An Open-Bigram Approach to Handwritten Word Recognition

Théodore Bluche[1,*], Christopher Kermorvant[1,2], Claude Touzet[3], Hervé Glotin[4]
1 A2iA SAS, Paris, France
2 Teklia SA, Paris, France
3 UMR Neurosciences Intégratives et Adaptatives, Aix-Marseille Université, France
4 UMR CNRS LSIS, Univ. Toulon, France
*** E-mail: tb@a2ia.com**

Abstract

Recent research in the cognitive process of reading hypothesized that we do not read words by sequentially recognizing letters, but rather by identifing open-bigrams, *i.e.* couple of letters that are not necessarily next to each other. In this paper, we explore an handwritten word recognition method based on open-bigrams. We trained Long Short-Term Memory Recurrent Neural Networks (LSTM-RNNs) to predict open-bigrams rather than characters, and we show that such models are able to learn the long-range, complicated and intertwined dependencies in the input signal, necessary to the prediction. For decoding, we decomposed each word of a large vocabulary into the set of constituant bigrams, and apply a simple cosine similarity measure between this representation and the bagged RNN prediction to retrieve the vocabulary word. We compare this method to standard word recognition techniques based on sequential character recognition. Experiments are carried out on two public databases of handwritten words (Rimes and IAM), an the results with our bigram decoder are comparable to more conventional decoding methods based on sequences of letters.

Author Summary

Introduction

Taking inspiration in Biology is sometimes very efficient. For example, deep neural networks (NN) – which are outperforming all other methods (including support vector machines, SVM) in image recognition – are based on a series of about five pairs of neurons layers, each pair involving sparsity in the activation pattern (a biological trait of the cortical map). The analogy continues with the modelization of the cortex as a hierarchy of cortical maps. Thanks to the analysis of reaction time in cognitive psychology experiments, the minimal number of cortical maps involved in a cognitive process is estimated to about ten, quite close to the number of layers of an efficient deep NN. In the case of handwritten word recognition, Dehaene et al. have proposed a biologically plausible model of the cortical organization of reading [1] that assumes seven successive steps of increasing complexity, from the retinal ganglion cells to a cortical map of the orthographic word forms (Fig. 1). One of the most recent successes of experimental psychology was the demonstration that human visual word recognition uses an explicit representation of letter position order based on letter pairs: the open-bigram coding [2–6].

As demonstrated in [7], open-bigrams (OB) allow an over-coding of the orthographic form of words that facilitates recognition. OB coding favors same length words (i.e., neighbors of similar lengths). In the context of learning to read, the existence of the OB layer just before the orthographic word representation has been used to explain the lack of efficiency of whole language method (today banned from reading teaching) compared to the phonics method which explicitly supervises the organization of the OB map (with syllables), where the global method does not (Fig. 2).

Since cognitive psychology has demonstrated the existence of the OB layer, the hypothesis has been put forward [7] that the orthographic representation of words may have evolved in order to take into account the topology of the OB space, instead of the topology of the single letter space. Our goal here

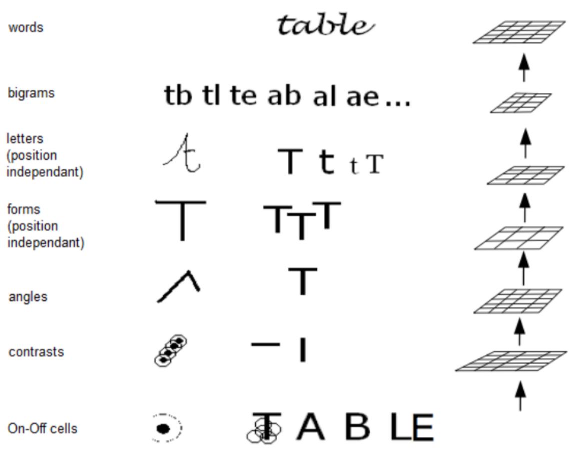


Figure 1: The cognitive process of reading, a seven steps procedure that includes an open-bigrams representation layer (adapted from [1]).

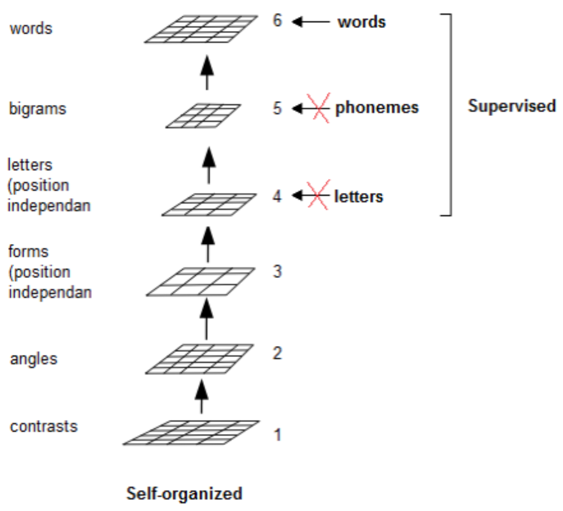


Figure 2: Additional information helps the organization of levels 4 and 5, when using a phonics method, but not a whole language method (today banned from reading teaching for lack of efficiency, adapted from [8]).

is to test this hypothesis, comparing OB vs sequential character recognition for word recognition. A state-of-art decoder based on a Long Short-Term Memory Recurrent Neural Networks (RNN) is used on two public databases of handwritten words (Rimes and IAM). The best published results on the full line recognition task of IAM [9] and Rimes [9, 10] were achieved by systems involving an RNN component.

In the first part of this paper, we will present the materials and methods. After giving a brief review of existing work on handwritten word recognition, we introduce the open-bigram representation of words. The following section details an open-bigram decoder and its application to French and English dictionnaries. The next section is devoted to the data preparation. We use the RIMES and IAM databases of handwritten words and a vocabulary of 50,000 words for each language to perform the recognition. Finally, we present our models to extract an open-bigram representation from the image, with Bidirectional Long Short-Term

Memory Recurrents Neural Networks (BLSTM-RNNs).

The second part shows the results obtained in recognition of open-bigrams. After demonstrating that RNNs are able to recognize sequences of open-bigrams, we test word recognition using bigram predictions. We compare open-bigrams vs sequence of characters, and show that the novel method we propose achieve results that are competitive to state-of-the-art models, despite its simplicity.

Finally we discuss and conclude.

Materials and Methods

In this section, we present the proposed method for handwritten word recognition based on open-bigrams. First, we give a brief overview of existing techniques for handwriting recognition, with a focus on systems based on hidden Markov models (HMMs), that are the state-of-the-art methods to which we compare our approach. Then, we introduce the open-bigram representation and the proposed decoder. Finally, we present the data selection and preparation, and the chosen method to propose bigram hypotheses from the images.

Standard Approaches for Handwritten Word Recognition

Whole-Word Models

In this first class of methods, the goal is to recognize a word directly as a whole, without relying on a segmentation or on an explicit representation of the parts. Often, a simplified representation of the word shape is extracted from the images and matched against a lexicon, for example with simplified upper and lower profiles [11], or with holistic features such as ascenders, descenders, loops and so on [12–14]. [15] feed holistic features to a three-level network. The feature level is connected to a character level, itself connected to a word level. The intermediate character level allows an iterative bottom-up and top-down approach to find the features in the image which confirm the recognition of a word.

The disadvantage of whole word modeling is the inherent limitation of the vocabulary size. The number of models grows linearly with the number of words. Moreover, for methods based on a matching to a prototype, a new prototype must be created with every added word. Classifiers must be retrained. The method of [15], with the intermediate character level, and hand-wired connections, could scale well with lexicon size, but the holistic approach might show its limitations with many different words with a similar shape. Most of these methods are applied to legal amount recognition in bank cheque processing, were the vocabulary does not exceed 40-50 words.

Part-Based Methods

In part-based methods, the image is divided into sub-regions corresponding to at most one character. Segmenting characters from cursive words, without knowing the word's identity, is challenging, due to the amiguous nature of handwritten text. Thus these systems often perform an over-segmentation of the image, or consider several segmentation alternatives. The goal is to merge different part to get complete characters, or to find the correct segmentation into characters among the various hypotheses.

The first step consists in finding possible segmentations from the image. It is often derived from heuristics about character connections. For example, [16] extract control points from vertical and horizontal extrema of the contour to extract simple strokes. [17] apply mathematical morphology operators to isolate regularities, which are potential segmentation points. Then, a shortest path computation between the leftmost and rightmost segments allow to select the segmentation points among candidates, and heuristics are applied to split, merge and order segments. In [18, 19], the segmentation also uses regularity and singularity principles. In [20], the segmentation process tries to find ligatures between characters from the contour, either by comparing with the average stroke width or by looking at concavity and convexity features. [21] look at the local minima of the upper contour. They shift the segmentation point when a loop or an unacceptable width is found. A similar method can be found in [22]. [23] cut the image horizontally into slices of width chosen according to the height of core region of the word. The width is adjusted so that the number of black pixels traversed by the cut is locally minimal. In [24], several horizontal segmentation points are found with different algorithms. [25] combine online information (such as the pen velocity) and offline information to generate segmentation hypotheses, and use heuristics to limit the number of merges in the segmentation graph. A survey of character segmentation techniques was proposed by [26].

For the actual recognition, we can differentiate the combination approach and the grapheme approach. In the combination approach, adjacent segments are merged to produce different character segmentation hypotheses, generally represented as a directed graph. In [16] the strokes are recognized by prototype alignment, and characters are hypothesized from combinations of strokes. The word is found with a best-first search. [20] represent characters by different code words obtained through a k-means clustering, and compute the distance from merged segments features to these code words. The character can also be predicted with k-nearest neighbors [24], or neural networks [23, 25]. The segmentation graph is then transformed into a prediction graph, in which the best path corresponds to the word recognition. Lexical constraints may be taken into account (e.g. in [24]). In [23], the character predictions are used as emission probabilities in a hidden Markov model, which also serves to train the neural network at the word level.

In the grapheme approach, the parts of characters (the graphemes) are directly modeled by the system, and a character is defined as a sequence of such units. Generally the characters are modeled by Hidden Markov Models (HMMs). [21, 22] extract simple features and use discrete HMMs. In [19], the graphemes in the training set are clustered into 100 classes, and a neural network is trained to predict the grapheme class. The predictions are fed to the HMM. [17] build a continuous HMM with Gaussian mixture models emission probabilities. To cope with possible under-segmentations (i.e. a grapheme corresponds to more than one character), some HMMs include a skip transition, e.g. in [19, 22].

Segmentation-Free Approach

In the segmentation-free approach, the recognition is accomplished without an explicit segmentation of the image, thus without relying on heuristics to find character boundaries, and limiting the risk of under-segmentation. This category of approaches is the most popular nowadays, receiving a lot of research interest, and achieving the best performance on standard benchmarks. In this section, we only present a brief overview of this kind of methods. They are mainly based on hidden Markov models and neural networks, which is the topic of the next chapter. More details on the modeling techniques will be presented there.

The first strategy consists in extracting a sequence of feature vectors with a sliding window [27]. The main parameters of the window are its width and shift, defining respectively the amount of context included in the feature vector and the overlap between two observations. Hidden Markov models are especially suited to transform the sequence of feature vectors into a sequence of characters. Each character is modeled by an HMM with several states. An emission model defines the probability of a state generating an observation, i.e. a feature vector. This probability model may be discrete (e.g. in [27]), continuous (usually a Gaussian mixture model, e.g. in [28]), or even estimated with more complex systems, such as neural networks [29]. A survey of Markov models for handwriting recognition can be found in [30].

In the second strategy, the system directly models the two-dimensional image input. For example, [31,32] use a two-dimensional HMM with Gaussian modeling of spectral features. [33] match keypoints (SURF) of an image to reference keypoints of each character class. The final sequence of characters is found with a dynamic programming search from the classification of keypoints. Other notable techniques are based on neural networks. More particularly convolutional neural networks, in which the two dimensional structure of the image is kept in several layers. In this structure, local receptors are defined and repeated at each position in the image. The outputs of each receptor can be arranged in two-dimensional maps, allowing to apply the concept again, and iteratively extract more complex representations. Subsampling operations, such as max pooling, reduce the size of the maps. At the end, a classification is performed at each horizontal position to provide character sequence hypotheses. This concept was for example applied in [25, 34], and used in the very successful Multi-Dimensional Recurrent Neural Networks [35].

In the models presented in this section, the input is not segmented. The segmentation of the image into characters is either performed after the recognition (e.g. in [33]), or a by-product of the recognition. For example, word or sentence recognition is achieved by concatenation of character models with HMMs, and the recognition consists in finding the most likely sequence of states, from which the character boundaries can usually be trivially inferred. To some extent, convolutional neural networks also allow to retrieve the general position of a character in the original image. Some systems, such as RNNs, model the sequence directly, and the segmentation might not be easily recovered.

State-of-the-Art Methods for Offline Handwritten Text Recognition

The problem of handwriting recognition is now widely approached with hidden Markov models [30] (the first applications of HMMs to that problem date from [27, 36]). These models are designed to handle sequential data, with hidden states emitting observations. There are well-known techniques to *(i)* compute the probability of an observation sequence given a model, *(ii)* find the sequence of states which is most likely to have produced an observed sequence, and *(iii)* find the parameters of the model to maximize the probability of observing a sequence [37]. The third problem is the actual training of these models, while the second allows to decode a sequence of observations. In this paradigm, the characters are each represented by a hidden Markov model. A simple concatenation of these character models produce word models. The advantages are twofold:

- from a few character models (around one hundred), we can build word models for potentially any word of the language. Thus this method is much more scalable to large vocabularies than systems attempting to model each word separately
- recognizing a word from character models does not require a prior segmentation of the word image into characters. Since the word model is a hidden Markov model of its own, the segmentation into characters is a by-product of the decoding procedure, which consists in finding the most likely sequence of states.

In HMMs, the observed information is not the state sequence, but a sequence of data $\mathbf{x} = x_1 \dots x_T$ generated by HMM states with an emission probability model.

An HMM λ is defined by the following elements:

- A set of states $\mathcal{Q} = \{s_1, \dots, s_N\}$. For convenience, we also define the set of all state sequences of length T : $\mathcal{Q}_T = \{\mathbf{q} = q_1 \dots q_T \ : \ q_i \in \mathcal{Q}, 1 \leq i \leq T\}$.
- A transition model, defined by the probabilities $p(q_t = s_i | q_{t-1} = s_j)$, for all $s_i, s_j \in \mathcal{Q}$.
- A probability distribution over initial states $p(q_1 = s), \forall s \in \mathcal{Q}$.
- An emission model, defined by the probabilities $p(x|s)$, where $s \in \mathcal{Q}$, and x is an observation.

The observation sequence $\mathbf{x} = x_1 \dots x_T$ is a sequence of feature vectors, extracted from the image using for example a sliding window. Each vector may contain the pixels in the window, or handcrafted features, such as histograms of gradients, or center of mass of the window.

The most standard emission model is a Gaussian mixture model (GMM):

$$p(x|s) = \sum_{m=1}^{M} c_{sm} \mathcal{N}(x; \mu_{sm}, \Sigma_{sm}) \tag{1}$$

where $\mathcal{N}(\cdot,\mu,\Sigma)$ is the multivariate Gaussian distribution with mean μ and covariance matrix Σ. c_{sm}, μ_{sm} and Σ_{sm} are respectively the weight, mean, and covariance of the m-th component of the GMM for state s.

In the hybrid neural network (NN)/HMM scheme [29], the emission model is derived from the outputs of a neural network, providing state posterior probabilities $p(s|x)$:

$$p(x|s) = \frac{p(s|x)}{p(s)}p(x) \tag{2}$$

This method using discriminative models tends to be more popular, and to give better results than GMMs. The neural network predicting HMM state posteriors $p(s|x)$ may for example be Multi-Layer Perceptrons (MLPs) [38, 39] or Recurrent Neural Networks (RNNs) [9, 40].

An Open-Bigram Representation of Words

The letter bigrams of a word w is the set of pairs of consecutive letters. The **open-bigram of order** d is the set of pairs of letters separated by d other letters in the word, which we call $\mathcal{B}_d(w)$:

$$\mathcal{B}_d(w) = \{w_i w_{i+d} : i \in \{1 \ldots |w| - d\}\} \tag{3}$$

The usual bigrams are open-bigrams of order 1. By extension, we call $\mathcal{B}_0(w)$ the set of letters in the word w. For example, for word `word`, we have:

$$\begin{aligned}\mathcal{B}_1(\texttt{word}) &= \{\texttt{or}, \texttt{rd}, \texttt{wo}\}\\ \mathcal{B}_2(\texttt{word}) &= \{\texttt{od}, \texttt{wr}\}\\ \mathcal{B}_3(\texttt{word}) &= \{\texttt{wd}\}\end{aligned}$$

The general open-bigram representation of a word is the union of

$$\mathcal{B}_{d_1,\ldots,d_n}(w) = \mathcal{B}_{d_1}(w) \cup \ldots \cup \mathcal{B}_{d_n}(w) \tag{4}$$

For example, $\mathcal{B}_{1,2,3}(\texttt{word}) = \{\texttt{od}, \texttt{or}, \texttt{rd}, \texttt{wd}, \texttt{wo}, \texttt{wr}\}$

We extend $\mathcal{B}$ into $\mathcal{B}'$ by including special bigrams for the letters at the begining and end of a word:

$$\mathcal{B}'(w) = \mathcal{B}(w) \cup \{\_w_0, w_{|w|}\_\} \tag{5}$$

So, for example,

$$\mathcal{B}'_{1,2,3}(\texttt{word}) = \{\texttt{\_w}, \texttt{d\_}, \texttt{od}, \texttt{or}, \texttt{rd}, \texttt{wd}, \texttt{wo}, \texttt{wr}\} \tag{6}$$

In this paper, we will call B the set of all bigrams, and W the set of all words. We will represent a word of the vocabulary $w \in W$ as a normalized binary vector $\mathbf{v}_{w\in W} \in \Re^{|B|}$

$$\mathbf{v}_w = \frac{[\delta(b \in \mathcal{B}(w))]_{b\in B}}{\sqrt{|\mathcal{B}(w)|}} \tag{7}$$

i.e. the vector with 0 everywhere and $1/\sqrt{|\mathcal{B}(w)|}$ at indices corresponding to bigrams of the word. The stacking of the vector representation of all the words in the vocabulary yields the vocabulary matrix $V \in \Re^{|W|\times|B|}$.

Note that in this representation, the bigrams form an unordered set. We *do not know*:

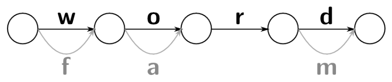
- where the bigrams are,
- what is the order of a given bigram, and
- how many times it occurs.

The goal is to build a word recognition decoder in the bigram space.

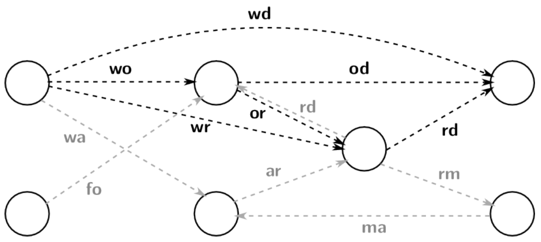
An Open-Bigram Decoder

While the trivial representation of a word is an ordered sequence of letters, the order in the bigram space is locally embeded in the bigram representation. Most state-of-the-art word recognition systems recognize sequences of letters, and organize the vocabulary for a constrained search as directed graphs, such as prefix trees, or Finite-State Transducers. On the other hand, we can interpret the bigram representation as encoding directed edges in a graph, although we will not explicitely build such a graph for decoding.

On Fig. 3, we show the graph for a representation of the word into a sequence of letters. Gray edges show the potential risk of a misrecognition in the letter sequences. On the bottom figure, we display the conceptual representation of bigrams as edges. We observe that a global order of letters can emerge from the local representation. Moreover, the consituant information of a word in the bigram space is redundant, potentially making this representation more robust to mispredictions of the optical model.



(a) Sequential representation



(b) Bigram representation

Figure 3: Word representation as an explicit sequence of letters (a), and as a set of bigrams (b). Grey edges show the potential impact of misrecognitions.

The optical model is the system which provides the predictions of bigrams from the image (or, in classical approach sequences of character predictions). That is, it provides a confidence measure that each bigram b is present in image x: $0 \leq p_b(x) \leq 1$. This is transformed in a vector in the bigram space:

$$\mathbf{q}_x = \frac{[p_b(x)]_{b \in B}}{\sqrt{\sum_b p_b^2(x)}} \tag{8}$$

For decoding, we chose the very simple cosine similarity between the query ($\mathbf{q}_x$) and a vocabulary word ($\mathbf{v}_w$). Since we normalized both vectors, this is simply the dot product:

$$d(\mathbf{q}_x, \mathbf{v}_w) = \mathbf{v}_w^T \mathbf{q}_x \tag{9}$$

so the similarity with all words of the vocabulary can be computed with a matrix-vector product:

$$D_V(x) = V^T \mathbf{q}_x \tag{10}$$

The recognized word is the one with maximum similarity with the query:

$$w^* = arg\max D_V(x) = arg\max_w \frac{\sum_{b \in \mathcal{B}(w)} p_b(x)}{\sqrt{|\mathcal{B}(w)| \sum_b p_b^2(x)}} \tag{11}$$

We carried out a few preliminary experiments to justify the open-bigram decoder. First, we considered the famous sentence with mixed up letters:

"aoccdrnig to a rscheearch at cmabrigde uinervtisy it deos not mttaer in waht oredr the ltteers in a wrod are the olny iprmoatnt tihng is taht the frist and lsat ltteers be at the rghit pclae the rset can be a toatl mses and you can sitll raed it wouthit porbelm tihs is bcuseae the huamn mnid deos not raed ervey lteter by istlef but the wrod as a wlohe".

Although the origin and validity of this statement when letters are put in the right order has been discussed [1], it is true that most of us can read it without trouble. For each word of more than one letter in this sentence, we computed the open-bigram representation ($d = 0..3$), and replaced it with the word having the highest cosine similarity in the English vocabulary described in the next section. The result was

"according to a ***researcher*** *at* ***abridged*** *university it does not matter in what* ***ordered*** *the letters in a word are the only important thing is that the first and last letters be at the right place the rest can be a total* ***messes*** *and you can still read it* ***outwith*** *problem this is because the human mind does not read every letter by itself but the word as a whole".*

Note that the word "cambridge" was not in the vocabulary. Although the task in this paper is not to recognize mixed up words, it shows the ability of our decoder to perform a reading task that we naturally do.

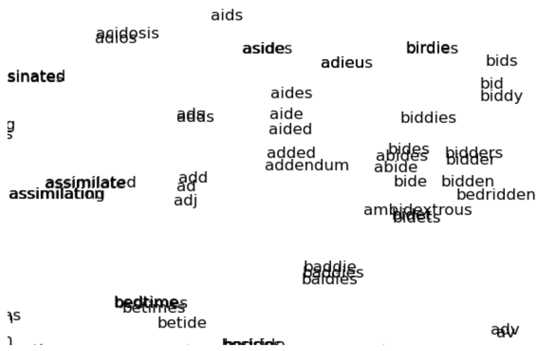


Figure 4: Visualisation of the bigram representation of the English vocabulary, for $d = 1..3$, with t-SNE [41].

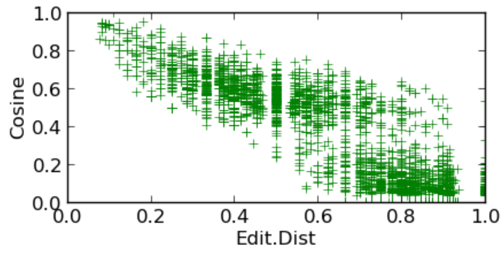
On Fig. 4, we show the English vocabulary in bigram space ($d = 1..3$), reduced to two dimensions with t-SNE [41]. We observe that words which are close in the bigram space also have a close orthographic form. On Fig. 5, we randomly selected pairs of words, and pairs of words with high cosine similarity in the French and English dictionnary, and plotted their cosine similarity in the bigram space against the edit distance between the two words, normalized by the length of the longest word. We note that words with high cosine similarity also have short edit distance, supporting the idea that the bigram representation encode some global letter order, and therefore might favor word recognition.

Data Preparation

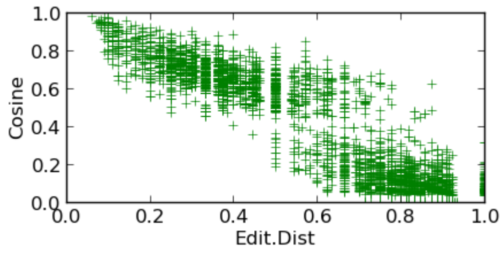
Databases

We carried out the experiments on two public handwritten word databases: Rimes [42] (French), and IAM [43] (English). We simplified the problem by limiting ourselves to words of at least two lowercase characters (a to z). This selection removed approximately 30% of the words. The Rimes database (from the ICDAR 2011 competition) do not have an official development set, so we split the training set and

[1] http://www.mrc-cbu.cam.ac.uk/people/matt.davis/cmabridge/



(a) English



(b) French

Figure 5: Relation between cosine similarity in bigram space and normalized edit distance, for the English (a) and French (b) vocabularies.

keep the last 10% of words for development. The number of words and bigrams of different orders in the different sets are reported on Table 1.

We applied deslanting [44], contrast enhancement, and padded the images with 10px of white pixels to account for empty context on the left and right of words. From the preprocessed images, we extracted sequences of feature vectors with a sliding window of width 3px. The features are geometrical and statistical features described in [45], which give state-of-the-art results in handwritten text line recognition [38].

Vocabularies

We downloaded word frequency lists for French and English from [46]. These lists were build from film subtitles [2] written by many contributors, and they contain many misspellings. We removed the misspelled words using GNU Aspell [47].

We selected 50,000 words for each language. They are the most frequent words (length ≥ 2) and made only of lowercase characters between a and z, making sure to also include all the words of the database. For example, the 50,000 most frequent French words fulfilling these condition miss about 200 words of the Rimes database, so we selected the most frequent 49,800 and added the missing 200.

Recognition of Open-Bigrams with Recurrent Neural Networks (RNNs)

To predict bigrams, we chose Bidirectional Long Short-Term Memory RNNs (BLSTM-RNNs) for their ability to consider the whole sequence of input vectors to make predictions. We trained one RNN for each order-d bigram, with the Connectionist Temporal Classification (CTC [48]) criterion. The CTC framework defines a sequence labeling problem, with an output sequence of labels, of smaller length than the input sequence of observations.

[2]http://opensubtitles.org

Table 1: Number of words/bigrams in different sets of Rimes and IAM databases. (in parenthese are the number of distinct tokens).

		Train	Valid.	Test
Rimes	Words	33,947	3,772	5,695
	Bigram $d = 0$	161,203 (26)	17,887 (25)	26,828 (25)
	Bigram $d = 1$	127,256 (312)	14,115 (240)	21,133 (258)
	Bigram $d = 2$	93,309 (425)	10,343 (336)	15,438 (352)
	Bigram $d = 3$	68,747 (436)	7,651 (327)	11,372 (368)
	Bigram $d = 1..3$	289,312 (517)	32,109 (420)	47,943 (452)
IAM	Words	38,584	5,977	18,394
	Bigram $d = 0$	177,101 (26)	26,133 (26)	82,320 (26)
	Bigram $d = 1$	138,517 (428)	20,156 (364)	63,926 (417)
	Bigram $d = 2$	99,933 (550)	14,179 (477)	45,532 (537)
	Bigram $d = 3$	68,961 (543)	9,361 (457)	30,537 (522)
	Bigram $d = 1..3$	307,411 (598)	43,696 (551)	139,995 (592)

We built the target sequences for training as sequences of bigrams, ordered according to the first letter of the bigram. For example, for $d = 2$, the target for `example` is `ea-xm-ap-ml-pe`. The CTC training criterion optimizes the Negative Log-Likelihood (NLL) of the correct label sequence. We set the learning rate to 0.001, and stopped the training when the NLL on the validation set did not decrease for 20 epochs. We kept the network yielding the best NLL on the validation set.

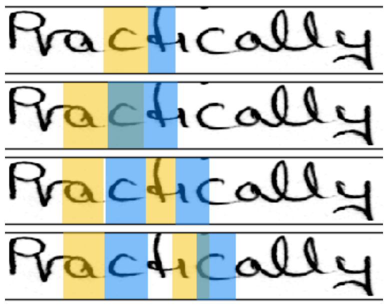


Figure 6: Hypothetical context needed in the input image to make two consecutive (yellow and blue) bigram predictions, for $d = 0$ (top, to predict `c`, then `t`) to 3 (bottom, to predict `ai`, then `cc`). As d increases, the contexts become more complex to model: they involve long range dependencies and are highly intertwined.

We trained one RNN for each order $d = 0$ to 3, including the special bigrams for word extremities or not. We will refer to each of these RNNs with rnn_d for order d ($rnn_{d'}$ when extremities are included).

The RNNs have seven hidden layers, alternating Long Short-Term Memory [49] recurrent layers in both direction and feed-forward layers. The first LSTM layers have 100 hidden LSTM units. The outputs of two consecitive timesteps of both directions are fed to a feedforward layer with 100 nodes, halving the size of the sequence. The next LSTM layers have 150 units, and are connected to a feed-forward layer with 150 units. The last two LSTM layers and feed-forward layer have 200 hidden units.

These RNNs are trained to predict sequences of open-bigrams of fixed order. Here, we are interested in a word representation as a bag of bigrams, which does not carry any information about the sequence in which the bigrams appear, the number of times each bigram appears, or the order of each individual bigram. That is, we are interested in a decoder which considers an unordered set of bigrams predictions for all bigram orders.

We **forget about the temporal aspect** of bigram predictions by taking the maximum value of a given bigram prediction by the RNN:

$$p_{d,b}(x) = \max_t rnn_d(x,t) \tag{12}$$

and we **forget about the bigram order** by taking the maximum output across different values of d:

$$p_b(x) = \max_d \max_t rnn_d(x,t) \tag{13}$$

It would have been more satisfying for this experiment to train an optical model to predict a set of bigrams for all orders. However, this work is focused on the decoder. Moreover, even the simpler task of predicting a sequence of bigrams of fixed order is challenging. On Fig. 6, we show the hypothetical context needed to make two consecutive predictions, for bigram order $d = 0..3$. RNNs are popular for handwriting recognition, and can consider a context size of variable length – but still local – to predict characters ($d = 0$).

For $d = 1$, the required context is still local (and would span two consecutive characters), but overlap, because each character is involved in two bigrams.

For $d > 1$, the context is even split into two areas (covering the involved characters) that might be far apart depending on d. Contexts for different predictions are entangled: the whole area between two characters forming a bigram is not relevant for this bigram (and might be of varying size), but will be important to predict other bigrams. It means that the RNN will have to remember a character observation for some time, until it sees the second character of the bigram, while ignoring the area in between for this bigram prediction, and remembering it since it will be useful in order to predict other bigrams.

The number of classes for bigrams is 26 times larger than the number of characters, making the classification problem harder, and the number of examples per class in training smaller.

Results

In this paper, we focused on a subset of Rimes and IAM word databases, which makes the comparison with published results difficult. Instead, we compared the bigram decoder approach to decoding with standard models, consisting of a beam search with Viterbi algorithm in the lexicon.

Baseline Systems based on HMMs and Viterbi Decoding

We built several models and used the same vocabulary as for the bigram decoder, and no language model (all words have the same prior probability). The first one is a Hidden Markov Model, with 5 (Rimes) or 6 (IAM) states per characters, and an emission model based on Gaussiam Mixture Models. This system is trained with the Maximum Likelihood criterion, following the usual Expectation-Maximization procedure. At each iteration, the number of Gaussians is increased, until no improvement is observed on the validation set. The forced alignments with the GMM/HMM system are used to build a labeled dataset for training a Multi-Layer Perceptron (MLP) with 4 hidden layers of 1,024 sigmoid units. We optimized the cross-entropy criterion to train the network to predict the HMM states from the concatenation of 11 input frames, with a learning rate of 0.008. The learning rate was halved when the relative improvement was smaller than 1% from one epoch to the next. The MLP is integrated in the hybrid NN/HMM scheme by dividing the predicted state posteriors $p(s|x)$ by the state priors $p(s)$, estimated from the forced alignments. It is then further trained with a sequence-discriminative criterion: state-level Minimum Bayes Risk [50] (sMBR) for 5 epochs. Finally, we also performed the sequential decoding with vocabulary constraints using rnn_0.

The input features for all systems are the same, described in "Data Preparation". The results of similar systems for handwritten text line recognition can for example be found in a previous work [38], where a comparison is made with state-of-the-art systems.

Table 2: Word Error Rates (%) with baseline systems and Viterbi decoding of character sequences.

		Valid.		Test	
	Model	Top1	Top10	Top1	Top10
Rimes	GMM/HMM	37.38	11.11	36.24	10.03
Viterbi	MLP/HMM	22.00	4.32	21.37	4.50
(Char. seq.)	MLP-sMBR/HMM	14.82	2.23	14.45	2.16
	rnn_0/HMM	10.79	3.74	10.03	3.18
IAM	GMM/HMM	27.64	7.81	37.96	15.34
Viterbi	MLP/HMM	16.90	4.02	26.26	9.81
(Char. seq.)	MLP-sMBR/HMM	11.73	2.63	19.97	6.55
	rnn_0/HMM	10.21	3.06	17.49	6.88

On Table 2, we report the results on the validation and test sets of Rimes and IAM. The reported numbers are percentages of word errors. *Top1* corresponds to the error when the correct word is not the recognized one, and *Top10* to the error when the correct word is not in the top ten alternatives (10-best list).

The best *Top1* word error rates are around 10% (17.5% on the test set of IAM), and the best *Top10* around 2-3% (6.5% for IAM-test), and consttitute the baseline performance to which the bigram approach is to be compared.

Evaluation of RNN Performance

We trained RNNs to predict sequences of bigrams of a given order. Their performance to accomplish this task can be measured, on the validation set, with the edit distance (edit.dist) between the recognized sequence and the true sequence, and with the percentage of sequences with at least one error: the Sequence Error Rate (SER).

The results are reported on Table 3. We observe that the performance decreases as d increase. Yet they remain in a reasonnable range compared to the simple case $d = 0$, and one should keep in mind that a small degradation should be expected from the larger number of classes and smaller number of training examples per class.

We have shown that RNNs can perform the apparently difficult task of recognizing sequences of open-bigrams. Thus, we may use the RNN bigram predictions in our decoder, rather than building bigram predictions from letter ones, which would have been less satisfying considering the supposed interest of the redundancy of the bigram representation that allows error corrections.

Measuring the Quality of Bigram Predictions

In the previous section, we presented RNNs to predict sequences of fixed order bigrams. However, we are interested in a decoder which considers an unordered set of bigrams predictions across bigram orders. We presented how to forget about the temporal aspect of bigram predictions and about the bigram order.

The edit distance and SER are not relevant measures for the proposed decoding. The decoder results are more affected by false alarms and missed bigrams in the final prediction vector. Since we keep a confidence value for all bigrams in the prediction vector, rather than using a binary vector (cf. Eq. 8), we modified the formulation of precision and recall.

Table 3: RNN results

	d	edit.dist(%)	SER(%)	Precision(%)	Recall(%)	F-measure
Rimes	0	8.3	22.7	95.00	93.42	0.942021
	1	11.4	20.8	89.86	87.61	0.8872
	1’	9.6	21.7	91.17	89.25	0.901998
	2	13.4	23.2	79.78	84.84	0.822315
	2’	11.1	26.2	82.84	85.84	0.843128
	3	14.8	23.7	74.80	83.37	0.788523
	3’	12.5	27.4	82.57	80.93	0.817402
	1,2,3	-	-	84.53	86.68	0.855922
	1’,2’,3’	-	-	84.03	88.48	0.86197
IAM	0	8.0	21.7	93.51	92.54	0.930205
	1	13.1	23.2	87.34	86.20	0.867681
	1’	10.4	23.3	89.28	88.48	0.888813
	2	16.7	26.2	77.71	82.33	0.799537
	2’	12.7	28.3	81.57	83.95	0.827408
	3	20.8	31.2	62.29	77.54	0.69081
	3’	14.6	31.8	76.17	78.56	0.773436
	1,2,3	-	-	80.53	84.26	0.823527
	1’,2’,3’	-	-	81.04	86.40	0.836315

A bigram $b \in \mathcal{B}(w)$ is correctly retrieved with confidance $p_b(x)$, and missed with confidence $(1 - p_b(x))$. Similarly, a bigram not in the representation $\mathcal{B}(w)$ of word w is falsely recognized with confidance $p_b(x)$, and correctly ignored with confidence $(1 - p_b(x))$. It gives us the following expressions for precision and recall

$$precision = \frac{\sum_{(x,w)} \sum_{b \in \mathcal{B}(w)} p_b(x)}{\sum_x \sum_{b' \in B} p_b(x)} \tag{14}$$

$$recall = \frac{\sum_{(x,w)} \sum_{b \in \mathcal{B}(w)} p_b(x)}{\sum_{w \in W} |\mathcal{B}(w)|} \tag{15}$$

which are the usual ones when $p_b(x) \in \{0, 1\}$.

The *F-measure* is calculated from precision and recall with the usual formula. The results for all RNNs, and for the combination of orders, are also reported on Table 3. We observe that the precision and recall results are correlated to the performance in terms of edit distance or sequence error rates. Namely, they decrease as the bigram order increases, which is not surprising, given that higher order bigrams are more difficult to recognize with these sequence models. We also see that including the special bigrams for word beginnings and endings generally improves the results. This is not surprising either: the RNNs are good at recognizing them.

Depsite this performance decrease, the precision remains above 70%, which limits the amount of noise that will be included in the bigram representation for recognition. Combining the recognition across orders, we obtain a precision of around 84% on Rimes and 80% on IAM. The recall tends to be higher than the precision, staying around or above 80% in all configurations. Across orders, the recall is above 88% on Rimes and 86% on IAM. The high recall will limit the amount of missing information in the bigram representation.

Overall, the F-measure for bigram recognition is above 80%, which is a good starting point, given that *(i)* the vocabulary used in decoding will add constraints and may help recovering from some mistakes in the bigram recognition, and *(ii)* the redundancy and order encoded in the bigram may limit the impact of misrecognitions.

Word Recognition using Bigram Predictions

On Table 4, we report the results of bigram decoding. For each word image in the validation and test sets, we computed the bigram predictions with the RNNs described above. We combined the different orders are explained previously, and either added the special bigrams for word boundaries and/or the single character predictions or not. We computed the cosine similarity to the bigram decomposition of all words in the vocabularies in the same representation space (*i.e.* same orders, and same choices for the inclusion of special bigrams and single characters) by computing the product of the vocabulary matrix V by the recognition vector. We sorted vocabulary words by descending similarities, and counted the number of times the correct word was not the most similar one, or in the top 10.

Table 4: Decoding results (% word error rates).

			Valid.		Test	
		Model	Top1	Top10	Top1	Top10
Rimes	Viterbi *(Char. seq.)*	Best in Table 2	10.79	2.23	10.03	2.16
	Cosine (bigrams)	$rnn_{1,2,3}$	25.58	7.32	24.37	6.50
		$rnn_{1',2',3'}$	12.43	5.22	12.27	4.76
		$rnn_{0,1,2,3}$	11.03	5.83	10.41	5.23
		$rnn_{0,1',2',3'}$	9.81	4.72	9.43	4.32
IAM	Viterbi *(Char. seq.)*	Best in Table 2	10.21	2.63	17.49	6.88
	Cosine (bigrams)	$rnn_{1,2,3}$	13.45	7.18	20.82	12.30
		$rnn_{1',2',3'}$	11.80	5.25	19.25	10.10
		$rnn_{0,1,2,3}$	11.98	6.17	19.61	11.03
		$rnn_{0,1',2',3'}$	11.09	4.92	18.39	9.62

We see that adding the sepcial bigrams for word boundaries improves the results, especially when single characters are not included in the representation. A possible explanation, besides the fact that they tend to be recognized more easily, could be that they provide a very useful information to disambiguate words having a similar bigram representation (*e.g.* `them` and `theme`). Adding single characters also improves the performance of the decoder, especially when the boundary bigrams are not included in the representation. The gain obtained with the single characters is about the same – sometimes a little better – as the gain with boundaries. It might be due to the much better recognition of the RNN for single characters (precision and recall over 90%), as well as the added redundancy and complementary information provided.

The best performance is achieved with both single characters and word boundaries, although the gain compared to adding only one of them is slight. The error rates are competitive or better than the best Top-1 error rates obtained by classical character sequence modeling and Viterbi decoding. The Top-10 results, however, are much worse, showing that the correct word was not close in the bigram representation to the bigram predictions of RNNs. We believe that an improvement of the RNNs and a more elaborate similarity measure than a mere cosine could lead to a drastic drop of the Top-10 error rate, and therefore to a much better Top-1 result.

Discussion

State-of-the-art systems, as well as most of the systems for handwritten word recognition found in the litterature, either try to model words as a whole, or as a sequence of characters. The latter currently gives the best results, is widely adopted by the community, and benefits from a lot of attention. In this paper, we have proposed a simple alternative model, inspired by the recent findings in cognitive neurosciences research on reading.

Acknowledgments

This work was supported by the French Research Agency under the contract Cognilego ANR 2010-CORD-013.

References

1. Dehaene S, Cohen L, Sigman M, Vinckier F (2005) The neural code for written words: a proposal. Trends in cognitive sciences 9: 335–341.

2. Whitney C, Bertrand D, Grainger J (2012) On coding the position of letters in words: a test of two models. Experimental psychology 59: 109.

3. Gomez P, Ratcliff R, Perea M (2008) The overlap model: a model of letter position coding. Psychological review 115: 577.

4. Grainger J, Van Heuven W (2003) Modeling letter position coding in printed word perception. The mental lexicon : 1–24.

5. Glotin H, Warnier P, Dandurand F, Dufau S, Lété B, et al. (2010) An adaptive resonance theory account of the implicit learning of orthographic word forms. Journal of Physiology-Paris 104: 19–26.

6. Dufau S (2008) Auto-organisation des représentations lexicales au cours de l'apprentissage de la lecture: approches comportementale électrophysiologique et neuro-computationnelle. Ph.D. thesis, Université de Provence.

7. Touzet C, Kermorvant C, Glotin H (2014) A Biologically Plausible SOM Representation of the Orthographic Form of 50000 French Words. In: Advances in Self-Organizing Maps and Learning Vector Quantization, Springer AISC 295. pp. 303–312.

8. Touzet C (2015) The Theory of neural Cognition applied to Robotics. International Journal of Advanced Robotic Systems, 12:74 .

9. Kozielski M, Doetsch P, Ney H, et al. (2013) Improvements in RWTH's System for Off-Line Handwriting Recognition. In: Document Analysis and Recognition (ICDAR), 2013 12th International Conference on. IEEE, pp. 935–939.

10. Pham V, Bluche T, Kermorvant C, Louradour J (2014) Dropout improves recurrent neural networks for handwriting recognition. In: 14th International Conference on Frontiers in Handwriting Recognition (ICFHR2014). pp. 285–290.

11. Parisse C (1996) Global word shape processing in off-line recognition of handwriting. IEEE transactions on pattern analysis and machine intelligence 18: 460–464.

12. Madhvanath S, Govindaraju V (1996) Holistic lexicon reduction for handwritten word recognition. In: Electronic Imaging: Science & Technology. International Society for Optics and Photonics, pp. 224–234.

13. Madhvanath S, Krpasundar V (1997) Pruning large lexicons using generalized word shape descriptors. In: Document Analysis and Recognition, 1997., Proceedings of the Fourth International Conference on. IEEE, volume 2, pp. 552–555.

14. Guillevic D, Suen CY (1998) Recognition of legal amounts on bank cheques. Pattern Analysis and Applications 1: 28–41.

15. Côté M, Lecolinet E, Cheriet M, Suen CY (1998) Automatic reading of cursive scripts using a reading model and perceptual concepts. International Journal on Document Analysis and Recognition 1: 3–17.

16. Edelman S, Flash T, Ullman S (1990) Reading cursive handwriting by alignment of letter prototypes. International Journal of Computer Vision 5: 303–331.

17. Chen MY, Kundu A, Srihari SN (1995) Variable duration hidden Markov model and morphological segmentation for handwritten word recognition. Image Processing, IEEE Transactions on 4: 1675–1688.

18. Baret O (1990) Régularités singulairtés de représentations et leur complémentarité: application à la reconnaissance de l'écriture manuscrite non contrainte. Ph.D. thesis.

19. Knerr S, Augustin E, Baret O, Price D (1998) Hidden Markov model based word recognition and its application to legal amount reading on French checks. Computer Vision and Image Understanding 70: 404–419.

20. Kim G, Govindaraju V (1997) A lexicon driven approach to handwritten word recognition for real-time applications. Pattern Analysis and Machine Intelligence, IEEE Transactions on 19: 366–379.

21. Morita M, El Yacoubi A, Sabourin R, Bortolozzi F, Suen CY (2001) Handwritten month word recognition on Brazilian bank cheques. In: Document Analysis and Recognition, 2001. Proceedings. Sixth International Conference on. IEEE, pp. 972–976.

22. El-Yacoubi A, Gilloux M, Sabourin R, Suen CY (1999) An HMM-based approach for off-line unconstrained handwritten word modeling and recognition. Pattern Analysis and Machine Intelligence, IEEE Transactions on 21: 752–760.

23. Tay YH, Lallican PM, Khalid M, Knerr S, Viard-Gaudin C (2001) An analytical handwritten word recognition system with word-level discriminant training. In: Document Analysis and Recognition, 2001. Proceedings. Sixth International Conference on. IEEE, pp. 726–730.

24. Favata JT, Srikantan G (1996) A multiple feature/resolution approach to handprinted digit and character recognition. International journal of imaging systems and technology 7: 304–311.

25. Bengio Y, LeCun Y, Nohl C, Burges C (1995) Lerec: A NN/HMM hybrid for on-line handwriting recognition. Neural Computation 7: 1289–1303.

26. Casey RG, Lecolinet E (1996) A survey of methods and strategies in character segmentation. Pattern Analysis and Machine Intelligence, IEEE Transactions on 18: 690–706.

27. Kaltenmeier A, Caesar T, Gloger JM, Mandler E (1993) Sophisticated topology of hidden Markov models for cursive script recognition. In: Document Analysis and Recognition, 1993., Proceedings of the Second International Conference on. IEEE, pp. 139–142.

28. Marti UV, Bunke H (2001) Using a statistical language model to improve the performance of an HMM-based cursive handwriting recognition system. International journal of Pattern Recognition and Artificial intelligence 15: 65–90.

29. Bourlard H, Morgan N (1994) Connectionist speech recognition: a hybrid approach Chapter 7, volume 247 of *The Kluwer international series in engineering and computer science: VLSI, computer architecture, and digital signal processing*. Kluwer Academic Publishers, 129–151 pp.

30. Plötz T, Fink GA (2009) Markov models for offline handwriting recognition: a survey. International Journal on Document Analysis and Recognition (IJDAR) 12: 269–298.

31. Chevalier S, Prêteux FJ, Geoffrois E, Lemaitre M (2005) A generic 2D approach of handwriting recognition. In: ICDAR. pp. 489–493.

32. Lemaıtre M, Grosicki E, Geoffrois E, Prêteux F (2008) Off-line handwritten word recognition based on a bidimensional Markovian approach with a large vocabulary .

33. Wang S, Uchida S, Liwicki M (2012) Part-based method on handwritten texts. In: Pattern Recognition (ICPR), 2012 21st International Conference on. IEEE, pp. 339–342.

34. LeCun Y, Bottou L, Bengio Y, Haffner P (1998) Gradient-based learning applied to document recognition. Proceedings of the IEEE 86: 2278–2324.

35. Graves A, Schmidhuber J (2009) Offline handwriting recognition with multidimensional recurrent neural networks. In: Advances in Neural Information Processing Systems. pp. 545–552.

36. Levin E, Pieraccini R (1992) Dynamic planar warping for optical character recognition. In: Acoustics, Speech, and Signal Processing, 1992. ICASSP-92., 1992 IEEE International Conference on. IEEE, volume 3, pp. 149–152.

37. Rabiner L, Juang BH (1986) An introduction to hidden Markov models. ASSP Magazine, IEEE 3: 4–16.

38. Bluche T, Ney H, Kermorvant C (2014) A Comparison of Sequence-Trained Deep Neural Networks and Recurrent Neural Networks Optical Modeling for Handwriting Recognition. In: International Conference on Statistical Language and Speech Processing. pp. 199–210.

39. Espana-Boquera S, Castro-Bleda MJ, Gorbe-Moya J, Zamora-Martinez F (2011) Improving offline handwritten text recognition with hybrid HMM/ANN models. Pattern Analysis and Machine Intelligence, IEEE Transactions on 33: 767–779.

40. Bluche T, Louradour J, Knibbe M, Moysset B, Benzeghiba MF, et al. (2014) The A2iA Arabic Handwritten Text Recognition System at the Open HaRT2013 Evaluation. In: 11th IAPR International Workshop on Document Analysis Systems (DAS). IEEE, pp. 161–165.

41. Van der Maaten L, Hinton G (2008) Visualizing data using t-SNE. Journal of Machine Learning Research 9: 85.

42. Augustin E, Carré M, Grosicki E, Brodin JM, Geoffrois E, et al. (2006) RIMES evaluation campaign for handwritten mail processing. In: Proceedings of the Workshop on Frontiers in Handwriting Recognition. 1.

43. Marti UV, Bunke H (2002) The IAM-database: an English sentence database for offline handwriting recognition. International Journal on Document Analysis and Recognition 5: 39–46.

44. Buse R, Liu ZQ, Caelli T (1997) A structural and relational approach to handwritten word recognition. IEEE Transactions on Systems, Man and Cybernetics 27: 847–61.

45. Bianne AL, Menasri F, Al-Hajj R, Mokbel C, Kermorvant C, et al. (2011) Dynamic and Contextual Information in HMM modeling for Handwriting Recognition. IEEE Trans on Pattern Analysis and Machine Intelligence 33: 2066 – 2080.

46. Lists FW. Hermit Dave. URL `http://invokeit.wordpress.com/frequency-word-lists/`.

47. Atkinson K. GNU Aspell. URL `http://aspell.net/`.

48. Graves A, Fernández S, Gomez F, Schmidhuber J (2006) Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In: International Conference on Machine learning. pp. 369–376.

49. Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural computation 9: 1735–1780.

50. Kingsbury B (2009) Lattice-based optimization of sequence classification criteria for neural-network acoustic modeling. In: IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2009). IEEE, pp. 3761–3764.

Figure Legends

Tables

Supporting Information Legends