Handwritten Text Recognition using Deep Learning and Word Beam Search

Kavitha Ananth [1], Kirubanand V B[2]
[1]Research Scholar ,Department of Computer Science CHRIST (Deemed to be University)
[2]Associate Professor , Department of Computer Science CHRIST (Deemed to be University), Banglore

Article History: Received: 11 January 2021; Accepted: 27 February 2021; Published online: 5 April 2021

Abstract— This paper offers a solution to traditional handwriting recognition techniques using concepts of Deep learning and Word Beam Search. This paper explains about how an individual handwritten word is classified from the handwritten text by translating into a digital form. The digital form when trained with the Connectionist Temporal Classification (CTC) loss function, the output produced is a RNN. This is a matrix containing character probabilities for each time-step. The final text is mapped using a CTC decoding algorithm by converting the character probabilities. The recognized text is constructed by a list of words from the dictionary by using the token passing algorithm. It is found the running time of token passing depends on the size of dictionary. Also the numbers like arbitrary character strings will not able to decode. In this paper the decoding search algorithm word beam search is proposed, in order to tackle these types of problems. This methodology support to constrain words similar to those contained in a dictionary. It allows the character strings such as arbitrary non-word between the words, and integrates into a word-level language model. It is found the running time is better when compared with the token passing. The proposed algorithm comprises of the decoding algorithm named vanilla beam search and token passing using the IAM dataset and Bentham data set.

Keywords—HTR(Handwritten Text Recognition),NN(Neural Network),CNN(Convolutional Neural Network), RNN(Recurrent Neural Network), CTC(Connectionist Temporal Classification), LM (Language Model), FSM(Finite State Machine)

1. Introduction

The Handwritten Text Recognition (HTR) is the process of converting handwritten text into digital text. However, Offline recognition is accomplished, post capturing the text. In offline recognition the text is captured as an image and the images are processed further. Challenges recognoized in HTR include the cursive nature of handwriting, and the variety of each character in size and shape and large vocabularies[1]. Improving the text recognition results the following in the extraction: spelling mistakes will be avoided, punctuation marks and arbitrary numbers will be allowed and allows to use a word-level language model. The labelings outputs are trained in a specific coding scheme using Neural Network. Decoding algorithms are used to calculate the final labelling. A prefix tree is used in WBS which is created by constraining the words in the respective recognized text. Four different methods are proposed to result the beams by a word-level LM : (1) beams are constrained only by the dictionary, (2) the output word is completely recognized by identify a score, (3) forecasting the score by evaluating next words (Ortmanns et. al [2] uses this idea in the circumstances of hidden Markov models) and (4) factors the possible words with the next set of words using the random sample. Each of the arbitrary non-word strings such as characters are identified as an operating state.

The algorithm explained in this paper is able to defeat the best path decoding, Vanilla Beam Search and passing token on the IAM [3] and Bentham [4] datasets. Additionally, the token passing is defeated by the running time.

The remaining paper is explained as follows: in Section II, a short introduction about CTC loss, VBS decoding and token passing is discussed. Later, in Section III discussed about the proposed model prefix trees and LMs. Section IV explains in detail about the proposed algorithm. Section V provides the detailed evaluation and compares the scoring modes results with other decoding algorithm. Finally, the paper is summarized by the conclusion.

2. State of The Art

First, the Connectionist Temporal Classification CTC operation is discussed. Later part, the two state-of-the-art decoding algorithms, namely VBS and token passing, are presented.

A. \Connectionist Temporal Classification

A Recurrent Neural Network (RNN) produces a progression of length T with C + 1 character probabilities per progression element, where C denotes the number of characters [5]. In this paper for the RNN output a pseudo-character which is called as blank is added additionally and it is denoted by "-". From the RNN output one character per time-step is chosen and they are concatenated together inorder to form a path π [5]. The product of all character probabilities on this path is defined as the probability of a path. One or more adjacent

occurences of the characters in the path are used for encoding the single characters from a labelling, by following the sequence of blanks [5]. A way to model this encoding is by using a Finite State Machine (FSM) [5]. The FSM is created as follows:first the labeling is elongated by inserting blanks and next for each character and blank a state is created. For each state a self loop is added and all the consecutive states are connected by a transition. For consecutive different characters a direct transition is included by skimming the blank. Fig 1 depicts a FSM which outputs for the labeling "ab" valid paths by transactions from a initial state to the final state (end) on an arbitrary path. e.g. " - a b - " or " a - b -" or "a b" among the others.

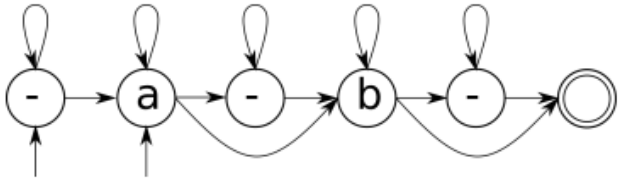


Fig. 1. FSM depicts valid paths (encodings) for the labeling "ab". The two left-most states are assumed as initial states while the right-most state is recognized as the final state.

B. Vanilla Beam Search

The VBS decoding algorithm is described in the paper of Hwang and Sung [2]. An illustration is shown in Fig 2. The output of RNN is a matrix of size T ×(C + 1) and this is passed into the decoding algorithm. The multiple candidates for the final labeling called beams are iteratively calculated. The each beam-labeling is extended by all possible characters during each time interval. In addition, the original beam is also copied to the next time-step.

Always only the best beams are kept at each time-step in order to avoid exponential growth of the tree: the Beam Width (BW) governs the number of beams to keep. If there are two beam-labeling are equal at a given time-step, then they are merged by summing up the probabilities first and then removing one of them.

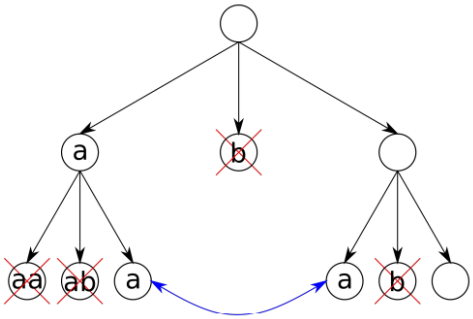


Fig. 2. Iteratively extended beam-labeling (from top to bottom) forms this tree. The two best-scoring beams are retained per duration time interval (i.e. BW = 2), remaining all are removed (red). All the equal labeling are merged (blue).

In order to score the extension of a beam-labeling by a character, a character-level LM can optionally be used. As soon as an Out-Of-Vocabulary (OOV) word occurs and identified the beam should be removed so that the constrained decoding algorithm is made possible [6]. Alternatively, there is a probability to have an OOV word in each beam which in turn restrict the procedure of using the constrained decoding methodology.

C. Token Passing

This algorithm is proposed by Graves et al. [7] , however, the subsequent discussion is based on one another publication from Graves [8]. A succession of dictionary words are exhibited as algorithm output when a RNN, a dictionary and a LM are given. Inheritance to the previous explained coding scheme CTC, a word model is created for every word, which in turn like a state machine which connects sequence of characters. In general a sequence of word is modeled by parallel inclusion of the multiple word-models, and all end-states are connected with all the begin-states. The tokens are used to implement the information flow, which are mutually transmitted from one state to another state. Each token detain the score and the history of already visited words. The chain of dictionary words are most likely searched by the algorithm and line up with the RNN output. Further it makes use of the word-transitions score with a word-level LM. The three different word-models which are displayed in parallel elucidate the time-complexity of this algorithm as O (T · W 2), where T represents the sequence length and W represents the dictionary size [8].

3. Proposed Model

The proposed WBS decoding algorithm model uses a prefix tree in order to query the characters. This can further extend the current beam-labeling. Additionally, words which have the current beam-labeling as prefix can also be queried. The beams on word-level is scored using a LM.

A. Prefix Tree

Trie, also called digital tree or prefix tree, is a kind of search tree is a basic tool in the domain of string processing [9]. Fig 3 shows a prefix tree containing 5 words. It is a tree-data structure and therefore consists of edges and nodes. Each edge is labelled by a character and points to the next node. A node has[9] a word flag which indicates if this node represents a word. A word is encoded in the tree by a path initiating from the root node and following the edges labelled with the corresponding characters of the word. By following a given prefix by querying characters is easy : First the node corresponding to the prefix is identified and then the outgoing edge labels are identified which determine the characters which follows the prefix. Alternatively it is possible to identify all the words which contain a given prefix: starting from the corresponding node.

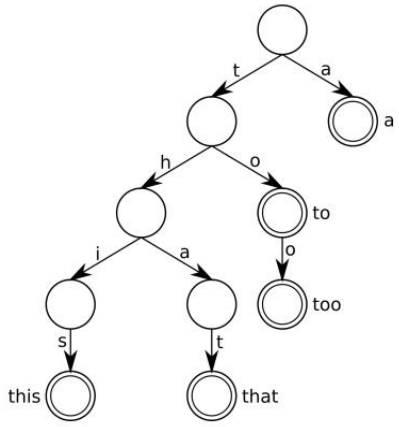


Fig. 3. Depicts the prefix tree containing words “a”, “to”, “too”, “this” and “that”.

Each time required to reach the prefix’s node in the prefix tree by following the labeled edges with the prefix’s characters, and then collecting the labels of the outgoing edges of this node. If the prefix “th” is given, then the possible next characters are “i” and “a”. From the prefix’s node, all descendant nodes are collected and represented as a word. If the prefix “th” is given, then the achievable words are “this” and “that”.

Aoe et al. [9] show an efficient implementation of this data structure. Identifying the node for a prefix with length L required O(L) time in their implementation. The time to find all words containing in a given prefix calculates on the quantity of nodes of the tree. An upper bound for the quantity of nodes is the quantity of words W times the utmost characters M of a word, therefore the time all words is O (W – M).

B. Language Model (LM)

A LM is able to predict upcoming words given previous words and it is capable to assign probabilities to given sequence of words [10] . It can be queried to give the probability P(w\h) that a word sequence (history) h is followed by the word w. This type of model is trained from a text by counting how often w follows h. The probability of a sequence is then $P(h) = P(w1) \cdot P(w2|w1) \cdot P(w3|w1, w2) \cdot ... \cdot P(wn|w1, w2, ..., wn-1)$ [10].

C. Proposed Algorithm: Word Beam Search.

The WBS decoding algorithm is a modification of VBS decoding and possess the following properties:

- Words are constrained to dictionary words
- Allows arbitrary number of non-word characters between words (such as numbers, punctuation marks)
- Optionally a word-level bigram LM can be integrated.
- Efficient running time when compared with the token passing

In order to constrain words to dictionary words and allowing arbitrary non-word characters between words, the concept of beam is used. In general each beam will be either in one of the two states. The beam is considered to be in the word state if a beam gets extended by a word character (typically “a”, “b”, …), otherwise it is in the non-word-state.

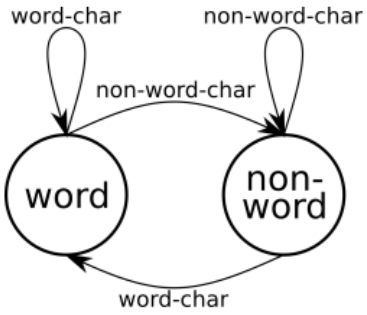


Fig. 4. Explains a beam can be either in one of the two states.

In general a beam is extended by a set of characters which depends on the beam-state. If the beam is in the non-word-state, the beam-labelling can be extended by all possible non-word-characters (typically " ", ".", ...). Additionally, it can be extended by a character which could occur as the first character of any word. All these additional characters are retrieved from the edges which leave the root node of the prefix tree. If the beam is in word-state, The prefix tree can be queried for a list of possible next characters if the beam is found to be in word-state. Fig 5 shows an example of a beam today in the word-state.

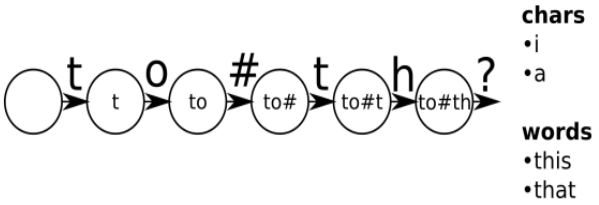


Fig. 5. A beam today in the word-state. The "#" character represents a non-word character. The prefix "th" is extended by "i" and "a" and this can be reach out to form the words "this" and "that". The prefix tree from Fig 3 is assumed in this example.

The corresponding node in the prefix tree for the last word-characters form the prefix "th" is found by succeeding the edges "t" and "h". The next characters, which are "i" and "a" in this example is determined by the outgoing edges of this node. Furthermore, the consequent characters also consist of all non-word-characters if the current prefix represents a complete word (e.g. "to"). Optionally, a word-level LM can be integrated. In the following text a bigram LM is assumed. In general if a beam contains more words, then more often it gets scored by the LM. The score gets normalized by the number of words in account of this. Four possible LM scoring-modes exist and names are assigned to them which are used throughout this paper: Words – it denotes only a dictionary but no LM is used. Algorithm 1 shows the pseudo-code for WBS decoding. The beam of the current time-step is considered as set B, and the probabilities for the beams are considered as P. The paths of a beam end with a blank are considered as probability P_b, and the paths of a beam end with non- blank is considered as probability P_{nb}. For LM the probability assigned P_{txt}. In general the P_{tot} is an abbreviation for $P_b + P_{nb}$. The algorithm iterates from t = 1 through t = T and creates a tree of beam-labelling as shown in Fig 5. Always an empty beam is denoted by ∅ and the last character of a beam is indexed by −1. All the best beams are obtained by sorting them with respect to P_{tot} . P_{txt} and required to keep the BW as best ones. The probability of seeing the beam-labelling at the current time-step is calculated for each of the beams. For paths ending with a blank and paths ending with a non-blank separate book keeping accounts are maintained for the CTC coding scheme. Depending on the beam-state, each beam gets extended by a set of possible next characters. Depending on the scoring-mode, the LM calculates a score P_{txt} when the beam is extended. The LM score is achieved by taking P_{txt} to the power of 1/num Words (b), where num Words (b) is the count of words contained in the beam b. Once the algorithm completes the iteration through time, the beam-labelling also get completed. The prefix tree is queried to provide a list of possible words if a prefix is not representing a complete word. There are two different ways to implement: by extending the beam-labelling by the most likely word (according to the LM), or if the list of possible words contains exactly one entry then the beam-labelling is completed.

4. Implementation

D. Handwritten Text Recognition

Handwritten Text Recognition (HTR) system comprise of the handwritten text in the form of scanned images as shown in Fig 6. The text recognition is done using neural network (NN) and trained with the connectionist

temporal classification (CTC) loss function. An image containing text is passed into the NN and the digitized text output is received. In general the NN simply generates the output characters as viewed in the image

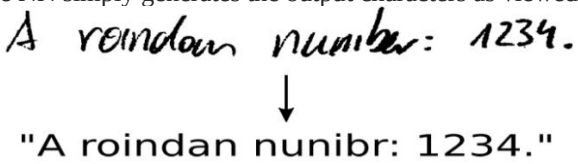


Fig. 6. Represents the Image of word taken from IAM Dataset

The NN model is completely an optical model and this might introduce errors due to identical characters similar to "o" and "a". Hence the Beam search algorithm is used which allows arbitrary strings of characters, which are needed to decode numbers and punctuation marks. The token passing limits its output to dictionary words, which avoids spelling mistakes.

Connectionist Temporal Classification (CTC) decoding algorithm is a Word beam search decoding. This is exclusively used for sequence recognition tasks like Handwritten Text Recognition (HTR) or Automatic Speech Recognition (ASR). Fig.7 illustrates how a HTR system works with its Convolutional Neural Network (CNN) layers, Recurrent Neural Network (RNN) layers and the final CTC (loss and decoding) layer. Right after the RNN layers the decoding algorithm word beam search is placed in order to translate the output, refer the red doted rectangle in the Fig. 7.

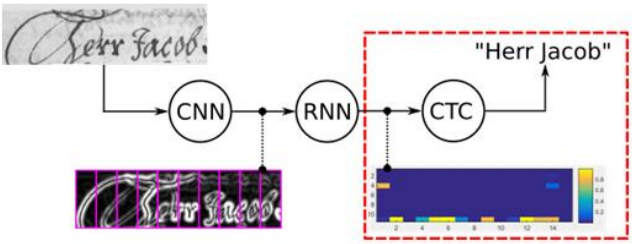
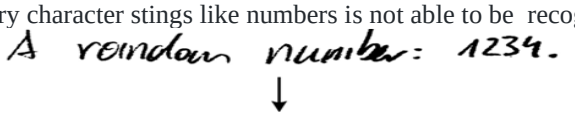


Fig. 7. HTR System with CNN-RNN-CTC

1) Hyperparameters

There are three hyper parameters we have to set: beam width, scoring mode and LM smoothing. The number of text candidates kept at each iteration step of the decoding algorithm is specified by the **beam width**. The trade-off between accuracy and performance is established by setting the beam width. For vanilla beam search (VBS) and word beam search (WBS) the character error rate (the lower, the better) depending on the beam width is plotted for the IAM dataset. In general a small beam width is sufficient for vanilla beam search, whereas a larger values for word beam search. The reason for this behavior is represented in the table containing the five top-scoring beams for both algorithms. In regard of WBS, always there is a uncertainty of the texts: the majority of the texts differ in the punctuation, and hence more number of beams are required in order to restore from the prediction of word wrongly. As a pragmatism, the validation set can be tuned with any value between 16 and 50.

The following sample Fig. 8 shows a typical use-case of the algorithm word beam search along with the results given by five different decoders. Since the noisy output of the optical model is used by the two decoders, Best path decoding and vanilla beam search the output generated is always the wrong words. The vanilla beam search can be extended by a character-level LM which enhance the result by allowing only likely character sequences. All the words generated as output is found to be right as the token passing uses word-level LM and a dictionary. However, the arbitrary character stings like numbers is not able to be recognized.



Best Path Decoding	"A roindan nunibr: 1234." ☒
Vanilla Beam Search	"A roindan numbr: 1234." ☒
Vanilla Beam Search LM	"A randan numbr: 1234." ☒
Token Passing	"A random number" ☒
Word beam Search	"A random number: 1234." ☑

Fig. 8. Result of various algorithms is given

It is found that only the word beam search algorithm is able to identify the words very accurately using a dictionary, additionally also the non-word characters are identified correctly.

The below illustration provides an analysis of the algorithm's input and the output by feeding the RNN output into the algorithm. The textual input enables or allows the word beam search decoding algorithm to create a LM and dictionary. The different settings helps to manage the LM scores from the beams (text candidates). Also controls the number of beams used per time-step. The output of the algorithm is the decoded text. We only keep the best-scoring beams per iteration. The score depends on the NN output. Alternatively with the help of different scoring modes we can also incorporate a word-LM (in the results section the abbreviations for these modes are used):

The LM should not be used at all, it should be constrained by using only the words **(W)** by a dictionary. So far this has been discussed. **Every time a word is fully recognized (N), a LM includes a score**. For example if there is a beam "My boo". As soon as a character "k" is added and the last word "book" is finished, then the LM scores seeing "My" and "book" as it is consequent to each other. **Looking into the future (N+F) :** the next consecutive occurrence of word can be recognized with the use of the prefix tree. A beam "I s" might be extended to "I see", "I saw", … The result can be achieved by totaling the LM-scores of all possible word instead of waiting until a word is finished in order to apply the LM-score to a beam. This would result in "I" and "see", "I" and "saw",… in the mentioned example. By using only the random subset (N+F+S) of the words the performance improvement can be achieved. Thus we arrived an algorithm which is capable to identify the text "A random number: 1234" correctly.

5. Result Analysis

The algorithm was able to correctly recognize the sample from Fig. 6. The goal is to compare how the algorithm performs on a complete dataset. The two HTR datasets Bentham and IAM are used for evaluation. The goal is to compare the evaluated result by comparing the performance of different decoding algorithms for a given neural network output. It is not only about achieving or outperforming state-of-the-art results on the mentioned datasets.

	Perfect LM	**Rudimentary LM**
Best Path	5.60 / 17.06	5.60 / 17.06
Token Passing	8.16 / 9.24	(not feasible)
VBS	5.35 /16.02	5.55 /16.39
WBS W	4.22 / 7.90	5.47 / 14.09
WBS N	4.07 / 7.08	6.15 /13.90
WBS N+F	4.05 / 7.36	6.76 /18.00
WBS N+F+S	4.06 /7.39	6.75 / 18.06

Table 1: For different algorithm results are given as CER [%] / WER [%] [W: no LM used, N: LM used, N+F: LM with forecasting, N+F+S: LM with forecasting and sampling] and training texts for the LM (perfect and rudimentary LM). CER/WER less value means we have a well performing algorithm.

The following error measures are chosen: Character Error Rate (CER) and Word Error Rate (WER) [12]. The Bentham HTR dataset is used. Table1 depicts the results.

The Bentham HTR dataset contains handwritten text from the time around the year 1800. The same is fed in NN output into all decoding algorithms. Thus a fair comparison is made possible. Two different training texts are used for the LM:

1. All the text from the test-set consisting of all words which is expected to be recognized are used. This is assumed as an ideal training text.
2. A basal training text is created from the text of the training-set concatenated with a word list consisting of 370000 words.

The error measures are referred by the character error rate (CER) and word error rate (WER). The edit distance between the ground truth text and the identified text is defined as the CER. In general it is normalized by the ground truth length. Word Error Rate (WER) is also defined in the similar way, but this is defined by the word-level.

6. Conclusion and future scope

A decoding algorithm for CTC-trained neural networks was proposed in this paper. This algorithm restricts words only to dictionary words, alternatively allows the arbitrary strings of characters between the words. Also this can optionally integrate a word-level LM and it is found to be faster than token passing. This comes with four different scoring-modes which control the effect of the LM and can also be used to govern the running time.

The algorithm proposed is evaluated by extensive use of the two HTR datasets Bentham and IAM. It is recognized Word Beam Search as one of the best algorithm to extract the words from the defined data set. The following score beam mode such as modes N, N+F or N+F+S can be used if there is a better training text for the LM. On the other hand, constraining the words is best instead of using the LM in order to use the beams to score (mode W). In general, this algorithm provides a huge efficiency when the abundant words to be recognized are known in advance. In case only a dictionary but no word-level LM is available, the Words mode is well suited which constrains the words of the beam-labeling. This provides a better effective and efficient solution as compared with other algorithm. This explains beam search algorithm can be a good choice for handwriting recognition.

7. Acknowledgement

This paper was completed successfully under the guidance of Dr. Kirubanand V.B., Christ (Deemed to be University), as a part of Computer Science Department.

References

1. H. Scheidl, S. Fiel Wien, and H. Scheidl Robert Sablatnig, "Handwritten Text Recognition in Historical Documents DIPLOMARBEIT zur Erlangung des akademischen Grades Visual Computing eingereicht von," pp. 90–96, 2011, [Online]. Available: http://www.ub.tuwien.ac.athttp//www.ub.tuwien.ac.at/eng.
2. S. Ortmanns, A. Eiden, H. Ney, and N. Coenen, "Look-ahead techniques for fast beam search," *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, vol. 3, pp. 1783–1786, 1997, doi: 10.1109/icassp.1997.598876.
3. U. V. Marti and H. Bunke, "A full English sentence database for off-line handwriting recognition," *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR*, pp. 709–712, 1999, doi: 10.1109/ICDAR.1999.791885.
4. J. A. Sanchez, A. H. Toselli, V. Romero, and E. Vidal, "ICDAR 2015 competition HTRtS: Handwritten Text Recognitio0n on the tranScriptorium dataset," in *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR*, 2015, vol. 2015-November, doi: 10.1109/ICDAR.2015.7333944.
5. A. Graves, M. Liwicki, S. Fernández, R. Bertolami, H. Bunke, and J. Schmidhuber, "A novel connectionist system for unconstrained handwriting recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 5, 2009, doi: 10.1109/TPAMI.2008.137.
6. A. Graves and N. Jaitly, "Towards end-to-end speech recognition with recurrent neural networks," in *31st International Conference on Machine Learning, ICML 2014*, 2014, vol. 5.
7. H. Scheidl, S. Fiel, and R. Sablatnig, "Word beam search: A connectionist temporal classification decoding algorithm," *Proceedings of International Conference on Frontiers in Handwriting Recognition, ICFHR*, vol. 2018-Augus, pp. 253–258, 2018, doi: 10.1109/ICFHR-2018.2018.00052.
8. A. Graves, "Supervised Sequence Labelling," 2012.
9. P. Brass, *Advanced Data Structures [Brass 2008-09-08]*. .
10. R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," *Journal of Machine Learning Research*, vol. 12, 2011.