Keyword Spotting in Online Handwritten Documents Containing Text and Non-Text using BLSTM Neural Networks

Emanuel Indermühle, Volkmar Frinken, Andreas Fischer and Horst Bunke
Institute of Computer Science and Applied Mathematics
University of Bern
CH-3012 Bern, Switzerland
Email: {eindermu, frinken, afischer, bunke}@iam.unibe.ch

***Abstract*—Spotting keywords in handwritten documents without transcription is a valuable method as it allows one to search, index, and classify such documents. In this paper we show that keyword spotting based on bidirectional Long Short-Term Memory (BLSTM) recurrent neural nets can successfully be applied on online handwritten documents with non-text content. It even works without preprocessing steps such as text vs. non-text distinction and text line extraction. We also propose a modification that can improve the precision with little effort.**

I. Introduction

Keyword spotting is known as the process to identify all positions of a given keyword in a document. In the ideal case handwritten documents would be transcribed completely and stored in digital form. However, there are several limitations in today's methods which result in inaccurate transcription and a high computational cost. In this situation, keyword spotting offer a valuable alternative as it keeps computational cost low while establishing the ability to search and index in handwritten documents [1]. An other scenario where keyword spotting can be useful is the classification of documents according to the presence of specific keywords.

In this paper we evaluate the spotting of keywords in online handwritten documents. This is achieved with bidirectional Long Short-Term Memory (BLSTM) recurrent neural nets.

Keyword spotting in handwritten text using BLSTM has been introduced in [2]. This publication, together with others, marks a shift in methodology from the well known template based approach for keyword spotting [1] to methods based on handwriting recognition. This shift eliminates the dependency on training templates for keywords to be spotted. Instead, a large amount of data can be used to train the system's models for individual characters, which will then be combined for the search of the keywords. BSLTM, originally applied for speech recognition [3], had a remarkable impact in the field of handwriting recognition recently [4]. It could improve recognition of online and offline handwriting, without the need for word segmentation, neither in the test nor in the training phase. The application of this system for keyword spotting as described in [2] demonstrates its flexibility.

Online handwritten text faces a growing significance due to the use of PDAs, Tablet PCs, and digital pens. The understanding of such documents is a highly valuable goal, e.g. for the scenario of a smart meeting room [5] where it is desired to search, browse, and organize handwritten notes taken with digital pens during a meeting. One important difference to the offline modality is the linear character of the data which binds elements related in temporal context more than the spacial arrangement does.

In the literature two works on keyword spotting for online handwriting can be found. In [6] a template approach is proposed. Using dynamic time warping, segmented words are compared to word templates. With this method, Jain et al. could achieve a precision of 92% at a recall rate of 90%. Zhang et al. [7] reported a keyword spotting method based on a character classifier. This method allows arbitrary keywords, but it has to deal with the character segmentation problem. Nonetheless, a precision of 94% for a recall of 87% have been achieved.

In these two papers like in most work on online handwriting, the focus lies on individual characters, words, or text lines. This makes sense, as with PDAs individual characters must be recognized and Tablet PCs allow handwritten input on a given line. With digital pens, by contrast, the analysis of the handwriting is done after the pen has been put in the cradle. This results in whole pages covered with handwritten text, drawings, structuring elements, etc. The analysis of such documents requires difficult preprocessing steps, such as text detection [8], [9] and line extraction [10], [11] which lead to an increase of the error rate. In this situation the method presented in this paper is a useful alternative, as it can be applied to online handwritten documents without the previously mentioned preprocessing steps. Keyword spotting in entire documents is also performed in [12]. The authors, however, use documents with text which is historical arabic handwriting, and hence offline. The template based methodology is adopted.

The remaining part of this paper is structured as follows. In Section II-A the preprocessing of the digital ink and feature extraction are described. Then in Section II-B and II-C keyword spotting with BLSTM is explained. We propose a useful modification of the system in Section III. The experiments and their results are shown in Section IV. Finally, we draw conclusions in Section V.

II. Keyword Spotting

The approach for keyword spotting applied in this paper is based on a fully-fledged text line recognizer in contrast with the template based methods mentioned in the introduction. The considered task is position retrieval. This means that, given a user query in terms of a keyword, all positions where the keyword in question occurs are returned. To achieve this goal, the documents have to be transformed into sequences of feature vectors. These vectors are then passed on to our keyword spotting system, which ranks all probable keyword occurrences according to their likelihood. Next we describe these steps in greater detail.

A. Preprocessing and Feature Extraction

The digital ink we are dealing with was generated by Anoto pens. In order to save disc space, this device compresses the ink by removing sample points holding redundant information. To get a uniform data stream, the points must be recovered again. Between two strokes, the pen does not touch the paper, which results in no recorded ink. Such gaps must be filled to have a consecutive sequence of sampling points. This step is done by interpolating a straight line.

Commonly used features for handwriting recognition of online documents, as described, for example, in [13], depend on text line segmentation. This type of features do not fit our requirements since no segmentation should be performed beforehand. What we actually need are features extracted in the original writing order. We propose seven features which satisfy this need. They are extracted from each sampling point i using the following four properties: the force f_i, the coordinates x_i and y_i, and the time stamp t_i. The list of the features is given as follows:

1) The pen force f_i, where 0 indicates no contact between pen and paper and 1 is the maximal force recorded. Anoto pens distinguish 256 different values.
2) Δx of the segment between point $i-1$ and $i+1$:
$$\Delta x = \frac{x_{i+1} - x_{i-1}}{d(i-1, i+1)} \tag{1}$$
where $d(i,j)$ is the Euclidean distance between sample point i and j.
3) Δy of the segment between point $i-1$ and $i+1$
$$\Delta y = \frac{y_{i+1} - y_{i-1}}{d(i-1, i+1)} \tag{2}$$
4) Change of angle at point i: $\Delta\phi = \phi_i - \phi_{i-1}$, where
$$\phi_i = \arccos(\Delta x_i) + \pi I_{\Delta y_i > 0} \tag{3}$$
and $I_{\Delta y_i > 0} \in \{0, 1\}$ is the indicator function which specifies if $\Delta y_i > 0$.
5) The Speed $\frac{d(i-1,i)}{(t_i - t_{i-1})r}$ is given by the Euclidean distance between point $i-1$ and i divided by time in terms of sampling intervals, where r is the sampling rate. This value is normalized to $[-1, 1]$ using the hyperbolic tangent.
6) Distance from the current sample point i to the nearest crossing point nx_i of the digital ink. As feature value $\frac{1}{d(i, nx_i)}$ is chosen and normalized to $[-1, 1]$ by the hyperbolic tangent.
7) Number of crossing points on the segment between points $i-1$ and i.

B. BLSTM Neural Networks

The considered keyword spotting system is based on a recently developed recurrent neural network, termed *bidirectional long-short term memory* (BLSTM) neural network [4]. Instead of simple nodes, the hidden layers are made up of so-called *long short-term memory* (LSTM) blocks. These memory blocks are specifically designed to address the *vanishing gradient problem*, which describes the exponential increase or decay of values as they cycle through recurrent network layers. This is done by nodes that control the information flow into and out of each memory block. The input layer contains one node for each of the seven features, while the hidden layer consists of the LSTM cells, and the output layer contains one node for each possible character plus a special ε node, to indicate "no character".

The network is *bidirectional*, which means that the sequence is fed into the network both ways, forward and backward. This is because the form of a handwritten character does not only depend upon the previous but also upon the following character. The *bidirectional* architecture is realized by two input and two hidden layers. One input and one hidden layer deal with the forward sequence, and the other input and hidden layer with the backward sequence. The output layer sums up the activation levels from both hidden layers at each position in the text. The output activations of the nodes in the output layer are then normalized to sum up to 1. Hence they can be treated as a vector indicating the probability for each letter to occur at a particular position. A path through this probability vector sequence therefore corresponds to a sequence of letters. The likelihood of these letters being written in the text can be computed by multiplying the corresponding values along the path. For more details about BLSTM networks we refer to [14] and [4].

C. Keyword Spotting using BLSTM Neural Networks

For keyword spotting, a modification of the sequence of probability vectors returned by the neural network is needed. A virtual output node, termed *any node*, is added with a static value of 1 for its activation. The special symbol *, corresponding to the *any node*, is then added to the beginning of the keyword to be spotted. In addition, between each letter of the keyword the ε symbol is inserted. The keyword *hello* will turn into $*\varepsilon\_\varepsilon h\varepsilon e\varepsilon l\varepsilon l\varepsilon o\varepsilon\_$ where _ indicates white space, which ensures that only complete words will be found.

To find the most probable occurrence of a given keyword in the sequence of output activations, the Connectionist Temporal Classification (CTC) Token Passing algorithm, a kind of dynamic programming, is used [14].

The CTC algorithm finds a mapping between the keyword and the output activations of the corresponding characters that maximizes the summed up log-likelihoods. This mapping preserves the sequential order of both sequences. A matrix of so called tokens $\vartheta(i,t)$ is initialized, where i is the index of the letter in the keyword and t is the time step of the output activation. The value of the tokens is defined by

$$\begin{aligned}\vartheta(0,t) &= 0\\ \vartheta(i,t) &= \infty \text{ if } t < i\\ \vartheta(i,t) &= \max_{j=\{0,1,2\}}(\vartheta(i-j,t-1)) + log(o(i,t))\end{aligned}$$

where $o(i,t)$ is the output activation of character i at time step t. Value $\vartheta(i-2,t-1)$ is only considered if the previous letter c_{i-1} is ε. At each time step the token for the last letter of the keyword (the white space) can therefore be interpreted as the log-likelihood that the keyword ends at this position.

The *any node* at the beginning of the keyword allows the CTC algorithm to wait with the mapping of the first character until it finds the optimal position in the stream. For more details on the keyword spotting algorithm we refer to [2]. An actual output activation of the BLSTM network can be seen in Figure 1.

The time complexity of the keyword spotting as described above is given by $O(nT)$ where n is the number of labels and T the length of the input sequence. This is considerably faster than a full translation for a text search which has a complexity of $O(m^2T)$ where m is the number of words in the dictionary.

At each time step the normalized log-likelihood of the last keyword letter, the start-, and the end position is recorded and will be referred to as *match*. The normalization is done by dividing the log-likelihood by the number of keyword letters. The matches are filtered by a threshold reducing their number to a manageable size. In addition, overlapping matches are eliminated taking the most likely one in a greedy fashion.

Based on a threshold T, the matches are divided into a set with log-likelihoods larger than T containing the positive matches, and a set containing the remaining, negative matches. In Figure 2 some matches and their normalized log-likelihoods can be seen.

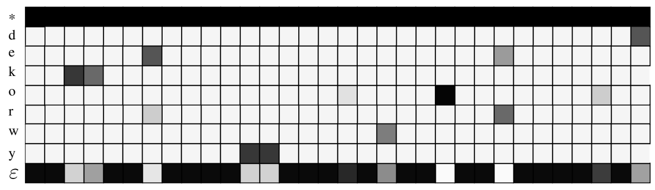


Figure 1: The activation level for some nodes in the output layer where input sequence constitutes the word *keyword*. The activation is close to 0 most of the time for normal letters and peaks only at distinct positions. In contrast, the activation level of the ε node is nearly constantly 1. The level for the *any node* * is 1 by definition.

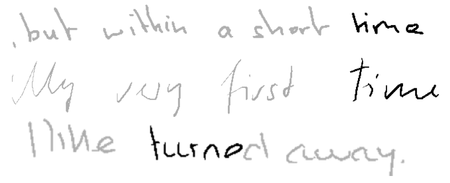


Figure 2: Examples of matches returned by the keyword spotting procedure when looking for the word *time*. Their log-likelihood are -0.38, -0.82, and -2.7. The first two matches are correct. The ink actually covered by the match is highlighted black.

III. Modification

In this section, we propose a method to improve the keyword spotting procedure as it was described in the section before.

Analyzing the false positive matches, the confusion between similarly written words could be identified as a major source of errors. To tackle this problem for a given keyword w, the set of all words W with a string edit distance lower than 3 are collected from a dictionary. In our case the dictionary contains all words of the data set. For each match the log-likelihood ll_v of all similar words $v \in W$ is evaluated.

The value used to separate positive and negative matches when compared with the threshold is given by

$$\alpha\, ll_w + (1-\alpha)(ll_w - \max_{v \in (W \cup \{w\})}(ll_v)) \tag{4}$$

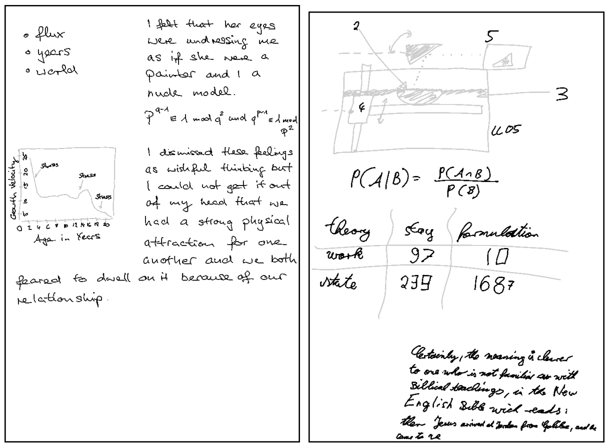
where for α a value between 0 and 1 is chosen according to results on the validation set.

IV. Experiments

A. Data

The proposed procedure for keyword spotting was evaluated on the IAMonDo-database[1] [15]. The dataset consists of

[1]The IAMonDo-Database is online available at http://www.iam.unibe.ch/fki/databases/iam-online-document-database



∘ flux
∘ years
∘ world

I felt that her eyes were undressing me as if she were a painter and I a nude model.

$p^{q-1} \equiv 1 \bmod q^2$ and $q^{p-1} \equiv 1 \bmod p^2$

I dismissed these feelings as wishful thinking but I could not get it out of my head that we had a strong physical attraction for one another and we both feared to dwell on it because of our relationship.

$P(A|B) = \frac{P(A \cap B)}{P(B)}$

theory	key	formulation
work	97	10
state	239	1687

Certainly, the meaning is clearer to one who is not familiar as with biblical teachings, in the New English Bible which reads: then Jesus arrived at Jordan from Galilee, and he came to re

Figure 3: Two sample documents from the dataset. Text ink is black, non-text ink is gray.

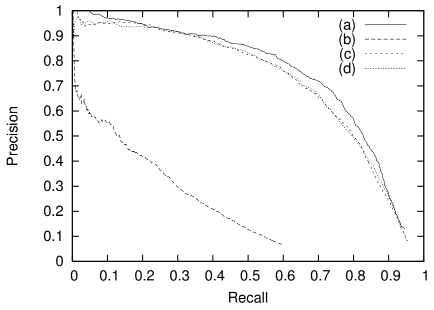


Figure 4: *Precision-recall* plot of the experiments: In (a) keyword spotting is applied on text lines only, in (c) on slices and in (b) and (d) on entire documents with mixed contents. In (a) and (b) nets are trained on text lines, while in (c) and (d) nets are trained on slices.

1,000 documents produced by 200 writers. The documents contain text in textblocks, lists, tables, and diagrams, as well as non-text in drawings and diagrams. About 72% of all strokes belong to text. Examples of these documents can be seen in Figure 3. Some of the documents are quite challenging regarding the proper extraction of text lines. The digital ink, which is the main part of the document, is stored in terms of groups of successive vectors consisting of X-, and Y coordinates, time, and pressure value. Detailed transcription and segmentation ground truth is available.

B. Setup

From the database 403 documents are used for training, 200 for validation and 203 for testing. The division of the data into these three subsets is given in [15].

The neural nets are trained in two different ways, ten for each way. The first set of neural nets is trained only on the text lines in the training set. For the second set the training documents are divided into *slices* containing exactly 40 text and non-text strokes each. A slice contains half of the strokes of the previous slice in the same document. If a slice contains a word only partially, strokes will be added to the end or removed from the beginning until the slice contains only complete words. The second set of nets are trained with these slices. To deal with non-text elements a special "non-text" label is introduced and assigned to the non-text strokes in the training set.

The text lines or slices in the validation set are used to stop the training iterations before over-fitting effects appear. More detail on the training of BLSTM neural network can be found in [4]. The 2,000 most frequent words from all documents are used as the keywords to be spotted. Stop words are ignored.

C. Evaluation

The ground truth of the documents (see [15]) is used to identify the correct *matches* among all matches that were returned by the keyword spotting system for the 2,000 words in the test documents. A match is defined *correct* if the text line which covers more than 50% of its width contains the keyword.

Knowing the correct matches, the number of *true positive* TP, *false positive* FP, *true negative* TN, and *false negative* FN ones can be identified given a threshold for the log-likelihood. With these values *precision* $= \frac{TP}{TP+FP}$ and *recall* $= \frac{TP}{TP+TN}$ are calculated. Varying the threshold, a *precision-recall* plot can be produced. A precision-recall curve gives us an idea about the noise in the returned results, given the percentage of how many true elements are found.

The *area under the curve* (AUC) of the *precision-recall* plot is a measure for the quality of the keyword spotting procedure. In this section we compare the results based on the AUC of the best network as a percentage with respect to 1.

D. Results

In the first experiment we show the capability of our system to spot keywords in text-only documents. Using the nets trained on text lines, keyword spotting is applied only to the text lines of the test documents. In this experiment an AUC of 74.2% is achieved.

When spotting keywords in entire documents the system must be able to ignore non-text content. The results achieved by the different sets of networks are quite diverse. With an AUC value of only 19.6%, the nets trained on text lines only seem not to be able to cope with the data pattern representing non-text. The nets trained on the *slices* are able to deal much better with this problem almost keeping the precision of keyword spotting on text-only data which is demonstrated by an AUC value of 71.3%. Figure 4 displays the correspondent *precision-recall* curves.

When applying the modification which takes the log-likelihood of similar words into account, the AUC can be boosted from 74.2% up to 78.3% on the text line system

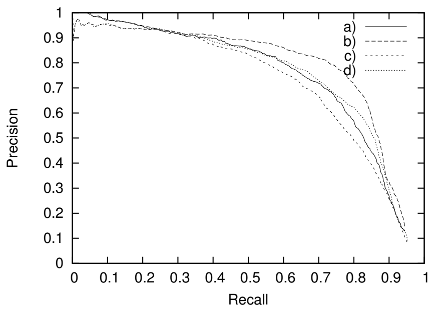


Figure 5: *Precision-recall* plot of the modification: In (a) and (b) the nets are trained on text lines and spotting is applied on text lines only as well. In (c) and (d) nets are trained on slices and spotting is applied on entire documents with mixed content. Curve (a) and (c) show keyword spotting in unmodified manner, while in (b) and (d) the modification mentioned in Section III is applied.

and from 71.3% to 74.7% on entire documents. An α value of 0.25 has proven to be best. Especially in the region of 80% recall the precision can be raised by more than 10%. See Figure 5 for the corresponding *precision-recall* plots.

V. Conclusion

In this paper, we applied word spotting driven by BLSTM nets to online handwritten documents with text and non-text content. We were able to demonstrate that almost no precision was lost compared to the application on perfectly segmented text lines containing only text. This allows one to index and search in raw documents as they are obtained from a digital pen. Moreover, we proposed a modification to the keyword spotting procedure which could further raise the precision.

In the future more enhanced modifications to further improve the selection of correct matches will be evaluated. Another line of research will be the spotting of non-text elements, such as arrows, boxes, and tables in online handwritten documents.

Acknowledgement

We thank Alex Graves for kindly providing us with the BLSTM Neural Network source code.

References

[1] R. Manmatha, C. Han, and E. M. Riseman, “Word spotting: A new approach to indexing handwriting,” in *Proc. of Conf. on Computer Vision and Pattern Recognition*. Washington, DC, USA: IEEE Computer Society, 1996, pp. 631–637.

[2] V. Frinken, A. Fischer, and H. Bunke, “A novel word spotting algorithm using bidirectional long short-term memory neural networks,” in *4th Workshop on Artificial Neural Networks in Pattern Recognition*, F. Schwenker and N. E. Gayar, Eds., vol. 5998. Springer, 2010, pp. 185–196.

[3] A. Graves and J. Schmidhuber, “Framewise phoneme classification with bidirectional lstm and other neural network architectures,” *Neural Networks*, vol. 18, no. 5-6, pp. 602 – 610, 2005.

[4] A. Graves, M. Liwicki, S. Fernandez, R. Bertolami, H. Bunke, and J. Schmidhuber, “A novel connectionist system for unconstrained handwriting recognition,” *IEEE Trans. PAMI*, vol. 31, no. 5, pp. 855–869, 2009.

[5] D. Moore, “The IDIAP smart meeting room,” IDIAP-Com, Tech. Rep., 2002.

[6] A. K. Jain and A. M. Namboodiri, “Indexing and retrieval of on-line handwritten documents,” in *Proc. 7th Int. Conf. on Document Analysis and Recognition*, 2003, pp. 655–659.

[7] H. Zhang, D.-H. Wang, and C.-L. Liu, “Keyword spotting from online chinese handwritten documents using one-vs-all trained character classifier,” in *Proc. 12th Int. Conf. on Frontiers in Handwriting Recognition*. Los Alamitos, CA, USA: IEEE Computer Society, 2010, pp. 271–276.

[8] A. K. Jain, A. M. Namboodiri, and J. Subrahmonia, “Structure in on-line documents,” in *Proc. 6th Int. Conf. on Document Analysis and Recognition*, 2001, pp. 844–848.

[9] E. Indermühle, H. Bunke, F. Shafait, and T. Breuel, “Text vs. non-text distinction in online handwritten documents,” in *Proc. of the 25th Annual ACM Symposium on Applied Computing*, vol. 1, 2010, pp. 3–7.

[10] E. H. Ratzlaff, “Inter-line distance estimation and text line extraction for unconstrained online handwriting,” in *Proc. 7th Int. Workshop on Frontiers in Handwriting Recognition*, 2000.

[11] Y. Li, Y. Zheng, D. Doermann, and S. Jaeger, “A new algorithm for detecting text line in handwritten documents,” in *Proc. 10th Int. Workshop on Frontiers in Handwriting Recognition*, Guy Lorette, Ed., Université de Rennes 1. La Baule (France): Suvisoft, 10 2006.

[12] R. F. Moghaddam and M. Cheriet, “Application of multi-level classifiers and clustering for automatic word spotting in historical document images,” in *Proc. 10th Int. Conf. on Document Analysis and Recognition*, 2009, pp. 511–515. [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/ICDAR.2009.104

[13] M. Liwicki and H. Bunke, “HMM-based on-line recognition of handwritten whiteboard notes,” in *Proc. 10th Int. Workshop on Frontiers in Handwriting Recognition*, 2006, pp. 595–599.

[14] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, “Connectionist temporal classification: Labelling unsegmented sequential data with recurrent neural networks,” in *23rd Int. Conf. on Machine Learning*, 2006, pp. 369–376.

[15] E. Indermühle, M. Liwicki, and H. Bunke, “IAMonDo-database: an online handwritten document database with non-uniform contents,” in *Proc. 9th Int. Workshop on Document Analysis Systems*, 2010, pp. 97–104.