University of Massachusetts Amherst

From the SelectedWorks of R. Manmatha

2003

Text Alignment with Handwritten Documents

E. Micah Kornfield
R. Manmatha, *University of Massachusetts - Amherst*
James Allan

Available at: https://works.bepress.com/r_manmatha/25/

Text Alignment with Handwritten Documents

E. Micah Kornfield, R. Manmatha and James Allan*
Center for Intelligent Information Retrieval
Department of Computer Science
University of Massachusetts Amherst
{emkorn, manmatha,allan}@cs.umass.edu

Abstract

Today's digital libraries increasingly include not only printed text but also scanned handwritten pages and other multimedia material. There are, however, few tools available for manipulating handwritten pages. Here, we propose an algorithm based on dynamic time warping (DTW) for a word by word alignment of handwritten documents with their (ASCII) transcripts. We see at least three uses for such alignment algorithms. First, alignment algorithms allow us to produce displays (for example on the web) that allow a person to easily find their place in the manuscript when reading a transcript. Second, such alignment algorithms will allow us to produce large quantities of ground truth data for evaluating handwriting recognition algorithms. Third, such algorithms allow us to produce indices in a straightforward manner for handwriting material. We provide experimental results of our algorithm on a set of 70 pages of historical handwritten material - specifically the writings of George Washington. Our method achieves 74.5% accuracy on line by line alignment and 60.5% accuracy when aligning whole pages at time.

1. Introduction

A number of today's digital libraries contain handwritten material. Some of these libraries include both handwritten material and ASCII transcripts. An example of such a digital library is the Newton Project (http://www.newtonproject.ic.ac.uk/) that proposes to create ASCII transcripts for Newton's handwritten manuscripts. A word by word alignment of the transcript and the handwritten manuscript would allow a person to easily find their place in the manuscript when reading the transcript. For example, one could display both the manuscript and the transcript and whenever the mouse is held over a word in the transcript, the corresponding word in the manuscript would be outlined using a box.

Creating such alignments is challenging since the transcript is an ASCII document while the manuscript page is an image. Handwriting recognition is not accurate enough to recognize such large vocabulary historical document collections. We therefore propose an alternative approach to aligning such material. The handwriting page image is automatically segmented. Features (for example box and text position, aspect ratio etc) are then computed for both the transcript and the page image. An algorithm based on dynamic time warping (DTW) is then used to align the words on the page image and the transcript. We compute alignments for whole pages and also for situations in which one can assume that the beginning and end positions of lines are known. We show results on a set of 70 pages from George Washington's handwriting. Performing alignment on a line by line basis we achieved an accuracy of 74.5% and when aligning pages without line breaks we achieved 60.5% when evaluating alignments based on our edit distance metric (see Section 7).

Alignment is difficult because every step in the above mentioned approach produces errors. Segmentation of handwriting is known to cause errors - both over and under segmentation occur. Since our corpus consists of scanned images of old historical documents, there are even more errors. The transcripts may themselves be erroneous although this is less common. In addition, the alignment algorithm itself produces errors.

Such alignments have other applications. One such application is the ability to create ground truth data for evaluating handwriting recognition and retrieval algorithms [8]. Effectively producing ground truth data for large collections of handwritten manuscripts is a manually intensive and laborious process that requires a person to first create a tran-

* This work was supported in part by the Center for Intelligent Information Retrieval and in part by the National Science Foundation under grant number IIS-9909073. Any opinions, findings and conclusions or recommendations expressed in this material are the author(s) and do not necessarily reflect those of the sponsor.

script based on the entire manuscript and then label individual words. The process of labelling can be avoided if alignment algorithms are available. Alignment also allows us to create an index for the manuscript. Specifically, this allows one to search the manuscript by searching its ASCII transcript. The alignment can then be used to highlight the search terms in the manuscript (as is done with conventional text search engines).

The remainder of this paper is organized as follows. Section 2 discusses related work and how our approach differs. We then continue by formally defining the problem and notation used for the rest of the paper in section 3. Several baseline algorithms are discussed in section 4. Our DTW algorithm is described in section 5. In section 6 we discuss the format of our data. Section 7 goes over different evaluation metrics for the alignment tasks. We conclude with experimental results in section 8 and future research paths in section 9.

2. Previous work

2.1. Historical Documents

Very little research has been done on aligning transcripts of historical documents. The only prior work that makes an attempt to perform such an alignment of words between transcripts and word images is by Tomai et al. [11]. The method they propose is to limit the lexicon of a handwriting recognizer by using the transcript. A ranked list of possible words from the lexicon is returned for each recognized word image. Several different likely segmentations of a line are made. The segmentation that has the highest probability given the transcript and previous alignments is then used. If a mapping cannot be performed with high enough confidence for a word then it is left out.

Tomai et al give a figure of 82.95% accuracy in mapping words to a page, However, this figure makes certain assumptions. First they exclude 32 of the 249 words due to their "extreme noisiness". Including all words, their accuracy is roughly 72%. Second, they mention that of the 180 words they map, 17 are exactly mapped and 163 'roughly mapped'. In the absence of other information, we are unable to decide what the term 'roughly mapped' means and will assume that all 180 words were accurately mapped from transcript to manuscript. Finally, the results are reported for a single page of handwriting.

There has also been work in the areas of Automatic Speech Recognition (ASR) [9] and machine translation [4] on alignment. We note that these problems are somewhat different. For example, in machine translation, the alignment is between ASCII text in two different languages and additional constraints in terms of dictionary and grammar are available that are not available for word images.

Our approach to the transcript problem assumes that images must be aligned with ASCII text from the transcript. Thus, it is a completely different approach from that of Tomai et al [11]. In addition, we report results on a large number of documents (70 versus 1 in the previous case).

2.2. Optical Character Recognition (OCR)

The OCR community [2] has done research into aligning transcripts with printed documents for the purposes of creating ground truth. For example [2] tries to find a geometric transformation between the document description and the image of the document which minimizes a cost function. This technique assumes that along with the transcript there is a page description that denotes where the words in the transcript appear on the page. The most information that might be available in existing transcripts of historical documents is where line breaks occur. This limited information does not appear to be sufficient to make use of the algorithm proposed.

Another technique that might be applicable to our task was proposed by Ho and Nagy [1]. Their proposed algorithm uses a predefined lexicon to help recognize characters. Ho and Nagy's algorithm is to segment a printed page into individual characters and cluster each of the segments. After clustering, character labels are assigned to the clusters by finding mappings that maximize a v/p ratio. The v/p ratio measures how well a set of mappings matches the lexicon. This technique is not directly applicable to our task because in general segmenting individual characters from handwritten manuscripts is very difficult. However, the idea of using the word-level language model from the transcripts to make assignments is appealing and should be investigated further.

3. Problem Definition and Notation

Given a digitized image of a page D_i (the set of all pages is denoted by D) we generate a segmentation $\beta(D_i)$ that produces a vector of word images $\{b_0, b_1, \ldots, b_n\}$. For clarity a segmentation actually produces bounding boxes for a digitized image, the pixels within a bounding box comprise a word image. We also have a transcript T_i that is a vector of ASCII words $\{w_0, w_1, \ldots, w_r\}$ for each page. For each $b_m \in \beta(D_i)$ we wish to select a set W_m of words from the transcript ($W_m \subseteq T_i \cup \{\}$) such that W_m contains the ASCII equivalent to what is represented by the word image b_m. An example of a handwritten page and a perfect alignment for the page is shown in Figure 1.

When performing alignment we can view a segmented document $\beta(D_i)$ as containing multiple lines. Transcripts, however, might not contain such line breaks. In general, when we refer to $\beta(D_i)$, we view the entire document as

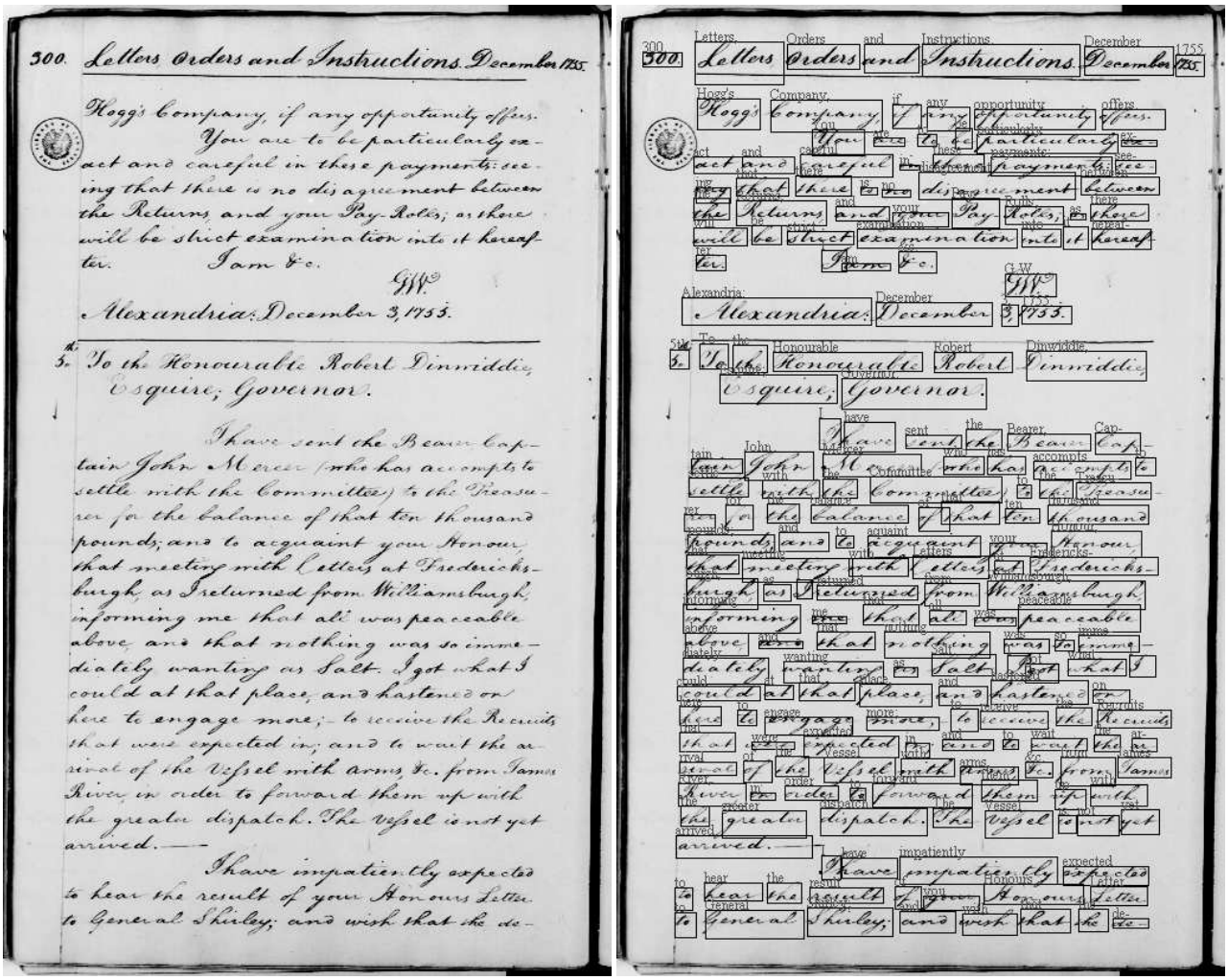

Figure 1. Handwritten Page and Perfect Alignment

one long line. This is accomplished by placing each successive line at the end of the previous line. For example, if we have two lines $\{b_1, \ldots, b_n\}$ and $\{b_{n+1}, \ldots, b_m\}$. We adjust every bounding box in $\{b_{n+1}, \ldots, b_m\}$ to have the same baseline (y-coordinate) as the first line ($\{b_1, \ldots, b_n\}$) and adjust the starting x-coordinate of each box in the second line by adding the x-coordinate of the end of image b_n.

Sometimes transcripts will have line break information. In this case it is useful to remove the abstraction of a single long line and refer to specific lines. We denote this as $\lambda_l(\beta(D_i))$ where l indicates that we are interested in only the bounding boxes on the l^{th} line. Similarly $\lambda_l(T_i)$ denotes we are interested only in the ASCII words on the corresponding line l of the transcript. $|\lambda(x)|$ gives the count of lines in either transcript or segmentation data.

4. Baseline Algorithms

Baseline algorithms are fairly simple, naive algorithms that give us a reference point for determining how well our algorithm performs.

4.1. Linear Alignment

Linear alignment is the simplest possible type of alignment one can imagine. If we have a set of bounding boxes $\{b_1, \ldots, b_M\}$ and a set of transcript words $\{w_1, \ldots, w_N\}$ we can do a forward alignment by assigning w_i to b_i where $1 \leq i \leq \min(M, N)$. Alternatively we can start from the end of the document and move to the beginning by assigning w_{N-i-1} to b_{M-i-1} where $1 \leq i \leq \min(M, N)$. Note that when $N \neq M$, these techniques leave some words or bounding boxes unassigned.

4.2. Alignment Using Character Position

Alignment using character position is done by aligning words and bounding boxes by calculating a normalized character position for either boxes or words and then finding the closest word to the position in the other set. In contrast to linear alignment we are now trying to align words and boxes based on their length, rather than counting from the beginning or end. We define:

$X_{start}(b)$ The starting x-coordinate of the bounding box for word image b.

$X_{end}(b)$ The ending x-coordinate of the bounding box for word image b.

$Y_{top}(b), Y_{bottom}(b)$ The corresponding quantities for the y-coordinate.

$\mu(\{b_i, \ldots, b_{i+n}\})$ is the width of a set of images and is equal to $X_{end}(b_{i+n}) - X_{start}(b_i)$. Note that if $n = 0$ then this definition is simply the width of the word image b_i. In addition, by defining width in this way, for any value of $n \geq 1$ spaces between word images are included in the width. The rationale behind this method of calculating width is that it will still provide accurate positions estimates even if a word fails to be segmented as long as a mistake was not made at the beginning or end of a line.

The alignment can work in one of two ways, either from text to images or from images to text.

When aligning from text to images we calculate for each $w_1, \ldots, w_N$ the character position ($CP(w_j)$) as:

$$CP(w_j) = \frac{\sum_{i=1}^{j}(|w_i| + 1)}{\mu(\{w_1, \ldots, w_N\})} \quad (1)$$

We then multiply $CP(w_j)$ by $\mu(\{b_1, \ldots, b_M\})$. The resulting product, p_j, is a position somewhere in the interval $0 \leq p_j \leq \mu(b_1, \ldots, b_M)$. Box b_l is assigned to word w_j if position p_j lies somewhere within the box i.e. such that $X_{start}(b_l) \leq p_j \leq X_{end}(b_l)$. If p_j falls between two bounding boxes, then it is assigned to the closest of the two boxes b_{l+1}, b_l by computing ($\arg_b \min(X_{start}(b_{l+1}) - p_j, p_j - X_{end}(b_l))$).

Alignment can also be performed by calculating the estimated character position in the images and multiplying it by the character width to get the position. The ratio is calculated as:

$$CP(b_j) = \frac{X_{end}(b_j)}{\mu(\{b_1, \ldots, b_M\})} \quad (2)$$

We then multiply by the width of the transcript ($\mu(\{w_1, \ldots, w_N\})$). The resulting product is a character position. If the character happens to be the space we arbitrarily pick the word preceding the space as the one to assign to box b_j.

4.3. Upper Bound Alignment

In upper bound alignment we try to assign the correct word with the correct box. Note that if a bounding box encircles two words, this alignment causes both ASCII words to be assigned to this box. This measure allows us to see what the maximum value of an evaluation metric we can expect, without performing the more complicated task of splitting ASCII words. It is generated automatically by assigning the complete word annotations to each box (see Section 6.2).

5. Dynamic Time Warping

Dynamic Time Warping (DTW) is an algorithm for aligning two time series by minimizing the "distance" between them. A time series is a list of samples taken from a signal, ordered by the time that the respective samples were obtained. For our alignment task, we view each ASCII word in a transcript and each box in a segmentation as the samples that make up the two times series we are concerned with.

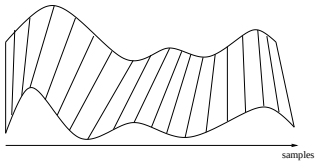

Figure 2. Two similar time series.

Rather than mapping samples that have the same time index to each other, DTW allows for the fact that one time signal may be warped with respect to the other. An example of an alignment for two series can be seen in Figure 2. The name Time Warping is derived from the fact that this alignment "warps" the time axes of the two series so that the corresponding samples more closely relate to our intuition of what a good alignment should be. The DTW cost between two time series $b_1 \ldots b_M$ and $w_1 \ldots w_N$ is DTW(M,N) which may be calculated using the following recurrence relation:

$$DTW(i,j) = \min \left\{ \begin{array}{l} DTW(i, j-1) \\ DTW(i-1, j) \\ DTW(i-1, j-1) \end{array} \right\} + d(b_i, w_j) \quad (3)$$

where $d(b_i, w_j)$ is our sample-wise cost measure:

$$d(b_i, w_j) = \sum_{k=1}^{|\delta|} \delta_k(b_i, w_j) \quad (4)$$

$\delta_k(b, w)$ is the k^{th} word-box cost feature used (see Section 5.1). The solution to the recurrence is calculated using dynamic programming.

The recurrence ensures that no samples are left out, as a result every word from the transcript will be assigned to at least one box and every box will have at least one word assigned to it.

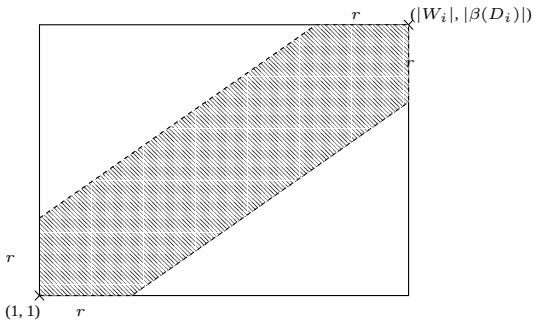

Figure 3. Sakoe-Chiba path constraint with width r.

An important aspect of DTW is that we constrain how much each of the time axes can be warped. This has a two-fold effect. First, it reduces computation time for the algorithm. Second, it disallows large warps. By a large warp, we mean either assigning a single word to a large number of boxes, or a large number of words to a single box. This constraint is known as a global path constraint. There are a variety of ways that the global path constraint can be implemented. We chose to use the *S*akoe-Chiba [10] band constraint that simply limits how far off the diagonal an alignment can move (see Figure 3).

Pseudocode for the algorithm is given in Figure 4. Assignments are made by back tracking through the dynamic programming table starting at point ($|T_i|$,$|\beta(D_i)|$) and finding the preceding minimum point as defined by the recurrence.

5.1. Word-Box Features

Word-box features are used in calculating the cost of assigning a word to a given bounding box in DTW. Any combination of the features listed below can be used when running dynamic time warping. We used two distinct types of features. The first type relies on computing scalar features over the word images and ASCII text. Once we have feature values corresponding to each word in the transcript and image in the segmentation, we can then calculate the cost of any word-box pair using a suitable cost measure. In this case δ_k from Equation (4) is defined as $cost(f_k(w_i), f_k(b_j))$ where f_k is a feature below and $cost$ is a cost function.

There are many possible cost functions that can be used. We considered two possible measures for our alignments. The first is the square of the Euclidean distance, $cost(x, y) = (x - y)^2$. The second is absolute difference $cost(x, y) = |x - y|$.

Aspect Ratio For an image b we calculate the aspect ratio as $\frac{Y_{bottom}(b) - Y_{top}(b)}{\mu(b)}$. To calculate aspect ratio for text, we render the text in a script font and perform the same computation on a bounding box of the rendered word. All values are normalized to be between 0 and 1 with a mean of 0.5.

Width For a word image width is calculated as $\frac{\mu(b_j)}{\sum_{b \in \beta(D)} \mu(b)}$. ASCII words are rendered and the same computation is performed on the rendered text images.

Character Position We use Equations (1) and (2) to compute character positions.

Ascender Count Some characters have ascenders that extend above other characters. For instance capital letters, "l" and "d" have ascenders. An estimation technique [3] is used to try to determine the number of ascenders for a word image. Characters with ascenders can be directly counted for words from the transcript. All values are normalized to be between 0 and 1 with a mean of 0.5.

Descender Count Some letters have descenders that extend below the baseline. For instance, "g" and "y" have descenders. The same techniques for finding ascenders is used for finding descenders in images and words. All values are normalized to be between 0 and 1 with a mean of 0.5.

The second type of cost feature does not explicitly extract two scalar values that can be compared with a simple cost function. Instead the cost for assigning a given word to an image is more complex. Two features we looked at were:

Image Matching DTW This feature is obtained by rendering all ASCII characters off-line, and performing DTW between the concatenation of features for each character in a word and an image word from the document as described in [7]. Image mapping is an attempt to see if we can get a high enough correlation between mock handwriting (the rendered text) and George Washington's handwriting. If there is such

Input: $T_i = (w_1, \ldots, w_{|T_i|})$ and $\beta(D_i) = (b_1, \ldots, b_{|\beta(D_i)|})$, cost function $d(\cdot, \cdot)$
Output: DTW matrix Δ Algorithm:

1. $\Delta(1,1) = d(w_1, b_1)$;
2. for m = 1:$|T_i|$
3. $\quad \Delta(m,1) = \Delta(m-1,1) + d(w_m, b_1)$;
4. for $n = 1 : |\beta(D_i)|$
5. $\quad \Delta(1,n) = \Delta(1,n-1) + d(w_1, b_n)$;
6. for $m = 1 : |T_i|$
7. $\quad$ for $n = 1 : |\beta(D_i)|$
8. $$\Delta(m,n) = \min \left\{ \begin{array}{l} \Delta(m,n-1) \\ \Delta(m-1,n) \\ \Delta(m-1,n-1) \end{array} \right\} + d(w_m, b_n);$$

Figure 4. Pseudocode for DTW (adapted from [12])

a correlation then using this feature will help match words between the transcript and the images.

Stop Word Matching Stop word matching (STM) gives a fixed penalty if we believe a word image contains a stop word ("a", "the", etc.) and the corresponding ASCII word is not aligned with the image. Our belief of the contents of a word image is based on trained clustering of all word images offline.

More specifically we have a set of labeled clusters C such that $c \in C$ has a label representing the words in the cluster (i.e. "the", "a", etc). c is composed of a set of word images. $\delta_S(w_i, b_j)$ is defined as follows: if $\exists c \in C$ such that $b_j \in c$ then if $w_i \neq label(c)$ add a fixed penalty. Otherwise add zero.

Clustering for stop word matching was done as follows:

1. Randomly arrange all word images we wish to cluster.
2. Using training data, build a cluster for each of words we are interested in recognizing. In our case we choose the top 30 occurring words in the training data.
3. Take the next image, b_i, to calculate its distance from each cluster: Find $\min_{c \in C}(dist(b_i, \psi(c)))$ and $\arg_c \min_{c \in C}(dist(b_i, \psi(c)))$. Where $dist$ is the DTW distance [7] between the centroid of the cluster ($\psi(c)$) and the image.
4. If the distance in step 3 is less then a threshold γ (obtained through experimentation) then assign the image b_i to cluster c and update the centroid. Otherwise discard the example.
5. If there are more images to cluster go to step 3.

6. Data

Our data consisted of 100 digitized pages from George Washington's archive. The pages were divided randomly into a set of 30 pages that are used for parameter estimation, cluster training and validation. The remaining 70 pages are used for evaluation. For each page we have two different types of segmentations with annotations and a line aligned transcript.

	# Boxes	# Lines
Automatic Segmentations (β^{auto})	17,696	2380
Manual Segmentations (β^{hand})	17,192	2404

Table 1. Number of bounding boxes and lines in our evaluation data

6.1. Segmentation (β)

The segmentation produces a list of bounding boxes that when applied to the image should isolate all the pixels that are part of a single word. For each bounding box we have the coordinate that defines a rectangle and an indicator of the line in the digital image the bounding box occurs on. The two different types of segmentation are described below. Figure 5 shows each type of segmentation. Table 1 contains the number of boxes and lines in the segmentations for the 70 pages of evaluation data.

Automatic Segmentations (β^{auto}) Automatic segmentations are those generated automatically by a program that is an improved version of [6]. These segmenta-

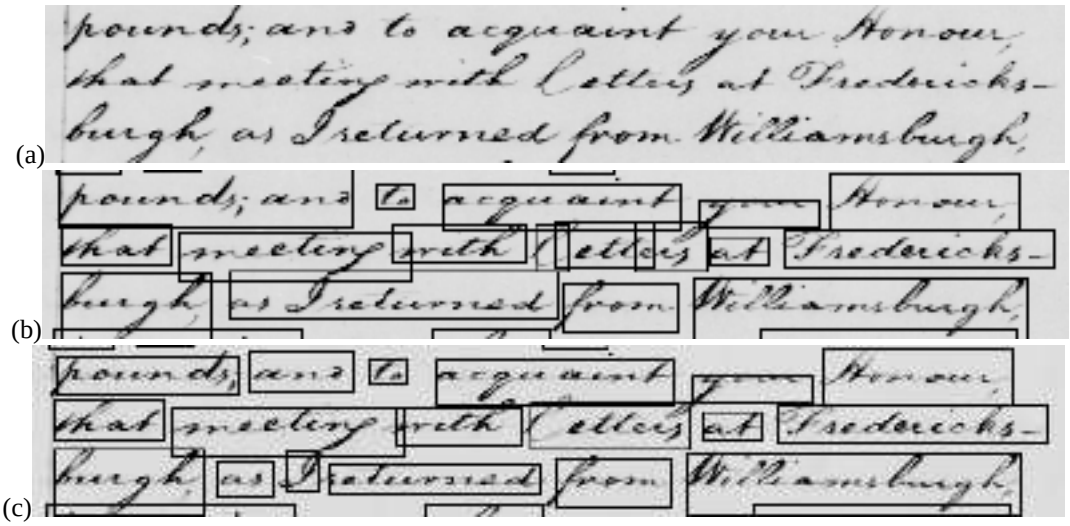

Figure 5. (a) 3 Lines from a sample image (b) Automatic Segmentation (c) Manual Segmentation

tions are not perfect and can contain four different types of mistakes:

1. Bounding boxes will sometimes be placed around artifacts on the page that are not real words.
2. Some words might have no bounding boxes placed around them.
3. Bounding boxes are sometimes placed around more then one word.
4. A word can sometimes be split into more then one bounding box, or only be partially included in a bounding box.

Manual Segmentations (β^{hand}) Manual segmentations are corrections of automatically segmented pages. For each page an annotator went through and made any corrections so that there is a one-to-one and onto mapping of words from the transcript to bounding boxes. Words in this case are strings made from all alphanumeric characters.

6.2. Annotations $(A[\beta(D)])$

Annotations consist of vectors of ASCII strings for each bounding box in a segmentation. These labels provide us with the truth value of the contents of each bounding box, that can be used to evaluate how well or poorly and alignment algorithm works.

For manually segmented documents an annotation is simply the ASCII text equivalent of the word in the bounding box. Automatically segmented pages have a slightly richer representation to account for possible errors in the segmentation. For each bounding box that contains one or more words, the string labels are the exact text that is located within the bounding box (if a bounding box only covers part of a word, only the part covered is included). If a bounding box only contains part of a word, then in addition to exactly what is contained inside the box, we also record the complete word that was split by the box.

6.3. Transcripts (T)

A transcript is an ASCII text file consisting of text that corresponds to a specific page. Each file is aligned in parallel, on the line level, with the two different segmentations above. Figure 6 contains example transcripts for the three lines contained in Figure 5. A transcript for a document is the same thing as an annotation for a hand segmented document image with some additional punctuation.

```
pounds; and to aquaint your Honour,
that meeting with Letters at Fredericks-
burgh, as I returned from Williamsburgh,
```

Figure 6. Example Transcript

7. Alignment Evaluation

Evaluation of the alignment is not straightforward. Evaluation metrics vary depending upon the goal of the alignment. For instance, if we are interested in generating training data for other handwriting recognition or retrieval al-

gorithms, then we wish to have exact annotations for each bounding box. Alternatively, to build an index directly from alignments and use it for retrieval, a less strict measure might give a better idea of the results we can expect when conducting retrieval. Described below are five different evaluation algorithms that we use to evaluate alignments we generate.

Our evaluation measures are all defined by giving a score, on a bounding box level and then averaging this score for all of the bounding boxes in all of the documents.

7.1. Exact Matching (σ_{exact})

For $b_j \in \beta(D_i)$ of a document we have an annotation, $S_j \in A[\beta(D_i)]$, and an alignment vector W_j (the words assigned to b_j). Exact matching gives a point (1) for b_j if $|S_j| = |W_j|$ and $\forall i : \{1 \leq i \leq |S_j|\} s_i = w_i$. That is, the two strings are equal if they are the same length and all corresponding characters are equal. So

$$\sigma_{exact}(b_j) = \begin{cases} 1 & |S_j| = |W_j|, \forall i : \{1 \leq i \leq |S_j|\} s_i = w_i \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

Exact matching is very strict. For a perfect score, it requires algorithms to not only give a reasonable alignment, but to trim words from the transcript to fit poorly segmented words and split words if a segmentation splits the word. This type of measure is probably best used when evaluating alignments for use as training data for other retrieval methods.

7.2. Edit Distance Matching (σ_{ED})

Exact match is a rigorous evaluation measure, and might not be suited to all applications of the alignment algorithm. We therefore propose a more relaxed definition of what it means to get an alignment for a bounding box correct. If we concatenate the strings in both our annotation for a bounding box and the aligned text for the box we can then use the value returned by Equation 6 for the two resulting strings to judge if a bounding box has the correct text in it.

$$\sigma_{ED}(s_1, s_2) = \begin{cases} 1 & \max(|s_1|, |s_2|) - ED(s_1, s_2) > \frac{\max(|s_1|, |s_2|)}{2} \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

where $ED(s_1, s_2)$ is the edit (Levenshtein) distance [5] between the two strings. The edit distance between two strings is given by the recurrence:

$ED(\text{“”}, \text{“”}) = 0$

$ED(s, \text{“”}) = ED(\text{“”}, s) = |s|$

$$ED(s_1{+}c_1, s_2{+}c_2) = \min \begin{pmatrix} ED(s_1, s_2) + \epsilon(c_1, c_2), \\ ED(s_1 + c_1, s_2) + 1, \\ ED(s_1, s_2 + c_2) + 1 \end{pmatrix}$$

where c_1, c_2 are characters and $\epsilon(c_1, c_2)$ returns zero if the characters are equal and 1 otherwise. Edit distance matching is more relaxed then exact matching. By counting bounding boxes as correct if the words mostly match, it better reflects the case of using alignments for direct retrieval. It also give a little bit of leeway in case of annotation and transcript discrepancies caused by typographical errors in the creation of either set. So if we define $\kappa(\{st_1, \ldots, st_n\})$ to be the concatenation of a set of strings then

$$\sigma_{ED}(b_j) = \sigma_{ED}(\kappa(S_j), \kappa(W_j)) \tag{7}$$

7.3. Precision-Recall ($\sigma_{Precision}$, σ_{Recall})

Recall and precision, are measure commonly used in the information retrieval domain. We can extend them to alignment evaluation, by calculating each of the metrics on a bounding box level. Precision is then defined as:

$$precision(S_j, W_j) = \frac{|S_j \cap W_j|}{|W_j|} \tag{8}$$

(the proportion of the words in the assignment that match the annotation) and recall as:

$$recall(S_j, W_j) = \frac{|S_j \cap W_j|}{|S_j|} \tag{9}$$

the proportion of the words in the annotation that are matched. So $\sigma_{precision}(b_j) = precision(S_j, W_j)$ and $\sigma_{recall}(b_j) = recall(S_j, W_j)$.

7.4. Tomai et al. Evaluation

The evaluation metric that is used by Tomai et al. [11], is slightly different in flavor then any of our proposed evaluation metrics. Instead of looking at bounding boxes and determining which words are placed correctly within a given box, they look at each transcript word and determine if the box it is mapped to contains the correct image. More formally for each word-box pair (w_i, b_j^{auto}), the mapping is considered correct if $w_i = S_k \in A[\beta^{hand}(D_i)]$ and

$$\begin{aligned} Y_{top}(b_j^{auto}) &\leq Y_{top}(b_k^{hand}) \\ Y_{bottom}(b_j^{auto}) &\geq Y_{bottom}(b_k^{hand}) \\ X_{start}(b_j^{auto}) &\leq X_{start}(b_k^{hand}) \\ X_{end}(b_j^{auto}) &\geq X_{end}(b_k^{hand}) \end{aligned}$$

. There score is calculated as the number of correct mappings divided by the size of the transcript.

7.5. Averaging

For any of the measures above, we can average the evaluation in three different ways: over documents Equation (10),

over lines Equation (11), or over boxes Equation (12).

$$\frac{\sum_{x=1}^{|D|}\left(\frac{\sum_{i=1}^{|\beta(D_x)|}\sigma(b_i)}{|\beta(D_x)|}\right)}{|D|} \tag{10}$$

That is, each page D_i is weighted equally. Recall that D is the set of handwritten document, D_i is a page, $\lambda_l(\beta(D_i))$ is a line and b_i is a word image. We can also weight each line equally:

$$\frac{\sum_{x=1}^{|D|}\sum_{i=1}^{|\lambda(D_x)|}\left(\sum_{b_y\in\lambda_i(\beta_i(D_x))}\sigma(b_y)\right)}{\sum_{x=1}^{|D|}|\lambda(D_x)|} \tag{11}$$

or each word image equally:

$$\frac{\sum_{x=1}^{|D|}\sum_{i=1}^{|\beta(D_x)|}\sigma(b_i)}{\sum_{x=1}^{|D|}|\beta(D_x)|} \tag{12}$$

8. Experiments

Our experiments consisted of performing alignments for both β^{auto} and β^{hand}. For discussion we include tables of our edit distance evaluation metric (see Section 7.2), tables for each of the evaluation metrics described in Section 7, are included at the end of paper in an appendix.

For each segmentation we attempted two types of alignment. The first type uses the line break information in the transcripts ("Line by Line" in the tables). The second aligns documents without any line breaks ("Full Page" in the table). The former task is easier because in each application of an alignment algorithm there are many fewer possible assignments.

The first 5 rows (before the dividing line) in each table are entries for our baseline algorithms (see Section 4 for details). The remaining rows are the results for our dynamic time warping algorithm, with slightly different feature sets (see Section 8.1 for a more detailed description). The second dividing line denotes the results between DTW runs that used perfect clusters for stop word matching (not achievable in the real world) and those that either do not use the stop word matching feature or those that used clusters produced with our image clustering algorithm. Every entry number in the table is a percentage of correct alignments.

8.1. β^{auto}

Experimentally, we found that a path width of 35 and absolute distance function work the best. We determined that all of the best DTW runs contained the same basic features: width, ascenders, descenders and aspect ratio. Including character position as a feature helped for the line by line task, but not for the full page. Stop word matching helped, only by a small amount. (If we cheated and used perfect clustering, stop words help a great deal.) These determinations held true on our evaluation data. We achieved a top accuracy of 74.5% on the line by line task and 60.5% on the full page task when using the edit distance metric for evaluation.

Each algorithm entry in the table for DTW only denotes the features that differentiate them from one another (each DTW used the basic features listed above). Clusters indicates that stop word matching was used as a feature. For each alignment algorithm we also calculated the standard deviation (placed to the right of each experimental column).

Box Level Averaging:

	Line by Line	**Std. Dev.**	**Full Page**	**Std. Dev.**
Baseline Algorithms				
Linear Aligner FRONT	48.3	50.0	4.3	20.2
Linear Aligner BACK	46.4	49.9	4.4	20.5
Char. Pos. ($T_i \rightarrow \beta(D_i)$)	58.3	49.3	6.5	24.7
Char. Pos. ($\beta(D_i) \rightarrow T_i$)	64.4	47.9	5.9	23.6
Upper Bound	87.3	33.3	87.3	33.3
Dynamic Time Warping (using Basic Features +)				
Char. Pos. Perfect Clusters	76.9	42.2	70.2	45.8
Perfect Clusters	74.6	43.5	70.2	45.7
Real Clusters	70.9	45.4	60.5	48.9
Char. Pos., Real Clusters	74.5	43.6	58.6	49.3
Char. Pos.	74.2	43.7	51.2	50.0
No Additional Features	70.5	45.6	57.6	49.4

Table 2. β^{auto} Edit Distance

Document Level Averaging:

	Line by Line	**Std. Dev.**	**Full Page**	**Std. Dev.**
Baseline Algorithms				
Linear Aligner FRONT	53.6	16.9	3.1	3.4
Linear Aligner BACK	48.7	9.6	4.2	5.4
Char. Pos. ($T_i \rightarrow \beta(D_i)$)	51.7	11.4	4.2	4.2
Char. Pos. ($\beta(D_i) \rightarrow T_i$)	68.2	10.4	6.8	5.4
Upper Bound	86.1	7.3	86.1	7.3
Dynamic Time Warping (using Basic Features +)				
Char. Pos., Perfect Clusters	77.6	7.6	71.5	10.9
Perfect Clusters	75.5	7.5	71.8	7.7
Real Clusters	72.2	7.8	61.9	9.8
Char. Pos., Real Clusters	75.3	7.5	58.7	17.5
Char. Pos.	75.4	7.6	51.3	21.2
No Additional Features	71.8	7.5	58.9	12.2

Table 3. Tomai et al's Evaluation for β^{auto}

Number of pages (out of 70) Greater then 72% with Tomai et al's Evaluation

	Line by Line	Full Page
Baseline Algorithms		
Linear Aligner FRONT	11	0
Linear Aligner BACK	0	0
Char. Pos. ($T_i \rightarrow \beta(D_i)$)	0	0
Char. Pos. ($\beta(D_i) \rightarrow T_i$)	26	0
Upper Bound	67	67
Dynamic Time Warping (using Basic Features +)		
Char. Pos., Perfect Clusters	56	37
Perfect Clusters	51	32
Real Clusters	33	10
Char. Pos., Real Clusters	50	13
Char. Pos.	51	11
No Additional Features	34	9

Table 4. Number of documents for which we performed better than Tomai et al.

In addition to our measures we also include results with Tomai et al.'s evaluation (see Section 7.4) in Table 3. We also analyze how many documents in our evaluation set DTW performs better on then Tomai et al. in Table 4. Table 4 indicates the number of pages that each of the algorithms discussed exceeded 72% (the adjusted reported accuracy of Tomai et al.'s algorithm on their data). Our algorithm performs favorably when compared to theirs when we use line break information from the transcript. We achieved an accuracy of 75.4% over a total of 70 pages compared to their result 72% on a single page. If we do not use line break information then our accuracy drops to 61.9% but there are still 10 pages on which we perform better than them. Additionally, we see from Tables 2 through 4 that dynamic time warping performs quite well in comparison with our baseline algorithms, increasing our percentage by $\approx 10\%$ for our best runs on the line by line experiment. When aligning pages a page at a time, the contrast is even more stark, we obtain a order of magnitude improvement over the base line measures. However, our results also show, that there is still room for improvement between the upper achievable score and our current results.

8.2. β^{hand}

Performing alignments against hand segmented pages, is a way to test if our algorithm is correct. Given a perfect segmentation, any admissible algorithm should achieve near

Box Level Averaging:

	Line by Line	Std. Dev.	Full Page	Std. Dev.
Baseline Algorithms				
Linear Aligner FRONT	100.0	0.0	100.0	0.0
Linear Aligner BACK	100.0	0.0	100.0	0.0
Char. Pos. ($T_i \rightarrow \beta(D_i)$)	69.9	45.9	7.0	25.5
Char. Pos. ($\beta(D_i) \rightarrow T_i$)	85.2	35.5	11.2	31.5
Upper Bound	100.0	0.0	100.0	0.0
Dynamic Time Warping (using Basic Features +)				
Char. Pos., Perfect Clusters	100.0	2.0	98.9	10.6
Perfect Clusters	99.9	3.4	97.7	14.9
Char. Pos., Real Clusters	99.5	7.4	94.2	23.4
Real Clusters	98.6	11.7	91.0	28.6
Char. Pos.	99.7	5.3	84.6	36.1
No Additional Features	99.3	8.6	88.2	32.2

Table 5. β^{hand} Edit Distance

100%. Table 13 indicates that DTW achieves this for line by line alignment (99.5%), and comes close for full page alignment (91.0%).

The results for this set of experiments did not have parameters re-estimated from the training data (i.e. we used the parameters from β^{auto}). If they did performance would have been better because the path for DTW would have presumably been constrained to just be along the diagonal and hence DTW would collapse into simple linear alignment.

9. Future Work

We have shown that dynamic time warping provides a correct algorithm in the sense that when given hand segmented documents it produces very good mappings. In addition, when using automatically segmented documents, it achieves high performance compared to other algorithms for mapping transcripts to handwritten documents.

As our results show, there can still be a large increase in alignment accuracy,for example, as our clustering gets better. Further investigations into clustering or other methods for recognizing very common words will help improve our results further. Other areas improvement for DTW include making the algorithm more flexible. Currently the algorithm must assign at least one word to every box, and each word in the transcript at least once. It would be very useful if our alignment algorithm could avoid these two constraints, because of the errors resulting from segmentation.

Ultimately, we foresee the segmentation and alignment system working as an iterative process where each iteration refines the output, until no changes occur.

Further areas of research exist in trying to leverage imperfect transcripts of documents. For instance, it might be

more expedient to read historical documents out loud and have an automatic speech recognition (ASR) system produce an ASCII transcript. Of course, ASR is not perfect and will introduce errors in the transcript. Developing algorithms to deal with the noisiness from both transcripts and segmentations will be even more challenging then the problem addressed in this paper.

Another challenging task to be addressed in the area of alignment is non-standard documents. For instance, it is not clear that our techniques that assume documents consist of prose, will also adapt to mathematical formulas and diagrams.

References

[1] T. Ho and G. Nagy. OCR with no shape training. In *Proceedings of 15th ICPR*, Barcelona, Sept 3–8, 2000.

[2] J. D. Hobby. Matching document images with ground truth. *International Journal on Document Analysis and Recognition*, 1(1):52–61, Feb. 1998.

[3] S. Kane, A. Lehman, and E. Partridge. Indexing George Washington's handwritten manuscripts. Technical Report MM-34, Center for Intelligent Information Retrieval, University of Massachusetts Amherst, 2001.

[4] M. Kay and M. Röscheisen. Text-translation alignment. *Computational Linguistics*, Volume 19(1):121–142, March 1993.

[5] V. I. Levenshtein. Binary codes capable of correcting spurious insertions and deletions of ones. *Russian Problemy Peredachi Informatsii*, pages 1:12–25, 1965. (Original in Russian. English translation in *Problems of Information Transmission* 1:8-17, 1965).

[6] R. Manmatha and N. Srimal. Scale space technique for word segmentation in handwritten documents. In *Scale-Space Theories in Computer Vision*, pages 22–33, 1999.

[7] T. Rath and R. Manmatha. Word image matching using dynamic time warping. In *Proceedings of CVPR-03*, Madison, WI, June 16-22, 2003.

[8] T. M. Rath, V. Lavrenko, and R. Manmatha. A statistical approach to retrieving historical manuscript images without recognition. Technical Report MM-42, Center for Intelligent Information Retrieval, University of Massachusetts Amherst, 2003.

[9] D. K. Roy and C. Malamud. Speaker identifi cation based text to audio alignment for an audio retrieval system. In *Proc. ICASSP '97*, pages 1099–1102, Munich, Germany, 1997.

[10] H. Sakoe and S. Chiba. Dynamic programming optimization for spoken work recognition. *IEEE Trans. on Acoustics, Speeh and Signal Processing*, 26:623–625, 1980.

[11] C. Tomai, B. Zhang, and V. Govindaraju. Transcript mapping for historic handwritten document images. In *8th International Workshop on Frontiers in Handwriting Recognition*, pages 413–418, Niagara-on-the-Lake, ON, August 6-8 2002.

[12] R. Triebel. Automatische erkennung von handgeschriebenen worten mithilfe des level-building algorithmus. Master's thesis, Institut fur Informatik, alber-Ludwigs-Universtät Freiburg, Dec. 1999. (In German).

Appendix: Remaing Evaluation Measures

Document Level Averaging:

	Line by Line	Std. Dev.	Full Page	Std. Dev.
Baseline Algorithms				
Linear Aligner FRONT	46.9	17.5	3.4	3.9
Linear Aligner BACK	43.1	10.6	3.5	4.7
Char. Pos. ($T_i \rightarrow \beta(D_i)$)	47.9	12.3	5.2	3.7
Char. Pos. ($\beta(D_i) \rightarrow T_i$)	56.3	12.4	4.4	4.1
Upper Bound	82.2	11.3	82.2	11.3
Dynamic Time Warping (using Basic Features +)				
Char. Pos., Perfect Clusters	69.1	11.2	63.2	13.1
Perfect Clusters	67.8	11.3	63.7	11.4
Real Clusters	64.4	11.4	54.8	12.2
Char. Pos., Real Clusters	66.7	11.2	52.1	17.8
Char. Pos.	66.2	11.1	45.5	20.9
No Additional Features	63.8	11.0	52.2	14.1

Line Level Averaging:

	Line by Line	Std. Dev.	Full Page	Std. Dev.
Baseline Algorithms				
Linear Aligner FRONT	47.5	38.7	4.0	15.6
Linear Aligner BACK	42.2	39.9	3.7	13.5
Char. Pos. ($T_i \rightarrow \beta(D_i)$)	49.5	31.0	5.8	16.1
Char. Pos. ($\beta(D_i) \rightarrow T_i$)	54.7	29.2	4.6	13.6
Upper Bound	82.2	22.0	82.2	22.0
Dynamic Time Warping (using Basic Features +)				
Char. Pos., Perfect Clusters	66.9	29.3	59.4	32.2
Perfect Clusters	65.7	29.9	59.7	31.4
Real Clusters	62.6	30.9	51.3	33.4
Char. Pos., Real Clusters	64.8	30.1	49.4	34.9
Char. Pos.,	64.2	29.9	43.4	35.5
No Additional Features	61.9	30.9	48.9	33.4

Box Level Averaging:

	Line by Line	Std. Dev.	Full Page	Std. Dev.
Baseline Algorithms				
Linear Aligner FRONT	44.8	49.7	3.5	18.3
Linear Aligner BACK	42.2	49.4	3.6	18.7
Char. Pos. ($T_i \rightarrow \beta(D_i)$)	46.3	49.9	5.2	22.1
Char. Pos. ($\beta(D_i) \rightarrow T_i$)	55.1	49.7	4.3	20.3
Upper Bound	80.8	39.4	80.8	39.4
Dynamic Time Warping (using Basic Features +)				
Char. Pos., Perfect Clusters	67.9	46.7	62.5	48.4
Perfect Clusters	66.6	47.2	62.7	48.4
Real Clusters	63.1	48.2	53.7	49.9
Char. Pos., Real Clusters	65.6	47.5	51.8	50.0
Char. Pos.	65.0	47.7	45.0	49.8
No Additional Features	62.6	48.4	51.0	50.0

Table 6. β^{auto} Exact Match

Document Level Averaging:

	Line by Line	Std. Dev.	Full Page	Std. Dev.
Baseline Algorithms				
Linear Aligner FRONT	50.3	18.3	3.7	4.1
Linear Aligner BACK	45.4	11.2	4.0	4.8
Char. Pos. ($T_i \rightarrow \beta(D_i)$)	52.9	12.1	6.3	4.1
Char. Pos. ($\beta(D_i) \rightarrow T_i$)	59.6	14.3	5.1	4.7
Upper Bound	83.3	11.1	83.3	11.1
Dynamic Time Warping (using Basic Features +)				
Char. Pos., Perfect Clusters	69.0	11.6	62.6	13.5
Perfect Clusters	66.9	11.4	62.8	11.4
Real Clusters	63.8	11.4	54.0	12.2
Char. Pos., Real Clusters	67.0	11.4	51.8	17.8
Char. Pos., Real Clusters	67.0	11.5	45.7	21.0
No Additional Features	63.5	11.1	52.1	14.1

Line Level Averaging:

	Line by Line	Std. Dev.	Full Page	Std. Dev.
Baseline Algorithms				
Linear Aligner FRONT	51.5	37.6	4.4	16.0
Linear Aligner BACK	44.9	36.6	4.3	14.4
Char. Pos. ($T_i \rightarrow \beta(D_i)$)	54.4	28.0	7.0	17.3
Char. Pos. ($\beta(D_i) \rightarrow T_i$)	58.9	30.8	5.8	16.1
Upper Bound	83.0	21.5	83.0	21.5
Dynamic Time Warping (using Basic Features +)				
Char. Pos.,Perfect Clusters	68.4	26.5	60.8	30.9
Perfect Clusters	66.7	27.3	60.9	30.0
Real Clusters	64.1	28.1	52.8	32.0
Char. Pos., Real Clusters	66.6	27.2	51.0	33.7
Char. Pos.	66.6	26.9	45.4	34.8
No Additional Features	63.8	27.9	51.0	32.3

Box Level Averaging:

	Line by Line	Std. Dev.	Full Page	Std. Dev.
Baseline Algorithms				
Linear Aligner FRONT	48.0	50.0	3.7	18.9
Linear Aligner BACK	44.2	49.7	4.1	19.8
Char. Pos. ($T_i \rightarrow \beta(D_i)$)	50.8	47.2	6.0	22.7
Char. Pos. ($\beta(D_i) \rightarrow T_i$)	57.9	49.4	5.0	21.7
Upper Bound	81.0	39.0	81.0	39.0
Dynamic Time Warping (using Basic Features +)				
Char. Pos., Perfect Clusters	69.7	44.9	64.7	46.9
Perfect Clusters	68.3	45.5	65.0	46.7
Real Clusters	65.3	46.6	56.4	48.6
Char. Pos., Real Clusters	67.8	45.7	54.1	48.9
Char. Pos.	68.0	45.5	47.8	49.0
No Additional Features	65.4	46.4	54.2	48.6

Table 7. β^{auto} Precision

Document Level Averaging:

	Line by Line	Std. Dev.	Full Page	Std. Dev.
Baseline Algorithms				
Linear Aligner FRONT	45.0	15.9	3.4	3.8
Linear Aligner BACK	40.8	9.7	3.7	4.6
Char. Pos. ($T_i \rightarrow \beta(D_i)$)	57.3	13.2	7.2	4.7
Char. Pos. ($\beta(D_i) \rightarrow T_i$)	53.4	11.7	4.6	4.1
Upper Bound	82.2	10.8	82.2	10.8
Dynamic Time Warping (using Basic Features +)				
Char. Pos., Perfect Clusters	68.6	11.9	62.5	13.8
Perfect Clusters	67.0	11.8	63.3	11.9
Real Clusters	64.2	11.8	55.3	12.5
Char. Pos., Real Clusters	66.7	11.8	52.3	18.1
Char. Pos.	66.6	11.9	46.4	21.4
No Additional Features	63.9	11.5	53.5	14.6

Line Level Averaging:

	Line by Line	Std. Dev.	Full Page	Std. Dev.
Baseline Algorithms				
Linear Aligner FRONT	47.0	36.3	4.0	15.2
Linear Aligner BACK	42.4	36.7	3.9	13.4
Char. Pos. ($T_i \rightarrow \beta(D_i)$)	58.0	28.1	7.8	18.7
Char. Pos. ($\beta(D_i) \rightarrow T_i$)	52.6	28.1	4.7	13.1
Upper Bound	82.1	22.1	82.1	22.1
Dynamic Time Warping (using Basic Features +)				
Char. Pos., Perfect Clusters	68.1	26.7	60.5	30.9
Perfect Clusters	66.7	27.3	61.0	30.1
Real Clusters	64.2	28.0	53.5	32.1
Char. Pos., Real Clusters	66.4	27.3	51.1	33.7
Char. Pos.	66.3	27.0	45.6	34.9
No Additional Features	64.0	27.9	51.7	32.4

Box Level Averaging:

	Line by Line	Std. Dev.	Full Page	Std. Dev.
Baseline Algorithms				
Linear Aligner FRONT	45.3	48.6	3.5	18.1
Linear Aligner BACK	41.7	48.1	3.7	18.6
Char. Pos. ($T_i \rightarrow \beta(D_i)$)	54.6	49.0	6.5	24.2
Char. Pos. ($\beta(D_i) \rightarrow T_i$)	54.4	48.2	4.3	19.8
Upper Bound	81.0	39.0	81.0	39.0
Dynamic Time Warping (using Basic Features +)				
Char. Pos., Perfect Clusters	69.4	44.9	64.5	46.8
Perfect Clusters	68.2	45.4	65.1	46.6
Real Clusters	65.4	46.5	57.0	48.7
Char. Pos., Real Clusters	67.5	45.6	54.4	49.0
Char. Pos.	67.8	45.4	48.3	49.2
No Additional Features	65.6	46.4	55.2	48.9

Table 8. β^{auto} Recall

Document Level Averaging:

	Line by Line	Std. Dev.	Full Page	Std. Dev.
Baseline Algorithms				
Linear Aligner FRONT	100.0	0.0	100.0	0.0
Linear Aligner BACK	100.0	0.0	100.0	0.0
Char. Pos. ($T_i \rightarrow \beta(D_i)$)	61.2	6.8	5.4	6.4
Char. Pos. ($\beta(D_i) \rightarrow T_i$)	85.4	3.6	10.7	6.5
Upper Bound	100.0	0.0	100.0	0.0
Dynamic Time Warping (using Basic Features +)				
Char. Pos., Perfect Clusters	100.0	0.2	98.7	1.3
Perfect Clusters	99.9	0.3	97.2	2.0
Char. Pos., Real Clusters	99.3	1.2	93.7	4.6
Real Clusters	98.3	1.7	90.2	4.9
Char. Pos.	99.7	0.7	83.8	19.5
No Additional Features	99.2	1.1	87.8	6.6

Line Level Averaging:

	Line by Line	Std. Dev.	Full Page	Std. Dev.
Baseline Algorithms				
Linear Aligner FRONT	100.0	0.0	100.0	0.0
Linear Aligner BACK	100.0	0.0	100.0	0.0
Char. Pos. ($T_i \rightarrow \beta(D_i)$)	64.0	29.7	6.0	18.3
Char. Pos. ($\beta(D_i) \rightarrow T_i$)	86.9	15.9	13.4	25.9
Upper Bound	100.0	0.0	100.0	0.0
Dynamic Time Warping (using Basic Features +)				
Char. Pos., Perfect Clusters	100.0	1.2	98.3	9.6
Perfect Clusters	99.9	1.8	96.8	12.6
Char. Pos., Real Clusters	99.5	4.1	93.4	19.0
Real Clusters	98.7	7.2	89.9	22.3
Char. Pos.	99.8	3.2	84.3	31.9
No Additional Features	99.4	5.2	87.6	25.0

Box Level Averaging:

	Line by Line	Std. Dev.	Full Page	Std. Dev.
Baseline Algorithms				
Linear Aligner FRONT	100.0	0.0	100.0	0.0
Linear Aligner BACK	100.0	0.0	100.0	0.0
Char. Pos. ($T_i \rightarrow \beta(D_i)$)	60.9	48.8	5.2	22.2
Char. Pos. ($\beta(D_i) \rightarrow T_i$)	85.2	35.5	10.7	30.9
Upper Bound	100.0	0.0	100.0	0.0
Dynamic Time Warping (using Basic Features +)				
Char. Pos., Perfect Clusters	100.0	2.2	98.7	11.4
Perfect Clusters	99.9	3.8	97.3	16.2
Char. Pos., Real Clusters	99.3	8.1	93.7	24.2
Real Clusters	98.4	12.6	90.3	29.6
Char. Pos.	99.7	5.5	84.2	36.5
No Additional Features	99.2	9.0	87.6	33.0

Table 9. β^{hand} Exact Match

Document Level Averaging:

	Line by Line	Std. Dev.	Full Page	Std. Dev.
Baseline Algorithms				
Linear Aligner FRONT	100.0	0.0	100.0	0.0
Linear Aligner BACK	100.0	0.0	100.0	0.0
Char. Pos. ($T_i \rightarrow \beta(D_i)$)	65.4	5.7	6.7	7.3
Char. Pos. ($\beta(D_i) \rightarrow T_i$)	85.4	3.6	10.5	6.5
Upper Bound	100.0	0.0	100.0	0.0
Dynamic Time Warping (using Basic Features +)				
Char. Pos., Perfect Clusters	100.0	0.2	98.6	1.4
Perfect Clusters	99.9	0.3	97.1	2.2
Char. Pos., Real Clusters	99.3	1.2	93.2	5.1
Real Clusters	98.2	1.8	89.4	5.5
Char. Pos.	99.7	0.8	82.8	20.4
No Additional Features	99.2	1.2	87.2	7.0

Line Level Averaging:

	Line by Line	Std. Dev.	Full Page	Std. Dev.
Baseline Algorithms				
Linear Aligner FRONT	100.0	0.0	100.0	0.0
Linear Aligner BACK	100.0	0.0	100.0	0.0
Char. Pos. ($T_i \rightarrow \beta(D_i)$)	69.8	24.3	7.7	19.7
Char. Pos. ($\beta(D_i) \rightarrow T_i$)	86.9	15.9	13.2	25.9
Upper Bound	100.0	0.0	100.0	0.0
Dynamic Time Warping (using Basic Features +)				
Char. Pos., Perfect Clusters	100.0	1.1	98.5	8.4
Perfect Clusters	99.9	1.6	97.1	11.0
Char. Pos., Real Clusters	99.5	3.7	93.7	18.0
Real Clusters	98.8	6.7	90.4	21.2
Char. Pos.	99.8	2.9	84.7	31.3
No Additional Features	99.4	4.7	88.3	23.6

Box Level Averaging:

	Line by Line	Std. Dev.	Full Page	Std. Dev.
Baseline Algorithms				
Linear Aligner FRONT	100.0	0.0	100.0	0.0
Linear Aligner BACK	100.0	0.0	100.0	0.0
Char. Pos. ($T_i \rightarrow \beta(D_i)$)	67.7	42.9	6.4	23.1
Char. Pos. ($\beta(D_i) \rightarrow T_i$)	85.2	35.5	10.6	30.8
Upper Bound	100.0	0.0	100.0	0.0
Dynamic Time Warping (using Basic Features +)				
Char. Pos., Perfect Clusters	100.0	1.8	98.9	10.0
Perfect Clusters	99.9	3.1	97.8	14.0
Char. Pos., Real Clusters	99.5	6.8	94.4	22.3
Real Clusters	98.6	10.9	91.3	27.2
Char. Pos.	99.7	4.9	84.9	35.3
No Additional Features	99.3	7.8	88.8	30.5

Table 10. β^{hand} Precision

Document Level Averaging:

	Line by Line	Std. Dev.	Full Page	Std. Dev.
Baseline Algorithms				
Linear Aligner FRONT	100.0	0.0	100.0	0.0
Linear Aligner BACK	100.0	0.0	100.0	0.0
Char. Pos. ($T_i \rightarrow \beta(D_i)$)	74.9	5.2	8.3	8.9
Char. Pos. ($\beta(D_i) \rightarrow T_i$)	85.4	3.6	10.5	6.5
Upper Bound	100.0	0.0	100.0	0.0
Dynamic Time Warping (using Basic Features +)				
Char. Pos., Perfect Clusters	100.0	0.1	99.1	1.0
Perfect Clusters	99.9	0.2	98.2	1.5
Char. Pos., Real Clusters	99.6	0.8	95.0	3.9
Real Clusters	98.9	1.2	92.5	4.1
Char. Pos.	99.8	0.6	85.3	19.4
No Additional Features	99.4	0.8	90.4	5.8

Line Level Averaging:

	Line by Line	Std. Dev.	Full Page	Std. Dev.
Baseline Algorithms				
Linear Aligner FRONT	100.0	0.0	100.0	0.0
Linear Aligner BACK	100.0	0.0	100.0	0.0
Char. Pos. ($T_i \rightarrow \beta(D_i)$)	77.1	21.1	9.6	23.4
Char. Pos. ($\beta(D_i) \rightarrow T_i$)	86.9	15.9	13.2	25.8
Upper Bound	100.0	0.0	100.0	0.0
Dynamic Time Warping (using Basic Features +)				
Char. Pos., Perfect Clusters	100.0	0.8	98.9	7.3
Perfect Clusters	99.9	1.1	97.9	9.3
Char. Pos., Real Clusters	99.7	2.8	94.8	16.5
Real Clusters	99.1	5.1	92.3	19.3
Char. Pos.	99.8	2.4	86.1	30.5
No Additional Features	99.6	3.8	90.3	22.0

Box Level Averaging:

	Line by Line	Std. Dev.	Full Page	Std. Dev.
Baseline Algorithms				
Linear Aligner FRONT	100.0	0.0	100.0	0.0
Linear Aligner BACK	100.0	0.0	100.0	0.0
Char. Pos. ($T_i \rightarrow \beta(D_i)$)	74.7	43.5	7.9	27.0
Char. Pos. ($\beta(D_i) \rightarrow T_i$)	85.2	35.5	10.6	30.8
Upper Bound	100.0	0.0	100.0	0.0
Dynamic Time Warping (using Basic Features +)				
Char. Pos., Perfect Clusters	100.0	1.7	99.1	9.4
Perfect Clusters	99.9	2.9	98.2	13.1
Char. Pos., Real Clusters	99.6	6.3	95.1	21.6
Real Clusters	98.9	10.2	92.5	26.3
Char. Pos.	99.8	4.6	85.8	35.0
No Additional Features	99.5	7.4	90.2	29.8

Table 11. β^{hand} Recall

Document Level Averaging:

	Line by Line	Std. Dev.	Full Page	Std. Dev.
Baseline Algorithms				
Linear Aligner FRONT	50.4	17.8	4.2	4.0
Linear Aligner BACK	47.2	10.6	4.3	5.1
Char. Pos. ($T_i \rightarrow \beta(D_i)$)	59.8	12.5	6.7	4.3
Char. Pos. ($\beta(D_i) \rightarrow T_i$)	65.3	12.0	6.0	4.9
Upper Bound	88.4	8.3	88.4	8.3
Dynamic Time Warping (using Basic Features +)				
Char. Pos., Perfect Clusters	77.9	10.1	70.7	13.3
Perfect Clusters	75.6	10.5	71.0	10.9
Real Clusters	71.9	10.4	61.4	12.0
Char. Pos. Real Clusters	75.5	10.0	58.8	18.8
Char. Pos.	75.2	10.2	51.5	22.4
No Additional Features	71.4	10.2	58.7	14.3

Line Level Averaging:

	Line by Line	Std. Dev.	Full Page	Std. Dev.
Baseline Algorithms				
Linear Aligner FRONT	51.3	38.8	4.9	16.6
Linear Aligner BACK	46.4	39.7	4.5	14.3
Char. Pos. ($T_i \rightarrow \beta(D_i)$)	61.7	29.7	7.4	18.6
Char. Pos. ($\beta(D_i) \rightarrow T_i$)	64.1	28.8	6.3	15.9
Upper Bound	88.5	17.1	88.5	17.1
Dynamic Time Warping (using Basic Features +)				
Char. Pos. Perfect Clusters	76.0	26.6	66.9	32.0
Perfect Clusters	73.7	28.0	67.0	31.1
Real Clusters	70.3	29.3	58.0	33.7
Char. Pos. Real Clusters	73.8	27.6	56.1	35.6
Char. Pos.	73.5	27.4	49.5	37.0
No Additional Features	69.8	29.3	55.4	34.2

Box Level Averaging:

	Line by Line	Std. Dev.	Full Page	Std. Dev.
Baseline Algorithms				
Linear Aligner FRONT	48.3	50.0	4.3	20.2
Linear Aligner BACK	46.4	49.9	4.4	20.5
Char. Pos. ($T_i \rightarrow \beta(D_i)$)	58.3	49.3	6.5	24.7
Char. Pos. ($\beta(D_i) \rightarrow T_i$)	64.4	47.9	5.9	23.6
Upper Bound	87.3	33.3	87.3	33.3
Dynamic Time Warping (using Basic Features +)				
Char. Pos. Perfect Clusters	76.9	42.2	70.2	45.8
Perfect Clusters	74.6	43.5	70.2	45.7
Real Clusters	70.9	45.4	60.5	48.9
Char. Pos., Real Clusters	74.5	43.6	58.6	49.3
Char. Pos.	74.2	43.7	51.2	50.0
No Additional Features	70.5	45.6	57.6	49.4

Table 12. β^{auto} **Edit Distance**

Document Level Averaging:

	Line by Line	Std. Dev.	Full Page	Std. Dev.
Baseline Algorithms				
Linear Aligner FRONT	100.0	0.0	100.0	0.0
Linear Aligner BACK	100.0	0.0	100.0	0.0
Char. Pos. ($T_i \rightarrow \beta(D_i)$)	70.1	5.9	7.3	7.6
Char. Pos. ($\beta(D_i) \rightarrow T_i$)	85.4	3.6	11.1	6.4
Upper Bound	100.0	0.0	100.0	0.0
Dynamic Time Warping (using Basic Features +)				
Char. Pos., Perfect Clusters	100.0	0.2	98.8	1.2
Perfect Clusters	99.9	0.3	97.6	1.8
Char. Pos., Real Clusters	99.4	1.0	94.1	4.4
Real Clusters	98.6	1.6	90.9	4.7
Char. Pos.	99.7	0.7	84.2	19.3
No Additional Features	99.2	1.1	88.4	6.4

Line Level Averaging:

	Line by Line	Std. Dev.	Full Page	Std. Dev.
Baseline Algorithms				
Linear Aligner FRONT	100.0	0.0	100.0	0.0
Linear Aligner BACK	100.0	0.0	100.0	0.0
Char. Pos. ($T_i \rightarrow \beta(D_i)$)	72.5	24.9	8.0	20.6
Char. Pos. ($\beta(D_i) \rightarrow T_i$)	86.9	15.9	13.8	25.8
Upper Bound	100.0	0.0	100.0	0.0
Dynamic Time Warping (using Basic Features +)				
Char. Pos., Perfect Clusters	100.0	1.0	98.5	9.2
Perfect Clusters	99.9	1.5	97.2	11.9
Char. Pos., Real Clusters	99.6	3.7	93.8	18.4
Real Clusters	98.9	6.4	90.7	21.6
Char. Pos.	99.8	3.0	84.7	31.6
No Additional Features	99.4	4.9	88.2	24.6

Box Level Averaging:

	Line by Line	Std. Dev.	Full Page	Std. Dev.
Baseline Algorithms				
Linear Aligner FRONT	100.0	0.0	100.0	0.0
Linear Aligner BACK	100.0	0.0	100.0	0.0
Char. Pos. ($T_i \rightarrow \beta(D_i)$)	69.9	45.9	7.0	25.5
Char. Pos. ($\beta(D_i) \rightarrow T_i$)	85.2	35.5	11.2	31.5
Upper Bound	100.0	0.0	100.0	0.0
Dynamic Time Warping (using Basic Features +)				
Char. Pos., Perfect Clusters	100.0	2.0	98.9	10.6
Perfect Clusters	99.9	3.4	97.7	14.9
Char. Pos., Real Clusters	99.5	7.4	94.2	23.4
Real Clusters	98.6	11.7	91.0	28.6
Char. Pos.	99.7	5.3	84.6	36.1
No Additional Features	99.3	8.6	88.2	32.2

Table 13. β^{hand} **Edit Distance**