An Acoustic Recognition End-to-end System on Under-resources Languages

Chun-Hao Wang
Department of Computer Science
National Taiwan University
chuchuhao831@gmail.com

Kuan-Hou Chan
Department of Computer Science
National Taiwan University
iepiechan@gmail.com

***Abstract*—With the rapid development of sequence learning techniques, the state-of-the-art speech recognition system have a great performance on English speech corpus with almost human ability. Compare to others under-resourced language which is less language-specific knowledge or less hand-labeling data is still an active field for research and still have no good performance. Instead to find a system can make a good performance on some language recognize problem, we present a end-to-end system that directly translate the acoustic sequence to label sequence without any language-specific knowledge so that we can adapt on any language without much resources. An experiment on the Cebuano speech corpus demonstrates its ability indeed learning some pattern from data.**

***Index Terns*—** under-resources language, minority languages, connectionist temporal classification (CTC), sequence learning, speech and language resources, automatic speech recognition (ASR), cross-lingual acoustic modeling.

I. Introduction

Speech recognition problem takes advantage of recent advance in algorithms and computer hardware, using the artificial neural networks (ANNs) techniques acquire a good performance on the task. According to the Grave's research[2],*a end-to-end system achieves a word error rate of 27.3% on the Wall Street Journal corpus with no prior linguistic information, 21.9% with only a lexicon of allowed words, and 8.2% with a trigram language model. Combining the network with a baseline system further reduces the error rate to 6.7%.* Not only to this end-to-end system acquire that perfect result, there also some graphical models such as hidden Markov Models (HMMs; Rabiner), conditional random field (CRFs; Lafferty) hybrid with ANNs and the aggregation of different models with additional language-specific feature, all of them do the great job. While all of these approaches performed excellent, but they usually have to use lots of language resources which from large hand-labeling acoustic sequence to label sequence pair to some knowledge like phoneme and feature theory except the Grave's RNN-CTC system. Grave's system can achieves low error rate in English corpus without prior linguistic information, and we assume such characteristic which regardless of linguistic information let us be able to apply the system on some other language.

So far, the speech recognition problem on under-resourced language is still an active field for research, and we expect that if we could apply a system without the language-specific knowledge (like the Grammer, Grapheme and Phonology) and less labeling data then we create a system as the speech-to-text machine on all the language in the world. We believe we can break the language barrier for human communication.

To define clarity, we seem this as a sequence learning problem which labeling unsegmented sequence data or to say unsupervised learning the temporal classification[4]. Input a time various speech sequence data and output a sequence of label, for example, we take the filter-bank extract acoustic features from the speech signal with a fixed frame size as input and turn them into a sequence of labels which is English Character as output.

In our experiment we use bidirectional long short-term memory as our neural unit with their powerful, general mechanism for modelling time series and connectionist temporal result (CTC) output to let the RNN-alone sequence learning become possible.

The next section introduce bidirectional long short-term memory (BLSTM) and connectionist temporal classification (CTC) and briefly describe how them be trained, output representation and build our recognition system. Section 3 describe the Cebuano speech corpus we use in our experiment. Section 4 present our experiment result and discuss them. Section 5 makes a conclusion and provide list of further work we are on the process to solve this problem.

II. Recognition system architecture

Artificial neural network is a box that take an input and return an output target. The architecture inside is a single or multiple layer with each layer consist a list of neurons. All these neurons and the weighted connection between the layer build that box. To let our box can learn the ability to response the correct output a we want, we define a proper objective function that represent the score of it's learning outcome. We try to adjust the connection weights to get the better score. Here, we are not going the detail of mathematical formalism for the reason that all of them can be found in the reference[4], what here is just some my own reviews of this box architecture. Last we combine the CTC output layer to the network and train them with the log-likelihood to let the system learn from sequences to sequences not only frames to frame. The reason we use bidirectional long short-term memory as our neurons is explain and following subsection and what CTC do in section 2-B.

A. *Bidirectional long short-term memory*

A standard recurrent neural network (RNN) computes the hidden vector sequences and output vector sequence by an input sequence, with it's cyclic architecture in neural connection it reveal the ability to memorize the temporal information for a better modeling. However researcher have found that Long Short-Term Memory (LSTM) architecture [5], which uses purpose-built memory cells to store information, is better at finding and exploiting long range context and solve some training problem with standard RNN[6]. Since the standards RNNs process sequence in temporal order, they do not use the future context which is obviously important in sequence learning system. There are two method to solve this, one is to add a time-window of the future context to the network input, for example, at time t, in standard RNNs, we take only data at time t, but with time-window, we concatenate t and t+1 data as input, which double the dimension and use the future data. However, as well as unnecessarily increasing the dimension of input weights, this suffer from the intolerance of distortions. Bidirectional recurrent networks (BRNN)[7] offer a more elegant solution. The basic idea of Banns is to present each training sequence to forward and backward to two separate recurrent hidden layers, both of which are connected to the same output layer. Briefly speaking, standard RNNs train temporally from time beginning of the sequence to the end, on the contrary, BRNNs train not only forward time ordering but also the backward time ordering to get the future context. Last we use the deep bidirectional LSTM[8] which means we take all advantage from the BRNNs, LSTM and deep structure in Artificial Neural Network as our system to do the speech recognition task.

B. *Connectionist temporal classification*

Although the BLSTM acquire a huge success on frame-level classifier in speech recognition, that still not solve our problem which is a sequence to sequence learning. Instead of training separate target for every frame and do the alignment between the audio and transcription sequence by the some other method, CTC provides an objective function that allows an RNN to be trained directly for sequence transcription tasks without any alignment. CTC works like a standard RNNs with HMMs system, the biggest difference compare to the hybrid system CTC train them together. The architecture of system is a standard RNN concatenation with a CTC output. There's two thing in CTC output layer, one is finding the label as decoding in HMMs; list of labels emission probability (used the same terminology in HMMs) we get by the standard RNNs then we find a best path, which is the most probability sequence we can get from a dynamic programming. Another, update the transition probability just like the forward-backward algorithm in HMMs. With CTC output layer we can get the target sequence. This let us define a objective function via principle of maximum likelihood and its derivatives with respect to the network outputs, the weight gradients can be calculated with standard backpropagation through time.

III. Cebuano data processing

Cebuano is an Austronesian language spoken in the Philippines by about 20 million people. It has the largest native language-speaking population of the Philippines despite not being taught formally in schools and universities. There are some linguistic research about the language but we dose not use in this project. The reason we choose Cebuano is that we know nothing about the Cebuano expect a 3 hours speech data with their target label. Each data pair represent a speech slice consist a sequence of speech data and a correspondent sequence of labels(character). The following section 3-A we describe how we process the speech data, ans some observation in Cebuano target and what we handle them in section 3-B.

A. *Processing speech data*

There's two ways we process the speech to our system input, one is 30-dimension mel-frequency cepstral coefficients(MFCCs) and 90-dimension DNN trained bottleneck feature. Both of these two front-end feature extract we use in our experiment. The MFCCs approach is the traditional ways to do the front-end job. The bottle-neck feature is a pre-MLP extract feature[9][10]. For the limitation of computing power, only those sequence within 500 frames was retained. Almost 100 sequence dropped in this process respect to the final number of sequence, which is 3917, for our training. Approximate 2% of data not being used. Same process on testing data, and number of them is 4157, 2% drop also.

B. *Observation in Cebuano target*

The difference between the training and testing target file, which are sequences of character as the training target, can be separate discuss in two aspect: characters and words. On the character view, there are 32 different characters includes English alphabet (a-z) and specific character (” ”, ””’”, ”-”, ”_”, ”.”, ”?”), the ” ” represent the space, ”.” is the blank (for CTC output) and ”?” means some noise that have no meaning. We don't know the usage of all other specific character, but we think they are important and should be include. These characters we names them the labels. On the words side, we do not know exactly what is a word in Cebuano but simply extract a word when a sequence of label meet the *space*. There are 3733 different words in training data and 3651 different words in testing data. But, the overlap of them only got 1506 words.

IV. Experiment results

In our experiment, the implementation of Bi-LSTM is using Lasagne-nntools[12] and the implementation of CTC is using mohammadpz/CTC-Connectionist-Temporal-Classification[13]. We are using single GTX-980 for our GPU acceleration. The architecture we use in our experiment have two different input model, one is 30-dimension MFCC abbreviate to MFCC, another is 90-dimension bottleneck feature abbreviate BNF. The dimension of output layer is 32. We only count the label error rate(LER), the edit distance between predict sequence and target sequence. All networks were trained with RMSprop using s learning rate 10^{-4} and a momentum of 0.9. Gaussian weight noise[11] with a standard deviation of 0.075 was injected during training to reduce overfitting. Number of neurons in each hidden layer is 250 except one with 4 hidden layer. Fig.1 and Fig.2 illustrate our learning result.

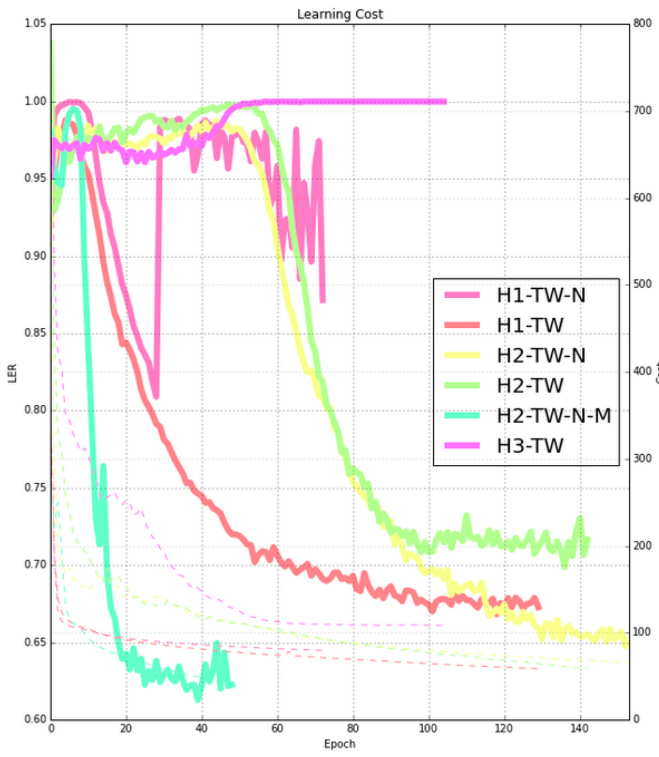

Fig. 1. Training Result with different model under time-window, the prefix number of H represent the number of hidden layers, TW is the teime window, N and M abbreviate to Gaussian weight Noise and Momentum. Color lines are corresponding to the left y-axis, the label error rate on validation data, dot lines are the CTC cost during training process, scale them on right y-axis. The x-axis record down the epoch of each model, we did not train the same epoch numbers for every model but stop them when the CTC cost start saturate.

At the beginning of our training, we use MFCC feature with concatenation the previous three and following three at each frame for the larger input size. Different model with and without the Gaussian noise. According to the figure 1, system can't learn anything if the input layers is larger then 3, so we do not put the layer larger then 3 in our figure. With only one hidden layer, model without noise seems better then the one with, however, this statement is not true with two hidden layer. Out best result with time-window is the 2 hidden layer model with noise, momentum and clipping, the label error rate is 0.62. Here we add clipping to prevent the too large or too small value when updating the parameter of our model[6] but we have not verify whether it should use with LSTM neurons.

Later of our training, due to the lots of paper mentions that as well as the distorted data being used, time-window approach suffer from the fact that the range of useful context is generally unknown. Comparison between whether use the time-window propose in fig. 2. Evidence indicate that there are not obviously progress or regress in our model. For there's no progress in training, there's no reason to use the time-window as our input. Last we try the bottleneck feature without time-window on our training. At first we got a huge success on label error rate of validation data, then we try the deeper model which 3 and 4 hidden layers, but all of them can't learn anything (the label error rate saturate around 1). Finally we reduce the number of neurons in each hidden layer and it start learn from the data. Three models we pick to do the experiment on test data, the H2-M-N model with LER is 0.62, the H2-M-BNF with LER is 0.44 and the H4(100)-M-BNF is 0.45. Three results are 0.71, 0.68 and 0.66 which almost have no difference. Additional beam-search is implement for test the word error rate (WER). Instead of combine language model, we try all 2000-best result to calculate WER, but still get nothing, so we stop our experiment on here.

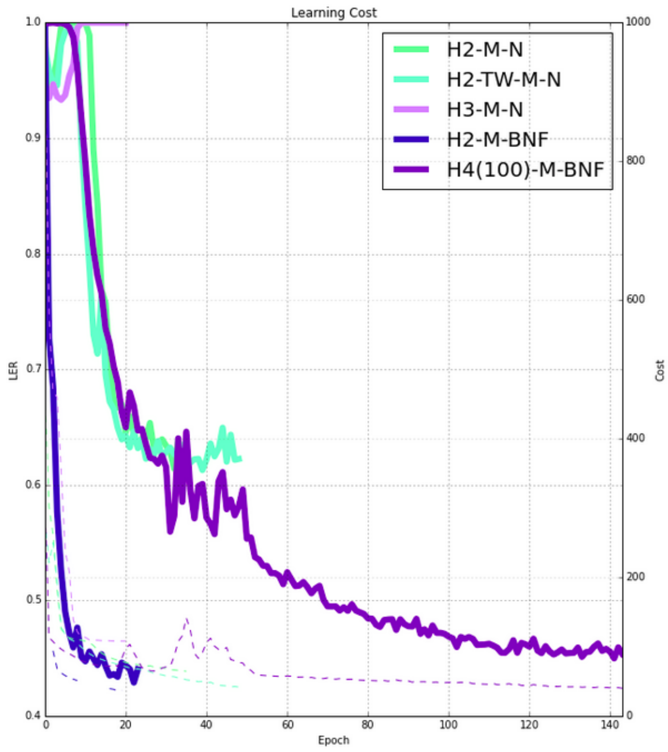

Fig. 2. Training Result without time-window, the TW, H, M, N is the same abbreviation as Fig. 1. New term BNF abbreviate to bottle-neck feature. There's (100) behind the H4 to remind that each hidden layer in that model only get 100 neurons instead of 250. Notice that the BNF model all train with Gaussian noise.

V. Discuss and Conclusion

According to our experiment result, there is no good enough label error rate for us to predict the word sequence and we stuck our score on testing data around 70% even the bottleneck feature exhibit the better score on training data. The first observation is that when we increase the number of total weight, the whole system become untrainable even with enough epoch, we attribute the untrainable evidence much to the less of training data. Second, them seems a overfitting on the bottleneck feature, however, we had separate the speaker on validation set, so, there's no reason to perform the problem, the overfitting most probably due to the extracting process. Although the experiment results is not indicate that the system is good enough to do a speech recognition test. The provided character-level transcription on the low-resource language, the RNN-CTC system have being proof that it indeed learn something on the language. The following examples is randomly picking decoding sentence in our Model.

The validation data with label error rate = 0.44
Target: o ngano diay muuban ka
Output: o manga diay mobanka
Target: ? ka didto sigi ra og tindog
Output: h bai to giy ngtndog
Target: ay usahay lang dili' man pod siya ingon nga dalo' piro
Output: ay salang wa' po sa ingaon dalo piro3

The testing data with label error rate = 0.71
target: mga pila na kadto katuig wala' pa siya katrabaho mas nauna pa gani' siya og graduate gani'
output: mangatila napo katoli ula a sa katrba bassawu na sa radde nadi
target: sa imo nga papa wala' diay ka na na
output: naimo ng pata wala diin ta nana
target: ? si lemon
output: idi

First glance at those data, obviously seem that almost all the sentence learn some pattern from the system. Nonetheless we still discover some sentence that learn nothing at all. On the testing result, we found that there are many sentence get the label error rate around 40%, at the mean time, also mant sentence get ehe LER around 1. Huge variance on LER but we have no good explanation except the not sufficient data. For the experiment here, we depart the word meaning on Cebuano language, so we don't think the less word overlap affect our result. In conclusion, result shows that even on under-resources language an end-to-end system is a good method to do the acoustic recognition task and we can keep going study how fix the system to get the better result.

Acknowledgment

Special thanks to our professor Hung-Yi Lee, who open a course named *Machine Learning and having it deep and structured (2015,Spring)* in National Taiwan University, all of my knowledge about neural network are learning from the class and project also inspired by the final project in that course which is build an ASR system on English.

References

[1] L. Besacier, *Automatic Speech Recognition for Under-Resourced Languages: A Survey*
[2] A. Grave, *Towards End-to-End Speech Recognition with Recurrent Neural Networks*
[3] A. Grave, *Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Network*
[4] A. Grace, *Supervised Sequence Labelling with Recurrent Neural Network*
[5] S. Hochreiter, *Long Short-term memory*
[6] R. Pascanu , *On the difficulty of training recurrent neural networks*
[7] M. Schuster , *Bidirectional Recurrent Neural Networks*
[8] A. Grace, *Bidirectional LSTM Networks for Improved Phoneme Classification and Recognition*
[9] F. Grezl, *Probabilistic and Bottle-Neck Feature for LVCSR of Meetings*
[10] S. Tomas, *Deep Neural Network Features and Semi-supervised Training For Low Resource Speech Recognition*
[11] Jim, *An Analysis of Noise in Recurrent Neural Networks Convergence and Generalization*
[12] Lasagne, craffel/nntools, *https://github.com/craffel/nntools*
[13] mohammadpz/CTC-Connectionist-Temporal-Classification, *https://github.com/mohammadpz/CTC-Connectionist-Temporal-Classification/tree/no_underflow*