SPOKEN TERM DETECTION WITH CONNECTIONIST TEMPORAL CLASSIFICATION: A NOVEL HYBRID CTC-DBN DECODER

Martin Wöllmer, Florian Eyben, Björn Schuller, Gerhard Rigoll

Institute for Human-Machine Communication, Technische Universität München, Germany
woellmer@tum.de

ABSTRACT

This paper proposes a novel system for robust keyword detection in continuous speech. Our decoder is composed of a bidirectional Long Short-Term Memory recurrent neural network using a Connectionist Temporal Classification (CTC) output layer, and a Dynamic Bayesian Network (DBN). The CTC network exploits bidirectional context information to reliably identify phonemes, whereas the DBN is able to discriminate between keywords and arbitrary speech while explicitly modeling substitutions, deletions, and insertions in the CTC phoneme output string. Our technique is vocabulary independent and does not require an explicit garbage model. Experiments show that our system architecture prevails over a standard Hidden Markov Model approach.

***Index Terms*—** Spoken Term Detection, Keyword Spotting, Dynamic Bayesian Networks, Connectionist Temporal Classification

1. INTRODUCTION

Keyword spotting aims at picking out several predefined keywords from continuous speech signals. In recent years it has found many applications, including voice command detection, information retrieval, and embodied conversational agents.

At present the most popular methodology for keyword spotting is using Hidden Markov Models (HMM) [1]. A major difficulty with HMM based systems is that they are forced to model the *garbage* (i. e. non-keyword) parts of the signal as well as the keywords themselves. However, a structure flexible enough to model all possible garbage words is likely to be able to model the keywords as well. For example, if phoneme level models are used, then garbage parts can be accurately captured by a model that connects all possible phonemes; however, such a model will also fit the keywords. One solution is to use whole word models for both, garbage and keywords, but this requires that all the keywords occur many times in the training corpus, and also means that new keywords cannot be added without training new models.

The architecture introduced in this paper overcomes these drawbacks by using a phoneme based recognition system with no explicit garbage model. The architecture is robust to phoneme recognition errors, and unlike methods based on large vocabulary speech recognizers, it does not require a language model: only the keyword phonemizations are needed.

Our system thereby consists of two major components: a bidirectional Long Short-Term Memory (BLSTM) recurrent neural net [2] and a Dynamic Bayesian Network (DBN). The BLSTM network can access long-range context information along both input directions and uses a Connectionist Temporal Classification (CTC) output layer [3] to localize and classify the phonemes. BLSTM has been proven to outperform standard methods of modeling context such as triphone HMMs [2], while CTC allows the network to be trained on data that has not been presegmented into phonemes. In the DBN layer of our hybrid model architecture the phoneme string detected by the CTC network is decoded so that keywords can be robustly recognized even if the pronunciation differs from the keyword phonemizations in the dictionary. The DBN uses hidden variables as well as the concept of *switching parents* [4] to discriminate between keywords and arbitrary speech and to explicitly learn and model typical phoneme confusions, deletions, and insertions that occur in the CTC layer.

DBNs (and other graphical models) offer a flexible statistical framework that is increasingly applied to speech recognition tasks [4]. Hybrid or Tandem architectures that combine discriminatively trained neural networks with Gaussian mixture modeling are widely used for speech recognition [5, 6]. However, BLSTM is a relatively new architecture that has so far been applied to keyword spotting in only three works: in [7] and [8] the *framewise* phoneme predictions of BLSTM (without CTC) were shown to enhance keyword spotting performance of discriminative and generative models, respectively; and in [9] a keyword spotter using only BLSTM-CTC was introduced. The disadvantage of the latter method is that it has a separate output unit for each keyword, which requires excessive amounts of training data for large vocabularies, and also means the network must be retrained when new keywords are added. The aim of this work is to combine the high-level flexibility of graphical models with the low-level signal processing power of BLSTM-CTC – without generative Gaussian mixture modeling.

The rest of the paper is structured as follows: Sections 2 and 3 briefly reviews the principle bidirectional Long Short-Term Memory networks and Connectionist Temporal Classification. Section 4 explains the DBN architecture of our hybrid keyword spotter, while experimental results on the TIMIT database are shown in Section 5. Concluding remarks are given in Section 6.

2. BIDIRECTIONAL LONG SHORT-TERM MEMORY

Analysis of the error flow in conventional recurrent neural nets led to the finding that long range context is inaccessible to standard RNNs since the backpropagated error either blows up or decays over time (vanishing gradient problem). This led to the introduction of Long Short-Term Memory (LSTM) RNNs [10]. An LSTM layer is composed of recurrently connected memory blocks, each of which contains one or more memory cells, along with three multiplicative 'gate' units: the input, output, and forget gates. The gates per-

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement No. 211486 (SEMAINE).

978-1-4244-4296-6/10/$25.00 ©2010 IEEE

form functions analogous to read, write, and reset operations. More specifically, the cell input is multiplied by the activation of the input gate, the cell output by that of the output gate, and the previous cell values by the forget gate. The overall effect is to allow the network to store and retrieve information over long periods of time, thereby overcoming the vanishing gradient problem.

Another problem with standard RNNs is that they have access to past but not future context. This can be overcome by using bidirectional RNNs [11], where two separate recurrent hidden layers scan the input sequences in opposite directions. The two hidden layers are connected to the same output layer, which therefore has access to context information in both directions. The amount of context information that the network actually uses is learned during training, and does not have to be specified beforehand.

Combining bidirectional networks with LSTM gives bidirectional LSTM, which has demonstrated excellent performance in phoneme recognition [2], keyword spotting [9], and emotion recognition from speech [12].

3. CONNECTIONIST TEMPORAL CLASSIFICATION

A major problem with the standard objective functions for RNNs is that they require individual targets for each point in the data sequence, which in turn requires the boundaries between segments with different labels (e.g. the phoneme boundaries in speech) to be pre-determined. The Connectionist Temporal Classification output layer [3] solves this problem by allowing the network to choose the location as well as the class of each label. By summing up over all sets of label locations that yield the same label sequence, CTC determines a probability distribution over possible labelings, conditioned on the input sequence.

A CTC layer has as many output units as there are distinct labels for a task, plus an extra *blank* unit for no label. The activations of the outputs at each timestep are normalized and interpreted as the probability of observing the corresponding label (or no label) at that point in the sequence. Because these probabilities are conditionally independent given the input sequence, the total probability of a given (framewise) sequence $l^{fr}_{1:F}$ of blanks and labels is

$$p(l^{fr}_{1:F}|x_{1:F}) = \prod_{f=1}^{F} o_f^{l_f^{fr}}, \tag{1}$$

where $x_{1:F}$ is a length F input sequence and o_f^k is the activation of output unit k at time f. In order to sum over all the output sequences corresponding to a particular labeling (regardless of the *location* of the labels) we define an operator $\mathcal{B}(\cdot)$ that removes first the repeated labels and then the blanks from the output sequence, so that e.g. $\mathcal{B}(AA--BBB-B) = \mathcal{B}(AAAB-BB) = ABB$. The total probability of the length T labeling $l_{1:T}$, where $T \leq F$, is then

$$p(l_{1:T}|x_{1:F}) = \sum_{l^{fr}_{1:F}:\mathcal{B}(l^{fr}_{1:F})=l_{1:T}} p(l^{fr}_{1:F}|x_{1:F}) \tag{2}$$

A naive calculation of (2) is unfeasible, because the number of $l^{fr}_{1:F}$ terms corresponding to each labeling increases exponentially with the sequence length. However, $p(l_{1:T}|x_{1:F})$ can be efficiently calculated with a dynamic programming algorithm similar to the forward-backward algorithm for HMMs (see [3]).

The CTC objective function O^{CTC} is defined as the negative log likelihood of the training set $\mathbb{S}$

$$O^{CTC} = -\sum_{(x_{1:F},l_{1:T})\in\mathbb{S}} \ln p(l_{1:T}|x_{1:F}) \tag{3}$$

An RNN with a CTC output layer can be trained with gradient descent by backpropagating through time the following partial derivatives of O^{CTC} with respect to the output activations:

$$\frac{\partial O^{CTC}}{\partial o_f^k} = \frac{-1}{p(l_{1:T}|x_{1:F})o_f^k} \sum_{t\in lab(l_{1:T},k)} \alpha_f(t)\beta_f(t), \tag{4}$$

where $lab(l_{1:T},k)$ is the set of positions in $l_{1:T}$ where the label k occurs. $\alpha_f(t)$ and $\beta_f(t)$ denote the forward and backward variables as defined in [3].

4. DYNAMIC BAYESIAN NETWORK ARCHITECTURE

This section introduces the Dynamic Bayesian Network that processes the CTC output phoneme label strings $l_{1:T}$ in order to detect keywords. Figure 1 shows the DBN model architecture that is used during *training*. The grey-shaded box represents the BLSTM-CTC layer which is composed of an input layer i_t^n, two hidden layers h_t^f and h_t^b (forward and backward direction), and an output layer o_t^n. In the DBN layer hidden variables are displayed as circles and observed variables are represented as squares. Straight lines correspond to deterministic conditional probability functions (CPFs) while zig-zagged lines denote random CPFs. Note that even though the BLSTM network produces an output activation for every feature frame index f, only the non-blank labels (synchronized with time index t) are forwarded to the DBN. The variables x_t denote the acoustic feature vectors. Within the DBN layer the following random variables are additionally defined for every time step t: q_t is the current phoneme index corresponding to the phoneme annotation of the training sequence, q_t^c is a simple count variable containing the current position within the ground truth phoneme string, and the binary variables d_t and i_t indicate deletions and insertions, respectively. Assuming a CTC output phoneme sequence of length T, the DBN structure in Figure 1 specifies the factorization

$$p(l_{1:T}, q_{1:T}, q_{1:T}^c, d_{1:T}, i_{1:T}) = \prod_{t=1}^{T} p(l_t|q_t) f(q_t|q_t^c) p(d_t) p(i_t) f(q_1^c|d_1, i_1) \prod_{t=2}^{T} f(q_t^c|q_{t-1}^c, d_t, i_t) \tag{5}$$

with $p(\cdot)$ denoting random conditional probability functions and $f(\cdot)$ describing deterministic CPFs. The probability of the observed label sequence $l_{1:T}$ can then be computed by summing over all hidden variables. The CPF $f(q_t^c|q_{t-1}^c, d_t, i_t)$ defines that the count variable q_t^c is incremented by one at every time step in case d_t and i_t are equal to zero. Otherwise, if there is a deletion ($d_t = 1$), q_t^c is incremented by two. On the other hand, an insertion implies that $q_t^c = q_{t-1}^c$. Thus, apart from training the CTC network, the goal of the training phase is to learn the CPFs $p(l_t|q_t)$, $p(d_t)$, and $p(i_t)$ (i. e. to learn substitution, deletion, and insertion probabilities).

Figure 2 shows the DBN *decoding* architecture for keyword spotting. Dotted lines within the DBN layer represent so-called *switching parent* dependencies which allow a variable's parents to change conditioned on the current value of the switching parent. Thereby a switching parent can not only change the set of parents but also the implementation (i. e. the CPF) of a parent. The DBN for decoding contains five additional hidden variables: w_t denotes the identity of the current word, w_t^{ps} is the position within the word, w_{tr} indicates a word transition, c_t represents a 'cut' variable that is equal to one as soon as there is a deletion at the *end* of a keyword, and a

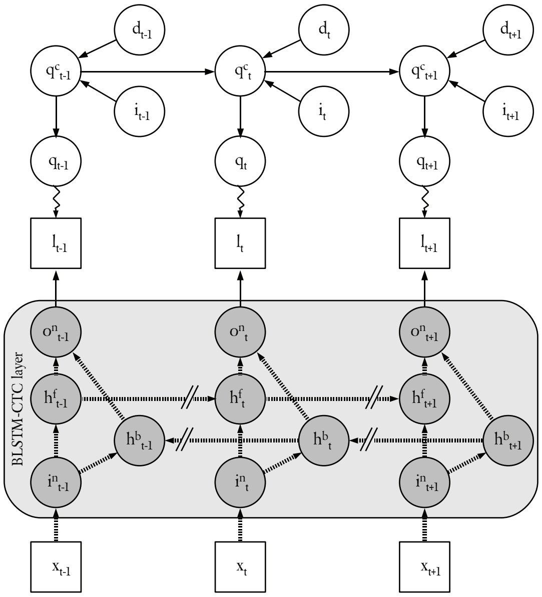

Fig. 1. Hybrid CTC-DBN architecture for training

hidden *garbage variable* g_t indicates whether the current word is a keyword or not. The DBN structure corresponds to the factorization

$$\begin{aligned} & p(l_{1:T}, q_{1:T}, w_{1:T}, w^{ps}_{1:T}, w^{tr}_{1:T}, d_{1:T}, i_{1:T}, c_{1:T}, g_{1:T}) = \\ & \prod_{t=1}^{T} p(l_t|q_t) f(w^{tr}_t|w_t, w^{ps}_t) f(c_t|w_t, w^{ps}_t) f(g_t|w_t) p(d_t|g_t) \\ & p(i_t|g_t) p(w_1) f(w^{ps}_1|d_1, i_1) p(q_1|w_1, w^{ps}_1, g_1, c_1, i_1) \qquad (6) \\ & \prod_{t=2}^{T} p(w_t|w_{t-1}, w^{tr}_{t-1}) f(w^{ps}_t|w^{ps}_{t-1}, w^{tr}_{t-1}, d_t, i_t) \\ & p(q_t|q_{t-1}, w_t, w^{ps}_t, g_t, c_t, i_t) \end{aligned}$$

The hidden variable w_t can take values in the range $0...K$ with K being the number of different keywords in the vocabulary. When $w_t = 0$ the model is in the *garbage state* which means that no keyword is uttered at that time. The variable g_t is then equal to one. w^{tr}_{t-1} is a switching parent of w_t: if no word transition is indicated, w_t is equal to w_{t-1}. Otherwise a word bigram specifies the CPF $p(w_t|w_{t-1}, w^{tr}_{t-1} = 1)$. In our experiments we simplified the word bigram to a zerogram which makes all keywords equally likely. However, we introduced differing a priori likelihoods for keywords and garbage phonemes in order to be able to adjust the trade-off between true positives and false positives. A word transition occurs whenever $w^{ps} = P$, whereas P is the number of phonemes contained in w_t. If the model is in the *garbage state*, w^{tr}_t is always equal to one, meaning that 'garbage words' are assumed to consist of only one phoneme. In case a keyword is detected, q_t is a deterministic function of w_t and w^{ps}_t. Otherwise, if garbage speech is observed, a trained phoneme bigram specifies the conditional probability $p(q_t|q_{t-1})$. The same holds for the case when an insertion occurs while a keyword is decoded ($i_t = 1$), or when the last phoneme of a keyword is deleted ($d_t = 1$ and $c_t = 1$). Similar to the variable q^c_t in the DBN for training, the increment of w^{ps}_t is controlled by the insertion and the deletion variable. The 'cut' variable c_t is equal to one if w^{ps}_t exceeds P, meaning that the last phoneme of a keyword has been deleted.

Note that the DBN decoder can not only potentially tolerate phoneme substitutions, deletions, and insertions – the CPF $p(q_t|q_{t-1}, w_t, w^{ps}_t, g_t, c_t, i_t)$ also strongly biases the DBN to choose $g_t = 0$ (i. e. detect a keyword) when a phoneme sequence corresponding to a keyword is observed. Decoding such an observation while in the garbage state $g_t = 1$ would lead to 'phoneme transition penalties' since the phoneme bigram $p(q_t|q_{t-1})$ contains probabilities less than one. By contrast, $p(q_t|w_t, w^{ps}_t, g_t = 0)$ is deterministic, introducing no likelihood penalties at phoneme borders.

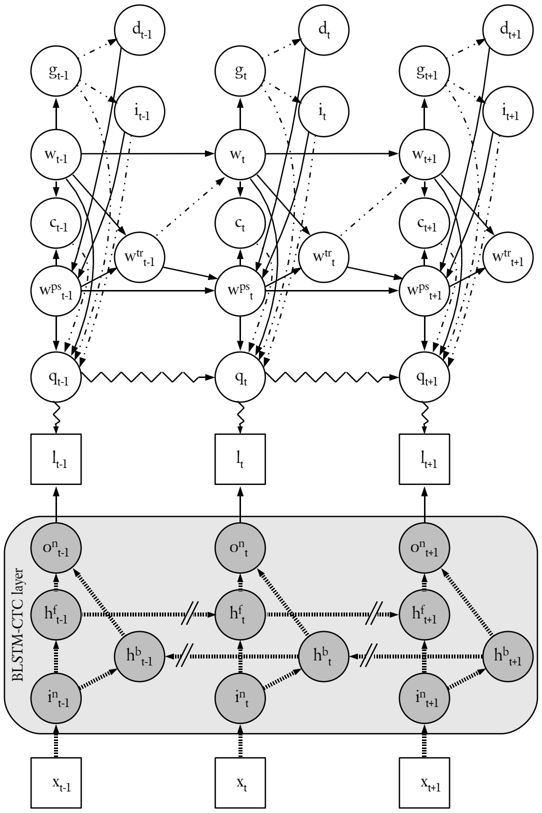

Fig. 2. Hybrid CTC-DBN architecture for keyword spotting

5. EXPERIMENTS AND RESULTS

Our hybrid CTC-DBN keyword spotter was trained and evaluated on the TIMIT speech corpus. The feature vectors consisted of cepstral mean normalized MFCC coefficients 1 to 12, log. energy, as well as first and second order delta coefficients. For the training of the BLSTM-CTC network, 200 utterances of the TIMIT training split were used as validation set while the net was trained on the remaining training sequences. The BLSTM input layer had a size of 39 (one for each MFCC feature) and the size of the output layer was 40 since we used the reduced set of 39 TIMIT phonemes plus one blank label. The network consisted of three hidden layers per input direction: a backpropagation layer composed of 78 hidden cells and two hidden LSTM layers containing 128 and 80 memory blocks respectively. Each memory block thereby consisted of one cell. To improve generalization, zero mean Gaussian noise with standard deviation 0.6 was added to the inputs during training. The network was

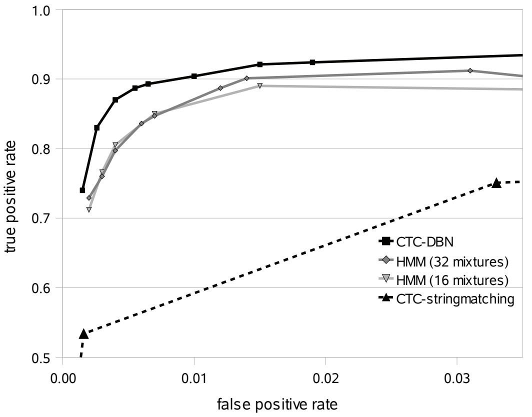

Fig. 3. Part of the ROC curve using the CTC-DBN, HMMs, and a simple string search on the CTC phoneme output

trained with online gradient descent, using a learning rate of 10^{-4} and a momentum of 0.9. We randomly chose 60 keywords from the TIMIT corpus to evaluate the keyword spotter. The dictionary thereby allowed for multiple pronunciations.

For comparison, a phoneme based keyword spotter using conventional (maximum likelihood) HMM modeling was trained and evaluated on the same task. Each phoneme was represented by three states (left-to-right HMMs) with up to 32 Gaussian mixtures. Thereby we used cross-word triphone models in order to account for contextual information. All phoneme HMMs were retrained using embedded training. For keyword detection we defined a set of keyword models and a garbage model. The keyword models estimate the likelihood of a feature vector sequence, given that it corresponds to the keyword phoneme sequence. The garbage model is composed of phoneme HMMs that are fully connected to each others, meaning that it can model any phoneme sequence. Via Viterbi decoding the best path through all models is found and a keyword is detected as soon as the path passes through the corresponding keyword HMM. In order to be able to adjust the operating point on the Receiver Operating Characteristics (ROC) curve, we used different a priori likelihoods for keyword and garbage HMMs, corresponding to the word zerogram used for the DBN.

Moreover we evaluated the benefit of the DBN decoder in comparison to a trivial phoneme string search on the raw CTC output. Figure 3 shows a part of the ROC curve for our CTC-DBN keyword spotter, the HMM based keyword spotter as well as for a simple string matching approach, tolerating a Levenshtein distance of 1 and 2 respectively. Note that due to the design of the decoder, the full ROC curve – ending at an operating point tpr=1 and fpr=1 – cannot be determined, since the model does not include a confidence threshold that can be set to an arbitrarily low value. It can be seen that the hybrid CTC-DBN decoder not only prevails over CTC string matching but also outperforms the HMM approach by up to 7 % (at a false positive rate of 0.4 %). Conducting the McNemar's test revealed that the performance difference between the CTC-DBN and the HMM is statistically significant at a common significance level of 0.01. For higher a priori keyword likelihoods the performance gap becomes smaller as more phoneme confusions are tolerated when seeking for keywords.

6. CONCLUSION AND OUTLOOK

We proposed a novel decoder for robust vocabulary independent keyword spotting. It combines a BLSTM recurrent neural network for context sensitive phoneme prediction with a Dynamic Bayesian Network designed to reliably detect keywords without requiring an explicit garbage model. Due to the CTC output layer, the system can be trained on data that has not been presegmented into phonemes. Our hybrid CTC-DBN recognizer was thereby shown to outperform a conventional HMM based approach. Future works on keyword detection might investigate the combination of triphone and BLSTM-CTC modeling.

7. REFERENCES

[1] R. C. Rose and D. B. Paul, "A hidden markov model based keyword recognition system," in *Proc. of ICASSP*, Albuquerque, NM, USA, 1990.

[2] A. Graves, S. Fernandez, and J. Schmidhuber, "Bidirectional LSTM networks for improved phoneme classification and recognition," in *Proc. of ICANN*, Warsaw, Poland, 2005, pp. 602–610.

[3] A. Graves, S. Fernandez, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: Labelling unsegmented data with recurrent neural networks," in *Proc. of 23rd Int. Conf. on Machine Learning*, Pittsburgh, USA, 2006.

[4] J. A. Bilmes, "Graphical models and automatic speech recognition," *Mathematical Foundations of Speech and Language Processing*, 2003.

[5] H. Hermansky, D. P. W. Ellis, and S. Sharma, "Tandem connectionist feature extraction for conventional HMM systems," in *Proc. of ICASSP*, Istanbul, Turkey, 2000, vol. 3, pp. 1635–1638.

[6] H. Ketabdar and H. Bourlard, "Enhanced phone posteriors for improving speech recognition systems," in *IDIAP-RR*, 2008, number 39.

[7] M. Wöllmer, F. Eyben, J. Keshet, A. Graves, B. Schuller, and G. Rigoll, "Robust discriminative keyword spotting for emotionally colored spontaneous speech using bidirectional LSTM networks," in *Proc. of ICASSP*, Taipei, Taiwan, 2009.

[8] M. Wöllmer, F. Eyben, A. Graves, B. Schuller, and G. Rigoll, "A Tandem BLSTM-DBN architecture for keyword spotting with enhanced context modeling," in *Proc. of NOLISP 2009*, Vic, Spain, 2009.

[9] S. Fernandez, A. Graves, and J. Schmidhuber, "An application of recurrent neural networks to discriminative keyword spotting," in *Proc. of ICANN*, Porto, Portugal, 2007, pp. 220–229.

[10] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[11] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Transactions on Signal Processing*, vol. 45, pp. 2673–2681, 1997.

[12] M. Wöllmer, F. Eyben, S. Reiter, B. Schuller, C. Cox, E. Douglas-Cowie, and R. Cowie, "Abandoning emotion classes - towards continuous emotion recognition with modelling of long-range dependencies," in *Proc. of Interspeech*, Brisbane, Australia, 2008, pp. 597–600.