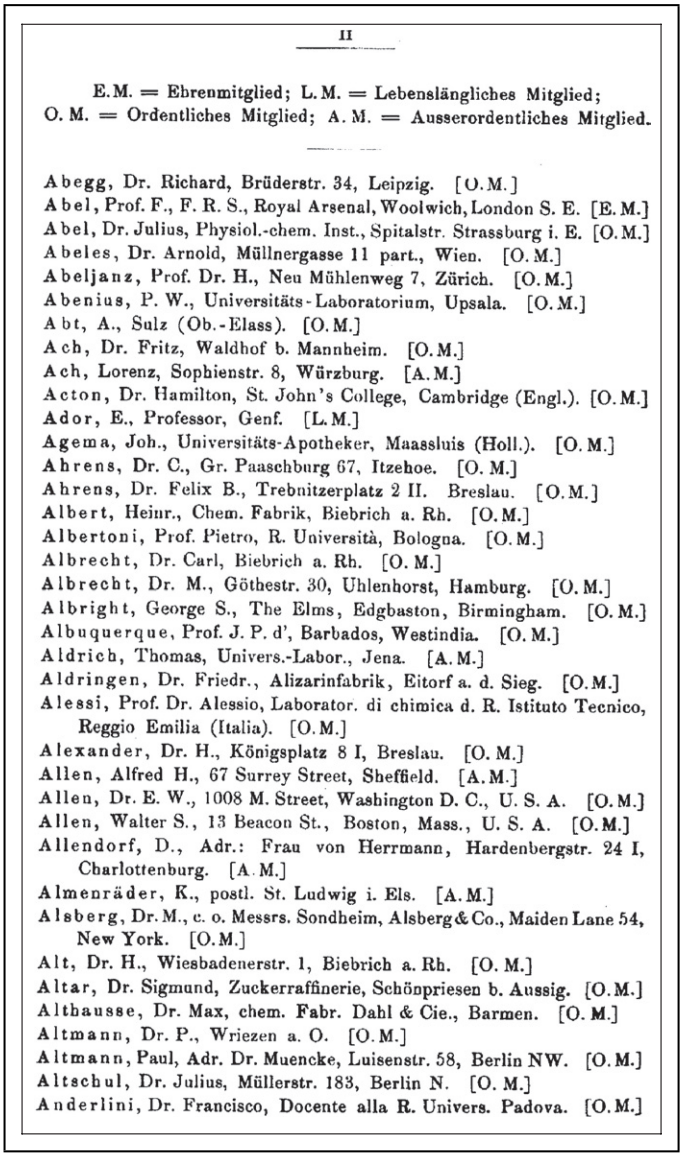
We developed our automatic coding procedure for two different projects. In the first project, we wanted to automate the coding of information on advertising firms that appeared in a U.S. trade directory published annually starting in 1918. In the second project, our goal was to code the information provided on chemists in annual membership lists of the German Chemical Society. In both cases, the quantity and the sequence of data provided on companies or individuals was not uniform. Hence it was not possible to process the data simply with the OCR software and then paste the text into a spreadsheet program. In one record, the street address of a person would be in column 2, and in another record, in column 3. We needed a program that could process this unordered data and consistently put the same kind of information in



Fittig, R., Dr. phil., Prof., Universität Tübingen.
Fitz, Alb., Dr., Waldstr. 32b, Carlsruhe.
Fleischer, A., Dr. phil., Prof., Univers. Klausenburg (Siebenbürgen).
Flemming, Hugo, Dr., Kalk bei Deutz.
Flight, Dr., British Museum, London.
Flückiger, Dr., Professor, Universität Strassburg, Els.
Forst, Stud. chem., Universitäts-Laboratorium, Bonn.
Forster, Karl, Chemiker, Juliuspromenade, Würzburg.
Franchimont, H., Dr., Wageningen, Prov. Gelderland, Niederlande.
Franck, A., Dr. phil., Fabrikbesitzer, Stassfurt.
Freire-Marreco, Algernon, Professor, Newcastle on Tyne.
Freese, C., Dr. phil., Gewerbeschule, Bielefeld.
Fresenius, H., Dr., Chemisches Laboratorium, Wiesbaden.
Frey, J. H., Dr. phil., Leipzig, Wintergartenstr. 15.
Friedburg, L. H., Dr. phil., Göttingen, Reinhäuser Chaussee 14.
Friedländer, S., Dr. phil., Proskau.
Friedrich, R., Stud., Täubchenweg 5 2 Tr., Leipzig.
Fritzsche, Th. von, Dr., Fabrikbesitzer, Frankfurt a. M.
Fuchs, C. W. C., Dr. phil., Professor, Meran (Tyrol).
Fuchs, E., Dr. phil., landw. Lehranstalt Cappeln, Schleswig.
Fuchs, Friedrich, Stud., Barbaragasse 17, Strassburg, Els.
Fudakowsky, H., Dr., Eriwanskastr. 4, Warschau.
Fumouze, V., Dr. med., Paris, Faubourg St. Denis 78.
Gabba, L., R. Istituto tecnico sup., Mailand (Italien).
Gaehtgens, Carl, Dr., Dorpat, Haus Mylius.
Gans, L., Dr. phil., Frankfurt a. M.
Gauhe, Dr. ph., adr. Hoesterey & Gauhe in Eitorf an der Sieg.
Gegerfelt, H. von, Gothenburg in Schweden.
Geilinger, Joh., Chemiker, Winterthur, Schweiz.
Geissler, H., Dr., Bonn.
Genz, B., Dr. phil., adr. K. Oehler in Offenbach a. M.
Gerber, N., Stud. chem., Helenenstr. 23, Strassburg Els.
Gerichten, von, Dr., Erlangen.
Gerland, B. W., Dr., Macclesfield (England).
Geromont, Fr., Hofbrauhaus, Würzburg.
Gerstl, R., 45. Harrington Street, London N. W.
Gessert, J., Dr. phil., Fabrikbesitzer, Elberfeld.
Geuther, Anton, Professor, Universität Jena.
Gilbert, Dr., Hamburg, Baumwallstr. 14.
Gill, C. H., F. C. S., King Henrys Road, Regents Park, London N. W.
Gintl, Dr., Professor am deutschen Polytechnicum, Prag.
Girard, Ch., 2 Rue Monge, Paris.
Gladstone, J. H., Dr. phil., F. R. S., London, 17 Pembridge Square.
Glaser, C., Dr. phil., Anilinfabrik, Ludwigshafen a. R.

Figure 1. Example page of regular members in 1874.

the same information category (column). There were two other reasons why the data was not uniform. First, the directories contained two different parts, regular members and honorary members. Second, the format in which the



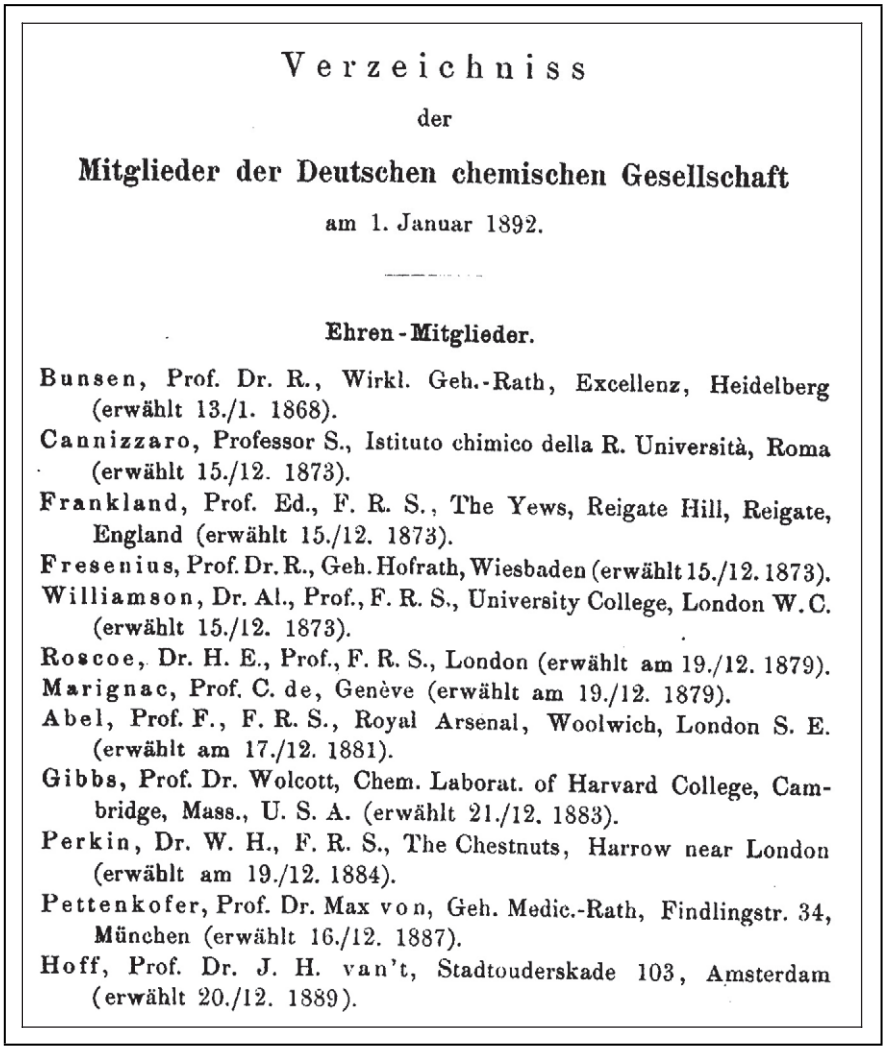
II

E.M. = Ehrenmitglied; L.M. = Lebenslängliches Mitglied;
O.M. = Ordentliches Mitglied; A.M. = Ausserordentliches Mitglied.

Abegg, Dr. Richard, Brüderstr. 34, Leipzig. [O.M.]
Abel, Prof. F., F. R. S., Royal Arsenal, Woolwich, London S. E. [E.M.]
Abel, Dr. Julius, Physiol.-chem. Inst., Spitalstr. Strassburg i. E. [O.M.]
Abeles, Dr. Arnold, Müllnergasse 11 part., Wien. [O.M.]
Abeljanz, Prof. Dr. H., Neu Mühlenweg 7, Zürich. [O.M.]
Abenius, P. W., Universitäts-Laboratorium, Upsala. [O.M.]
Abt, A., Sulz (Ob.-Elass). [O.M.]
Ach, Dr. Fritz, Waldhof b. Mannheim. [O.M.]
Ach, Lorenz, Sophienstr. 8, Würzburg. [A.M.]
Acton, Dr. Hamilton, St. John's College, Cambridge (Engl.). [O.M.]
Ador, E., Professor, Genf. [L.M.]
Agema, Joh., Universitäts-Apotheker, Maassluis (Holl.). [O.M.]
Ahrens, Dr. C., Gr. Paaschburg 67, Itzehoe. [O.M.]
Ahrens, Dr. Felix B., Trebnitzerplatz 2 II. Breslau. [O.M.]
Albert, Heinr., Chem. Fabrik, Biebrich a. Rh. [O.M.]
Albertoni, Prof. Pietro, R. Università, Bologna. [O.M.]
Albrecht, Dr. Carl, Biebrich a. Rh. [O.M.]
Albrecht, Dr. M., Göthestr. 30, Uhlenhorst, Hamburg. [O.M.]
Albright, George S., The Elms, Edgbaston, Birmingham. [O.M.]
Albuquerque, Prof. J. P. d', Barbados, Westindia. [O.M.]
Aldrich, Thomas, Univers.-Labor., Jena. [A.M.]
Aldringen, Dr. Friedr., Alizarinfabrik, Eitorf a. d. Sieg. [O.M.]
Alessi, Prof. Dr. Alessio, Laborator. di chimica d. R. Istituto Tecnico, Reggio Emilia (Italia). [O.M.]
Alexander, Dr. H., Königsplatz 8 I, Breslau. [O.M.]
Allen, Alfred H., 67 Surrey Street, Sheffield. [A.M.]
Allen, Dr. E. W., 1008 M. Street, Washington D. C., U. S. A. [O.M.]
Allen, Walter S., 13 Beacon St., Boston, Mass., U. S. A. [O.M.]
Allendorf, D., Adr.: Frau von Herrmann, Hardenbergstr. 24 I, Charlottenburg. [A.M.]
Almenräder, K., postl. St. Ludwig i. Els. [A.M.]
Alsberg, Dr. M., c. o. Messrs. Sondheim, Alsberg & Co., Maiden Lane 54, New York. [O.M.]
Alt, Dr. H., Wiesbadenerstr. 1, Biebrich a. Rh. [O.M.]
Altar, Dr. Sigmund, Zuckerraffinerie, Schönpriesen b. Aussig. [O.M.]
Althausse, Dr. Max, chem. Fabr. Dahl & Cie., Barmen. [O.M.]
Altmann, Dr. P., Wriezen a. O. [O.M.]
Altmann, Paul, Adr. Dr. Muencke, Luisenstr. 58, Berlin NW. [O.M.]
Altschul, Dr. Julius, Müllerstr. 183, Berlin N. [O.M.]
Anderlini, Dr. Francisco, Docente alla R. Univers. Padova. [O.M.]

Figure 2. Example page of regular members in 1892.

data on regular members were presented changed between 1871 and 1914. These sorts of variations are very typical when a researcher wishes to gather information from the same or similar sources over a longer period.



Verzeichniss

der

Mitglieder der Deutschen chemischen Gesellschaft

am 1. Januar 1892.

Ehren-Mitglieder.

Bunsen, Prof. Dr. R., Wirkl. Geh.-Rath, Excellenz, Heidelberg (erwählt 13./1. 1868).

Cannizzaro, Professor S., Istituto chimico della R. Università, Roma (erwählt 15./12. 1873).

Frankland, Prof. Ed., F. R. S., The Yews, Reigate Hill, Reigate, England (erwählt 15./12. 1873).

Fresenius, Prof. Dr. R., Geh. Hofrath, Wiesbaden (erwählt 15./12. 1873).

Williamson, Dr. Al., Prof., F. R. S., University College, London W. C. (erwählt 15./12. 1873).

Roscoe, Dr. H. E., Prof., F. R. S., London (erwählt am 19./12. 1879).

Marignac, Prof. C. de, Genève (erwählt am 19./12. 1879).

Abel, Prof. F., F. R. S., Royal Arsenal, Woolwich, London S. E. (erwählt am 17./12. 1881).

Gibbs, Prof. Dr. Wolcott, Chem. Laborat. of Harvard College, Cambridge, Mass., U. S. A. (erwählt 21./12. 1883).

Perkin, Dr. W. H., F. R. S., The Chestnuts, Harrow near London (erwählt am 19./12. 1884).

Pettenkofer, Prof. Dr. Max von, Geh. Medic.-Rath, Findlingstr. 34, München (erwählt 16./12. 1887).

Hoff, Prof. Dr. J. H. van't, Stadtouderskade 103, Amsterdam (erwählt 20./12. 1889).

Figure 3. Example page of second type of data: honorary members, 1892.

What has changed from the 1870s format is how the directory presents the type of membership a person holds with bracketed initials. [O.M.] stands for "regular member." [A.M.] stand for "new member." The automatic coding procedure needed to be able to deal with this change in format.

The second type of data in the directories presented honorary members and the date when they were elected. The automatic coding program needed to deal with this change in format.

b. Converting the jpeg image to editable text in OmniPage

OmniPage is designed to handle this in large batches, thus reducing significantly human labour. Here are some pointers for dealing efficiently with large batches (20 pages and more). We recommend going through this procedure once with just two or three images to get acquainted with the process.

Before doing anything else, set up the working environment using the menus as follows. Note that these procedures are specific to the latest version of OmniPage Professional 15. They may vary slightly in later, and indeed earlier, versions of the software.

1. Tools→Options→Text Editor→No Wrap ON. This setting conserves the line breaks of the original document.
2. Select the language of the original document: Tools→Options→ OCR. If more than one language is used throughout the document(s), select all those languages.
3. After starting the program, go to *View* and select Page Image (where your jpegs will appear), Text Editor (where the processed editable text will appear) and Thumbnail Images (where your jpegs will appear as thumbnails).
4. Go to the left lower side of the Text Editor screen to find three buttons. Click Plain Text View to get a .txt-file-like window where the processed text will appear with minimal layout.
5. Group images that are similar because they either have the same font or come from the same book, year or series. This prevents different character forms from mixing, which is important for making adequate use of IntelliTrain. IntelliTrain is a training function in OmniPage to help recognise badly shaped characters (see below IV b: *Semi-automatic Proofreading*).
6. Step 1: Load all the jpeg images in one go. Click the 'step one' button that should have 'Load Files' selected underneath it. Go to the group of images and add them all (button: add all images). Then save 'the project' as OPD (OmniPage Document): File→Save OPD. This is a file that will store all jpegs, the result of the later conversion, any changes during proofreading, etcetera. This will be the working file from now on.
7. Now that all the images are loaded in one file, all images can be found by scrolling along the side bar of the Thumbnail window. Click any one of them and it will appear in the Page Image window. The text editor is still blank and will stay blank for now.
8. Make sure that only the information that needs to be processed is processed. Omnipage normally does this automatically, but you might have text in your left or right margins such as handwritten notes that should not be processed. Indeed the capture of this sort of material may confuse the

software and lead to the layout of the scanned text changing. If there is text to be excluded *select* the parts you want to process as *process zone*. Starting with the first image, click the orange coloured *a,* and move the cursor to the image. Then select the text that needs to be processed. Repeat this with all images. OmniPage saves this selection instantly.

9. Step 2: Now select all the thumbnails. Select the first, scroll down, hold shift and click the last. Look at the Step 2 button and make sure it has 'custom (User defined)' selected underneath it when you decided to select the process zone by hand. If not, choose 'automatic'. Then click the Step 2 button. OmniPage now starts to process all the selected images. After it finishes the first image, OmniPage will open a proofreading window similar to popular Microsoft Office programs. During proofreading OmniPage will continue to process the rest of the images.

V. MANUAL CORRECTION OF THE OPTICAL CHARACTER RECOGNITION

The result of a processed image into editable text is unlikely to be flawless. Below you see what the regular member page from the chemist's directory (1892, p. 2), which we have seen earlier, now looks like when it comes out of the OCR process.

A b egg, Dr. Richard, Brüderstr. 34, Leipzig. [O.M.]
Abel, Prof. F., F. R. S., Royal Arsenal, Woolwich, London S. E. [E. M.]
Abel, Dr. Julius, Physiol.-chem. Inst., Spitalstr. Strassburg i. E. [O.M.]
A b e l e s, Dr. Arnold, Miillnergasse 11 part., Wien. [0.M.]
Abeljanz, Prof. Dr. H., Neu Mühlenweg 7, Zürich. [O.M.]
A b e n i u s, P. W., Universitäts -Laboratorium, Upsala. [0.M.]
A bt, A., Sulz (Ob. - Elass). [0.M.]
Ach, Dr. Fritz, Waldhof b. Mannheim. [0.M.]
Ach, Lorenz, Sophienstr. 8, Würzburg. [A. M.]
Acton, Dr. Hamilton, St. John's College, Cambridge (Engl.). [O.M.]
A d o r, E., Professor, Genf. [L. M.]
Agema, Job., Universitäts-Apotheker, Maassluis (Holl.). [0. M.]
Ahrens, Dr. C., Gr. Paaschburg 67, Itzehoe. [0. M.]
Ahrens, Dr. Felix B., Trebnitzerplatz 2 II. Breslau. [O.M.]
Albert, Heinr., Chem. Fabrik, Biebrich a. Rh. [0.M.]
Albertoni, Prof. Pietro, R. Universitä, Bologna. [O.M.]
Albrecht, Dr. Carl, Biebrich a. Rh. [0. M.]
Albrecht, Dr. M., Göthestr. 30, Uhlenhorst, Hamburg. [O. M.]
Albright, George S., The Elms, Edgbaston, Birmingham. [0.M.]
Albuquerque, Prof. J. P. d', Barbados, Westindia. [0. M.]

Aldrich, Thomas, Univers.-Labor., Jena. [A. M.]
Aldringen, Dr. Friedr., Alizarinfabrik, Eitorf a. d. Sieg. [O.M.]
Alessi, Prof. Dr. Alessio, Laborator. di chimica d. R. Istituto Teenico, Reggio Emilia (Italia). [0.M.]
Alexander, Dr. H., Königsplatz 8 I, Breslau. [0. M.]
Allen, Alfred H., 67 Surrey Street, Sheffield. [A. M.]
Allen, Dr. E. W., 1008 M. Street, Washington D. C., U. S. A. [O. M.]
Allen, Walter S., 13 Beacon St., Boston, Mass., U. S. A. [O.M.]
Allendorf, D., Adr.: Frau von Herrmann, Hardenbergstr. 24 I, Charlottenburg. [A. M.]
Almenräder, h., postl. St. Ludwig i. Els. [A.M.]
A l sb erg, Dr. M., c. o. Messrs. Sondheim, Alsberg& Co., Maiden Lane 54, New York. [O. M.]
Alt, Dr. H., Wiesbadenerstr. 1, Biebrich a. Rh. [0. M.]
Altar, Dr. Sigmund, Zuckerraffinerie, Schönpriesen b. Aussig. [O.M.]
Althausse, Dr. Max, chem. Fabr. Dahl & Cie., Barmen. [0. M.]
Altmann, Dr. P., Wriezen a. O. [0.M.]
Altmann, Paul, Adr. Dr. Muencke, Luisenstr. 58, Berlin NW. [O.M.]
Altschul, Dr. Julius, Müllerstr. 183, Berlin N. [0. M.]
Anderlini, Dr. Francisco, Docente alla R. Univers. Padova. [O.M.]

There are obviously some mistakes in this OCR text output, as for example, in line four where there are incorrect spaces added between the first and second letter of the last name (A b e l e s , Dr.). One can correct these mistakes in the text file before running the coding script on it.

a. How to proofread text in Omni Page

Automatic proofreading: As mentioned, OmniPage opens a semi-automatic proofreading function for every page it processes. It will lead the user through any decisions that OmniPage deems 'to close to call'. This means that characters that could not be recognised as, for example, either a **u** or an **n**, are presented to the user, so a human eye can decide. These dubious characters are numerous with bad quality images (or original prints for that matter), but actual misrecognitions are almost absent with good quality images.

Regrettably, this function has a flaw. It does not lead the user through everything that would be dubious from a human perspective. That flaw renders this function very useful if you are not planning on checking your results closely afterwards, that is, if you are content with a superficial conversion that might contain some falsely recognised characters. In our case, we needed flawless results because of how we intended to use the data in later analyses. We had to minimise the margin for mistakes. In these circumstances, we would advise the

user to skip the semi-automatic proofreading (which uses up quite some time), and to continue with the advanced functions that assist in proofreading by hand.

Proofreading by hand: OmniPage has a good function for proofreading by hand. To stop the semi-automatic proofreading, find the *close* button in the proofreading window.

1. Go to the Text Editor Window. There you see the editable text result of the OCR process. Now, left click twice in the middle of any line. You will see the corresponding line of the original image appear right above the editable text. Proofreading this way avoids constantly moving between two parts of the screen, or worse, between a piece of paper and a screen, constantly searching for the right line.
2. Now move the cursor over the corresponding line. Clicking the line will make the little window zoom in or out. Find the most comforting setting for your eyes: we recommend the same size as the editable text.
3. You might encounter a recurrence of the same misrecognised character. In our case, OmniPage almost always mistook the O for a 0 (zero). Unfortunately, we had a lot of them.
   i. The solution: Go to Tools → Training File → Select [none] → Click *Set as current*. This creates a training file in which shapes of original characters will be stored corresponding to a digital character of the user's choice. Make sure to make a new training file with every new batch of images – characters are different in every print.
   ii. Go to one of the 'bad' characters in the Text Editor window. Select only that letter and click right. Then click 'Train Character. . . ' A dialog box appears.
   iii. The window shows the character build up out of dots, and a suggested 'digital' character that is most likely incorrect. Fill in the right character under 'Correct' and left-click 'Train'.
   iv. Check on your training file and correct it, if necessary. Go to Tools → Training File → Edit. The window shows the trained characters. The training can be changed here. Check this file continuously when you are training many characters. It is easy to make mistakes that have big consequences for the output of your text. Consider what might happen if according to the training file, every 'e' should be changed to a 'c'.
4. When you finish proofreading a page and you select 'Next' a dialog box will appear with the remark: *Training data has been generated. It will be applied to new pages and re-recognized texts. Do you want to apply it to other existing pages in the document?* Choose "*Yes, from the current page on*" to make sure your previous corrections are not changed in any way!
5. For further details on the training file (also called IntelliTrain), use the Help-function of Omni Page.

Next you see the corrected text that will serve as the raw material for the automatic coding procedures.

Abegg, Dr. Richard, Brüderstr. 34, Leipzig. [O.M.]
Abel, Prof. F., F. R. S., Royal Arsenal, Woolwich, London S. E. [E. M.]
Abel, Dr. Julius, Physiol.-chem. Inst., Spitalstr. Strassburg i. E. [O.M.]
Abeles, Dr. Arnold, Müllnergasse 11 part., Wien. [O.M.]
Abeljanz, Prof. Dr. H., Neu Mühlenweg 7, Zürich. [O.M.]
Abenius, P. W., Universitäts -Laboratorium, Upsala. [O.M.]
Abt, A., Sulz (Ob. - Elass). [O.M.]
Ach, Dr. Fritz, Waldhof b. Mannheim. [O.M.]
Ach, Lorenz, Sophienstr. 8, Würzburg. [A. M.]
Acton, Dr. Hamilton, St. John's College, Cambridge (Engl.). [O.M.]
Ador, E., Professor, Genf. [L. M.]
Agema, Joh., Universitäts-Apotheker, Maassluis (Holl.). [O. M.]
Ahrens, Dr. C., Gr. Paaschburg 67, Itzehoe. [O. M.]
Ahrens, Dr. Felix B., Trebnitzerplatz 2 II. Breslau. [O.M.]
Albert, Heinr., Chem. Fabrik, Biebrich a. Rh. [O.M.]
Albertoni, Prof. Pietro, R. Università, Bologna. [O.M.]
Albrecht, Dr. Carl, Biebrich a. Rh. [O. M.]
Albrecht, Dr. M., Göthestr. 30, Uhlenhorst, Hamburg. [O. M.]
Albright, George S., The Elms, Edgbaston, Birmingham. [O.M.]
Albuquerque, Prof. J. P. d', Barbados, Westindia. [O. M.]
Aldrich, Thomas, Univers.-Labor., Jena. [A. M.]
Aldringen, Dr. Friedr., Alizarinfabrik, Eitorf a. d. Sieg. [O.M.]
Alessi, Prof. Dr. Alessio, Laborator. di chimica d. R. Istituto Teenico,Reggio Emilia (Italia). [O.M.]
Alexander, Dr. H., Königsplatz 8 I, Breslau. [O. M.]
Allen, Alfred H., 67 Surrey Street, Sheffield. [A. M.]
Allen, Dr. E. W., 1008 M. Street, Washington D. C., U. S. A. [O. M.]
Allen, Walter S., 13 Beacon St., Boston, Mass., U. S. A. [O.M.]
Allendorf, D., Adr.: Frau von Herrmann, Hardenbergstr. 24 I,Charlottenburg. [A. M.]
Almenräder, K., postl. St. Ludwig i. Els. [A.M.]
Alsberg, Dr. M., c. o. Messrs. Sondheim, Alsberg& Co., Maiden Lane 54, New York. [O. M.]
Alt, Dr. H., Wiesbadenerstr. 1, Biebrich a. Rh. [O. M.]
Altar, Dr. Sigmund, Zuckerraffinerie, Schönpriesen b. Aussig. [O.M.]
Althausse, Dr. Max, chem. Fabr. Dahl & Cie., Barmen. [O. M.]
Altmann, Dr. P., Wriezen a. O. [O.M.]
Altmann, Paul, Adr. Dr. Muencke, Luisenstr. 58, Berlin NW. [O.M.]

Altschul, Dr. Julius, Müllerstr. 183, Berlin N. [O. M.]
Anderlini, Dr. Francisco, Docente alla R. Univers. Padova. [O.M.]

b. Saving corrected editable text as .txt file

Before the raw material can be coded, we have to save the text. Once all the pages of editable text in the file were corrected, we saved them all at once as one text file.

1. Click the button 'Step 3'. Then find the phrase: "Files of type", where you will find a pull-down menu in which a saving format can be selected. Select 'TXT format with line breaks'.
2. Choose a file name. Choose a directory. Make sure that *File Options* (in this window) is set to *Create one file for all pages* and *Page Range* (also this window) is set to *All pages.* Then click "OK". Now one .txt file is saved with all the processed text.

VI. AUTOMATIC CODING OF RAW TEXT

(a) Writing a program to code the raw text

To convert the available information on each chemist into categories such as name, title, institutional affiliation, street address, city, country, type of membership, one has to process this data either by hand (which would take a long time) or by writing a program that automatically does the categorisation based on predefined rules.

Instead of writing our own computer program that would automatically code the raw text file of each printed page, we decided to hire a programmer (Alfonzo Forgione) to write the program for us. We came to the conclusion that hiring an experienced programmer of text coding programs was cheaper than learning to do this ourselves.

The main program written to process the raw text was developed with Microsoft Corporation Visual Studio 2005 development suite (http://msdn2.microsoft.com/en-us/vstudio/aa973782.aspx).[5] The programming language employed was Visual Basic 2005. Microsoft's Visual Studio provides an integrated development environment (IDE), to create stand-alone, mobile and Web-based application using the .NET technology.

The main component used by Visual Studio is the .NET Framework version 2.0. Since it is installed on all Windows-based platforms, it is very convenient to use. The new Visual Studio's enhanced support for writing code beyond components is a major advance. For Visual Basic, the tool suggests ways to correct several hundred syntax errors and warnings. For all languages, there is an

extensive library of reusable code snippets ranging from beginning programming constructs (like for loops) to sophisticated idioms that can even assist experts in several ways – working with XML, for example.

Essentially the program processes each text line and moves text elements (words or abbreviation) into different fields. Appendix I gives a description of the processing rules used to classify the data properly. (Appendix III provides the actual code of the program.)

It's very important to determine early in the design phase of the coding program all the different formats in which the printed text appears. The chemists' data appeared in three slightly different formats that required somewhat different processing rules. Much more significant were the text format differences in our second project where we coded automatically directories of U.S. advertising firms and their clients. The three formats posed very different design challenges for the coding program. To achieve very accurate coding results, we needed to write three distinct processing rules for these advertising directories.

The coding program was developed in an iterative process. The programmer developed a version of the program and then the research team told him what mistakes the program was making in parsing the data. We went back and forth a couple of times until the program reached a high level of accuracy.

One of the complications that we encountered was that the accented German characters did not show up properly in the output file of the program. This problem was made even more acute because the programmer and the individual members of the research team were in different parts of the world where computers used alternative language codes. In principle, automatic text processing can be done with any language in the world. The important step in having the output of the program show the correct language characters is to specify a language code for text characters that can handle different languages. The most common one is UTF8. We used UTF7 for our project. Our programmer added the following settings to the 'write line commands' in the script to create properly formatted characters: "System.Text.Encoding.UTF7." (See Appendix III)

The program creates a .csv output file that can be opened in a spreadsheet program such as Excel. One can now do final hand corrections to the data before analysing the data.

(b) How to open newly created .csv files in excel

When the information in the .txt files is transformed into a .csv file (comma-separated value), it can be viewed in the Microsoft Office program Excel. We used Excel 2007 to open these documents. Follow these specific guidelines to let Excel display the files as intended. Again, the specific commands may vary

between different Office versions, and the same versions running on different platforms.

1. Open Excel 2007 and go to the Office Button (or Start button) → Open file and find the .csv file you want and open it. Do not open the file by simply clicking on it: it will not open correctly.
2. You will get a window that is called 'Text import Wizard,' which will lead you through three steps. We only needed the first two.
3. Under the title 'Original data type' *delimited* should be selected. Find the phrase 'File Origin' and change the corresponding text-encoding to your needs. As outlined above the programmer set the output of the script to text-encoding UTF7. Excel wouldn't display all special characters like accents [e.g. â] and diaeresis [e.g. ü] (which are very common in German) when setting the 'File Origin' to UTF7. Excel would only display the special characters when selecting "*65001 Unicode (UTF8)*" as file origin. Click Next.
4. Select Tabs and Comma's as your delimiters, and click Finish. The columns should be displayed correctly now. Before going further, check if everything is as expected. Then save this file as an excel file (in Excel 2007: .xlsx files), not as a .csv file again. CSV files have limited functionality in Excel and you might end up with unsaved changes after manipulating them.

VII. MANUAL CORRECTION OF CODING RESULTS

Even the best automatic coding scripts for unstructured data will not always put the information into the proper category. If researchers want to achieve fully accurate results, it is necessary to check the coding results manually in the spreadsheet file.

We encountered a number of other specific problems.

a. Spacing before the first letter

Some fields had a space before the beginning of the first letter. This causes problems when one wants to sort data, perform statistical analyses with it or process it in other ways. To get rid such unwanted spaces, Excel offers the TRIM function to handle this problem. This function creates a double set of all the data that will be deleted later on. For example, there is a space before "Geh Rath".

Assuming that Demonds; Paul is in Cell A1, place the following in A4: =TRIM(A1). Copy the cell A4 to A5. The cell A5 will now say 'Geh. Rath.' without the unnecessary space before the first letter. Checking the formula in A5, you should find =TRIM(A2) there. Keep copying the cell A4 to the right until all

Table 1. Example of CSV File

LastName;FirstName	Titles	Institute	Street	Town/City/Country	M.Type
Abegg; Richard	Dr.		Brüderstr. 34	Leipzig.	[O.M.]
Abel; F.	Prof. F. R. S.	Royal Arsenal		Woolwich S. E. London	[E. M.]
Abel; Julius	Dr.	Physiol.-chem. Inst.		Spitalstr. Strassburg i. E.	[O.M.]
Abeles; Arnold	Dr.		Müllnergasse 11 part.	Wien.	[O.M.]
Abeljanz; H.	Prof. Dr.		Neu Mühlenweg 7	Zürich.	[O.M.]
Abenius; P. W.		Universitäts -Laboratorium		Upsala.	[O.M.]
Abt; A.		Sulz (Ob. - Elass).			[O.M.]
Ach; Fritz	Dr.	Waldhof b. Mannheim.			[O.M.]
Ach; Lorenz			Sophienstr. 8	Würzburg.	[A. M.]
Acton; Hamilton	Dr.	St. John's College		Cambridge (Engl.).	[O.M.]
Ador; E.	Professor			Genf.	[L. M.]
Agema; Joh.		Universitäts -Apotheker		Maassluis (Holl.).	[O.M.]
Ahrens; C.	Dr.		Gr. Paaschburg 67	Itzehoe.	[O.M.]
Ahrens; Felix B.	Dr.		Trebnitzerplatz 2 II.	Breslau.	[O.M.]
Albert; Heinr.		Chem. Fabrik		Biebrich a. Rh.	[O.M.]
Albertoni; Pietro	Prof.	R. Università		Bologna.	[O.M.]
Albrecht; Carl	Dr.			Biebrich a. Rh.	[O.M.]
Albrecht; M.	Dr.		Göthestr. 30	Uhlenhorst Hamburg.	[O.M.]
Albright; George S.		The Elms	Edgbaston	Birmingham.	[O.M.]

Albuquerque; J. P. d'	Prof.	Barbados		Westindia.	[O.M.]
Aldrich; Thomas		Univers.-Labor.		Jena.	[A. M.]
Aldringen; Friedr.	Dr.	Alizarinfabrik		Eitorf a. d. Sieg.	[O.M.]
Alessi; Alessio	Prof. Dr.	Laborator. di chimica d. R. Istituto Teenico		Reggio Emilia (Italia).	[O.M.]
Alexander; H.	Dr.		Königsplatz 8 I	Breslau.	[O.M.]
Allen; Alfred H.			67 Surrey Street	Sheffield.	[A. M.]
Allen; E. W.	Dr.		1008 M. Street Washington D. C. U. S.	A.	[O.M.]
Allen; Walter S.			13 Beacon St. Boston Mass. U. S.	A.	[O.M.]
Allendorf; D.		Adr.: Frau von Herrmann	Hardenbergstr. 24 I	Charlottenburg.	[A. M.]
Almenräder; K.		postl. St. Ludwig i. Els.			[A.M.]
Alsberg; M.	Dr.	c. o. Messrs. Sondheim	Alsberg& Co. Maiden Lane 54	New York.	[O.M.]
Alt; H.	Dr.		Wiesbadenerstr. 1	Biebrich a. Rh.	[O.M.]
Altar; Sigmund	Dr.	Zuckerraffinerie		Schönpriesen b. Aussig.	[O.M.]
Althausse; Max	Dr.	chem. Fabr. Dahl & Cie.		Barmen.	[O.M.]
Altmann; P.	Dr.			Wriezen a. O.	[O.M.]
Altmann; Paul		Adr. Dr. Muencke	Luisenstr. 58	Berlin NW.	[O.M.]
Altschul; Julius	Dr.		Müllerstr. 183	Berlin N.	[O.M.]
Anderlini; Francisco	Dr.	Docente alla R. Univers. Padova.			[O.M.]

Table 2. Example of Using the Trim Function in Excel

Demonds; Paul	Geh. Rath	=TRIM (A1)
Justin; Paul	Apotheker.	

your designated columns are covered. Now to the rows. Select the row of cells that contain the just copied TRIM function. Right click and copy. Now select all the rows beneath this row, until the last row of the original data is reached. Right click on the selection and paste. To the right of the original data now is the trimmed data. However, this data is only the outcome of a formula, we need to replace the original data with the result without functions in the background.

To do this, select the new set of data. Right click and copy. Than select the original set of data. Right click and special paste: a window will open. Click 'value' and click Ok. The original data is replaced with the result of the TRIM function.

b. Inconsistent recognition of characters

After we had manually corrected the data for a particular year of the chemist's directory, we combined the different yearly spreadsheet files into one spreadsheet containing all 18,000 records on individual chemists. When we sorted the 18,000 entries alphabetically by last name, we realised that some entries were not in proper alphabetical order. We determined that the problem was the font we were using: Arial. This font doesn't show a difference between an 'l' and an 'I'. We changed the font to Times New Roman and were able to identify about 80 'I"s that should have been l's (we probably made a mistake in the Intellitrain function of Omni Page Pro). We used the search and replace function in Excel, restricting the procedure on the name column in the spreadsheet.

It is important to note this problem can be identified and remedied right after the OCR step in our procedure by switching the font in the electronic text file that has uniquely formed characters.

Searching for incorrectly recognized characters should be done in Excel, where it is possible to define the area where you want to search. This way you can prevent, for example, finding all the I's in the document.

VIII. CONCLUDING REMARKS

This paper presented a complete method for using automatic techniques to code printed text pages of semi-regularised information. For over two decades, social scientists and historians have used electronic text analysis

techniques. Commercial programs such as WordStat allow researchers to count the frequency of particular words or combinations of words in an electronic text. But until now, very few researchers have used automatic techniques to code unstructured printed directory information because the tools for this were not widely available.

In developing a complete method for automatic coding of printed text, we discovered that present-day consumer digital cameras are much better than high-end scanners to obtain pictures of printed pages quickly and without the wear and tear associated with scanners. We also found that high-end OCR software such as OmniPage is much more cost-effective to achieve accurate text recognition and to process large amounts of data. There are many tools for writing programs that automatically code text into a series of variables. Because the main component used by Visual Studio development suite is the .NET Framework version 2.0, which is installed on all Windows-based platforms, using the Visual Studio development suite to write our automatic coding program proved very convenient – one can easily share the program among researchers.

The automatic techniques that we developed in this paper require non-trivial expenditures of time. This means they are most useful for researchers who want to code large amounts of printed texts. Coding a few pages can be done much more quickly by hand. However, with large amounts of printed material, the automatic techniques are wonderfully labour and cost saving.

APPENDIX I: DESCRIPTION OF CHEMISTS DATA AUTOMATIC CODING PROGRAM

GENERAL DESCRIPTION:

The purpose of the script is to read and process comma delimited text files generated by OCR software that will convert photographed images from Chemist directories listings.

TEXT FORMATS:

As was described previously in Section IV, the text files will be separated by format:

1. The format of the files in **Directory1** folder includes the pages from the directory that include the election to honorary membership dates at the end of every record. i.e. (12/10/1878)
2. The format in the files in **Directory2** folder includes the pages after the first page that include the "Membership Type" field encapsulated by the "[ ]" brackets, as well as pages that do not include this field.

All records will have a "Carriage Return", as the record end terminator will terminate all the text files.

SPECIAL AUXILIARY FILES:
The Script uses two special files that contain a list of countries and a list of titles. These files are used to identify these items in the text files and help parse and classify the data for the output file formatting. The files are called "Countries.txt" and "Titles.txt"

PROCESSING RULES:
The processing design of the script will use the following rules and conditional formatting:

- Parse the input file into 6 fields:
  - Field1: LastName; FirstName
  - Field2: Titles
  - Field3: Institute
  - Field4: Street
  - Field5: Town/City/Country
  - Field6: Membership Type
- Fix lines that have very few fields.
- If the record format does not include brackets "[", check Field1 for titles like "Professor", "Chemiker" or "Dr.". If found, move them to Field2 for further processing.
- Check Fields3 and Fields4 for titles like "F.R.S." or "Professor" and move the data to Fields2
- Automatically fix and compress the extra spaces in the FirstName field.
- Fix data for records that contain brackets "[". The information in brackets should always be in Field5. If it is in Field3 or Field4, move it to Field5.
- Fix records with too many entries. More than 6, 7 or 8 fields are concatenated.
- Check for specific Titles on several fields based on an array read from an auxiliary "Titles" file that will include a list of the most common titles used in the directories. Move then to the titles directory.
- Automatically move any information on fields that have numeric values in them to the address field.
- If Fields 3 and 4 are empty, the data from Field 2 will be move to Field4.
- If Field 4 is empty, the data from Field 3 will be moved to Field4.
- Final fix for Titles in the wrong location. Check for "Chemiker", "Prof." and "Dr. phil." and move to title field.
- Check for specific countries on several fields based on an array read from an auxiliary "Countries" file that will include the names of the most common countries used in the directories. Move them to the "Town/City/Country" field.
- If Field4 contains numbers, move the data to Field3.

- If Field4 is fewer than four characters long, the script will append the last word from Field3 to the beginning of Field4.
- The extra spaces and any "0"'s on the type of membership space will be fixed.

OUTPUT:

The output of the script will generate several "CSV" files that will be read into Excel.

If there are any other errors not defined, the record will be written to a file called "InvalidFormat.txt"

APPENDIX II: DETAILED INSTRUCTIONS TO USE CHEMIST DIRECTORY PROCESSING PROGRAM

a. Change Input directory to the appropriate path as needed.
b. Make sure that the "Countries" and "Titles" file paths are set correctly.
c. Click on the process button for the directory that will be processed.
d. The Script will process each file as follows:
    i. Read each file from the input directory until done.
    ii. Process each line from the file and format into \Done directory using the same filename with extension .CSV
    iii. Move the completed files to \Processed directory.
e. If there are any format errors on the input file, the data will be saved to the file "InvalidFormat.txt". These entries should be manually fixed.

APPENDIX III: MACHINE INSTRUCTIONS FOR AUTOMATIC CODING PROGRAM

```
Public Class Form1

    Private Sub ButtonD2_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles ButtonD2.Click
        Dim StdFormat As Integer() = {5, 10, 11, -1}
        Dim ErrorFormat As Integer() = {5, 5, -1}
        Dim FilePaths As System.Collections.ObjectModel.ReadOnlyCollection(Of String)
        Dim AllText, Filecount, InvalidFile, DoneFile, ProcessedFile, FileHeader, FileName, MoveFile As String
        Dim xCounter, fCounter, i, ii, b, c, t, Fixed, TitleLoc, MemberRec As Integer
        Dim affil, FirstName, Field1, Field2 As String

        Try
            'Make Sure Directory Structure Exists
            If Not IO.Directory.Exists(TextBox1.Text) Then
                IO.Directory.CreateDirectory(TextBox1.Text)
            End If
            If Not IO.Directory.Exists(TextBox1.Text & "\done") Then
                IO.Directory.CreateDirectory(TextBox1.Text & "\Done")
            End If
            If Not IO.Directory.Exists(TextBox1.Text & "\Processed") Then
                IO.Directory.CreateDirectory(TextBox1.Text & "\Processed")
            End If
            FilePaths = My.Computer.FileSystem.GetFiles(TextBox1.Text)

            'Read Countries Information into Array
            '*************************************************************************
            Dim FileReader As System.IO.StreamReader
            FileReader = My.Computer.FileSystem.OpenTextFileReader(TextBox3.Text, System.Text.Encoding.UTF7)
            Dim Countries(999) As String
            c = 0
            Countries(c) = FileReader.ReadLine()
            Do While Not Countries(c) Is Nothing
                c = c + 1
                Countries(c) = FileReader.ReadLine()
                If Countries(c) = "" Then
                    Exit Do
                End If
            Loop
            c = c - 1
            FileReader.Close()

            'Read Titles Information into Array
            '*************************************************************************
            'Dim FileReader As System.IO.StreamReader
            FileReader = My.Computer.FileSystem.OpenTextFileReader(TextBox4.Text, System.Text.Encoding.UTF7)

            Dim Titles(999) As String
            t = 0
            Titles(t) = FileReader.ReadLine()
            Do While Not Titles(t) Is Nothing
                t = t + 1
                Titles(t) = FileReader.ReadLine()
                If Titles(t) = "" Then
                    Exit Do
                End If
            Loop
            t = t - 1
            FileReader.Close()


            'Initialize Processing Variables
            '*************************************************************
            Label3.Visible = False
```

```
InvalidFile = TextBox1.Text & "\InvalidFormat.txt"
ProcessedFile = TextBox1.Text + "\Processed\"
xCounter = 0
fCounter = 0
FileHeader = "LastName;FirstName, Titles, Institute, Street, Town/City/Country, MembershipType" & vbCr & vbLf
Filecount = FilePaths.Count()
ProgressBar1.Maximum = Filecount
ProgressBar1.Visible = True

'LOOP for processing all files in Directory
'*************************************************************************************************
For Each file As String In FilePaths

    Dim foundFileInfo As New System.IO.FileInfo(file)
    FileName = foundFileInfo.Name
    Label3.Visible = True
    Label3.Text = "Processing File: " + FileName + "....."
    DoneFile = TextBox1.Text & "\Done\" & Replace(FileName, ".txt", ".csv")
    MoveFile = TextBox1.Text & "\Processed\" & FileName
    AllText = FilePaths.Item(xCounter)
    ProgressBar1.Value = xCounter + 1

    Using MyReader As New _
        Microsoft.VisualBasic.FileIO.TextFieldParser(FilePaths.Item(xCounter), System.Text.Encoding.UTF7)
        MyReader.TextFieldType = FileIO.FieldType.Delimited
        MyReader.SetDelimiters(",")

        Dim CurrentRow, CurrentRow2 As String()
        Dim Fields(999) As String
        'Write File header
        If Not My.Computer.FileSystem.FileExists(DoneFile) Then
            My.Computer.FileSystem.WriteAllText(DoneFile, FileHeader, True)
        End If

        While Not MyReader.EndOfData
            Try
                Dim RowType As String = MyReader.PeekChars(3)
                If String.Compare(RowType, "Err") = 0 Then
                    ' If this line describes an error, the format of the row will be different.
                    MyReader.SetFieldWidths(ErrorFormat)
                    CurrentRow = MyReader.ReadFields
                    MyReader.SetFieldWidths(StdFormat)
                Else
                    ' Otherwise parse the fields normally
                    CurrentRow = MyReader.ReadFields
                    fCounter = CurrentRow.Length
                    For i = 0 To 100
                        Fields(i) = ""
                    Next

                    MemberRec = 0
                    'Read Fields into Array using the comma as delimiter
                    For i = 0 To fCounter - 1
                        Fields(i) = CurrentRow(i)
                        Fields(i) = Replace(Fields(i), "[", ",[")
                        If Fields(i) = Nothing Then
                            Fields(i) = ""
                        End If
                        If Fields(i).Contains(",[") Then
                            CurrentRow2 = Fields(i).Split(",")
                            Fields(i) = CurrentRow2(0)
                            Fields(i + 1) = CurrentRow2(1)
                            MemberRec = 1
                        End If
                        If Fields(i) Is Nothing Then Fields(i) = ""
                    Next
```

```
'Fix Very Short line
If fCounter = 3 And Not Fields(3).Contains("[") Then
   Fields(4) = Fields(2)
   Fields(2) = ""
End If

'Fix Title on Field1
If MemberRec = 0 Then
   If Fields(1).Contains("Professor") Then
      Fields(2) = "Professor " + Fields(2)
      Fields(1) = Fields(1).Replace("Professor", "")
   ElseIf Fields(1).Contains("Chemiker") Then
      Fields(2) = "Chemiker " + Fields(2)
      Fields(1) = Fields(1).Replace("Chemiker", "")
   ElseIf Fields(1).Contains("Dr.") Then
      Fields(2) = "Dr. " + Fields(2)
      Fields(1) = Fields(1).Replace("Dr.", "")
   End If
End If

If Fields(2).Contains("F. R. S.") Then
   Fields(1) = Fields(2) + " " + Fields(1)
   Fields(2) = Fields(3)
   Fields(3) = ""
End If
If Fields(3).Contains("F. R. S.") Then
   Fields(1) = Fields(3) + " " + Fields(1)
   Fields(3) = Fields(4)
   Fields(4) = ""
End If
If Fields(2).Contains("Professor") Then
   Fields(1) = "Professor" + " " + Fields(1)
   Fields(2) = Fields(2).Replace("Professor", "")
End If



'Fix FirstName ExtraBlanks
FirstName = ""
For b = 0 To Len(Fields(0)) - 1
   If Fields(0).Substring(b, 1) <> " " Then
      FirstName = FirstName + Fields(0).Substring(b, 1)
   End If
Next
Fields(0) = FirstName

'Fix Data based on position
If Fields(4).Contains("[") Then
   Fields(5) = Fields(4)
   Fields(4) = Fields(3)
   Fields(3) = ""
   If fCounter = 4 Then fCounter = fCounter + 1
End If

If Fields(3).Contains("[") Then
   Fields(5) = Fields(3)
   Fields(3) = ""
   Fields(4) = ""
   If fCounter = 3 Then fCounter = fCounter + 2
End If

'Fix Data with too many entries
Select Case fCounter
   Case 6
      Fields(3) = Fields(3) + " " + Fields(4)
```

```
            Fields(4) = Fields(5)
            Fields(5) = Fields(6)
            Fields(6) = ""
        Case 7
            Fields(3) = Fields(3) + " " + Fields(4)
            Fields(4) = Fields(5) + " " + Fields(6)
            Fields(5) = Fields(7)
            Fields(6) = ""
            Fields(7) = ""
        Case 8
            Fields(3) = Fields(3) + " " + Fields(4) + " " + Fields(5)
            Fields(4) = Fields(6) + " " + Fields(7)
            Fields(5) = Fields(8)
            Fields(6) = ""
            Fields(7) = ""
            Fields(8) = ""
        Case Is > 8
            'Error -- Write to InvalidFormat File...
    End Select

    TitleLoc = 1
    If Fields(4).Contains("[") = False And Fields(5).Contains("[") = False Then
        TitleLoc = 2
    End If

    'Fix Title - FirstName - LastName
    '*********************************
    Fixed = 0
    For i = 0 To t
        If Fields(TitleLoc).Contains(Titles(i)) Then
            Fields(TitleLoc) = Fields(TitleLoc).Replace(Titles(i), "")
            Fields(0) = Fields(0) + ";" + Fields(1)
            Fields(1) = Titles(i)
            If Trim(Fields(2)) = "" Then
                Fields(2) = Fields(3)
                Fields(3) = ""
            End If
            Fixed = 1
            Exit For
        End If
    Next
    If Fixed = 0 Then
        Fields(0) = Fields(0) + "; " + Fields(1)
        Fields(1) = ""
    End If
    If Trim(Fields(1)) = "F. R. S. Prof." Then
        Fields(1) = "Prof. F. R. S."
    End If

    'If Column 2 Contains numbers move to Column 3
    For b = 0 To Len(Fields(2)) - 1
        If IsNumeric(Fields(2).Substring(b, 1)) Then
            If Trim(Fields(3)) <> "" Then
                Fields(4) = Fields(3) + " " + Fields(4)
                Fields(3) = ""
            End If
            Fields(3) = Fields(2) + " " + Fields(3)
            Fields(2) = ""
            'If Town is Empty move data from Intitute
            If Trim(Fields(4)) = "" Then
                Fields(4) = Fields(2)
                Fields(2) = ""
            End If
            Exit For
        End If
    Next
```

```
'Fix Last two fields if empty
If Trim(Fields(3)) = "" And Trim(Fields(4)) = "" Then
   If Fields(2).Length < 20 Then
      Fields(3) = Fields(2)
      Fields(2) = ""
   End If
End If
If Trim(Fields(4)) = "" Then
   Fields(4) = Fields(3)
   Fields(3) = ""
End If

'Final Fix for Titles in the wrong Fields
If Fields(4).Contains("Chemiker") Or Fields(4).Contains("Institut") Then
   Fields(2) = Fields(4)
   Fields(4) = ""
End If
If Fields(3).Contains("Prof.") Then
   Fields(3) = ""
End If
If Fields(2).Contains("Prof.") Then
   Fields(1) = Fields(1).Trim + " " + "Prof."
   Fields(2) = Fields(2).Replace("Prof.", "")
End If
If Fields(3).Contains("Dr. phil.") Then
   Fields(1) = Fields(1).Trim + Fields(3)
   Fields(3) = Fields(3).Replace("Dr. phil.", "")
End If


'Fix Countries using data from "Counties.txt" file
For i = 0 To c
   If Fields(4).Contains(Countries(i)) Then
      Fields(4) = Fields(4).Replace(Countries(i), "")
      Fields(3) = Fields(3) + " " + Fields(4)
      Fields(4) = Countries(i)
      'Move Institute to Town or Address to Town
      If Trim(Fields(3)) = "" Then
         Fields(4) = Fields(2) + " " + Fields(4)
         Fields(2) = ""
      Else
         Fields(4) = Fields(3) + " " + Fields(4)
         Fields(3) = ""
      End If
   End If
Next

'If Column 4 Contains numbers move to Column 3
For b = 0 To Len(Fields(4)) - 1
   If IsNumeric(Fields(4).Substring(b, 1)) Then
      Fields(3) = Fields(3) + " " + Fields(4)
      Fields(4) = ""
      'If Town is Empty move data from Intitute
      If Trim(Fields(4)) = "" Then
         Fields(4) = Fields(2)
         Fields(2) = ""
      End If
      Exit For
   End If
Next

'If Column 4 is less than 4 in length the move last word from column 3
If Len(Trim(Fields(4))) < 4 Then
   Field1 = ""
   Field2 = Trim(Fields(3))
```

```
                    If Field2.Contains(" ") Then
                        ii = Len(Field2)
                        Do While ii > 0
                            ii = ii - 1
                            Field1 = Field2.Substring(ii, 1) + Field1
                            If Field2.Substring(ii, 1) = " " Then
                                Fields(3) = Fields(3).Replace(Field1, "")
                                Fields(4) = Trim(Field1) + " " + Fields(4)
                                Exit Do
                            End If
                        Loop
                    Else
                        Fields(4) = Fields(3) + " " + Fields(4)
                        Fields(3) = ""
                    End If
                End If


                'Write Lines to Output File
                '*****************************************************************************
                'For Each newstring As String In CurrentRow
                For i = 0 To fCounter + 1
                    If Fields(i).Contains("[") Then Fields(i) = Fields(i).Replace("0", "O")
                    My.Computer.FileSystem.WriteAllText(DoneFile, Fields(i).Trim & " , ", True, System.Text.Encoding.UTF7)
                Next
                'Write Line two if necessary
                'For Each newstring2 As String In CurrentRow2
                'My.Computer.FileSystem.WriteAllText(DoneFile, newstring2 & " , ", True)

                My.Computer.FileSystem.WriteAllText(DoneFile, vbCr & vbLf, True)
                fCounter = 0
                End If

            Catch ex As Microsoft.VisualBasic.FileIO.MalformedLineException
                'MsgBox("Line " & ex.Message & " is invalid.  Skipping")
                For Each newString As String In CurrentRow
                    My.Computer.FileSystem.WriteAllText(InvalidFile, newString, True, System.Text.Encoding.UTF7)
                Next
                My.Computer.FileSystem.WriteAllText(InvalidFile, vbCr & vbLf, True)
            Catch ex As IOException
                For Each newString As String In CurrentRow
                    My.Computer.FileSystem.WriteAllText(InvalidFile, newString, True, System.Text.Encoding.UTF7)
                Next
                My.Computer.FileSystem.WriteAllText(InvalidFile, vbCr & vbLf, True)
            End Try

        End While
    End Using

    'Move Original file to the \Done directory
    My.Computer.FileSystem.MoveFile(file, MoveFile)

    ProgressBar1.Update()
    xCounter = xCounter + 1
  Next
Catch


End Try
ProgressBar1.Visible = False
Label3.Text = "All file(s) Processed....."
Label3.Visible = True
'MsgBox("Done!")
End Sub

Public Sub New()
```

```
    ' This call is required by the Windows Form Designer.
    InitializeComponent()

    ' Add any initialization after the InitializeComponent() call.

End Sub

Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
    Label3.Visible = False

End Sub

Private Sub OpenFileDialog1_FileOk(ByVal sender As System.Object, ByVal e As System.ComponentModel.CancelEventArgs)

End Sub

Private Sub Button3_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button3.Click
    Me.DialogResult = System.Windows.Forms.DialogResult.OK
    Me.Close()
End Sub

Private Sub Label1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Label1.Click

End Sub

Private Sub ButtonD1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles ButtonD1.Click
    Dim StdFormat As Integer() = {5, 10, 11, -1}
    Dim ErrorFormat As Integer() = {5, 5, -1}
    Dim FilePaths As System.Collections.ObjectModel.ReadOnlyCollection(Of String)
    Dim AllText, Filecount, InvalidFile, DoneFile, ProcessedFile, FileHeader, FileName, MoveFile As String
    Dim xCounter, fCounter, i, ii, b, d, Fixed As Integer
    Dim affil, Day, Month, Year, FirstName As String


    Try
        'Make Sure Directory Structure Exists
        If Not IO.Directory.Exists(TextBox2.Text) Then
            IO.Directory.CreateDirectory(TextBox2.Text)
        End If
        If Not IO.Directory.Exists(TextBox2.Text & "\done") Then
            IO.Directory.CreateDirectory(TextBox2.Text & "\Done")
        End If
        If Not IO.Directory.Exists(TextBox2.Text & "\Processed") Then
            IO.Directory.CreateDirectory(TextBox2.Text & "\Processed")
        End If
        FilePaths = My.Computer.FileSystem.GetFiles(TextBox2.Text)

        'Initialize Processing Variables
        Label3.Visible = False
        InvalidFile = TextBox2.Text & "\InvalidFormat.txt"
        ProcessedFile = TextBox2.Text + "\Processed\"
        xCounter = 0
        fCounter = 0
        FileHeader = "LastName-FirstName, Titles, Institute, Street, Town/City/Country, ElectedDate" & vbCr & vbLf
        Filecount = FilePaths.Count()
        ProgressBar1.Maximum = Filecount
        ProgressBar1.Visible = True

        'LOOP for processing all files in Directory
        '*************************************************************************************************
        For Each file As String In FilePaths

            Dim foundFileInfo As New System.IO.FileInfo(file)
            FileName = foundFileInfo.Name
            Label3.Visible = True
```

```
Label3.Text = "Processing File: " + FileName + "....."
DoneFile = TextBox2.Text & "\Done\" & Replace(FileName, ".txt", ".csv")
MoveFile = TextBox2.Text & "\Processed\" & FileName
AllText = FilePaths.Item(xCounter)
ProgressBar1.Value = xCounter + 1

Using MyReader As New _
   Microsoft.VisualBasic.FileIO.TextFieldParser(FilePaths.Item(xCounter), System.Text.Encoding.UTF7)
   MyReader.TextFieldType = FileIO.FieldType.Delimited
   MyReader.SetDelimiters(",")

   Dim CurrentRow, CurrentRow2 As String()
   Dim Fields(99) As String
   'Write File header
   If Not My.Computer.FileSystem.FileExists(DoneFile) Then
      My.Computer.FileSystem.WriteAllText(DoneFile, FileHeader, True)
   End If

   While Not MyReader.EndOfData
      Try
         Dim RowType As String = MyReader.PeekChars(3)
         If String.Compare(RowType, "Err") = 0 Then
            ' If this line describes an error, the format of the row will be different.
            MyReader.SetFieldWidths(ErrorFormat)
            CurrentRow = MyReader.ReadFields
            MyReader.SetFieldWidths(StdFormat)
         Else
            ' Otherwise parse the fields normally
            CurrentRow = MyReader.ReadFields
            fCounter = CurrentRow.Length
            For i = 0 To fCounter - 1
               Fields(i) = CurrentRow(i)
               Fields(i) = Replace(Fields(i), "(", ",(")
               If Fields(i) Is Nothing Then Fields(i) = ""
            Next
            'Check if needed to read second row
            If InStr(CurrentRow(fCounter - 1), "(") = 0 Then
               CurrentRow2 = MyReader.ReadFields
               fCounter = fCounter + CurrentRow2.Length
               For ii = 0 To CurrentRow2.Length - 1
                  Fields(ii + i) = CurrentRow2(ii)
                  Fields(ii + i) = Replace(Fields(ii + i), "(", ",(")
               Next
            End If
            'Break Affiliation to it's own field
            affil = Fields(fCounter - 1)
            CurrentRow2 = affil.Split(",")
            Fields(fCounter - 1) = CurrentRow2(0)
            Fields(fCounter) = CurrentRow2(1)

            'Fix FirstName ExtraBlanks
            FirstName = ""
            For b = 0 To Len(Fields(0)) - 1
               If Fields(0).Substring(b, 1) <> " " Then
                  FirstName = FirstName + Fields(0).Substring(b, 1)
               End If
            Next
            Fields(0) = FirstName

            'If column contains Prof. concatenate
            If Fields(2).Contains("Prof.") Then
               Fields(1) = Fields(2) + " " + Fields(1)
               Fields(2) = ""
            End If

            'Fix Data based on position
```

```
If Fields(3).Contains("(") Then
   Fields(5) = Fields(3)
   Fields(4) = Fields(2)
   Fields(3) = ""
   Fields(2) = ""
   If fCounter = 3 Then fCounter = fCounter + 2
End If

If Fields(4).Contains("(") Then
   Fields(5) = Fields(4)
   Fields(4) = Fields(3)
   Fields(3) = Fields(2)
   Fields(2) = ""
   If fCounter = 4 Then fCounter = fCounter + 1
End If

If Trim(Fields(3)) = "" And Trim(Fields(4)) = "" Then
   Fields(3) = Fields(2)
   Fields(2) = ""
End If

If Trim(Fields(4)) = "" Then
   Fields(4) = Fields(3)
   Fields(3) = ""
End If
If Fields(4).Contains("Prof.") And Fields(2).Trim = "" Then
   Fields(2) = Fields(4)
   Fields(4) = ""
End If

'Fix Title - FirstName - LastName
Fixed = 0
If Fields(1).Contains("Prof. Dr.") And Fixed = 0 Then
   Fields(1) = Fields(1).Replace("Prof. Dr.", "")
   Fields(0) = Fields(0) + ";" + Fields(1)
   Fields(1) = "Prof. Dr."
   Fixed = 1
End If
If Fields(1).Contains("Prof.") And Fixed = 0 Then
   Fields(1) = Fields(1).Replace("Prof.", "")
   Fields(0) = Fields(0) + ";" + Fields(1)
   Fields(1) = "Prof."
   Fixed = 1
End If
If Fields(1).Contains("Dr.") And Fixed = 0 Then
   Fields(1) = Fields(1).Replace("Dr.", "")
   Fields(0) = Fields(0) + ";" + Fields(1)
   Fields(1) = "Dr."
   Fixed = 1
End If
If Fields(1).Contains("Professor") And Fixed = 0 Then
   Fields(1) = Fields(1).Replace("Professor", "")
   Fields(0) = Fields(0) + ";" + Fields(1)
   Fields(1) = "Professor"
   Fixed = 1
End If
If Fixed = 0 Then
   Fields(0) = Fields(0) + "; " + Fields(1)
   Fields(1) = ""
End If
If Fields(2).Contains("F. R. S.") Then
   Fields(1) = Fields(1) + " " + Fields(2)
   Fields(2) = Fields(3)
   Fields(3) = ""
End If
If Fields(3).Contains("F. R. S.") Then
```

```
    Fields(1) = Fields(1) + " " + Fields(3)
    Fields(3) = Fields(4)
    Fields(4) = ""
End If

If Fields(3).Contains("Professor") Then
    Fields(1) = Fields(1) + " " + "Professor"
    Fields(3) = Fields(3).Replace("Professor", "")
End If

'Generic move up fields if Fields(2) is blank
If Fields(2).Trim = "" Then
    Fields(2) = Fields(3)
    Fields(3) = ""
End If

'If Column 2 Contains numbers move to Column 3
For b = 0 To Len(Fields(2)) - 1
    If IsNumeric(Fields(2).Substring(b, 1)) Then
        Fields(3) = Fields(2) + " " + Fields(3)
        Fields(2) = ""
        Exit For
    End If
Next

'Fix Data with too many entries
Select Case fCounter
    Case 6
        If Fields(3).Trim = "" Then
            Fields(3) = Fields(3).Trim + Fields(4)
            Fields(4) = Fields(5)
            Fields(5) = Fields(6)
            Fields(6) = ""
        Else
            Fields(5) = Fields(6)
            Fields(6) = ""
        End If
    Case 7
        Fields(3) = Fields(3) + " " + Fields(4)
        Fields(4) = Fields(5) + " " + Fields(6)
        Fields(5) = Fields(7)
        Fields(6) = ""
        Fields(7) = ""
    Case 8
        Fields(3) = Fields(3) + " " + Fields(4) + " " + Fields(5)
        Fields(4) = Fields(6) + " " + Fields(7)
        Fields(5) = Fields(8)
        Fields(6) = ""
        Fields(7) = ""
        Fields(8) = ""
    Case Is > 8
        'Error
End Select


'Write Line one
'For Each newstring As String In CurrentRow
For i = 0 To fCounter
    'Fix Date Format
    If Fields(i).Contains("(") Then
        d = Fields(i).IndexOf("/")
        Day = Str(Val(Fields(i).Substring(d - 3, 3)))
        Month = Str(Val(Fields(i).Substring(d + 1, 2)))
        d = Fields(i).IndexOf(")")
        Year = Str(Val(Fields(i).Substring(d - 4, 4)))
        Fields(i) = Trim(Month) + "/" + Trim(Day) + "/" + Trim(Year)
```

```
                    End If
                    My.Computer.FileSystem.WriteAllText(DoneFile, Fields(i).Trim & " , ", True, System.Text.Encoding.UTF7)
                Next
                'Write Line two if necessary
                'For Each newstring2 As String In CurrentRow2
                'My.Computer.FileSystem.WriteAllText(DoneFile, newstring2 & " , ", True)

                My.Computer.FileSystem.WriteAllText(DoneFile, vbCr & vbLf, True)
                fCounter = 0
              End If

            Catch ex As Microsoft.VisualBasic.FileIO.MalformedLineException
              'MsgBox("Line " & ex.Message & " is invalid.  Skipping")
              For Each newString As String In CurrentRow
                My.Computer.FileSystem.WriteAllText(InvalidFile, newString, True)
              Next
              My.Computer.FileSystem.WriteAllText(InvalidFile, vbCr & vbLf, True)
            End Try

          End While
        End Using


        My.Computer.FileSystem.MoveFile(file, MoveFile)
        REM My.Computer.FileSystem.WriteAllText("bigfile.txt", allText, True)
        ProgressBar1.Update()
        xCounter = xCounter + 1
      Next
    Catch


    End Try
    ProgressBar1.Visible = False
    Label3.Text = "All file(s) Processed....."
    Label3.Visible = True
    'MsgBox("Done!")
  End Sub
End Class

'Function IsDigitsOnly(ByVal Value As String) As Boolean
'   IsDigitsOnly = Not Value Like "*[!0-9]*"
'End Function

'Function IsNumber(ByVal Value As String) As Boolean
'    ' Leave the next statement out if you don't
'    ' want to provide for plus/minus signs
'    If Value Like "[+-]*" Then Value = Mid$(Value, 2)
'    IsNumber = Not Value Like "*[!0-9.]*" And _
'    Not Value Like "*.*.*" And _
'    Len(Value) > 0 And Value <> "." And _
'    Value <> vbNullString
'End Function
```

END NOTES

[1] For more technical information on the camera and reviews, go to http://reviews.cnet.com/Fujifilm_FinePix_F20/4505-6501_7-31974907.html?tag=prod.txt.1 [11 July 2008]

[2] When forced to work in conditions with little or no natural light (e.g. the basement of a library), one member of our research team (Sekou Bermiss) developed an alternative method in which he used a piece of plexiglass to keep the book pages flat. This method doesn't require the creation of a complex clasping-like contraption, but the action of placing and removing the plexiglass adds time to the process. We have detailed the alternative method on the internet where other practitioners can share their experience with different arrangements. For details, see http://www.economic-evolution.net/index.php/wiki/Photograhing_Printed_Materials/. [11 July 2008]

[3] [11 July 2008]

[4] For information on the software go to: http://www.nuance.com/omnipage/professional/. [11 July 2008] Since we bought the software, a new version (16) has appeared. We have not worked with the software, but we expect it to function in the same way. Here you can find out what is new in Version 16: http://www.nuancestore.com/v2.0-img/operations/scansoft/site/html/omnipage16/omnipage_pro16_whatsnew_standard.html [11 July 2008]

[5] Recently a new version of the development software has come out: Visual Studio 2008. It operates the same way as version 2005, but like all programming software undergoes functional enhancements to improve the ease of programming.