Steven Hill*, Zhimin Zhou, Lawrence Saul, and Hovav Shacham

# On the (In)effectiveness of Mosaicing and Blurring as Tools for Document Redaction

**Abstract:** In many online communities, it is the norm to redact names and other sensitive text from posted screenshots. Sometimes solid bars are used; sometimes a blur or other image transform is used. We consider the effectiveness of two popular image transforms — mosaicing (also known as pixelization) and blurring — for redaction of text. Our main finding is that we can use a simple but powerful class of statistical models — so-called hidden Markov models (HMMs) — to recover both short and indefinitely long instances of redacted text. Our approach borrows on the success of HMMs for automatic speech recognition, where they are used to recover sequences of phonemes from utterances of speech. Here we use HMMs in an analogous way to recover sequences of characters from images of redacted text. We evaluate an implementation of our system against multiple typefaces, font sizes, grid sizes, pixel offsets, and levels of noise. We also decode numerous real-world examples of redacted text. We conclude that mosaicing and blurring, despite their widespread usage, are not viable approaches for text redaction.

**Keywords:** redaction, mosaic, pixelation, blur, hidden markov models

DOI 10.1515/popets-2016-0047
Received 2016-02-29; revised 2016-06-02; accepted 2016-06-02.

## 1 Introduction

In many online communities, it is the norm to redact names and other sensitive text from posted screen shots. Reddit, for example, enforces a rule against the posting of personally identifying information. Many different techniques are used for redaction; the techniques applied and the tools used vary by community.[1]

Redaction has importance beyond adhering to community norms. Images ineffectively redacted and posted by members of at-risk communities (e.g., Reddit's /r/CreepyPMs) could render those users vulnerable to retribution. Accordingly, it is important to study and understand the effectiveness of widely used techniques for image redaction.

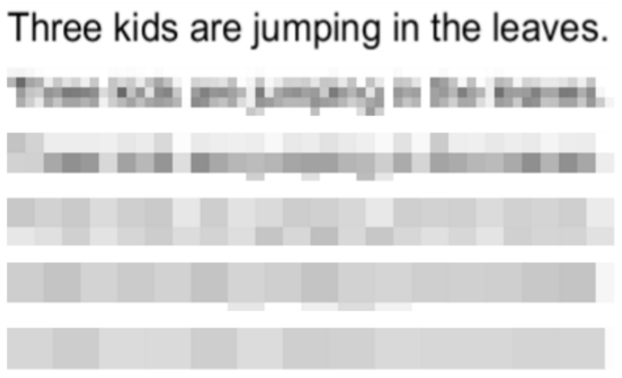

Three kids are jumping in the leaves.

**Fig. 1.** An example of mosaiced text (24p Arial font) with various grid sizes. From top to bottom: 1p, 6p, 12p, 18p, 24p, and 30p.

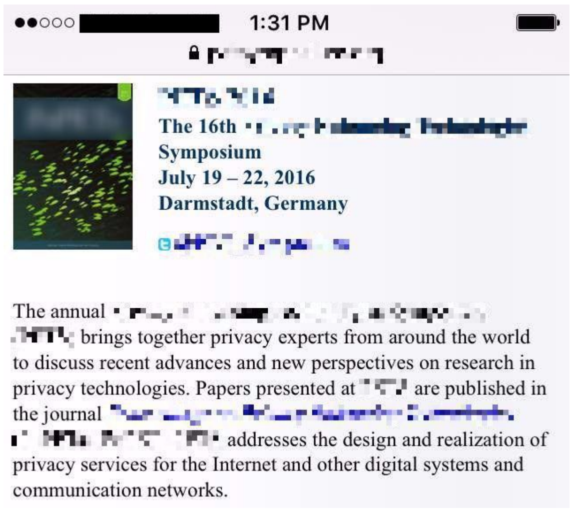
In this paper, we study the effectiveness of two popular techniques for the redaction of text: *mosaicing*, also called pixelization, and *blurring*. A mosaiced image is obtained by superposing a rectangular grid over the original image and averaging the color values of the pixels within each grid cell. Figure 1 shows the effect of mosaicing on an example sentence with various grid sizes. A blurred image is obtained by convolving the image with a two-dimensional Gaussian. Mosaicing and blurring are available in both Photoshop and GIMP, and they are two of the three filters supported by the popular Facepixelizer tool.[2] Figure 2 shows an iOS screenshot redacted by mosaicing and blurring using Facepixelizer.

Mosaicing and blurring occupy a middle ground between invertible image transformations, such as Photoshop's twirl filter [16], that can be reversed to recover the original image, and those that entirely obscure the redacted portion, for example covering it by a solid black bar. Both mosaicing and blurring are lossy, so they cannot, in general, be reversed to recover the original image. But if the original image has predictable regularities — as occur in text — then enough information may remain

***Corresponding Author: Steven Hill:** UC San Diego, E-mail: s5hill@eng.ucsd.edu

1 See, e.g., "Facebook Redacting," which documents over a dozen redaction techniques in use. Online: http://www.holyjuan.com/2010/12/facebook-redacting.html. Last visited February 28, 2016.

2 Online: http://www.facepixelizer.com/. Last visited February 28, 2016.


●●○○○ [redacted] 1:31 PM

[illegible]

[illegible]

The 16th [illegible]
Symposium
July 19 – 22, 2016
Darmstadt, Germany

[illegible]

The annual [illegible] brings together privacy experts from around the world to discuss recent advances and new perspectives on research in privacy technologies. Papers presented at [illegible] are published in the journal [illegible] addresses the design and realization of privacy services for the Internet and other digital systems and communication networks.

**Fig. 2.** A (cropped) screenshot redacted by mosaicing, blurring, and a black bar, using Facepixelizer.

after filtering to recover the redacted text, or at least to narrow down its space of possibilities.

In a 2007 blog post [19], Venkatraman described a brute force algorithm for recovering mosaiced text. His approach was to render every possible character sequence of mosaiced text and then identify the message whose mosaiced image is nearest to the one under investigation. A naïve implementation of Venkatraman's algorithm would take exponential time in the length of the redacted passage. In Section 3.2 we describe a version that runs in linear time and obtains the same results in practice, through effective search space pruning. Venkatraman's approach can also be adapted to recovering blurred text.

While we have yet to witness a large-scale attack on redacted text, it is important to recognize the widespread potential for abuse. Simple online searches return scores of images for redacted names, phone numbers, email addresses, passwords, credit card numbers, personal checks, private conversations, and other sensitive information. Mosaicing and blurring have also been used for the redaction of high-profile government documents and celebrity social media. It is self-evident that an attacker could exploit such sensitive information for malicious purposes (e.g., identity theft, blackmail).

**Our results.** We present an alternative approach to recovering mosaiced or blurred text that is more effective than Venkatraman's approach. We show that a simple but powerful class of statistical models — so-called hidden Markov models (HMMs) — can be used to recover instances of redacted text. Our approach borrows from the success of HMMs for automatic speech recognition, where they are used to recover sequences of phonemes from speech waveforms. Here we use HMMs in an analogous way to recover sequences of characters from images of redacted text. We recall the necessary background on HMMs in Section 2.

In practice, our approach works as follows. Given a snippet of redacted text, we assume that its basic parameters — for example, its typeface, font size, and granularity of resolution — can be manually identified[3] from the pixelated image as well as any surrounding text that is not redacted. After these parameters have been identified, the rest of our approach is fully automated, consisting of three steps. First, we generate a large data set of mosaiced or blurred text with these same parameters. Next, using this data set, we estimate an HMM that models the joint distribution over known character sequences and their mosaiced images. Finally, equipped with this statistical model, we use the Viterbi algorithm in HMMs — an instance of dynamic programming — to infer the redacted text of the original snippet. Interestingly, our approach to recovering blurred text first applies a mosaic to the blurred region, highlighting the fact that mosaicing can serve both as a lossy transformation and a form of image error correction.

We evaluate implementations of our approach against multiple typefaces, font sizes, mosaic grid offsets, and levels of image degradation for mosaicing (Section 3) and blurring (Section 4). In each case, we compare against our implementation of Venkatraman's approach. Broadly speaking, both approaches do equally well in "easy" cases: where the grid size or blur radius is small compared to the font size, and the redacted image has not been degraded. However, our HMM approach substantially outperforms Venkatraman's approach in "hard" cases. Our approach is more resilient to noise: it recovers readable text even when the image has been degraded by JPEG compression at 0% quality, where Venkatraman's approach does not. Where the underlying text conforms to a language model, our approach can exploit that model in addition to local information; as a result, it can recover readable text from English sentences rendered in an 18p font to which a 24p mosaic has been applied, whereas Venkatraman's approach cannot.

We conclude that hidden Markov models allow near-perfect recovery of text redacted by mosaicing or blur-

3 This assumption can be relaxed, however, at the expense of estimating an HMM for each possible setting of these parameters.

ring for many common fonts and parameter settings, and that mosacing and blurring are not effective choices for textual document redaction. We believe that a solid bar provides the best choice for all consequential redactions of text in an image. An even better approach, if the source document is available, is to replace any sensitive text with a placeholder, then render the modified document to a new image; unlike a black bar, this approach would obscure the length of any redacted passage.

**Related work.** Even drawing a solid black bar over text to be redacted may not always suffice to obscure it. At the Eurocrypt 2004 rump session, Naccache and Whelan used font metrics to recover text redacted by a solid black bar. Given a short list of a priori possible phrases, they measured which phrase fit best into the redacted space [14]. The implications of the Naccache-Whelan attack were considered in two papers by Lopresti and Spitz [9, 10]. Ho and Chang observed further that information about redacted text can remain in JPEG compression artifacts [7].

The recovery of mosaiced text can be viewed as a super-resolution technique for images containing textual data; see Mancas-Thillou and Mirmehdi for a survey [12]. Previous work in superresolution for text appears to be incomparable with our approach; it applies to more general deformations but requires a larger amount of residual information in the processed image.

Mosaicing may also be ineffective for redacting faces from images and video. Newton, Sweeney, and Malin showed that face recognition software can be used to recognize mosaiced faces [15] from still images. Likewise, Cavedon, Foschini, and Vigna[1] used superresolution techniques to recover mosaiced faces from video, assuming that the subjects on video did not move much between consecutive frames. (This possibility was earlier noted by Dufaux [4].) See Padilla-López, Chaarouni, and Flórez-Revuelta for a recent survey on effective image redaction [17]. Ford and Mayron considered the effectiveness of redaction in satellite images [6]; among their contributions is an algorithm for automatically detecting mosaiced or blurred map tiles that could be adapted to detecting redacted images in other contexts.

A great deal of work has also been done on the general problem of image deblurring. In 2010, Chen was able to produce high-quality deblurred results with low computation costs [3]. There is also commercial software[4] that attempts to deblur images whose blur parameters are provided as user input. Our approach differs in that we are specifically aiming to recover text that has been redacted by mosaicing.

Hidden Markov models of English text have been used in other security research — for example, to recover keystrokes from their acoustic emanations [21], and to recover spoken text from the size of encrypted VoIP packets [20]. More recently, work has also been done in exploiting biases in RC4 [22, 23].

## 2 Hidden Markov models

In this section we review the basic ideas of hidden Markov modeling [5, 18], then describe in detail how we use HMMs to decode mosaiced text.

### 2.1 Background

A hidden Markov model is a probabilistic graphical model (see Figure 3) of hidden and observed random variables that evolve over time. We denote the hidden variable (or *state*) at time $t$ by $S_t$ and the observed variable (or *observation*) at time $t$ by $O_t$. Likewise we use $S_{1:t} = \{S_1, S_2, ..., S_t\}$ and $O_{1:t} = \{O_1, O_2, ..., O_t\}$ to denote sequences of these states and observations.

The hidden states in HMMs form a simple Markov process: the state $S_t$ is conditionally independent of past states $S_{1:t-2}$ given the immediately preceding state $S_{t-1}$. Likewise, the observation $O_t$ is conditionally independent of previous states $S_{1:t-1}$ and observations $O_{1:t-1}$ given the hidden state $S_t$. Given these assumptions of conditional independence, it is simple matter to compose the HMM's joint distribution over state and

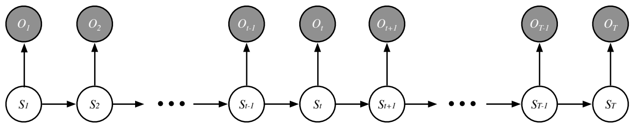


**Fig. 3.** Hidden Markov model for recovery of redacted text. The hidden state $S_t$ labels the characters that are present in the $t$th sliding window of redacted text; the observation $O_t$ is a quantized descriptor of the pattern of pixel intensities in the $t$th sliding window. The Viterbi algorithm in HMMs infers the most probable sequence of hidden states (unshaded) from the observations (shaded).

4 Online: http://smartdeblur.net/. Last visited February 28, 2016.

observation sequences:

$$P(S_{1:T}, O_{1:T}) = P(S_1)\prod_{t=2}^{T} P(S_t \mid S_{t-1}) \prod_{t=1}^{T} P(O_t \mid S_t) \tag{1}$$

where $T$ is the overall sequence length. In discrete HMMs, as we consider here, both the hidden states $S_t \in \{1, 2, ..., n\}$ and the observations $O \in \{1, 2, ..., k\}$ are drawn from finite alphabets. The joint distribution in eq. (1) is then parameterized by the following elementary probabilities:

$$\pi_i = P(S_1 = i), \tag{2}$$
$$a_{ij} = P(S_{t+1} = j | S_t = i), \tag{3}$$
$$b_{i\ell} = P(O_t = \ell | S_t = i), \tag{4}$$

where $i, j \in \{1, 2, \ldots, n\}$ and $\ell = \{1, 2, \ldots, k\}$. The parameters $\pi, \mathbf{a}, \mathbf{b}$ are known respectively as the HMM's start distribution, transition matrix, and emission matrix.

There are efficient algorithms for probabilistic inference and learning in HMMs. Most important for our application is the Viterbi algorithm, which computes the most likely sequence of hidden states given a particular sequence of observations. More precisely, for an HMM with parameters $\pi, \mathbf{a}, \mathbf{b}$, the Viterbi algorithm computes the hidden state sequence of maximum posterior probability:

$$S^*_{1:T} = \underset{S_{1:T}}{\arg\max} P(S_{1:T} \mid O_{1:T}) \tag{5}$$

The required computation in eq. (5) is a simple instance of dynamic programming; it decodes the optimal state sequence in time $O(n^2 T)$. Note that this procedure has no dependence on the size of the observation alphabet, $k$.

HMMs have been widely used for problems in automatic speech recognition [18] and bioinformatics [5]. Our own use of them most closely recalls their previous application to automatic handwriting recognition [8]. In our problem, the observations are not segments of cursive handwriting, but segments of mosaiced text; see Figure 4. In both these problems, though, the goal is the same: to decode the sequence of characters (represented by the model's hidden states) that correspond to the observed message.

While the main challenge in handwriting recognition is to model the variability of cursive penmanship, the main challenge in our problem is to overcome the lower resolution of mosaiced text. Our problem is simplified in one respect, though: in practical applications, we can often identify the font of redacted text by the surrounding text that is not redacted. Moreover, even when this is not the case, we can generally assume that the font characteristics do not change midway through the text that we wish to recover. A practical strategy is therefore to build a separate HMM-based recognizer for each individual font of interest. The following subsections describe in a general way how this is done, while Section 3 gives further details for several particular types of recognizers (e.g., for digit strings, for email addresses, for natural language).

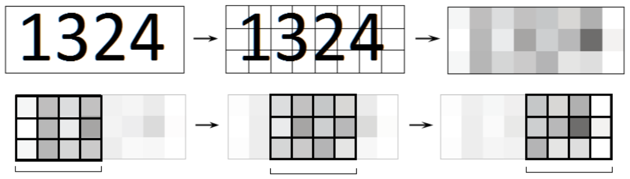


**Fig. 4.** *Top:* an image of mosaiced text is produced by averaging grids of pixels. *Bottom:* the redacted text is segmented by sliding windows that align with grid boundaries (though not necessarily character boundaries). Each window is characterized by its vector of grid values; by quantizing these vectors, we obtain sequences of observations that can be modeled by discrete HMMs.

## 2.2 States

The states in our HMMs label the characters that appear in narrow, sliding windows of redacted text. Note that the boundaries of these windows do not generally align with character boundaries; see the top panel of Figure 4. Thus it is not possible to label each window by a single character. For example, in the bottom panel of Figure 4, the leftmost window would be correctly labeled by the pair of digits $(1, 3)$, the middle window by the triple of digits $(3, 2, 4)$ (because this window just barely overlaps the leftmost part of the numeral *four*), and the right window by the pair of digits $(2, 4)$. Since the sliding windows are of fixed width, some states may represent an interior *slice* of a single wide character (e.g., the letter **M**), while others may represent a concatenation of several narrow characters (e.g., the letters **i** and **l** followed by a comma).

The number of states $n$ in our HMMs depends on two factors. First and foremost, it depends on the application at hand: for example, the number of states may be very large if we are attempting to recover generic text (consisting of lower and upper-case letters, punctuation, digits, etc), or it may be relatively small if we are merely attempting to recover phone numbers or bank accounts (consisting only of digits). Second, the number of states depends on the width of the sliding window used to extract slices of mosaiced text. In general, the wider the window, the more directly the HMM can model the dependencies between adjacent characters in text. But wider windows necessarily increase the size of the HMM's state space — because a state must be allocated for each possible subsequence of characters that can appear within one window. There is a trade-off here: though HMMs with larger state spaces may yield better models, the parameters of these HMMs will require larger amounts of data to be reliably estimated. (Recall that the transition matrix is of size $n \times n$.) The window size of each system must be chosen with this tradeoff in mind. For real-world applications, we found that this tradeoff was easily managed. The HMMs in Section 3 had state spaces as small as $n = 110$ (for recognizing digit strings) and as large as $n = 14593$ (for recognizing email addresses).

The start distribution $\pi_i = P(S_1 = i)$ and transition matrix $a_{ij} = P(S_{t+1} = j|S_t = i)$ of these HMMs are easily estimated from data. In particular, *because we generate the data ourselves by rendering known snippets of text*, it is a simple matter to align the sliding windows of these mosaiced images with their correct underlying state sequences. The parameters $\pi_i$ and $a_{ij}$ are then estimated, respectively, from the empirical frequencies of start-states and state-state transitions. The snippet of text in Figure 4, for example, would generate one count each for the initial state $(1, 3)$ and for the transitions $(1, 3) \to (3, 2, 4)$ and $(3, 2, 4) \to (2, 4)$. We obtain maximum likelihood estimates for the HMM's start distribution and transition matrix by accumulating such counts over the entire data set and taking appropriate ratios to ensure that the parameters represent properly normalized probabilities.

## 2.3 Observations

The observations in our HMMs are quantized descriptors of the sliding, pixelated windows that contain redacted text; see Figure 4. Each window is characterized by its vector of grid values. It would be possible to model the distribution over such vectors directly, using continuous-density HMMs. By quantizing these vectors, however, we obtain sequences of observations that can be modeled by discrete HMMs, which are simpler to estimate. Moreover, as we show in the next sections, discrete HMMs already suffice to demonstrate the ineffectiveness of text redaction by mosaicing and blurring.

We use the $k$-means clustering algorithm [11] for vector quantization. The $k$-means algorithms takes as input a collection of real-valued vectors and returns as output a set of $k$ representative prototypes in the same vector space. Quantized descriptors are then obtained by mapping every vector to its nearest prototype (as measured by Euclidean distance). In our case, the inputs to the $k$-means algorithm are the vectors of gridded pixel values obtained from sliding windows of redacted text. Sequences of these vectors are then mapped to sequence of discrete observations $\{O_1, O_2, \ldots, O_T\}$, where $O_t \in \{1, 2, \ldots, k\}$.

A question naturally arises how to choose $k$. Larger values of $k$ yield finer quantizations of the observation space, but they also lead to larger emission matrices $b_{i\ell} = P(O_t = \ell|S_t = i)$; recall that the emission matrix is of size $n \times k$. Once again this creates a trade-off: though HMMs with finer descriptors (i.e., larger $k$) may yield better models, the emission matrices of these HMMs will require larger amounts of data to be reliably estimated. The value of $k$ must be chosen with this tradeoff in mind. We determined $k$ automatically by evaluating the performance of HMMs on held-out validation sets of data; this worked well for our applications. The HMMs in Section 3 had alphabets as small as $k = 900$ (for recognizing digit strings) and as large as $k = 7300$ (for recognizing email addresses).

Once prototypes have been computed by the $k$-means algorithm, it remains to estimate the emission matrix $b_{i\ell} = P(O_t = \ell|S_t = i)$. We do this by assigning each window of gridded pixel values to its nearest prototype and aligning the resulting observation sequences $\{O_{1:T}\}$ with their corresponding state sequences $\{S_{1:T}\}$. (Once again, the latter are known to us because we generate the data ourselves from known snippets of text.) We accumulate the counts of co-occurrences between states and observations in an $n \times k$ matrix; the $i\ell^{\text{th}}$ element in this matrix records how many times, over the entire data set, a window mapped to the $\ell^{\text{th}}$ quantized descriptor was aligned with the character(s) represented by the $i^{\text{th}}$ state of the HMM. Finally, we obtain maximum likelihood estimates of the emission probabilities from the normalized values of these counts.

It should be noted that other methods exist for clustering, such as density or linkage-based models, but they are generally slower or more complicated than the $k$-means algorithm. As mentioned previously, it is also possible, using continuous-density HMMs, to model directly the observations of windowed images of text. We experimented briefly with such HMMs, using mixtures of multivariate Gaussian distributions for the emission densities. In general, we found that these more sophisticated approaches were not necessary to decode redacted images of text; the simpler models (e.g., $k$-means clustering, discrete HMMs) described in this section were quite sufficient.

## 3 Experiments for Mosaicing

We studied the effectiveness of HMMs for recovery of redacted text in a number of different settings. In particular, we experimented with different fonts, mosaic parameters, levels of noise, and message types. We also compared our HMMs to a brute-force heuristic [19] proposed for recovery of redacted text. (This latter approach does not require any statistical modeling.) In this section, we discuss the general methodology of our experiments and describe the brute-force heuristic in more detail. Then we present empirical findings for three different types of redacted text — U.S. bank account numbers, email addresses, and whole sentences. We also discuss the challenges of decoding real-world examples of redacted text from the Internet.

### 3.1 Methodology

Each HMM that we train is designed to recognize redacted text of a particular typeface, font size, mosaic grid size, and offset in pixels from the mosaic's origin. In addition, for each HMM it is necessary to select a window size and shift parameter for analyzing the redacted text. Once these particulars are specified, it is possible to generate a large data set of training examples to estimate the parameters of the HMM. (We discuss the specifics of these data sets in subsequent sections.) Our general procedure for estimating each HMM involves the following steps:

1. Render multiple instances of text as images. This was done using the Java Advanced Window Toolkit (AWT).
2. Create mosaiced text by pixelating the images. This was done using the built-in mosaicing feature of GIMP.
3. Divide the instances of mosaiced text into a training set (for estimating the HMM's start distribution, transition and emission matrices), a held-out validation set (for tuning all the other parameters of the recognizer), and a test set (for evaluating the HMM's performance on unseen examples).
4. Segment the images of redacted text into sliding, overlapping windows. The number of windowed segments is determined by the window size and shift parameters, which also must be specified by the user.
5. Quantize the vectors of gridded pixel values in each window using the $k$-means algorithm. (Appropriate values of $k$ can be determined by the HMM's performance on the held-out instances in the validation set.)
6. Estimate the model parameters on the instances of redacted text in the training set, as described in the previous section.
7. Evaluate the accuracy of the HMM on the instances of redacted text in the test set (and/or apply the HMM to a real-world example with matched typeface, grid size, and other characteristics).

In addition, we can introduce other optional steps into this procedure — such as the addition of noise or distorting artifacts (e.g., JPEG compression) — to evaluate the robustness of the HMMs when the test instances of redacted text do not perfectly conform to the training instances.

We use a simple variant on edit distance[5] to evaluate the accuracy of recovered text. In particular, let $x$ and $x'$ denote the original and recovered strings of text, respectively. Then we define

$$\text{score}(x') = (100\%) \times \left(1 - \frac{\text{EditDistance}(x, x')}{\text{Length}(x)}\right)$$

as the score of the recovered text. This score provides a useful measure of accuracy: a score of 100% indicates a perfect match to the original text, while a score of 0% indicates that the recovered text differs by a number of insertions, deletions, and substitutions equal to the length of the original text itself. As a visual aid, Table 5 shows the scores for some examples of recovered text. We see that a score of 70 or above is easily understood as natural language, while below that it becomes very difficult. In general, the higher the score, the less effective the redaction.

5 The edit distance between two strings computes the number of insertions, deletions, and substitutions required to transform one string into the other.

| Score (%) | Recovered Text |
|---|---|
| 100 | Nobody is practicing water safety and wearing preservers. |
| 90 | Nobody is practicing water safety and wesaring prerservers. |
| 80 | Nobody is peracticing waler safey ard wearing preservemas. |
| 70 | Nokedy is praching water safey anct waring pracarvers. |
| 60 | Nokedy is pracking watbr watwong anoftean ing pracarvers. |
| 50 | Nobody is jersetiking w sler baPesng anowir wsaling junsthrsuns. |
| 40 | Nobody is g machoing bre barbon aro f stalig g mackeroris |

**Table 1.** Examples of scores of recovered text. The higher the score, the less effective the redaction.

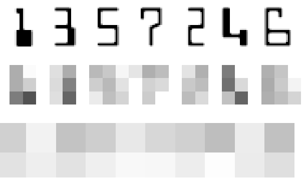


**Fig. 5.** Top row: a seven-digit string rendered in 24p MICR Encoding font. Middle and bottom rows: mosaiced images with grid sizes of 8p and 18p, respectively.

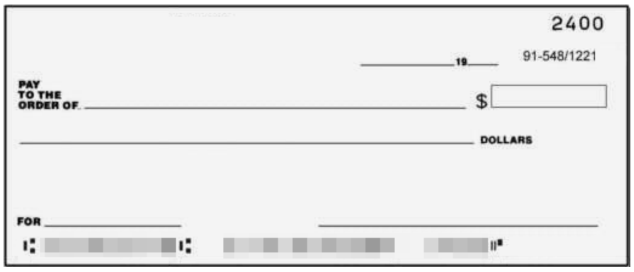


**Fig. 6.** Toy check with routing and account number redacted. Recreated from http://levittownnow.com/2016/01/04/cops-concert-organizer-wrote-bad-check/.

## 3.2 An Enumerative Approach

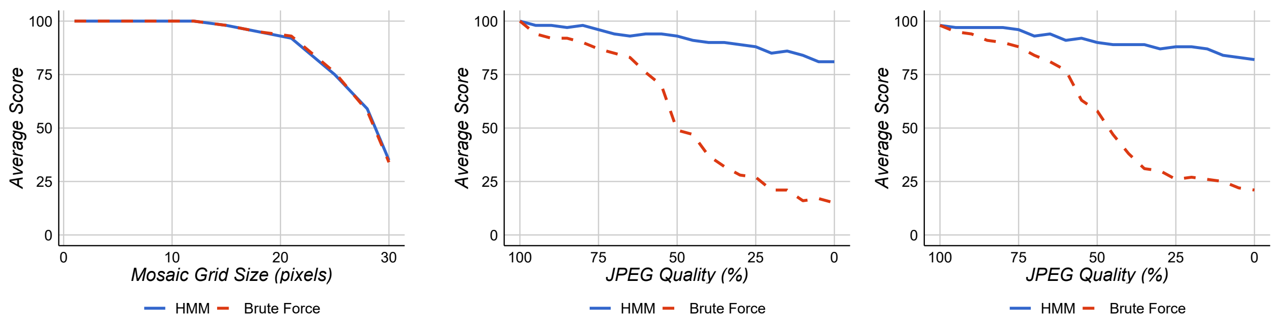
We compare our approach to a brute-force method suggested by Venkatraman [19]. The brute-force method does not involve any statistical modeling; instead, given an image of redacted text, it enumerates all possible strings that fit the observed horizontal and vertical dimensions of text, generates an image of mosaiced text for each of these strings, and outputs the string whose mosaiced image most closely matches the original one. This method is straightforward to implement for strings of very short length, and for strings of indefinite length, we consider a greedy variant of the method that searches for the best match in sub-exponential time. Pseudocode for this approach is shown below.

```
1: function BruteForceDecode(Mosaic m)
2:     gridsOfM := the grids of $m$
3:     sol := $\epsilon$                                    ▷ Empty string
4:     for $t = 1 \ldots \mathrm{len}(gridsOfM)$ do
5:         grids := the tiles from $0 \to t$.
6:         cs := the sequence of characters such that
           the length of $sol+cs$ is greater than or equal
           to the length of grids, and that the Euclidean
           distance between mosaic($sol + cs$)
           and grids is minimized.
7:         $sol := sol + cs'$ such that $cs'$ is the longest
           substring of cs that the length of $sol + cs'$
           is less than or equal to the length of grids.
8:     end for
9:     return sol
10: end function
```

Function *BruteForceDecode* is based on the same enumerative approach, but it takes into account that the effects of mosaicing are localized: i.e., the pixelation of one character in a string does not affect the pixelation of other distant characters. In this approach, a window size of text is chosen for which it is feasible to enumerate all possible strings. The function then recovers characters of the redacted text, one grid cell at a time, as it scans the mosaiced image from left to right. Specifically, the function starts by finding the best matching characters in the leading window of the mosaiced text; this is done by a brute-force search. After this search, it fixes the characters of the match that lie in the leftmost grid cell. Then it shifts the analysis window by one grid cell to the right, and the process repeats, but with all subsequent matches conditioned on the results of previous ones. This approach greedily prunes the search space of possible strings as the analysis window slides across the image of redacted text. It is not strictly guaranteed to reproduce the same result as an exhaustive search. However, it will succeed to do so if the window size is sufficiently large and the evidence within each window is unambiguous.

## 3.3 U.S. Bank Account Numbers

Our first application, also drawn from Venkatraman [19], is to analyze mosaiced images of digit strings that appear on U.S. checks. This is the simplest application that we consider in this paper; here, the redacted text consists exactly of seven evenly spaced digits. Figure 5 shows an example of such text both before and after it has been redacted by mosaicing.



**(a)** Score vs mosaic grid size **(b)** Score vs pre-mosaic JPEG quality **(c)** Score vs post-mosaic JPEG quality

**Fig. 7.** Accuracy of text recovered by HMM-based and brute-force recognizers for mosaiced images of seven-digit strings in 24p MICR Encoding font. Panel (a) shows the accuracy versus mosaic grid sizes in the absence of JPEG compression. Panels (b) and (c) show the accuracy versus JPEG quality with a fixed mosaic grid size of 14p. See text for details.

To train HMMs for this application, we randomly generated 20,000 seven-digit numbers and rendered their strings in 24 point MICR Encoding font. Note that the number of examples in our data set (20,000) is smaller than the total number ($10^7$) of possible digit strings by several orders of magnitude. We split the 20,000 examples into three sets: 10,000 for training, 5,000 for validation, and 5,000 for testing. The 15,000 examples in the training and validation sets are then mosaiced at various grid sizes and used to estimate HMMs that recognize mosaiced text.

Figure 7(a) compares the performance of the HMM-recognizer and brute-force method when they are evaluated on the mosaiced images of digit strings in the test set. Note that this evaluation corresponds to the idealized setting in which the mosaiced test images are not subject to any additional forms of noise or distortion. In this setting, the two methods yield comparable results, both of them recovering the redacted text over a wide range of grid sizes. Not surprisingly, the accuracies of both methods fall off when the mosaic grid size (as measured in pixels) approaches the font size. It seems that irresolvable ambiguities arise when the mosaic grid size is large enough to span multiple digits; moreover, as the grid size approaches the dimensions of the full text, the mosaicing filter (just like redaction by a black bar) results in a complete loss of information. (Later we will see that the HMM-recognizers are more resilient to larger grid sizes when recovering text from natural language.)

Next we consider a more realistic evaluation. It is extremely rare for images to be released on the Internet in a lossless format. More typically, images are found in JPEG format, with some degree of quality sacrificed for smaller file sizes. Figure 8 shows how the image quality of text is affected by JPEG compression at different levels. Of course, other forms of distortion — if the image was obtained by scanning a hard-copy document — are also possible.

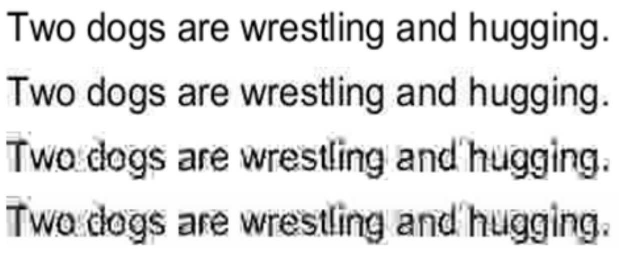
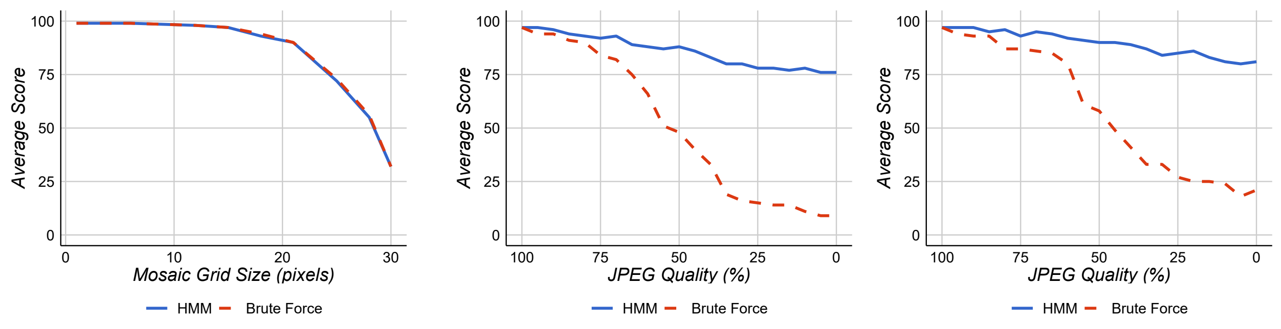

Two dogs are wrestling and hugging.
Two dogs are wrestling and hugging.
Two dogs are wrestling and hugging.
Two dogs are wrestling and hugging.

**Fig. 8.** Loss of quality through JPEG compression. Shown are JPEQ qualities of 100%, 66%, 33%, and 0% respectively.

We studied this more realistic setting by subjecting the test images in our data set to different levels of JPEG compression. Note that this is a mismatched testing scenario for both the HMM-based recognizers (which are trained on uncompressed images) and the brute-force method (which is comparing to uncompressed images). For completeness, we also experimented with JPEG compression that was applied either before or after the redaction by mosaicing. In the case of post-mosaicing compression, we obtained pixelated images for testing by computing the median pixel value in each grid cell.

Figures 7(b-c) show the results of these experiments. In the presence of JPEG artifacts, the HMM-based and brute-force recognizers are no longer comparable. In particular, the HMM-based recognizers are fairly robust to JPEG artifacts, whereas the brute-force recognizers,



**(a)** Score vs mosaic grid size **(b)** Score vs pre-mosaic JPEG quality **(c)** Score vs post-mosaic JPEG quality

**Fig. 9.** Accuracy of text recovered by HMM-based and brute-force recognizers for mosaiced images of email addresses in 24p Arial font. Panel (a) shows the accuracy versus mosaic grid size in the absence of JPEG compression. Panels (b) and (c) show the accuracy versus JPEG quality with a fixed mosaic grid size of 14p.

which search for exact matches of grid cell values, are easily misdirected. It seems that the probabilistic underpinnings of HMMs allow them to cope with greater uncertainty.

It is natural to ask why the HMM-based recognizers outperform the brute-force approach in this situation. Recall that the HMM's inference procedures, based on dynamic programming, are able to search efficiently over the whole space of possible digit strings. For the brute-force recognizers, however, we approximated the search by considering a greedy algorithm that runs in linear time. One possible explanation, therefore, is that the greedy approximation is introducing errors that would be overcome by a truly exhaustive search. However, we do not believe this to be the case. In particular, we verified that the brute-force recognizers obtained the same average accuracies even when the sliding window was doubled and quadrupled in size. We also verified that in the presence of JPEG artifacts, the greedy approach produced lower-error matches than the original digit strings.

The above suggests to look elsewhere for the weaknesses of the brute-force approach. It seems instead that the brute-force recognizers are mainly compromised by the distance function they use to compute matches. When exact matches are possible — when the redacted text has been mosaiced but not subject to other forms of noise or distortion — simple Euclidean distance between grid cells is an effective criterion for matching. This does not appear to be the case, however, in less idealized settings. Venkatraman notes that other distance metrics are likely to be more effective [19], though the design of such metrics is not straightforward. It is worth pointing out, then, that the HMM-based recognizers do not rely on simple template-matching via Euclidean distance. In fact, the quantized descriptors from the $k$-means algorithm and the emission matrices in HMMs are highly adaptive to the underlying density of mosaiced images. This appears to be a significant advantage of the HMM-based recognizers.

## 3.4 Email addresses

Another prominent application of mosaicing is the redaction of email addresses. Many users on social media (e.g, Twitter, Facebook, message boards) post images of email that they wish to share, but pixelate the email addresses to protect the privacy of the sender and/or recipient.

To begin, we show by a detailed example how to recover such email addresses with our approach. To mimic a real-world application, we took a screenshot of an email that we received from Google Wallet and asked a third party to mosaic out the email address. The resulting image is shown in Figure 10.

Before training an HMM, it was necessary to perform some manual forensics on the image; in particular, we needed to identify the font's typeface, size, and color, the grid size of the mosaicing filter, and the offsets (in pixels) of the text from the origin of the mosaic. The font characteristics were easily identified from Google Gmail, and the grid size by visual inspection. However, it required a trial-and-error approach to identify the offsets. (In fact, this is often the most time-consuming part of the recovery process.)

Once we identified the basic font and mosaicing parameters, the rest of the recovery process was fully automated. As data for our HMM, we randomly generated

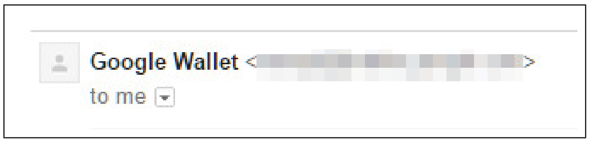


**Fig. 10.** Example of an email address redacted by mosaicing. Our HMM-based recognizer recovers the text `noreply@wallet.google.com`, which is the correct address.

| Randomly Generated Email Address |
|---|
| o4srjgz1d9ta@6yp20 |
| hr9@c09z4g8razbtaa |
| innuz@wmvq4lphj36 |
| 9ybwj4wqzxb@64jd.3.qgrrt |
| gyna2h@7xt1ajwp2cb6 |
| mo3of8p43ej2h@ehtmo |

**Table 2.** Examples of randomly generated email addresses used to train our HMM-based recognizers.

| SICK Sentence |
|---|
| The young boys are playing outdoors and the man is smiling nearby. |
| Two young women are sparring in a kickboxing fight. |
| Nobody in snowsuits is lying in the snow and making snow angels. |
| People wearing costumes are gathering in a forest. |
| A group of people in a large Asian restaurant is eating. |
| Pink bellbottoms and a pink scarf aren't to be worn by women. |

**Table 3.** Examples of SICK sentences

20,000 email addresses that matched a regular expression of the form `[a-zA-Z0-9._]+@[a-zA-Z0-9.]+`. Table 2 shows a set of these randomly generated email addresses. We rendered the email addresses in 24p Arial, applied a mosaic filter at the pre-determined grid size, and split these images into a training set of 10,000 images (for estimating the model parameters of the HMM), a validation set of 5,000 images (for estimating other parameters of the recognizer, such as the degree of image quantization, $k$), and a test set of 5,000 images (for evaluating more systematically the model's accuracy). Once the model was trained, we used it to decipher the redacted text in Figure 10.

Figure 11 shows the most likely sequence of hidden states (representing pairs or triples of adjacent characters) inferred by the HMM for this example. In this example, it is important to note that the mosaiced image was saved in JPEG format with lower quality than the original PNG of the screenshot. Perhaps for this reason the brute-force approach was unable to recover the email address; in particular, for this example it returned gibberish.

We also performed a more systematic evaluation of the HMM-based and brute-force recognizers on this data set. Figure 9 show results from the same suite of experiments that we ran on digit strings in the previous section. Here again we see that the HMM-based and brute-force recognizers perform equally well when the mosaiced images are not compressed, recovering email addresses over a wide range of mosaic grid sizes. However, the HMM-based recognizers are much more resilient to JPEG artifacts. As the results are similar to those of the previous section, we skip a more detailed analysis.

## 3.5 Natural language

For our last application we study the effectiveness of mosaicing as a redaction strategy for extended portions of text. The framework here is identical to those of previous sections, except that the strings we are trying to recover consist of semantically meaningful words. As we shall see, the HMM-based recognizers are very well equipped to leverage this additional structure in the text. In particular, because they explicitly model the statistics of likely character sequences, they are able to recover natural prose in more adverse settings than random digit strings or email addresses.

The SICK corpus [13] is a collection of 20,000 English sentences, originally used for the construction of distributional semantic models. Table 3 lists several sentences from the corpus. For the experiments in this section, we used 10,000 sentences for training, 5,000 for validation, and 5,000 for testing.

As a first set of experiments, each sentence was rendered in 18p Arial font and mosaiced with varying grid sizes. Figure 12 compares the accuracy of HMM-based and brute-force recognizers on the mosaiced images of these sentences. As in the previous experiments on digit strings and email addresses, we see that both types of recognizers degrade in performance as the mosaic grid size becomes larger than the font size. But now, *even in the absence of JPEG compression*, there is a significant difference between the two recognizers at larger grid sizes; the HMM-based recognizers perform much better. In fact, the brute-force recognizers exhibit a fairly catastrophic drop in performance when the grid size exceeds the font size.

Why do the two methods perform so differently at larger grid sizes? One likely explanation is that the

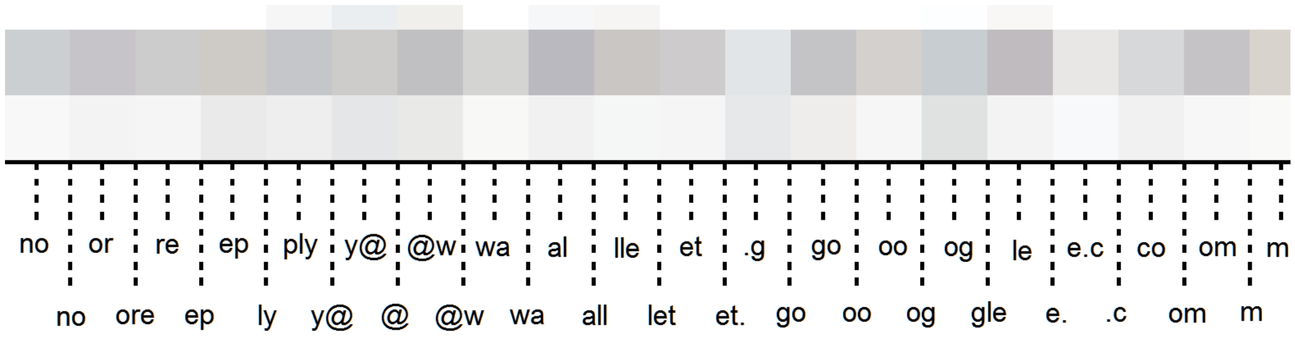


**Fig. 11.** The most likely sequence of hidden states inferred by the HMM for the mosaiced email address in Figure 10. Concatenating overlaps, we obtain the correct address `noreply@wallet.google.com`.

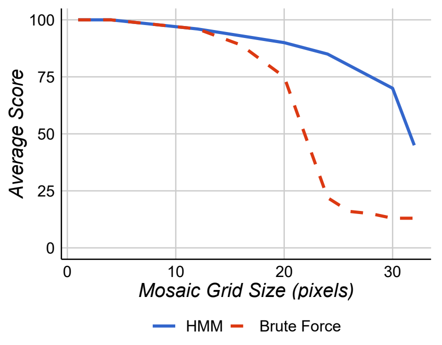


**Fig. 12.** Accuracy of text recovered by HMM-based and brute-force recognizers for mosaiced images of sentences from the SICK corpus in 18p Arial font.

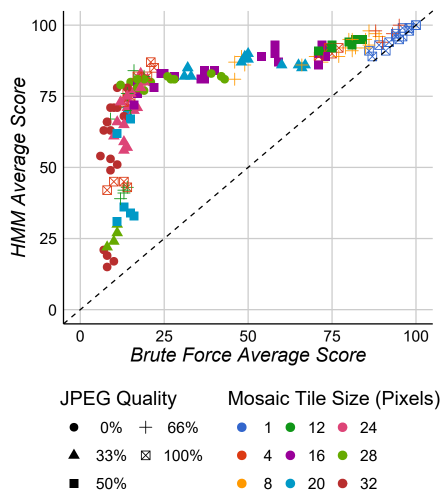


**Fig. 13.** Accuracy of recovered sentences across multiple experimental configurations (font typeface, mosaic grid size, JPEG quality) by HMM-based and brute-force recognizers. The HMM accuracies (y-axis) always exceed the brute-force accuracies (x-axis). See text for details.

HMMs are exploiting the statistics of likely character sequences. In particular, the transition matrix elements in the HMM (as learned from sentences in the SICK corpus) reflect these statistics. Consider the followed contrived example. Imagine that a mosaic grid is exactly wide enough to average the pair of letters ‘qu’. Notice in this case that a sliding window which contains a mosaiced image of ‘qu’ is indistinguishable from a sliding window which contains a mosaiced image of ‘uq’. The brute-force method, which enumerates both of these possibilities, has no way to distinguish between them. On the other hand, the HMM will assign a much higher likelihood to the former — assuming that the character ‘q’ was observed to precede the character ‘u’ in the SICK corpus many more times than the other way around. These sorts of ambiguities in the brute-force recognizer seem likely to multiply as the grid size increases, causing a propagation of errors in its recovery process.

We ran a comprehensive suite of experiments on mosaiced images of sentences from the SICK corpus, evaluating the accuracy of HMM-based and brute-force recognizers against multiple fonts (Arial, Comic Sans, Verdana, Courier), mosaic grid sizes, and levels of JPEG compression. Figure 13 summarizes the results of all these experiments in a scatterplot, with the accuracy of the brute-force recognizers plotted along the x-axis, and the accuracy of the HMM-based recognizers plotted on the y-axis. The diagonal line in the plot separates cases where the HMM-based recognizers outperformed the brute-force approach. We note that all the results occur above the diagonal; we did not observe a single scenario in which the HMM-based recognizers were outperformed by a non-statistical, enumerative approach.

| Font | Average Score (%) on SICK Corpus | | | | | | |
|---|---|---|---|---|---|---|---|
| | 8p | 12p | 16p | 20p | 24p | 28p | 32p |
| **Arial** | 92 | 93 | 93 | 93 | 93 | 94 | 96 |
| **Comic Sans MS** | 92 | 93 | 93 | 93 | 94 | 95 | 96 |
| **Courier New** | 91 | 92 | 92 | 92 | 93 | 93 | 94 |
| **Verdana** | 92 | 92 | 93 | 93 | 93 | 94 | 96 |

**Table 4.** The average scores on the SICK corpus as font type and size vary. Mosaic grid size is the same as the font size. These experiments were run with no JPEG quality loss.

Lastly we explored how well the HMM-based recognizers recover redacted text of varying font size. These results are shown in Table 4. For these experiments, we fixed the mosaic grid size to be the same as as font size (e.g., 16p grid size for 16p Arial font). The table shows that the accuracies are fairly constant over a 4x difference in font size; as expected, however, there is a small improvement for larger font sizes due to the finer clarity of their rendered text. We also found the HMM-based recognizers to recover the redacted text in real-world examples over a similar range of font sizes.

## 3.6 Real-world issues

We concede that one time-consuming element of the recovery process (which applies to both the brute-force and HMM-based approaches) is that we must first identify the correct horizontal and vertical offsets of mosaiced images, as measured in pixels, with respect to the start position and baseline of the text. This may require a laborious manual effort, so it is natural to ask whether the results are sensitive to incorrect estimates of these offsets. Figure 14 shows a confusion matrix of results when HMMs trained on mosaiced images with one vertical offset were tested on images with another vertical offset. As expected, the highest scores are obtained along the diagonal, when the training and testing offsets are perfectly matched. However, the scores fall off quickly with every pixel shift of these offsets. The results suggest that it is necessary to identify the correct offsets before the beginning of the modeling process. Another possibility would be to train multiple HMMs, one for each pairing of horizontal and vertical offsets (from zero pixels to the mosaic grid size), and evaluate them all on the redacted text of interest.

In addition to the synthetic examples described in this section, we have collected many real-world uses of mosaicing (and blurring) “in the wild” where these methods are used to redact sensitive text from online

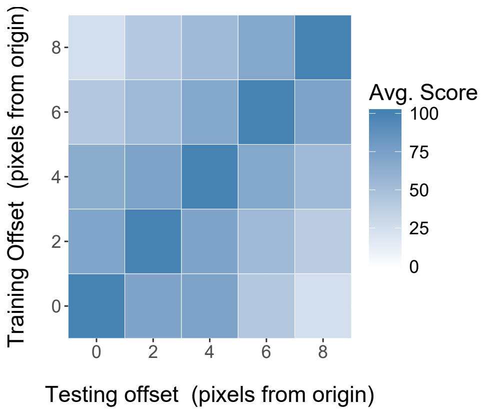


**Fig. 14.** Confusion matrix when HMMs trained with one vertical offset are tested on mosaiced images with another vertical offset. The sentences in these experiments were drawn the SICK corpus and rendered in 24p Arial font with a mosaic grid size of 14p.

screenshots. We have been successful in recovering the redacted text using our approach from each such image that we have examined; the most difficult step, as noted above, has typically been reproducing the parameters by which the original image was generated. While we understand the value of demonstrating our approach on other than synthetic examples, we think it is unethical to make the published version of this paper an archival means for the dissemination and preservation of information posted by people who meant to obscure it. However, we invite researchers who wish to study these real-world examples to contact us directly.

It is natural to ask whether similar results could have been obtained with fewer training examples, or whether better results could be obtained with even more. It is known that larger data sets generally yield better results for problems in natural language processing. For example, even larger data sets would be desirable to estimate the transition probabilities of longer-range $n$-gram models. For recovery of more constrained text, however, this does not seem necessary. Indeed, we obtained similarly accurate HMMs for check numbers with only 1K examples and for email addresses with only 5K examples.

## 4 Extension to Blurring

Another popular redaction method (perhaps more so than mosaicing) is blurring. In the most common type of blur, the original image of text is convolved with a two dimensional Gaussian whose standard deviation deter-

| Score (%) | Recovered Text |
|---|---|
| 100 | Nobody is practicing water safety and wearing preservers. |
| 90 | Nobody is practicing water safety and wesaring prerservers. |
| 80 | Nobody is peracticing waler safey ard wearing preservemas. |
| 70 | Nokedy is praching water safey anct waring pracarvers. |
| 60 | Nokedy is pracking watbr watwong anoftean ing pracarvers. |
| 50 | Nobody is jersetiking w sler baPesng anowir wsaling junsthrsuns. |
| 40 | Nobody is g machoing bre barbon aro f stalig g mackeroris |

**Table 5.** Examples of scores of recovered text. The higher the score, the less effective the redaction.

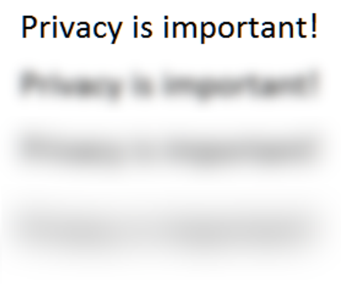


**Fig. 15.** Example of Gaussian-blurred text with blur radii (from top to bottom) of 0p, 15p, 30p, and 45p.

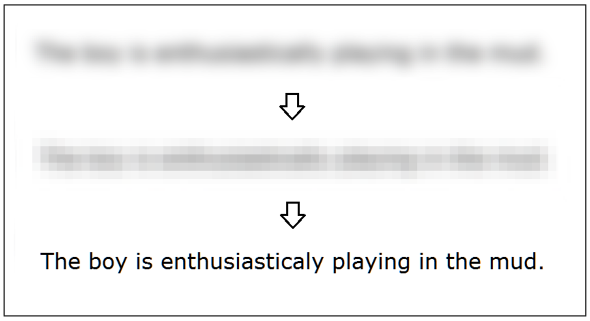


**Fig. 16.** A blurred image is mosaiced, and then the most likely text inferred by an HMM. We can see that the word *enthusiastically* is missing an '*l*', but otherwise the recovery is accurate.

mines the radius of the blur. Figure 15 shows an example of blurred text with different blur radii.

Though blurring and mosaicing both reduce the visual resolution of images, the former does not reduce the absolute number of pixels in the image. For exactly this reason, mosaicing can be a useful form of post-processing for reducing the variability of blurred images before attempting to model them. Consider, for example, an image of text that has been blurred with two slightly different blur radii. The two blurred images will have slightly different pixel values at the original resolution of the image. But if the two images are mosaiced with a grid size near their blur radius, the resulting mosaics will be identical or nearly so. We are accustomed to regarding mosaicing as an inherently lossy (therefore destructive) operation, but in this context, it is serving a useful purpose. In particular, it may be prohibitively expensive to build statistical models of blurred text for every conceivable blur radius. By working with mosaiced images, however, we can restrict ourselves to increments of the blur radius that are roughly equal to mosaic grid sizes.

With the above idea in mind, we have all the machinery in place from the previous section to build an HMM-based recognizer for blurred text. In particular, given an image of blurred text, we can roughly estimate its blur radius by mosaicing the image at different grid sizes and noting the grid size where the blurred image begins to degrade further due to the mosaicing. (There has been work [2] to calculate the blur radius analytically, or at least to estimate it more precisely, which we did not pursue.) Let us assume, also, that the font metrics (e.g., typeface, size, color) of the text can be determined from auxiliary information. Then we can follow the same procedure as before to recover the blurred text:

1. Render many images of known text in the same typeface, font size, color, etc.
2. Blur the images of text from step 1 at the estimated blur radius.
3. Mosaic the blurred images from step 2 at an appropriate grid size, so as to reduce their dimensionality without incurring a significant loss of visual information.
4. Train an HMM on the images of blurred, mosaiced text from step 3.
5. Recover the unknown blurred text by mosaicing it at the grid size from step 3 and inferring the most likely sequence of hidden states in the HMM from step 4.

Figure 16 shows an application of this approach, where an HMM recovers the text from a blurred image after it has been mosaiced. The recovery is nearly perfect despite a considerable degree of blurring.

Within this framework we also performed a more systematic set of experiments on the sentences in the SICK corpus. The text in these experiments was rendered in 24p Verdana font and convolved by Gaussian filters with different blurring radii. In addition, all blurred images were post-processed with a mosaic of grid size 8p. Our results are summarized in Figure 17, which shows the accuracy of recovered text versus the

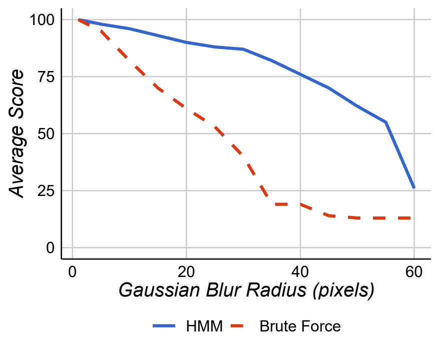


**Fig. 17.** Results on SICK sentence blurred data, generated as 24p Verdana font using 8p mosaic tiles post blur

radius of Gaussian blurring. The results are similar to those of the previous section. As one expects, the accuracy of recovered text decreases with increasing blur radius, but here again the HMM-based recognizers are more resilient to lower-resolution redactions than the brute-force recognizers.

Recall from Table 5 that an accuracy score of 70% corresponds roughly to the threshold of sensical vs nonsensical text. From Figure 17, we see that the HMM-based recognizers achieve this score up to a Gaussian blur of radius 45p, whereas the brute-force recognizers achieve this score only up to a blur of radius 15p. Figure 15 shows that a 15p blur is nearly readable to the human eye. Many real-world examples of blurring are much more severe.

## 5 Conclusion

In this paper we have demonstrated the ineffectiveness of mosaicing and blurring as tools for text redaction. Our approach, based on HMMs, is able to recover text perfectly for many common fonts and parameter settings. Moreover, in more challenging problems for recovery, our approach substantially outperforms a brute-force strategy based on exhaustive search and simple template-matching. Our results have shown that HMMs enjoy the advantages of a probabilistic, data-driven approach: when noise is present in the images, they benefit by modeling the underlying density of pixelated text, and when words are present in the text, they benefit by incorporating the statistics of likely character sequences.

Mosaicing and blurring are popular forms of redaction because they have a certain aesthetic appeal to the naked eye. The images that these methods produce are highly suggestive of text; as a result, they do not disrupt the visual appearance of documents to the same extent as cut-out or black box methods for redaction. But while mosaicing and blurring are lossy transformations, they preserve far more information than most users realize. Our goal in this paper has been to demonstrate, through the use of statistical models, just how much information these methods leave on the page. Given the widespread adoption of these methods in online communities, we hope that our results will raise a greater awareness of their weaknesses.

## 6 Acknowledgements

We thank Geoffrey Voelker, Stefan Savage, and the reviewers for many insightful comments. A special thanks is also owed to our shepherd, Aylin Caliskan-Islam, for her in-depth review and constructive suggestions. This work was supported by National Science Foundation grant NSF-1237264.

## References

[1] Cavedon, L., Foschini, L., and Vigna, G. Getting the face behind the squares: Reconstructing pixelized video streams. In *WOOT* (2011), pp. 37–45.

[2] Chen, F., and Ma, J. An empirical identification method of gaussian blur parameter for image deblurring. *Signal Processing, IEEE Transactions on 57*, 7 (2009), 2467–2478.

[3] Chen, X., Yang, J. and Wu, Q. Image deblur in gradient domain *Optical Engineering, Optical Engineering 49*, 11 (2010), 117003–117003.

[4] Dufaux, F. Video scrambling for privacy protection in video surveillance: recent results and validation framework. In *SPIE Defense, Security, and Sensing* (2011), International Society for Optics and Photonics, pp. 806302–806302.

[5] Eddy, S. What is a hidden markov model? *Nature biotechnology 22*, 10 (2004), 1315–1316.

[6] Ford, R., and Mayron, L. M. All Your Base Are Belong to US. In *Proceedings of NSPW 2012* (2012), ACM, pp. 105–14.

[7] Ho, N. Z.-Y., and Chang, E.-C. Residual information of redacted images hidden in the compression artifacts. In *Information Hiding* (2008), Springer, pp. 87–101.

[8] Hu, J., Brown, M. K., and Turin, W. Hmm based online handwriting recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 18*, 10 (1996), 1039–1045.

[9] Lopresti, D., and Spitz, A. L. Quantifying information leakage in document redaction. In *Proceedings of the 1st ACM workshop on Hardcopy document processing* (2004), ACM, pp. 63–69.

[10] Lopresti, D. P., and Spitz, A. L. Information leakage through document redaction: attacks and countermeasures. In *Electronic Imaging 2005* (2005), International Society for Optics and Photonics, pp. 183–190.

[11] MacQueen, J., et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability* (1967), vol. 1, Oakland, CA, USA., pp. 281–297.

[12] Mancas-Thillou, C., and Mirmehdi, M. An introduction to super-resolution text. In *Digital Document Processing*. Springer, 2007, pp. 305–327.

[13] Marelli, M., Menini, S., Baroni, M., Bentivogli, L., Bernardi, R., and Zamparelli, R. A sick cure for the evaluation of compositional distributional semantic models. In *Proceedings of the Ninth International Language Resources and Evaluation Conference* (2014), pp. 216–223.

[14] Naccache, D., and Whelan, C. 9/11: Who alerted the cia?(and other secret secrets). *Rump session, Eurocrypt* (2004).

[15] Newton, E. M., Sweeney, L., and Malin, B. Preserving privacy by de-identifying face images. *Knowledge and Data Engineering, IEEE Transactions on 17*, 2 (2005), 232–243.

[16] Nizza, M. Interpol untwirls a suspected pedophile. http://thelede.blogs.nytimes.com/2007/10/08/interpol-untwirls-a-suspected-pedophile/, 2007.

[17] Padilla-López, J. R., Chaaraoui, A. A., and Flórez-Revuelta, F. Visual privacy protection methods: A survey. *Expert Systems with Applications 42*, 9 (2015), 4177–4195.

[18] Rabiner, L. R. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE 77*, 2 (1989), 257–286.

[19] Venkatraman, D. Why blurring sensitive information is a bad idea. https://dheera.net/projects/blur, 2014.

[20] White, A. M., Matthews, A. R., Snow, K. Z., and Monrose, F. Phonotactic reconstruction of encrypted voip conversations: Hookt on fon-iks. In *Security and Privacy (SP), 2011 IEEE Symposium on* (2011), IEEE, pp. 3–18.

[21] Zhuang, L., Zhou, F., and Tygar, J. D. Keyboard acoustic emanations revisited. *ACM Transactions on Information and System Security (TISSEC) 13*, 1 (2009), 3.

[22] Vanhoef, M., Piessens, F. All your biases belong to us: Breaking RC4 in WPA-TKIP and TLS. In *24th USENIX Security Symposium (USENIX Security 15)*, 2015

[23] Bricout, R., Murphy, S., Paterson, K., and Merwe, T. Analysing and Exploiting the Mantin Biases in RC4. In *Cryptology ePrint Archive, Report 2016/063*, 2016