DISCRIMINATIVE TRAINING FOR IMPROVING LETTER-TO-SOUND CONVERSION PERFORMANCE

Yi-Ning Chen[1], Peng Liu[1], Jia-Li You[1,2], and Frank K. Soong[1]*

[1]Microsoft Research Asia, Beijing, China
[2]Institute of Acoustic, Chinese Academy of Sciences, Beijing, China
[1]{ynchen, pengliu, frankkps}@microsoft.com, [2]youjiali@mails.gucas.ac.cn

ABSTRACT

In this paper, we propose to use discriminative training (DT) for improving Letter-to-Sound (LTS) conversion performance. LTS is a critical component in both ASR and TTS for predicting the correct pronunciation of a word not included in the lexicon. For TTS applications, predicting the proper pronunciation of an out-of-vocabulary person/place name, especially a name with foreign origin can be challenging. We utilize discriminative training, which has been successfully used in speech recognition, to sharpen the baseline N-grams of grapheme-phoneme pairs. We address the problem in a unified framework of discriminative training. Two criteria, Maximum Mutual Information (MMI) and Minimum Phoneme Error (MPE), are investigated. Experimental results show that DT yields a small (3.8-4.6% relative) but consistent error reduction across all databases tested. In addition, we observe that by pinpointing the local errors in a finer resolution, we can obtain a better discriminative model.

Index Terms— Discriminative Training, Letter-to-Sound, Graphoneme

1. INTRODUCTION

The letter-to-sound conversion, also known as grapheme-to-phoneme or spelling-to-pronunciation conversion, is to predict the pronunciation of a given word strictly from its spellings. The LTS conversion is very important for handling out-of-vocabulary (OOV) words for both automatic speech recognition and TTS synthesis.

Currently, speech or language researchers has put more intensive effort into large-scale tasks, such as training acoustic model with thousands-of-hours data [1] or training language model with 7-gram and millions of words [2]. Predicting the pronunciation of words that are not in the given dictionary is then critical. In many applications, OOV can be a serious problem when the LTS performance is not good. For example, in a directory assistance task, the name list in yellow pages, many words can be OOV.

There are many reported research work in LTS [3-12]. Their algorithms can be roughly divided into non-metric methods (including rule based algorithm and decision tree based algorithm) and statistical methods.

A good review of non-metric method can be found in [3-5]. We can separate the approaches to two categories. The first and the oldest one is to use manually written rules. These rules can be written by system developers or language experts. The second one is data-driven learning algorithms, such as Pronunciation by Analogy (PbA) [3], decision tree [6, 7], and Transformation Based Learning (TBL) [5]. These algorithms have been adopted in many applications. The footprint of the model is in general small and the LTS decoding is fast.

Statistical methods use N-gram language models. Based on the way it is implemented, it can be divided into phoneme N-gram based algorithm [4, 8] and grapheme-phoneme pair (graphoneme) N-gram [8-12]. Although the statistical algorithms are relatively more complex and slower in decoding than the rule based ones, their performances tend to be better [5, 9]. In this paper, we also use graphoneme N-grams as our baseline system.

Discriminative training is widely used in speech recognition [13-18]. Algorithms like Minimum Classification Error (MCE) [13], MMI [14], MPE [15], and Minimum Divergence (MD) [16] are used to train acoustic model for better acoustic discriminations. In the language model part like N-gram, there are also some pioneering work. Chen et al use DT in Input Method Engine (IME) [17]. Guo et al use DT to improve the N-gram language model for speech recognition [18]. Recently, various criterions are viewed in a unified framework which is optimizing accuracy weighted posterior probability, and several optimization algorithms are proposed and proven effective in dealing with the framework [15, 16].

[*] This work is done when Jia-Li You visits Microsoft Research Asia as an intern

The success of discriminative training in language model motivates us to try it on improving N-gram based LTS module. We address this problem and come up with concrete algorithm to discriminatively train graphoneme base N-grams in the sense of minimum error training. We test these algorithms in different databases, starting with a well-tuned baseline system. Consistent improvements are obtained among the corpuses. The relative error reduction is from 3.8 to 4.6%. In addition, we observe the effect of localizing errors from word level to phone level for helpful guidelines in our future work.

The structure of this paper is as following. In Section 2, we review our baseline system, or the grapheme-phoneme pair N-gram model. In Section 3, we present our discriminative training of graphoneme N-grams in detail. In Section 4, we introduce the experimental setups and evaluation results of our discriminative training. Finally, we present our conclusions in Section 5.

2. GRAPHONEME N-GRAM MODEL

In this section, the baseline LTS system built upon graphoneme N-gram model is introduced. The same N-gram model has been adopted by other researchers to obtain state of the art LTS performance [5, 9-12].

Given a word *w*, we have a spelled letter sequence $\boldsymbol{L}_1^N$ and a phoneme sequence $\boldsymbol{Q}_1^M$ for the corresponding pronunciation. In it, *N* is the total number of letters and *M* is the total number of phonemes of the corresponding word.

We need to train a statistical model, $P(\boldsymbol{Q}_1^M \mid \boldsymbol{L}_1^N)$, to characterize the relation between letters and phonemes of a word. In most languages, the mapping between letters and phonemes is not strictly one-to-one. Instead, a many-to-many mapping needs to be statistically established from data. Give an alignment, the word can be modeled by a grapheme-phoneme (graphoneme) string $\boldsymbol{G}_1^E$. E is the total number of graphoneme. Given the alignment, $\boldsymbol{G}$ is determined by $\boldsymbol{Q}$ and vice versa.

The probability of $\boldsymbol{Q}$ given $\boldsymbol{L}$ is shown in eq. (1). In it, the *H* is the space of all phone hypotheses for a given word W.

$$P(\boldsymbol{Q} \mid \boldsymbol{L}) = \frac{P(\boldsymbol{Q},\boldsymbol{L})}{\sum_{\boldsymbol{Q}\in H} P(\boldsymbol{Q},\boldsymbol{L})} = \frac{P(\boldsymbol{G})}{\sum_{\boldsymbol{G}\in H} P(\boldsymbol{G})} \tag{1}$$

In decoding process, the criterion is to finding the phone sequence $\boldsymbol{Q}^*$ that maximizes $P(\boldsymbol{Q} \mid \boldsymbol{L})$. It is equivalent to maximize $P(\boldsymbol{G})$ as (2).

$$\boldsymbol{Q}^* = \arg\max_{\boldsymbol{Q}\in H}\left(P(\boldsymbol{Q} \mid \boldsymbol{L})\right) = \arg\max_{\boldsymbol{G}\in H}\left(P(\boldsymbol{G})\right) \tag{2}$$

An example of the word "phone" is shown as follows:

Word: phone
Letter: p h o n e
Phoneme: f ow n

After the alignment, we get the graphoneme sequence,

Graphoneme: ph:f o:ow n:n e:#

where we use # to denote the NULL (phoneme).

In the case of N-gram,

$$\boldsymbol{Q}^* = \arg\max_{\boldsymbol{Q}\in H}\left(P(\boldsymbol{G})\right) = \arg\max_{\boldsymbol{Q}\in H}\left(\prod_{e=1}^{E} P(g_e \mid \boldsymbol{G}_{e-1}^{e-N+1})\right) \tag{3}$$

Here $\boldsymbol{G}_{e-1}^{e-n+1}$ is the history of g_e. We use $\boldsymbol{\theta}$ for the whole set of model parameters where each element corresponds to the statistical N-gram model of a graphoneme string, $\boldsymbol{G}_e^{e-N+1}$.

2.1. Training

We complete each graphoneme sequence for a word by adding the start <G> and end </G>. Based on these sequences, a graphoneme N-gram model is estimated with back-off smoothing and absolute discounting [19].

2.2. Decoding

The decoding is to find $\boldsymbol{Q}^*$ that maximizes the likelihood in equation (3). In order to estimate this probability efficiently we use a Viterbi search algorithm.

3. DISCRIMINATIVE TRAINING OF N-GRAM

In this section, we apply the framework of minimum error training to the graphoneme N-gram models.

3.1. Unified view of minimum error training

Recently, various discriminative training criteria are formulated in a unified framework to minimize posterior weighted errors at different unit levels [15, 16].

$$F(\boldsymbol{\theta}) = \sum_r \left\{ f\left[\sum_{\boldsymbol{W}\in H_r} P_\theta(\boldsymbol{W} \mid \boldsymbol{O}_r) A(\boldsymbol{W}_r, \boldsymbol{W}) \right] \right\} \tag{4}$$

where $\boldsymbol{\theta}$ denotes the set of all model parameters; *r* represents the index of training token; $\boldsymbol{O_r}$ and $\boldsymbol{W}_r$ denote the corresponding observation and the reference, respectively; $\boldsymbol{W}$ is a hypothesis in the search space $H_{\boldsymbol{r}}$.

In the above equation, $P_\theta(\boldsymbol{W} \mid \boldsymbol{O}_r)$ is the generalized posterior probability of hypothesis $\boldsymbol{W}$ given $\boldsymbol{O}_r$. $A(\boldsymbol{W}_r, \boldsymbol{W})$ is an accuracy measure of a $\boldsymbol{W}$ given its reference $\boldsymbol{W}_r$ and *f* is a smoothing function. For pattern classification problems with an "elastic" scale, the design of accuracy term is not trivial. Actually, it can be defined globally at the sentence level or be localized to either word or phone levels. For example, configurations of several popular discriminative training criteria in speech recognition are listed in Table 1.

Table 1. Comparison between minimum error training criteria.

	f	$A(\boldsymbol{W}_r, \boldsymbol{W})$
MMI	log	$\delta(\boldsymbol{W} = \boldsymbol{W}_r)$
MCE	1	$\delta(\boldsymbol{W} = \boldsymbol{W}_r)$
MPE	1	$\lvert\boldsymbol{W}\rvert - \mathrm{LEV}(\boldsymbol{W}, \boldsymbol{W}_r)$

Here, $LEV(\ ,\ )$ is the Levenshtein distance between two symbol strings; $|\cdot|$ is the number of symbols in a string.

For training letter-to-sound language models, eq. (4) can be written as:

$$F(\boldsymbol{\theta}) = \sum_r \left\{ f\left[\sum_{\boldsymbol{Q} \in H_r} P_\theta(\boldsymbol{Q} \mid \boldsymbol{L}_r) A(\boldsymbol{Q}_r, \boldsymbol{Q}) \right] \right\} \quad (5)$$

According to eq. (1), it can also be rewritten as:

$$F(\boldsymbol{\theta}) = \sum_r \left\{ f\left[\sum_{\boldsymbol{G} \in H_r} P_\theta(\boldsymbol{G}) A(\boldsymbol{G}_r, \boldsymbol{G}) \Big/ \sum_{\boldsymbol{G} \in H_r} P_\theta(\boldsymbol{G}) \right] \right\} \quad (6)$$

In practice, the N-Best decoded results can be used for representing the hypothesized search space H.

To investigate the effect of error resolution, we compare MMI and MPE in our work. Correspondingly, the errors are defined at word and phone level, respectively.

3.2. Optimization algorithm

Given the objective function in (6), we optimize it with respect to the LTS model parameter set θ. Each element of it corresponds to the N-gram of a string, $\boldsymbol{G}_e^{e-N+1}$. In this work, we adopt the gradient descent approach, which adjusts the model in an iterative manner:

$$\theta_{t+1} = \theta_t - \varepsilon \frac{\partial F}{\partial \theta} \quad (7)$$

θ_{t+1} is the parameter of next iteration, θ_t is the parameter of the current iteration, ε is the step size, and $\frac{\partial F}{\partial \theta}$ is the gradient. The derivative can be calculated in equations (8), (9), and (10).

$$\frac{\partial F}{\partial \theta} = \sum_r \left\{ \frac{\partial f(d)}{\partial d} d \left(\frac{\sum_{\boldsymbol{G} \in H_r} \left(\frac{\partial P_\theta(\boldsymbol{G})}{\partial \theta} A(\boldsymbol{G}_r, \boldsymbol{G}) \right)}{\sum_{\boldsymbol{G} \in H_r} \left(P_\theta(\boldsymbol{G}) A(\boldsymbol{G}_r, \boldsymbol{G}) \right)} - \frac{\sum_{\boldsymbol{G} \in H_r} \frac{\partial P_\theta(\boldsymbol{G})}{\partial \theta}}{\sum_{\boldsymbol{G} \in H_r} P_\theta(\boldsymbol{G})} \right) \right\} \quad (8)$$

Where,

$$d = \sum_{\boldsymbol{G} \in H_r} P_\theta(\boldsymbol{G}) A(\boldsymbol{G}_r, \boldsymbol{G}) \Big/ \sum_{\boldsymbol{G} \in H_r} P_\theta(\boldsymbol{G}) \quad (9)$$

$$\frac{\partial P_\theta(\boldsymbol{G})}{\partial \theta} = P_\theta(\boldsymbol{G}) I(\theta) \quad (10)$$

$I(\theta)$ is the count of the graphoneme string, $\boldsymbol{G}_e^{e-n+1}$, occurred in a training word token.

4. EXPERIMENTAL RESULTS

4.1. Experiment settings

We used three databases (or pronunciation dictionaries), NetTalk, CMUDICT and NAME. The NetTalk dictionary is publicly available and was originally developed and manually aligned by Sejnowski and Rosenberg for training the NetTalk neural network [20]. It has 19,802 word entries. Some words are with multiple pronunciations and the total number of entries is 20,008. A one-to-one alignment between the letter string and the phoneme string is given in the dictionary. The CMU dictionary consists of more than 119,000 words, with corresponding more than 127,000 pronunciations [21]. NAME is a proprietary, name dictionary of 83,042 words. For each corpus, we use 90% data for training and the rest 10% for testing. In the selecting, each lexicon is alphabetically ordered and every 10th sample is selected to form the testing set.

4.2. Result of Baseline systems

In the baseline system, several parameters are investigated, including: the order of N-gram, the smoothing algorithm, the cut-off threshold and unit numbers. We used 4-grams as the language model, Knesser-Ney algorithm for smoothing [19]. We tuned other parameters according to different corpus, and got the results shown in table 1. The performances are comparable with other state of the art LTS systems [6,7,9,10].

Table 1. Baseline result of letter-to-sound.

	Word Error Rates
NetTalk	34.3%
CMUDICT	26.7%
CMUDICT(*)	34.0%
NAME	55.2%

Here, CMUDICT(*) refers to the fact that stressed vowels are marked in the dictionary. Among all databases, NAME gets the worst error rate, which is consistent with the discovery made by Black [6]. In their research, 79% of LTS errors of CMUDICT are from names.

4.3. The iteration curve of training and testing set

The word error rate evolution of training and testing sets in the MPE-based discriminative training procedure is shown in Fig. 1. The NAME is used as the training and testing corpus.

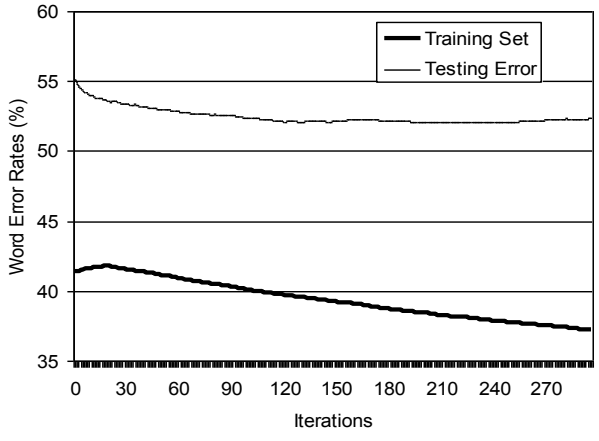

Figure 1. Word error rates of each iteration using MPE.

From the curve, we can observe that the word error rate of the training set decreases steadily with more iterations.

However, the error rate of the testing set decreases only in the beginning and then saturates.

4.4 Discriminative training results in different corpus

Table 2. Error rates and relative error reductions of discriminative training.

	Error Rates / Relative Error reduction		
	Baseline	MMI	MPE
NetTalk	34.3%	32.7% / 4.7%	32.9% / 4.1%
CMUDICT	26.7%	25.7% / 3.7%	26.0% / 2.6%
CMUDICT(*)	34.0%	32.7% / 3.8%	32.4% / 4.9%
NAME	55.2%	53.4% / 3.3%	52.0% / 5.8%
Average	37.6%	36.1% / 3.8%	35.8% / 4.6%

From these results, we can find that small (3.8-4.6%) but consistent improvements are obtained in the discriminative training under different training criteria (MMI and MPE) and across different databases. It should be noted that the baseline LTS module has been continuously tuned to get the best possible performance while the discriminative training of LTS N-grams is just our first attempt. We also tested all corpuses with different baseline settings and we obtained some consistent improvements after discriminative training. On average, MMI training yields a relative 3.8% reduction of errors and MPE training yields a slightly better, 4.6%, error reduction. That confirm with the similar findings in speech recognition with MPE, i.e., by localizing or refining the error resolution, we can train a better discriminative model.

4. CONCLUSION

In this paper, we proposed to use discriminative training for improving letter-to-sound conversion performance. It is built upon a baseline LTS module of graphoneme N-grams. By using MMI and MPE criteria in refining the graphoneme N-gram models, small but consistent LTS performance improvements are obtained. A 3.8% relative reduction of errors is obtained with MMI and 4.6% with MPE. In the future, we will test other discriminative training criteria like large margin for improving LTS performance further.

5. ACKNOWLEDGEMENTS

The authors would like to thank Mei-Yuh Hwang for providing the baseline LTS modules and Ashley Chang for the databases. Discussions with Min Chu, Zheng Chen, and Jian Wu are helpful.

6. REFERENCES

[1] Evermann, G., Chan, H. Y., Gales, M. J. F., Jia, B., Liu, X., Mrva, D., Sim, K.C., Wang, L., Woodland, P. C., and Yu, K., "Development of the 2004 CU-HTK English CTS Systems Using More Than Two Thousand Hours of Data," *In Proc. of 2004 Rich Transcription Workshop (RT-04)*, 2004.

[2] Brants, T., Popat, A. C., Xu, P., Och, F. J., and Dean, J., "Large Language Models in Machine Translation", *In Proc. of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pp. 858–867, Prague, 2007.

[3] Damper, R. I., Marchand, Y., Adamson, M. J., and Gustafson K., "A Comparison of Letter-to-Sound Conversion Techniques for English Text-to-Speech Synthesis," *In Proc. of the Institute of Acoustic*, 20(6), pp. 245-254, 1999.

[4] Taylor, P., "Hidden Markov Models for Grapheme-to-Phoneme Conversion," *In Proc. of Interspeech 2005*, Lisbon, Portugal, pp. 1973-1976, 2005.

[5] Polyakova, T., and Zablotskii, V. "Learning from Errors in Grapheme-to-Phoneme Conversion" *In Proc. of Interspeech 2006,* Pittsburgh, USA, pp. 2442-2445, 2006.

[6] Black, A., Lenzo, K. and Pagel, V., "Issues in Building General Letter to Sound Rules," *In Proc. of the 3rd ISCA workshop on speech synthesis*, Jenolah Caves, Australia, pp. 77-80, 1998.

[7] Jiang, L., Hon, H.-W., and Hang, X.-D., "Improvements on a trainable letter-to-sound converter," *In Proc. of Eurospeech*, Rhodes, Greece, pp 605-608, 1997.

[8] Jelinek, F. "Statistical Methods for Speech Recognition," MIT Press, 1998.

[9] Chen, S. F., "Conditional and Joint Models for Grapheme-to-Phoneme Conversion," *In Proc. of Eurospeech 2003,* Geneva, Switzerland, pp.2033-2036, 2003.

[10] Galescu, L., and Allen, J., "Bi-directional Conversion Between Graphemes and Phonemes Using a Joint N-gram Model," *In Proc. of the 4th ISCA Tutorial and Research Workshop on Speech Synthesis*, Perthshire, Scotland, 2001.

[11] Vozila, P., Adams, J., Lobacheva, Y., and Thomas, R., "Grapheme to phoneme conversion and dictionary verification using graphonemes," *In Proc. of Eurospeech 2003* Geneva, Switzerland, pp. 2469-2472, 2003.

[12] Bisani, M., and Ney, H., "Investigations on Joint-MultiGram Models for Grapheme-to-Phoneme Conversion," *In Proc. of ICSLP*, pp. 105-108, 2002.

[13] Juang, B.-H., Chou, W., and Lee, C.-H., "Minimum Classification Error Rate Methods for Speech Recognition," IEEE Trans. SAP, pp.257-265, Vol.5, No.3, May 1997.

[14] Valtchev, V., Odell, J.J., Woodland, P.C., and Young, S.J., "MMIE Training of Large Vocabulary Speech Recognition Systems, " *Speech Communication*, Vol. 22, pp. 303-314, 1997.

[15] Povey, D., "Discriminative Training for Large Vocabulary Speech Recognition," *Ph.D. thesis*, Cambridge University, 2004.

[16] Du, J., Liu, P., Soong, F.K., Zhou, J-L., and Wang, R-H., "Minimum Divergence Based Discriminative Training," *In Proc. Interspeech,* pp 2410-2412, 2006.

[17] Chen, Z, Li, M-J, and Lee, K-F, "Discriminative Training on Language Model," *In Proc. of International Conference on Spoken Language Processing*, Beijing, October 16-20, 2000.

[18] Kuo, H.-K., Fosler-Lussier, E., Jiang, H., and Lee, C.-H., "Discriminative Training of Language Models for Speech Recognition," *In Proc. of ICASSP*, Orlando, Floridam, pp. 325-328, 2002.

[19] Ney, H., Essen, U., and Knesser, R., "On Structuring Probabilistic Dependencies on Stochastic Language Modeling", *Computer Speech and Language*, vol. 8, no.1, pp. 1-83, 1994.

[20] Sejnowski, T.J., "The NetTalk Corpus: Phonetic Transcription of 20,008 English Words," 1988.

[21] Weide, R., "The CMU Pronunciation Dictionary," Carnegie Mellon University, 1998.