Neurocomputing 139 (2014) 84–96

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Autoencoder for words

Cheng-Yuan Liou [a,*], Wei-Chen Cheng [b], Jiun-Wei Liou [a], Daw-Ran Liou [c]

[a] Department of Computer Science and Information Engineering, National Taiwan University, Taiwan, ROC
[b] Institute of Statistical Science, Academia Sinica, Taiwan, ROC
[c] Sibley School of Mechanical and Aerospace Engineering, Cornell University, United States

ARTICLE INFO

Article history:
Received 13 March 2013
Received in revised form
31 August 2013
Accepted 16 September 2013
Available online 8 April 2014

Keywords:
Minimum entropy coding
Writing style similarity
Elman network
Polysemous word
Semantic indexing

ABSTRACT

This paper presents a training method that encodes each word into a different vector in semantic space and its relation to low entropy coding. Elman network is employed in the method to process word sequences from literary works. The trained codes possess reduced entropy and are used in ranking, indexing, and categorizing literary works. A modification of the method to train the multi-vector for each polysemous word is also presented where each vector represents a different meaning of its word. These multiple vectors can accommodate several different meanings of their word. This method is applied to the stylish analyses of two Chinese novels, Dream of the Red Chamber and Romance of the Three Kingdoms.

1. Introduction

In this work, we study an auto-encoder for the words in literary works. This encoder is accomplished by a devised training method that operates on Elman network [1,2]. Elman network is a super-Turing machine with powerful computation capability [3]. It can be trained to accomplish difficult tasks. We expect that it is powerful enough to capture the diversity of the language and accomplish the encoding task.

The output of the network is a renewed code vector after each training pass. Each dimension of the code may be looked at as a labeled attribute for the word that provides a partial meaning to its word. Manually constructed attributes for a large number of words have been developed for the purpose of indexing of documents, such as the associative method in [4]. It is costly and often brings contradictions to label words with multiple attributes manually by a team of linguists.

The stylish analyses based on statistics on the frequency of occurrence (or co-occurrence) of certain word (words) or phrase (phrases) have been developed with varying degrees of success [5–7]. These analyses must be manually operated repeatedly with consistent and confirmed results. Such statistical methods cannot be applied to document retrieval and semantic indexing. The trained codes obtained in this paper are useful in semantic indexing, see [8]. The non-negative matrix factorization [9] accomplished the stylish task by iteratively training the neural weights with improved non-negative values. The factorization is also based on statistics of words in a whole document. It cannot be used to pick a portion of a document.

Since Elman network is designed for semantic categorization of words [1,2], it cannot support the encoding task. This paper redesigned Elman network to encode the words in semantic space. The achieved codes can be used in ranking, indexing, and categorizing literary works. In the next section, we introduce the detailed design of the single-code encoder and Elman network mentioned above. Reduction of entropy is studied and many experimental results for the novel “The Adventures of Peter Pan” by James Matthew Barrie are recorded and illustrated in the section.

In the third section, we perform the more general approach to deal with the multi-code for the polysemous word. To achieve this goal, we associate with each word with a set of distinct codes such that these codes can accommodate several different meanings of their word. The trained codes are applied to and offer a new approach to the stylish analysis.

2. Single-code encoder by Elman network

In this section, we illustrate the detailed structure and training method for the single-code encoder.

2.1. Encoder and redesigned network

Elman network is a simple recurrent network that has a context layer, see Fig. 1. It was designed to find the hidden structure in sequential patterns [1]. Let L_o, L_h, L_c, and L_i be the number of neurons in the output layer, the hidden layer, the

* Corresponding author.
E-mail address: cyliou@csie.ntu.edu.tw (C.-Y. Liou).

http://dx.doi.org/10.1016/j.neucom.2013.09.055
0925-2312/

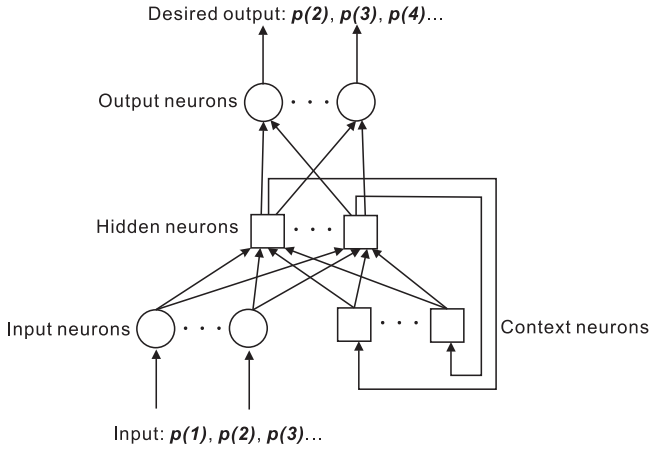

Fig. 1. Illustration of Elman network.

context layer, and the input layer, respectively. In the network, L_h is equal to L_c. During operation, at each training step t, the output of the hidden layer will be loaded to the context layer and together with the input layer to activate the hidden layer at time $t+1$.

Let the two weight matrices between layers be W_{oh} and W_{hic}, where W_{oh} is an L_h+1 by L_o matrix and W_{hic} is an L_i+L_c+1 by L_h matrix. Consider a sequence of input patterns, $\{p(t), t=1,2,3,\ldots\}$, the output vector of the hidden layer is denoted as $h(p(t))$ when $p(t)$ is fed to the input layer. $h(p(t))$ is an L_h by 1 column vector with L_h elements. Let $o(p(t))$ be the output vector of the output layer when $p(t)$ is fed to the input layer. $o(p(t))$ is an L_o by 1 column vector. The function of the hidden layer is $h(p(t))=\varphi(W_{hic}^T[in(p(t))])$, where $[in(p(t))]$ is an L_i+L_c+1 by 1 column vector and φ is a sigmoid activation function that operates on each element of a vector [10]. The input vector $[in(p(t))]$ has the form $[in(p(t))]=[p^T(t), h^T(p(t-1)), 1]^T$. The sigmoid function used in this paper is $\varphi(x)=[2/(1+\exp(-0.5x))]-1$. The function of the output layer is $o(p(t))=\varphi(W_{oh}^T[h^T(p(t)), 1]^T)$.

The back-propagation (BP) training algorithm [10] is commonly employed to train the weights, W_{oh} and W_{hic}, to reduce the difference between $o(p(t))$ and its desired output. The next input pattern $p(t+1)$ is served as the desired output [1]. For example, consider a sequence of word stream, $\{p(1), p(2), p(3), \ldots\}$, the input at time $t=1$ is $p(1)$, and its desired output at time $t=1$ is $p(2)$. The input at time $t=2$ is $p(2)$, and its desired output at time $t=2$ is $p(3)$. All the attempts are aimed at minimizing the error between the network's output and its desired output, $\|o(p(t))-p(t+1)\|^2$, to satisfy the prediction $o(p(t))\approx p(t+1)$.

Consider a corpus with N different words $\{c_n;\ n=1,2,\ldots,N\}$, each word c_n initially has a random lexical code vector with R dimensions, $c_n=[c_{n1}, c_{n2}, \ldots, c_{nR}]^T$. R is equal to L_i in this paper. The input word stream is encoded accordingly, $p(t)=c_i;\ c_i\in\{c_n; n=1,2,\ldots,N\}$. This stream is the same as that used in Elman's method. The redesigned network [8,11] minimizes the prediction error $\|o(p(t))-p(t+1)\|^2$ between the network's output $o(p(t))$ and its desired output $p(t+1)$ to satisfy the prediction $o(p(t))\approx p(t+1)$ by the BP algorithm. The weights W_{hic} and W_{oh} are updated after each word presented.

Set one epoch as when all T words in the corpus were presented, where T is the total number of words in the corpus, $T>N$. We renew the codes every k epochs. Let g be the number of epochs in the training process. The first renewal, $r=2$, is operated after the training of the first k epochs. We operate a "renewal" after each k epochs. We call each k epochs a "pass". After the training in each k epochs, a new raw code is calculated as follows:

$$c_n^{raw}=\frac{1}{freq_n}\sum_{\{t|p(t)=c_n^r\}} o(p(t-1)),\quad n=1\sim N, \tag{1}$$

where $freq_n$ is the number of times that the nth word c_n appears in the current training pass. Note that Elman averaged all the hidden output vectors for each word c_n, but we averaged all its prediction vectors instead.

All renewed codes are normalized before using them in the next pass by the equation

$$c_n^{r+1}=c_n^{nom}=\|c_n^{ave}\|^{-1}c_n^{ave}, \tag{2}$$

where the norm function is $\|c_n\|=(c_n^Tc_n)^{0.5}$. This equation sets the norm of each code vector as 1 and is able to prevent a diminished solution, $\{\|c_n\|\sim 0,\ n=1\sim N\}$, usually derived by the BP algorithm. c_n^{ave} is the nth column of the matrix $C_{R\times N}^{ave}$,

$$C_{R\times N}^{ave}=C_{R\times N}^{raw}-\frac{1}{N}C_{R\times N}^{raw}\begin{bmatrix}1 & \cdots & 1\\ \vdots & 1 & \vdots\\ 1 & \cdots & 1\end{bmatrix}_{N\times N}. \tag{3}$$

In (3), $C_{R\times N}$ is a code matrix and has the form $[c_1, c_2, \ldots, c_N]$. Eq. (3) makes each row of the code matrix $C_{R\times N}^{ave}$ become zero-mean.

The initial codes, $c_n^{r=1}$, before the first renewal is randomly assigned under the restriction that different words have different codes and they are normalized. Then use the BP algorithm to reduce the prediction error $\|o(p(t))-p(t+1)\|^2$ in each training step.

Note that the training step is operated after each word presented and the renewing step is operated after each k epochs. Iterations on these two steps constitute the training process. After the training process, we expect that each element of the code vector c_n contains a well learned attribute of the word c_n.

2.2. Properties of the encoder

Data description: We use the novel "The Adventures of Peter Pan by James Matthew Barrie" as the corpus to illustrate the encoder properties. We concatenated all sentences in it to form the training word stream. Note that the stream Elman used is formed with randomly sampled sentences from a manually constructed sentence pool. To reduce the amount of vocabularies, we apply several grammar rules to separate the suffix from the word such as "-s", "-ing", and "-ed". For example, "playing" becomes "play + ING"; "lights becomes "light + NVs"; "children's" becomes "children+ 's"; "turned" becomes "turn + Ved". After preprocessing, there are 3805 different root words including the mark that is added to represent the end of a sentence. The total number of sentences is 3101 and the total number of words is $T=54{,}999$.

Experiment setting: The network has $L_i=$ 15 input neurons, $L_o=15$ output neurons, $L_h=30$ hidden layer neurons, and $L_c=30$ context layer neurons. The numbers of neurons, L_i and $L_o=15$, are

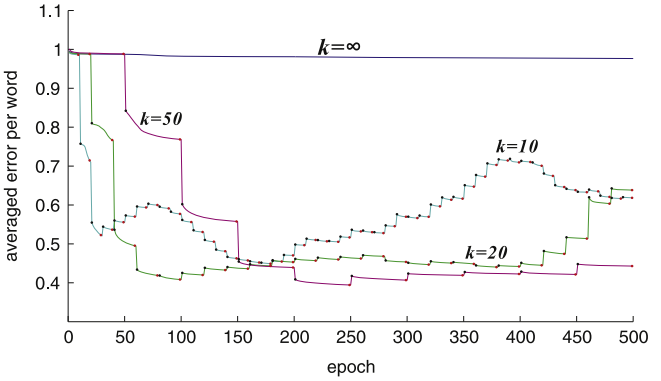

Fig. 2. Error curves under different re-encoding conditions, $k=\infty$, $k=10$, $k=20$, and $k=50$. The curve from red point to black point shows how error is reduced by re-encoding. The curve from black point to red point shows how error is reduced by BP weight adjustment. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)