Neurocomputing 132 (2014) 113–120

Contents lists available at ScienceDirect

Neurocomputing

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

Probabilistic speech feature extraction with context-sensitive Bottleneck neural networks

Martin Wöllmer*, Björn Schuller

Institute for Human-Machine Communication, Technische Universität München, Theresienstr. 90, 80333 München, Germany

ARTICLE INFO

Article history:
Received 22 February 2012
Received in revised form
11 June 2012
Accepted 19 June 2012
Available online 22 October 2013

Keywords:
Probabilistic feature extraction
Bottleneck networks
Long Short-Term Memory
Bidirectional speech processing

ABSTRACT

We introduce a novel context-sensitive feature extraction approach for spontaneous speech recognition. As bidirectional Long Short-Term Memory (BLSTM) networks are known to enable improved phoneme recognition accuracies by incorporating long-range contextual information into speech decoding, we integrate the BLSTM principle into a Tandem front-end for probabilistic feature extraction. Unlike the previously proposed approaches which exploit BLSTM modeling by generating a discrete phoneme prediction feature, our feature extractor merges continuous high-level probabilistic BLSTM features with low-level features. By combining BLSTM modeling and Bottleneck (BN) feature generation, we propose a novel front-end that allows us to produce context-sensitive probabilistic feature vectors of arbitrary size, independent of the network training targets. Evaluations on challenging spontaneous, conversational speech recognition tasks show that this concept prevails over recently published architectures for feature-level context modeling.

1. Introduction

Considering the unsatisfying word accuracies that occur whenever today's automatic speech recognition (ASR) systems are faced with 'unfriendly' scenarios such as conversational and disfluent speaking styles, emotional coloring of speech, or distortions caused by noise, the need for novel concepts that go beyond main stream ASR techniques becomes clear [1,2]. Since systems that are exclusively based on conventional generative Hidden Markov Models (HMM) appear to be limited in their reachable modeling power and recognition rates, the combination of Markov modeling and discriminative techniques such as neural networks has emerged as a promising method to cope with challenging ASR tasks. Hence, Tandem front-ends that apply multi-layer perceptrons (MLP) or recurrent neural networks (RNN) to generate probabilistic features for HMM processing are increasingly used in modern ASR systems [3–6].

Such Tandem systems apply neural networks to map from standard low-level speech features like Mel-Frequency Cepstral Coefficients (MFCC) or Perceptual Linear Prediction (PLP) features to phoneme or phoneme state posteriors which in turn can be used as features within an HMM framework. Usually, the quality of those probabilistic features heavily depends on the phoneme recognition accuracy of the underlying neural network. As phoneme recognition is known to profit from context modeling, an obvious strategy to consider contextual information is to use a stacked sequence of past and future vectors as input for an MLP that generates phoneme predictions [7]. However, extensive experiments in [8] have shown that flexible context modeling *within* the neural network leads to better phoneme recognition results than processing fixed-length feature vector sequences. Bidirectional Long-Short Term Memory (BLSTM) recurrent neural networks based on the concept introduced in [9] and refined in [10,8] were shown to outperform comparable context-sensitive phoneme recognition architectures such as MLPs, RNNs, or triphone HMMs, as they are able to model a self-learned amount of context via recurrently connected memory blocks. Thus, it seems promising to exploit the concept of BLSTM in Tandem ASR systems.

First attempts to use BLSTM networks for speech recognition tasks can be found in the area of keyword spotting [11–13]. In [14] it was shown that also continuous speech recognition performance can be enhanced when using a discrete feature, that indicates the current phoneme identity determined by a BLSTM network, in addition to MFCC features. Further performance gains could be demonstrated in [15] by applying a multi-stream HMM framework that models MFCC features and the discrete BLSTM phoneme estimate as two independent data streams. An enhanced BLSTM topology for multi-stream BLSTM-HMM modeling was presented in [6], leading to further ASR improvements.

An interesting alternative to using the logarithmized and decorrelated activations of the *output* layer of recurrent neural networks (RNN) or multi-layer perceptrons (MLP) as probabilistic

* Corresponding author. Tel.: +49 89 289 28550; fax: +49 89 289 28535.
E-mail address: woellmer@tum.de (M. Wöllmer).

0925-2312/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.neucom.2012.06.064

features is the extraction of activations in a narrow *hidden* layer within the network as the so-called 'Bottleneck' (BN) features [16]. This implies the advantage that the size of the feature space can be chosen by defining the size of the network's Bottleneck layer which makes the dimension of the feature vectors independent of the number of network training targets. The linear outputs of the Bottleneck layer are usually well decorrelated and do not have to be logarithmized.

In this paper, we present and optimize a novel approach towards BLSTM feature generation for Tandem ASR. First, we replace the discrete phoneme prediction feature used in [6] by the continuous logarithmized vector of BLSTM output activations and merge it with low-level MFCC features. By that we obtain extended context-sensitive Tandem feature vectors that lead to improved results when evaluated on the COSINE [17] and the Buckeye [18] corpora. Then, we show how ASR performance can be further enhanced by uniting the concepts of BLSTM and Bottleneck feature extraction.

The structure of our paper is as follows: First, in Section 2, we explain the BLSTM technique and provide an overview over the previously proposed BLSTM-based ASR systems. Next, we introduce our BN-BLSTM front-end (Section 3) and the used spontaneous speech corpora (Section 4). Finally, in Section 5, we present our experiments and results.

2. Background

2.1. Bidirectional Long Short-Term Memory RNNs

Even though the recurrent connections in RNNs allow to model contextual information, which makes them a more effective sequence labeling framework than, for example, MLPs, it is known that the range of context that standard RNNs can access is limited [19]. The reason for this is that the influence of a certain input on the hidden and output layer of the network either blows up or decays exponentially over time while cycling around the recurrent connections of the network. In literature, this problem is referred to as the so-called *vanishing gradient problem*. The effect of this decaying sensitivity is that RNNs have difficulties in learning temporal dependencies for which relevant inputs and targets are separated by more than ten time steps [19], i.e., the network cannot *remember* the previous inputs over longer time spans so that it is hardly possible to model input-target dependencies that are not synchronous. One of the most effective techniques to overcome this problem is the Long Short-Term Memory architecture [9], which is able to store information in linear memory cells over a longer period of time.

An LSTM hidden layer is composed of multiple recurrently connected subnets which will be referred to as *memory blocks* in the following. Every memory block consists of self-connected *memory cells* and three multiplicative *gate* units (input, output, and forget gates). Since these gates allow for write, read, and reset operations within a memory block, an LSTM block can be interpreted as (differentiable) memory chip in a digital computer. Fig. 1 contains an illustration of the architecture of a memory block comprising one memory cell.

If α_t^{in} denotes the activation of the input gate at time t *before* the activation function f_g has been applied and β_t^{in} represents the activation *after* application of the activation function, the input gate activations (forward pass) can be written as

$$\alpha_t^{\text{in}} = \sum_{i=1}^{I} \eta^{i,\text{in}} x_t^i + \sum_{h=1}^{H} \eta^{h,\text{in}} \beta_{t-1}^h + \sum_{c=1}^{C} \eta^{c,\text{in}} s_{t-1}^c \tag{1}$$

and

$$\beta_t^{\text{in}} = f_g(\alpha_t^{\text{in}}), \tag{2}$$

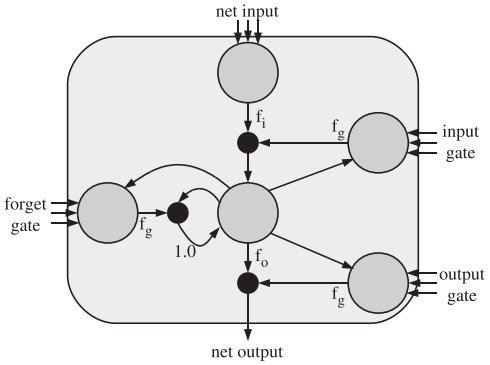

Fig. 1. LSTM memory block consisting of one memory cell: the input, output, and forget gates collect activations from inside and outside the block which control the cell through multiplicative units (depicted as small circles); input, output, and forget gate scale input, output, and internal state; f_i, f_g, and f_o denote activation functions; the recurrent connection of fixed weight 1.0 maintains the internal state.

respectively. The variable η^{ij} corresponds to the weight of the connection from unit i to unit j while 'in', 'for', and 'out' refer to input gate, forget gate, and output gate, respectively. Indices i, h, and c count the inputs x_t^i, the cell outputs from other blocks in the hidden layer, and the memory cells, respectively, while I, H, and C are the number of inputs, the number of cells in the hidden layer, and the number of memory cells. Finally, s_t^c corresponds to the *state* of a cell c at time t, meaning the activation of the linear cell unit.

Similarly, the activation of the forget gates before and after applying f_g can be calculated as follows:

$$\alpha_t^{\text{for}} = \sum_{i=1}^{I} \eta^{i,\text{for}} x_t^i + \sum_{h=1}^{H} \eta^{h,\text{for}} \beta_{t-1}^h + \sum_{c=1}^{C} \eta^{c,\text{for}} s_{t-1}^c \tag{3}$$

$$\beta_t^{\text{for}} = f_g(\alpha_t^{\text{for}}). \tag{4}$$

The memory cell value α_t^c is a weighted sum of inputs at time t and hidden unit activations at time $t-1$:

$$\alpha_t^c = \sum_{i=1}^{I} \eta^{i,c} x_t^i + \sum_{h=1}^{H} \eta^{h,c} \beta_{t-1}^h. \tag{5}$$

To determine the current state of a cell c, we scale the previous state by the activation of the forget gate and the input $f_i(\alpha_t^c)$ by the activation of the input gate:

$$s_t^c = \beta_t^{\text{for}} s_{t-1}^c + \beta_t^{\text{in}} f_i(\alpha_t^c). \tag{6}$$

The computation of the output gate activations follows the same principle as the calculation of the input and forget gate activations, however, this time we consider the *current* state s_t^c, rather than the state from the previous time step:

$$\alpha_t^{\text{out}} = \sum_{i=1}^{I} \eta^{i,\text{out}} x_t^i + \sum_{h=1}^{H} \eta^{h,\text{out}} \beta_{t-1}^h + \sum_{c=1}^{C} \eta^{c,\text{out}} s_t^c \tag{7}$$

$$\beta_t^{\text{out}} = f_g(\alpha_t^{\text{out}}). \tag{8}$$

Finally, the memory cell output is determined as

$$\beta_t^c = \beta_t^{\text{out}} f_o(s_t^c). \tag{9}$$

The overall effect of the gate units is that the LSTM memory cells can store and access information over long periods of time and thus avoid the vanishing gradient problem. For instance, as long as the input gate remains closed (corresponding to an input gate activation close to zero), the activation of the cell will not be overwritten by new inputs and can therefore be made available to the net much later in the sequence by opening the output gate.