tranScriptorium:
A Guide for the Humanist[1]

The aim of the *tranScriptorium* project is to produce software applications that will 'read' historical handwritten documents and will produce transcripts for these documents. There are many millions of pages of handwritten documents, the basis for much research in the arts and humanities, and yet the vast majority of this material—including manuscripts that have been digitized—remains inaccessible, except to the individual scholar who has the time and resources to visit the archive where the material is deposited and the palaeographic skills required to survey and read it. The creation of a process that will allow such material to be machine-read will have immensely beneficial consequences for scholarship and research. The vision behind *tranScriptorium* is that, beginning with digital images of handwritten documents, the software application will generate a typewritten transcription that not only reproduces the words themselves but also their layout on the page, and is fully searchable. The following account explains, in non-technical language, the various steps that are being taken in the *tranScriptorium* project order to turn this vision into reality.

It should be noted that the machine-reading of handwriting presents far more challenges than that of machine-reading type. The latter is relatively standardised by nature, and consists of a common alphabet across a significant number of languages. In type, the letters are separated, and the words themselves are separated by a significant space. Type, moreover, when it appears in printed documents, tends to follow a standard layout pattern, with extra-body text information, such as page numbers and running headings, being placed regularly on the page and in a regular format. In other words, one book very much looks like another. The machine-reading of type through Optical Character Recognition (OCR) is a well-developed technology that now presents relatively few difficulties.

Machine-reading of handwritten documents cannot, however, operate at the character level, because in typical handwritten documents, the letters of each word are joined together.

[1] By Philip Schofield, Bentham Project, Faculty of Laws, University College London, with assistance from Tim Causer and Kris Grint, and with thanks to Richard Davis, Melissa Terras, and Joan Andreu Sánchez for comments, advice, and information, and to the partners in the *tranScriptorium* consortium for the illustrations. The partners in the consortium are the Polytechnic University of Valencia (Spain), the University of Innsbruck (Austria), the National Centre for Scientific Research 'Demokritos' (Greece), the Institute for Dutch Lexicology (Netherlands), University College London (UK), and the University of London Computer Centre (UK). The *tranScriptorium* project is funded from 1 January 2013 to 31 December 2015 by the European Union's Seventh Framework Programme for Research, Technological Development, and Demonstration, under grant agreement no. 600707.

Moreover, individual words can often not be recognized because of the inconsistency of the gaps between them, or conversely the lack of any gap. Hence, the software needs to operate not at the character level, nor at the word level, but with longer sequences of text, such as lines, sentences, paragraphs, or even complete pages. This fact introduces a further complication in that, while each Western European language uses the same, relatively small number of characters, each language has a different set of words. Furthermore, languages change over time, and even individual authors from the same historical period and speaking the same language have differing vocabularies. Instead of dealing with the comparatively extremely small number of characters in the alphabet, the software to read handwritten documents needs to deal with many thousands of words. Furthermore, the layout of a handwritten document can follow any pattern that human ingenuity has been able to devise.

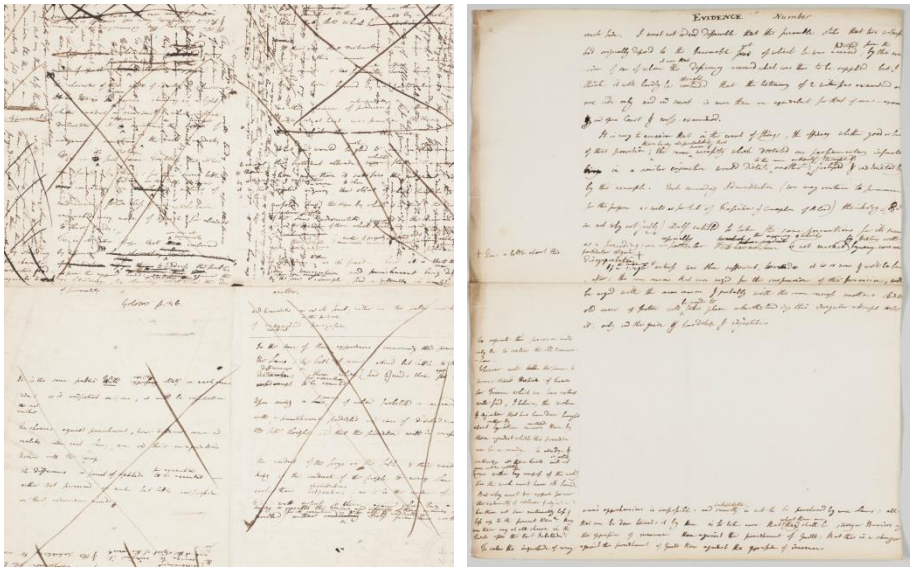

Figure 1. Examples of Bentham manuscripts (UC lxxix. 47 and UC li. 27).

In order to read and make sense of a document, there are two threshold requirements: we need to be able to recognize each word (though it is not necessary to have each individual word segmented, that is demarcated from every other), and we need to be able to recognize the order in which those words are organized. Hence, in order to produce machine-read handwritten documents, the software needs not only to recognize the words themselves, but the order in which those words should be read. The requirement to understand the order and organization of the words includes the requirement to recognize non-body text words, such as page numbers, as well as titles, headings, and marginalia. Some handwritten documents present much greater challenges than others, for a whole variety of reasons: because, for instance, of complex layouts, interlineations and crossings out, difficult handwriting, the use

of non-standard abbreviations and technical words, and the use of foreign languages embedded within a native language (for instance French, Latin, and Ancient Greek within a predominantly English manuscript). The current basic techniques for word-reading emerge not from OCR technology, but from Automatic Speech Recognition technology. The problem of Handwritten Text Recognition is analogous to recognizing speech in a badly degraded audio file.

It should be pointed out, before explaining the processing of a document in more detail, that the software needs to be 'trained' by being shown examples of similar documents in which the layout has been ascertained, including line detection, and the transcripts for those lines produced by human transcribers. These initial transcripts are referred to as the 'ground truth', and it is probably only slightly misleading to think of the ground truth being to the software the equivalent of what a diplomatic transcript is to the humanist. An important point is that the greater the quantity of ground truth that is used for training, the greater the accuracy of the machine-read text. Another feature of *tranScriptorium* is that the software learns from human checking of the machine-read transcripts that it has provided, creating a sort of virtuous circle. Suppose that the training documents consist of 50 pages. A second batch of 50 pages is machine-read, producing a word-error-rate of 30% (i.e. 3 out of every 10 words are misread). The second batch is corrected by a human transcriber. A third batch of 50 pages is machine-read, producing a word-error-rate of 25%, and so on. It should be remarked that most of these errors usually correspond to words that have not appeared in the training material, and so can not be recognised by the system.

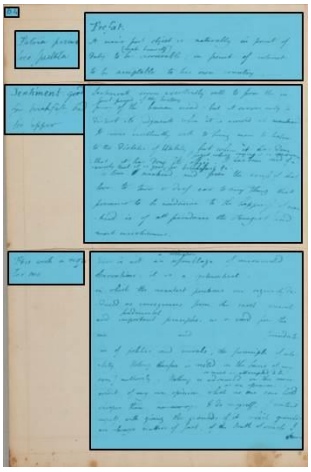

Figure 2. Marking the different elements of the layout.

The first stage in the process is to mark out the layout of the document. This process is termed layout analysis. In other words, the software locates the zones with text and then divides the document into areas or regions that represent different portions of the page: hence, a particular page may contain a margin, a page number, a heading, a title, as well as body text, which may itself appear in two or more columns. This process is carried out automatically, whereupon the user may correct any errors that have arisen.

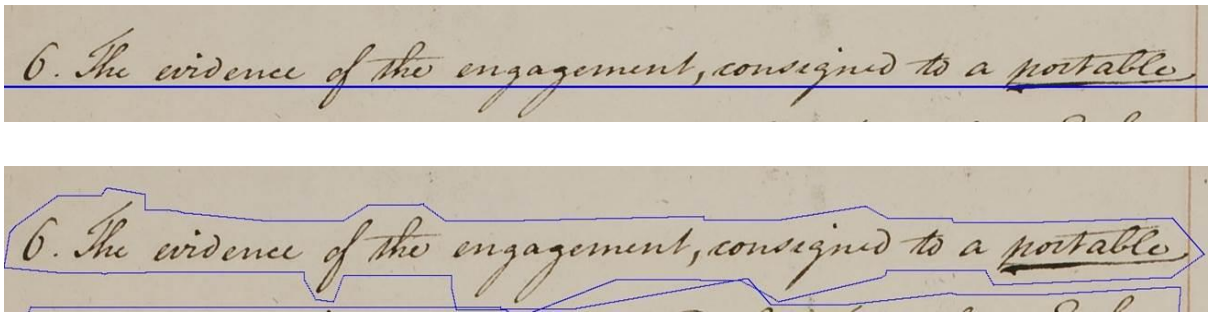

Figure 3. Line detection in *tranScriptorium*: baseline v. polygon.

The second stage is to identify the individual lines in the document. There are two main two alternatives for line identification. The first technique is the equivalent of drawing a line under the words, with each line so drawn representing a line of text. These lines are like the lines in a school exercise book, with the bottom of the characters sitting on the line, with the exception of the tails of j, g, etc. that fall beneath the line. The second technique is to draw a polygon around the line. A problem in handwritten documents is that the lower strokes of one line (j, g, etc) often interfere with the upper strokes (b, d, etc.) of the line immediately beneath. The software has to recognize that it is dealing with two separate lines, and hence separate the overlapping characters at the most appropriate point. The reason for this is that, when we come to that part of the process that reads the word itself, it will 'see' the image insofar as it is included in the polygon of the relevant line. An important difference between the techniques is that the former is more economic (in time and effort), while the latter is not appropriate in relation to certain categories of manuscript collections. A further problem with handwritten documents is interlineation. An author will typically make a correction by crossing out a word or words, and writing an alternative word or words above the line. An author will also typically make an addition by writing an additional word or words above the line. The software has to recognize that such interlineations are not separate lines (to read them as separate lines would usually render the text nonsensical or else possibly alter its meaning). Interlineations represents a significant challenge, and one that may require

some user correction once the machine-read transcript has been generated, though progress is being made by the *tranScriptorium* team towards finding a solution.

Having got to this point in the process, the next task is to prepare the image for recognition (i.e. machine reading).

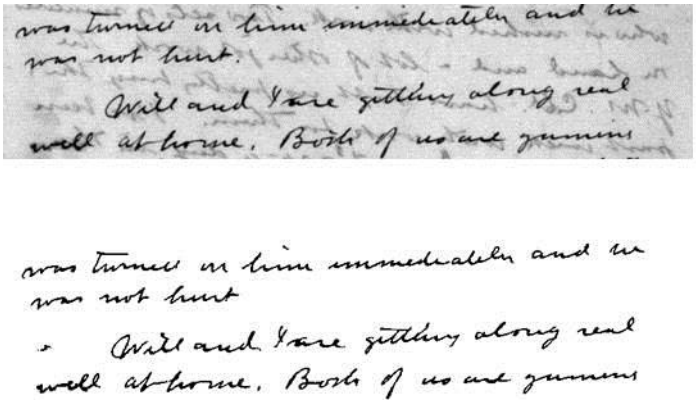

Figure 4. Image processing – removal of bleed-through and binarization.

The third stage is to 'clean' the document image. The paper or other material on which the text is written forms the background of the image which the software analyses. The paper may have become discoloured in an uneven way, contain bleed through from the reverse side of the page, or simply have become stained with dirty marks of various sorts. For the software, this unevenness in the background creates 'noise' that may interfere with its recognition of the text lines. Hence, techniques are used to 'smooth out' the background of the images so that it is closer to a uniform colour and hence remove the 'noise'. At this stage, techniques are employed both to emphasize and to make homogenous the grey level—that is the density of the colour—of the pen strokes.

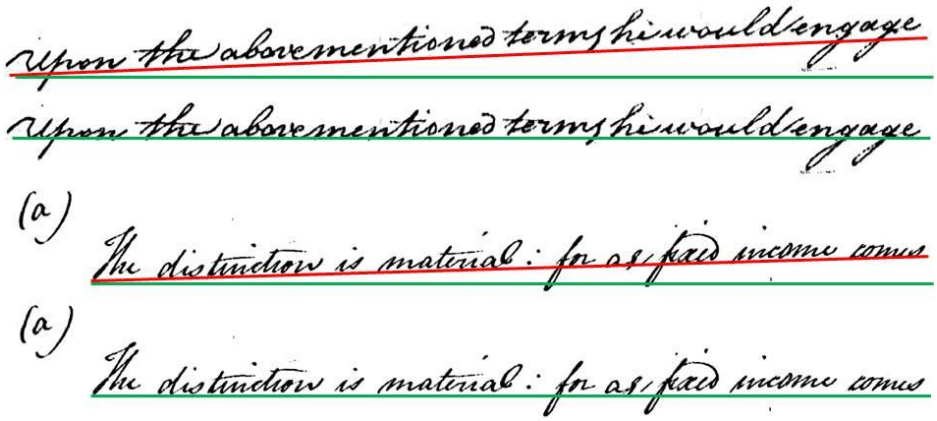

Figure 5. Skew correction.

The fourth stage is to remove ‘skew’ from the lines and ‘slant’ from the words. Skew is where a line does not cross the page at a right angle to the sides of the page, but starts lower and ends higher as it progresses, or *vice versa*, or begins by going higher and finishes by going lower, and so on. Slant is where the upstrokes of the characters in a word are not perfectly upright in relation to the line, but are at an angle of more or less than a right angle. Experiment has shown that recognition of words is more accurate when skew and slant is normalised.

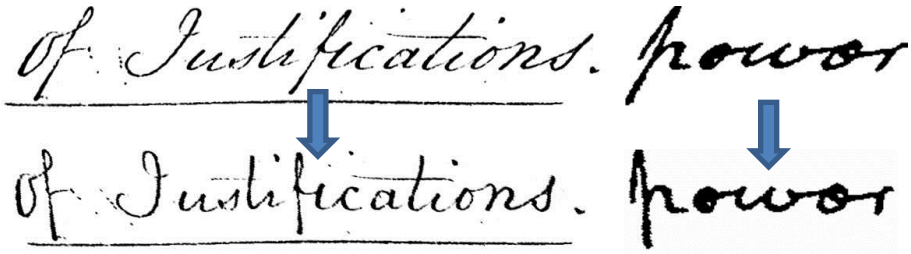

Figure 6. Slant correction.

We have now arrived at the fifth stage, the point where machine reading of the text lines themselves can begin. The software, in order to recognize a handwritten text line, must, however, have a Language Model. The Language Model helps the recognition system to read text correctly. It uses probability to predict a word. The most common Language Model is known as an n-gram, where the probability of a word depends on the probability of its following the previous word or words. In a 2-gram or bigram (a type of n-gram), if the word ‘sinister’, for instance, is often followed by ‘interest’ in the corpus under consideration, then the software is programmed to recognize that if it finds the word ‘sinister’, then it should consider it as a strong probability that the following word is ‘interest’. A more complex model is a 3-gram or trigram, where the probability of a word depends on the probability of its following the previous two words. Hence, if the software finds the two words ‘greatest happiness’, then it will consider it as a strong probability that the next word is ‘of’, as in the phrase ‘the greatest happiness of the greatest number’. The user is eventually presented with variant readings in a descending order of probability. In addition to the language model, a lexicon from which to identify unknown words is very important. A different Language Model and a different lexicon will be required for each separate language, and for the same language at a different point of time in its historical development. For instance, there is no point asking the software to recognize German text if the only lexicon to which it has access in the training phase is English. Fewer errors will be obtained if the software reading an

eighteenth-century English document is not trying to match the words with a lexicon drawn from twenty-first century English.

The sixth stage, and complementary to the fifth stage, is the actual attempt to recognize the text itself: Handwritten Text Recognition (HTR). As we read a line of handwritten text, our eye moves along the line and we recognize each word in turn. The HTR software operates in an analogous way by scanning the line from left to right, but taking into account the context of each word through the use of the Language Model. In the HTR process it is not necessary to divide the lines into single words, since the HTR system looks for a match between the line image and a hypothesised text line. The software may take into account thousands of text lines for matching the line image, and the best one is selected according to stochastic criteria. Alternatively, a list of best hypotheses can be obtained, and these are presented in a descending order of probability. These hypotheses can be merged in a graph that is termed a Word-Graph.

A further resource that is included in *tranScriptorium* is the facility to search images of handwritten documents for particular words. The production of an accurate transcript of a large corpus of material is extremely expensive in that the transcript needs to be checked by an expert, and there are some bodies of material that simply do not justify such a large investment of resources. A scholar, however, may still wish to search that material for references, for instance, to a particular person, place, or other object. Rather than engage in the time-consuming task of reading the material from beginning to end, the scholar may resort to Key Word Spotting (KWS), a technique whereby he or she can search for particular words. Within *tranScriptorium*, two approaches are being considered for KWS. In the first approach, the user searches for words by writing the desired string in plain text (‘query by string’). The software interrogates the Word-Graph in order to determine the most likely words that match the search. The user has the ability to choose between a greater number of candidate words with less certainty or a smaller number of words with greater certainty. In the second approach, the user searches a word by providing an image of the word (‘query by example’). In this second approach, first, the pen strokes that make up each word are ‘smoothed out’. If you magnify a word, you will see that the ink does not create a totally dark space on the paper, nor does it create perfectly straight edges. Hence, the software processes the image in order to make straighter the edges of the pen strokes that make up the letters, and ‘blackens’ any small areas that are contained within the boundary of those pen strokes that have not been ‘coloured’ by the ink. Second, the software reduces the word (remember

that the image of the word has been corrected to remove skew and slant, and background 'noise' has been removed, in order to facilitate this stage of the process) to a series of significant points, for instance at the middle and extremities of the pen strokes that make up the letters that make up the word. The software then attempts to find a best match with the patterns of such significant points that it contains in its database, and which are linked to words in the lexicon.

The whole document is processed in this way. The process is, perhaps, analogous to a production line in which a series of operations are performed on a raw material in order to transform it into something more useful.

We now have a machine-read transcript, but this is not the end of the matter. Further training of, and improvement of the results produced by, the software are achieved through the intervention and collaboration of users.

Let us assume that an archive (or 'content provider') possesses a major collection of handwritten documents of significant historical importance, and that it also possesses digital images of the documents. The archive, as a service to historians, decides that it wishes to acquire a searchable transcript of its collection. It gains access to the *tranScriptorium* software, and begins by creating an accurate transcript, properly segmented at page, line, and word level, of 50 pages. The *tranScriptorium* software is now trained. The archive then feeds a further 50 pages of unsegmented and untranscribed images, but of the same layout and in the same language, into the *tranScriptorium* software, which then automatically produces an appropriate transcript. The archive has, in fact, a PDF that mimics the layout of the document, as well as the words it contains. There will be errors in the machine-read transcript. The archive at this point has two options (not mutually exclusive) in order to correct the transcripts, and thereby improve the software by 'teaching it'. The first is to use professional transcribers. The second is to use crowdsourced volunteers, that is members of the general public who participate in the transcription because of their interest in the subject-matter and willingness to contribute to the goal of making historical material accessible.

In the first case, where the archive uses its own staff to check the machine-read transcript, the interface with the software will permit the user to alter the segmentation, whether at page, line, or word level – that is alter the layout of the document – as well as to amend any mistranscribed words. In the second case, where the archive asks unpaid

volunteers to correct the transcript, the interface may be simplified in that it may only allow amendments to the words themselves, and not to the layout.

The amendments made either by the professional transcribers or the volunteers are used to 'train' the software, and so the next 50 pages of 'raw' manuscripts that are presented to the *tranScriptorium* software will be transcribed with fewer errors. The number of manuscripts that can be presented to the *tranScriptorium* software is, of course, unlimited. The benefits to arts and humanities scholars of having internet access to masses of searchable handwritten documents are equally unlimited.