www.ijiarec.com

ISSN:2348-2079

Volume-5 Issue-1

International Journal of Intellectual Advancements and Research in Engineering Computations

Recurrent neural network based language model for large vocabulary continuous Tamil language speech recognition system

Sundarapandiyan S[#] Research Scholar, Dr.Shanthi N[*] Professor,

Department of Computer Science and Engineering, Nandha Engineering College, Erode, Tamilnadu, India

[#]s_sundarapandiyan@yahoo.com [*]shanthimoorthi@yahoo.com

Abstract **- Language processing make easier path to various speech recognition system to enhance the precision of various Language interaction systems. In this paper we project the work done on updating and improvement of the language model component of a continuous speech recognition system. This experiment proposes a recurrent neural network based language model for Tamil speech recognition system, which aims at reduction of perplexity, word error rate and generating more meaningful text. The RNN language model operates as a predictive model for the next word given in the previous schema. In our experiments, we show that the recurrent neural network language model outperforms the n-gram model and feed forward language model on various Tamil language datasets.**

Index Terms---Tamil Language, Automatic Speech Recognition, Recurrent Neural Network, N-Gram Language Model.

I. INTRODUCTION

Automatic speech recognition is the process by which a computer maps an acoustic speech signal to text. Speech Recognition systems are made up of acoustic model and language model .An acoustic model contains a statistical representation of the distinct sounds that make up each word in the Language Model.

Language modeling is the task of estimating the probability distribution of linguistic units such as words and sentences. The probability distribution itself is referred to as a language model. Language models have been used in a variety of NLP tasks including speech recognition, document classification, optical character recognition, and statistical machine translation. The design of speech recognition system requires careful attention to language models of various stages in order to improve the accuracy of the speech recognition system.

The main classification of language models are unigram language model, n-gram language model and neural network based language models .A unigram language model can be treated as the combination of a bunch of one-state finite automatons, it is also known as a grammar file. It contains sets of predefined combinations of words only. Unigram language model approximation is

$$Puni = (t1t2t3) = P(t1)P(t2)P(t3) \quad (1)$$

In this model, the probability to hit each word depends on its own instinct, so we have only one-state finite automatons as units. The words bigram and trigram language model denoted as n-gram language models with n=2 and n=3, respectively. In n-gram model, the probability of observing the sentence w1,..., wm is approximated as

$$P(w1, w2, \ldots\ldots, wm) = \prod_{i=1}^{m} P(wi|w1, w2, \ldots\ldots, wi-1) = P(wi|w_{i-(n-1)}, \ldots\ldots w_{i-1}) \quad (2)$$

From (2) the N-gram language model uses the last (n-1) words to compute the likelihood of the current word. A bigram language model uses the previous one word to predict the next word and a trigram language model uses the previous two words.

Next type of language model is Neural Network based language modelarchitecture can be divided into two types: recurrent and non-recurrent networks. An important non-recurrent neural network consists of architectures in which cells are organized into layers, and only unidirectional connections are permitted between adjacent layers. This is known as a feed forward multi-layer

perceptron (MLP) architecture. On the other hand, recurrent neural networks are characterized by both feed forward and feedback paths between the layers. The feedback paths are enable the activation at any layer either to be used as an input to a previous layer or return to that same layer after one or more time steps [1].

Each network training pattern consists of on context and a target word. The output probabilities are calculated by performing a forward pass through the network with the given context. The output corresponding to the target is used to calculate perplexity.

II. RELATEDWORK

In this section we describes the related works belongs to the language modelling techniques. Statistical language modelling is to learn the joint probability function of sequences of words in a language [2].Multi- Class Composite N-gram, is proposed to avoid a data sparseness problem for spoken language, [3]. A new recurrent neural network based language model (RNN LM) with applications to speech recognition [4].

III. PROBLEM DEFINITION

In n-grams language model the calculation of occurrence of the word w that appears in the context (h), and the normalized observations of h is, expressed by

$$P(w|h) = \frac{C(h,w)}{C(h)} \qquad (3)$$

In Recurrent neural networks histories of the occurrence of h are similar, but n-grams assume exact match of h. n-grams have problems with representing patterns over more than a few words[5] .With increasing order of the n-gram model, the number of possible parameters increases exponentially. There will never be enough of training data to estimate parameters of high-order N-gram models[6] .In RNNLM the sparse history h is projected into some continuous low-dimensional space, where similar histories get clustered .Parameters sharing among similar histories is allowed in Recurrent neural network based language model. The model is more robust; fewer parameters have to be estimated from the training data .The RNN is a Generative Model. In this paper we apply this recurrent neural network based language model on Tamil language.

IV.RECURRENT NEURAL NETWORKBASED LANGUAGE MODEL

A. Recurrent Neural Networks Model Description

The RNN is similar approaches to feed-forward networks, except that recurrency between hidden and input layer is being added. RNN has an input layer x, hidden layer s (also called context layer or state) and the output layer y is shown in Fig 1. At each time, the RNN receives an input, updates its hidden state, and makes a prediction [7].

Input to the network in time t is x(t), output is denoted by y(t), and s(t) is state of the network (hidden layer) [8]. U is the matrix of weights between input and hidden layer, V is the matrix of weights between hidden and output layer. Input vector x(t) is formed by concatenating vector w representing current word, and output from neurons in context layer s at +time t − 1. Input, hidden and output layers are then computed as follows:

$$x(t) = w(t) + s(t-1) \qquad (4)$$

The hidden layer s at time t can be represented as the function f (z) taking the sum of product of the input layer u and each component of the weight vector u

$$S_j(t) = f(\sum_i x_i(t) u_{i_j} \qquad (5)$$

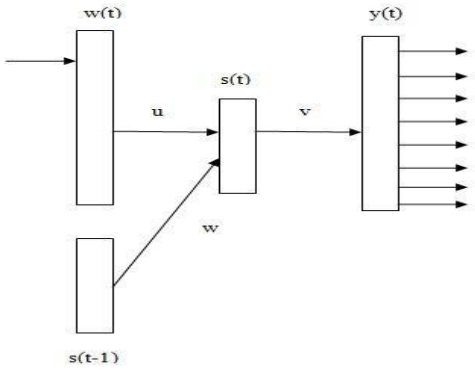

Figure 1: Simple Recurrent Neural Network

Where i - Number of input layer, j - Number of output layer $f(z)$ - Sigmoid function

$$f(z) = \frac{1}{1+e^{-z}} \qquad (6)$$

The output layer y can be represented as the function g(z) taking the sum of the product of the hidden layer s and the weight vector v

$$y_k(t) = g(\sum_j s_j(t)v_{kj} \quad (7)$$

Where the g(z) is a softmax function that makes sure the posterior probabilities sum up to 1.

B. System Architecture

Figure2 shows the architecture of Recurrent Neural Network language Model based Speech Recognition System. Initially the speech signal is converted into a sequence of feature vectors based on spectral and temporal measurements. Then the feature vector is given as input to the acoustic model. The acoustic model trains the system and calculates the posterior probability from the acoustic features [9]. This experiment uses recurrent neural network based language model for producing the probability of sequence of words from the text corpus. The recurrent neural network language model produced probability is known as prior probability. This prior probability and posterior probability from the acoustic model is given to the decoder. The Decoder finds the most probable word which is recognized word.

C. Training of Recurrent Neural Network Language Model

The training is performed using Stochastic Gradient Descent (SGD). All the training data are iteratively gone through, and the weight matrices are updated for U, V and W on line (after processing every single word).Training is performed in several epochs (usually 5-10)[10].Gradient of the error vector in the output layer e0 (t) is computed using a cross entropy criterion

$$e_0(t) = d(t) - y(t) \quad (8)$$

Where d(t) is a target vector that represents the word w(t + 1).Weights V between the hidden layer s(t) and the output layer y(t) are updated as

$$v(t+1) = v(t) + s(t)e_0(t)T_\alpha \quad (9)$$

Where α - Learning rate T - Number of time steps Next, gradients of errors are propagated from the output layer to the hidden layer

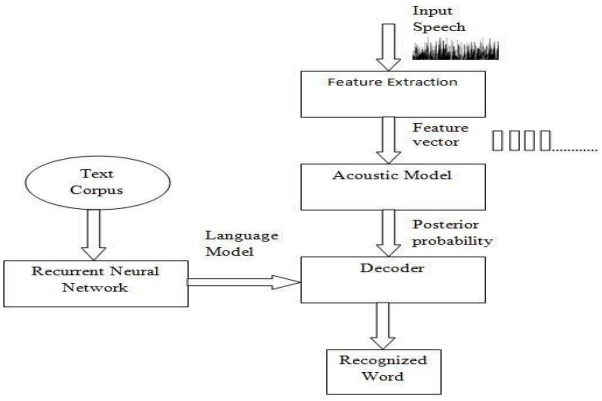

Figure 2: Architecture of Recurrent Neural Network language Model based Speech Recognition System

$$e_h(t)=d_h(e_0(t)^T V, t) \quad (10)$$

Where the error vector is obtained using function d_h that is applied by element-wise

$$d_{hj}(x,t) = xs_j(t)(1 - s_j(t)) \quad (11)$$

Weights U between the input layer w(t) and the hidden layer s(t) are then updated as

$$U(t+1) = U(t) + w(t)e_h(t)^T\alpha \quad (12)$$

Note that only one neuron is active at a given point and time in the input vector w (t). Which can be seen in (13), the weight change for neurons with zero activation is none, and thus the computation can be hastened by updating the weights that correspond only to the active input neuron.

D. Training of RNNLM Back propagation through Time

The recurrent weights W are updated by unfolding them in time and training the network as a deep feed forward neural network. The process of propagating errors back through the recurrent weights is called back propagation Through Time. (BPTT)[11]. Fig3 shows the Recurrent neural network train with three time steps back in time. Error propagation is done recursively as follows (note that the algorithm requires the states of the hidden layer from the previous time steps to be stored):

$$e_h(t-\tau-1) = d_h(e_h(t-\tau)^T W, t-\tau-1) \quad (13)$$

Where τ - current step in time

The unfolding can be applied for numerous time steps for every training data. However the error gradients quickly vanish as they get propagated back in time

(in rare cases the errors can explode), so several steps of unfolding are sufficient. The recurrent weights W are updated as

$$W(t+1) = (t) + (\sum_{z=0}^{T} s(t-z-1)e_h\,(t-z)^T\alpha \quad (14)$$

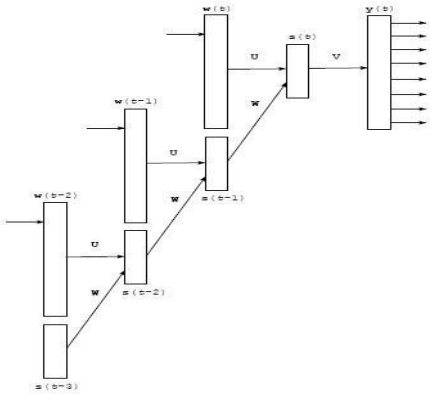

Figure 3: Recurrent neural network (here 3steps back in time)

E. Probability Estimation

The previous word $wi-1$ is fed to the input of the net using 1-of-n encoding together with the information encoded in the state vector $si-1$ from processing the previous words. By propagating the input layer we obtain the updated state vector si so that we can write:

$$P(w_i|h_i) = P_{rnn}\big(w_i\big|w_{i-1,}s_{i-1}\big)P_{rnn}(w_i|s_i) \quad (15)$$

Usually, the posterior probability of the predicted word is estimated by using a softmax activation function on the output layer, which has the size of the vocabulary. The posterior probability for any given wi can be read immediately from the corresponding output.

F. Classes

Computing complete probability distribution over all V words can be very complex. We instead assign all words from V to a single class. Compute probability distribution over words that belong to the specific class. By using classes, we can achieve speedups on large data set more than 100 times. The fig4 shows the factorization of an output layer along with class layer c(t).

G. Joint Training with Maximum Entropy (ME) Model

With increasing amount of training data, we need to increase size of the hidden layer to secure maximum performance .By joint training of RNNLM with a maximum entropy model. We can afford to have much smaller hidden layers. The ME model can be viewed as a direct weight matrix between the input and the output layers.

H. Rescoring

Further accuracy improvements are obtained by running a RNN rescoring [12]. In n-best list rescoring, the RNN model re-estimated a log-likelihood score for each n-best hypothesis.

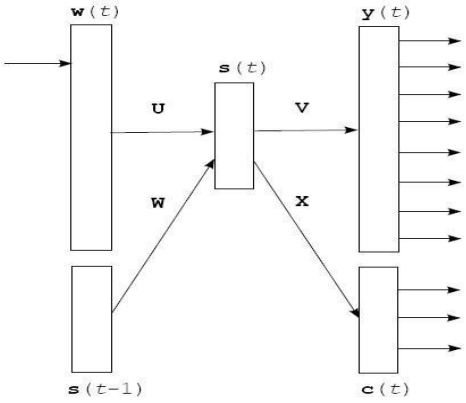

Figure 4: Output layer with class layer c (t)

Further accuracy improvements are obtained by running a RNN rescoring [12]. In n-best list rescoring, the RNN model re-estimated a log-likelihood score for each n-best hypothesis

$$\log L(s) = n.wp + \sum_{i=1}^{n} asc_i \sum_{i=1}^{n} \log p_x\,(w_i|h_i) \quad (16)$$

Where *n*- Number of words, wp- Word insertion penalty, asc_i-Acoustic score for word, hi - The history $w1,\_wi$, *lms* - The language model scale applied in the generation of the input lattices.

P_X-Combined probability estimate of standard 4-gram and RNN models, which is obtained by linear interpolation.

$$P_x(w_i|h) = \lambda P_{rnn}(w_i|h) + (1-\lambda)P_{ng}(w_i|h) \quad (17)$$

Where λ- Set to some value so that that this probability distribution sum to 1.

V.EXPERIMENTAL SETUP

The experiment used in the LDC-IL Tamil text corpus.80% of the data is used for training and the remaining data is used for testing. For training the recurrent neural network based language model we need to specify the parameters hidden layer size, number of class and BPTT. The hidden layer size depends on the size of the training data for less than 1M words, 50-200 neurons for 1M-10M words use 200-300 neurons. Here the experiment is done using

40 hidden neurons. Using larger hidden layer usually does not degrade the performance, but makes the training progress as slower. Here the BPTT is specified as three, will back propagate error through recurrent connections for two steps back in time [12]. Class value should be 200 for speed up training progress.

VI. EXPERIMENTAL R E S U L T

In this section we describe our experimental results that were obtained. The performance of any language model is measured in terms of perplexity; word error rate (WER).Perplexity is a measure of how well the model predicts the occurrence of words in a given text. The perplexity of a language model X is given by *N*

$$PP(X) = -\frac{1}{N}\sum_{i=1}^{N} \log P\left(w_i \middle| H_{q-1}\right) \quad (18)$$

Where N - Total number of words in the test set. Perplexity also indicates the average branching produced by the language model X. WER can be computed by the equation.

$$WER = \frac{S+D+I}{N} \quad (19)$$

Where *S*-Number of substitutions, *D* - Number of the deletions, *I* - Number of the insertions, *N*-Number of words in the reference.

TABLE 1: COMPARISON RECURRENT NEURAL NETWORK LANGUAGE MODEL AMONG DIFFERENT LANGUAGE MODELING TECHNIQUE

Language model techniques	**Speech Corpus**	**WER**
N-gram language model	LDC-IL	27.1%
Feed Forward Neural	LDC-IL	25.6%
Recurrent Neural Network	LDC-IL	24.9%

Thus, Word Error Rate (WER) indicates how well the model performs for a speech recognition task. Generally, the relation between word error rates of a speech recognizer is lower. Perplexity usually translates to lower word error rates. We have performed three experiments in each of N-gram, Feed forward and, recurrent neural networks. The table 1 shows the word error rate and perplexity obtained by the experiments.

VII.CONCLUSION

We propose that the recurrent neural network model outperforms the n-gram model on various datasets including tamil .For a good model of language, meaningful sentences should be more likely than the ambiguous ones .RNN LMs can generate much more meaningful text than n-gram models trained on the same data .However, in our experiments we have achieved almost twice perplexity reduction over n-gram models by using a recurrent neural network based language model. Thus from the practical point of view, the main problem is to perform fast training of these models on very large corpora. Despite its simple design, the RNN can be used to train very good RNN language models on corpora with hundreds of millions of words.

ACKNOWLEDGMENT

We wish to express our gratitude to the Linguistic Data Consortium – Indian Languages for providing the standard LDC-IL text Tamil corpus with transcription.

REFERENCES

[1] M.M.El Choubassi,H.E.El Khoury, C.E.Jabra Alagha, J.A.Skaf and M.A.Al-Alaoui, *Arabic Speech Recognition Using Recurrent Neural Networks*,,IEEE transaction 2003.

[2] Yoshua Bengio,Rejean Ducharme,Pascal Vincent,Christian Jauvin, *A Neural Probabilistic Language Model*,,Journal of Machine Learning Research 3 (2003).

[3] Hirofumi Yamamoto,Shuntaro Isogai,Yoshinori Sagisaka, *Multi-Class Composite N-gram Language Model for Spoken Language Processing Using Multiple Word Clusters* ,ACL'01 Proceedings of the 39th Annual Meeting on Association for Computational Linguistics 2001.

[4] Tomas Mikolov,Martin Karafia,LukasBurget1,Jan Honza Cernocky,Sanjeev Khudanpur, *Recurrent neural network based language model* ,INTERSPEECH 2010.

[5] Andriy Mnih, Geoffrey E.Hinton, *A Scalable Hierarchical Distributed Language Model*,,NIPS 2008.

[6] Jianfeng Gao,Hisami Suzuki,and Yang Wen,*Exploiting headword dependency and predictive clustering for language modeling*, ,ACL-02 conference on Empirical methods in natural language processing
,2002.

[7] Ilya Sutskever,James Martens,Geoffrey Hinton, *Generating Text with Recurrent Neural Networks*, ,ICML 2011.

[8] Tomas Mikolov,Martin Karafia,Lukas Burget1,JanHonza Cernocky,Sanjeev Khudanpur, *Variational approximation of long-span language models for LVCSR*, ,IEEE Signal Processing Society 2011.

[9] M.A.Anusuya,S.K.Katti, *Speech recognition by machine: A Review*,,IJCSIS 2009.

[10] Tomas Mikolov,Anoop Deoras,Daniel Povey,Lukas Burget,Jan Honza Cernocky,*Strategies for Train- ing Large Scale Neural Network Language Models*,IEEE Signal Processing Society 2011.

[11] Tomas Mikolov,Stefan Kombrink,Anoop Deoras,Lukas Burget,Jan Honza Cernocky , *RNNLM - Recurrent Neural Network Language Modelling Toolkit*,,IEEE -ASRU 2011.

[12] Stefan Kombrink,Tomas Mikolov,Martin Karafiat,Lukas Burget *Recurrent Neural Network based Language Modelling in Meeting Recognition*,,International Speech Communication Association 2011.