Automatic Speech Recognition System Channel Modeling

Qun Feng Tan, Kartik Audhkhasi, Panayiotis G. Georgiou, Emil Ettelaie, Shrikanth Narayanan

Signal Analysis and Interpretation Lab (SAIL)
Electrical Engineering Department
University of Southern California, Los Angeles, CA

{qtan,audhkhas}@usc.edu, georgiou@sipi.usc.edu, ettelaie@usc.edu, shri@sipi.usc.edu

Abstract

In this paper, we present a systems approach for channel modeling of an Automatic Speech Recognition (ASR) system. This can have implications in improving speech recognition components, such as through discriminative language modeling. We simulate the ASR corruption using a phrase-based machine translation system trained between the reference phoneme and output phoneme sequences of a real ASR. We demonstrate that local optimization on the quality of phoneme-to-phoneme mappings does not directly translate to overall improvement of the entire model. However, we are still able to capitalize on contextual information of the phonemes which a simple acoustic distance model is not able to accomplish. Hence we show that the use of longer context results in a significantly improved model of the ASR channel.

Index Terms: Automatic Speech Recognizer, Channel Modeling, Finite State Transducer

1. Introduction

In this paper, we present a model of ASR channel errors. An accurate system of phoneme-sequence generation from text input (a phoneme pseudo-ASR) will allow for evaluation of alternative language models, without the need for acoustic data. Such a model can be useful in generating data for training discriminative language models [1], and also for studying the channel characteristics of the ASR.

As is well known, an ASR system can be viewed as a noisy channel, since its output (e.g. the 1-best hypothesis) is a noisy version of the clean reference transcription. Kurata et al. [1] proposed to model this channel using a simple phoneme weighted finite state transducer (FST), coined "pseudo-ASR". This model is constructed with each phoneme mapping to another phoneme or epsilon/NULL transition with a certain probability. These transition probabilities are computed from trained single-Gaussian monophone acoustic models, by finding the Bhattacharya distance between the Gaussians at the central state of the 3-state left-to-right Hidden Markov Model (HMM). The transition probability is computed as follows:

$$P(p_j|p_i) = \frac{exp(-BD_{ij})}{\sum_k exp(-BD_{ik})} \quad (1)$$

where p_i and p_j represents two phonemes, and BD_{ij} is the Bhattacharya distance between corresponding acoustic models. In fact, most of the approaches to distortion modeling [2, 3, 4] utilize the ASR confusion matrix, or some distance between acoustic models (Kullback-Leibler divergence, Mahalanobis distance or Bhattacharya distance). Hence, little or no contextual information is utilized while training the distortion model.

We propose an alternative way to do this channel modeling using a Statistical Machine Translation (SMT) system to learn the mappings between the reference phoneme sequence and the ASR-corrupted phoneme sequence. By doing so, we are able to capitalize on contextual information in the phoneme sequences. The SMT system has been used previously to learn the letter-to-phoneme rules [5], and has been demonstrated to be significantly better than single phoneme to letter alignments.

The paper is organized as follows: Section 2 gives the motivation behind the chosen methodology. Section 3 provides a detailed description of our modeling approach. Our experimental setup and results are presented in Section 4, along with interpretations. The paper concludes in Section 5, with a discussion of possible applications and extensions of this work.

2. Methodology

The main motivation of the paper stems from the following rationale: instead of modeling just the relation between phonemes p_j and p_i as $P(p_j|p_i)$, why not model $P(p_{j_1}^{j_n}|p_{i_1}^{i_m})$, by explicitly taking into consideration the phoneme context when learning the mappings. $p_{j_1}^{j_n}$ and $p_{i_1}^{i_m}$ are compact representations for the phoneme sequences $\{p_{j_1}, ..., p_{j_n}\}$ and $\{p_{i_1}, ..., p_{i_m}\}$ respectively. $p_{i_1}^{i_m}$ is the input phoneme sequence and $p_{j_1}^{j_n}$ is the ASR corrupted phoneme sequence. Note that the phoneme set can also include the NULL phoneme which represents an insertion or deletion error. In practical ASR systems, we will have

$$0 \leq i_1 \leq j_1 \leq i_m \leq j_n \quad (2)$$

to maintain causality. However, we relax this assumption in our case since we will be learning the mappings from the entire training data set, which will allow for learning non-causal mappings. Note that our prediction problem can be cast as the following:

$$p_{j_n} = \arg\max_{p_{j_1}^{j_m}} P(p_{j_1}^{j_n}|p_{i_1}^{i_m}) \quad (3)$$

Note that (3) can be reformulated using Bayes' theorem:

$$p_{j_n} = \arg\max_{p_{j_1}^{j_m}} P(p_{i_1}^{i_m}|p_{j_1}^{j_n})P(p_{j_1}^{j_n}) \quad (4)$$

where $P(p_{j_1}^{j_n})$ is given by the language (phonotactic) model (LM) on the target corrupted phonemes.

We see that the Machine Translation framework proposed by Brown *et. al.* [6] lends itself well in representing the above and it is therefore our chosen approach for learning multiple phoneme-phoneme mappings.

3. System Description

Having explained the theoretical motivation behind this work, we proceed to outline the actual implementation of the system.

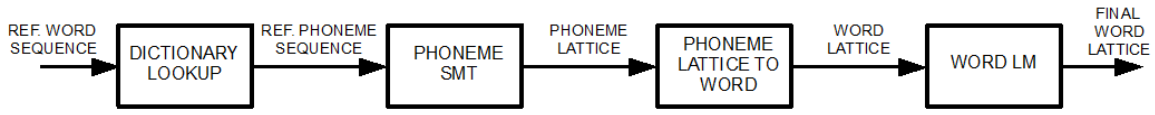

Figure 1: *The block diagram of the proposed ASR system corruption channel model.*

The training stage follows this procedure:

- Statistical Phoneme Transducer (SPT) model:
  - Phoneme decoding using Phoneme-ASR
  - Train a Statistical Phoneme Transducer (SPT) on clean phoneme sequence as reference and ASR-corrupted phoneme sequence as target
- Build phoneme-word transducer using pronunciation dictionary
- Build word transition transducer (FST encoded LM)

Using the above modules the channel model can be represented as in Fig. 1:

- Text-Phoneme conversion
- Decoding through the SPT. Output a phoneme lattice
- Phoneme-Word transducer. Output a word lattice
- Decode through LM transducer (FST-LM). Output the final word-lattice or N-best list

The system can be represented as a single FST with the composition of the phoneme lattice from the SPT with the Phoneme-Word transducer and the Language Model transducer in order. The best word sequence can be obtained by the Viterbi algorithm over the composed lattice, giving us the N-best lattice.

The text-to-phoneme conversion can be carried out by simple dictionary look up approach. While our initial experiments assume no OOVs present, the extension to handle them is straightforward: the mappings can be adequately learnt by building another transducer system on the letter-to-sound rules [5].

An SPT is trained using the MOSES [7] system on parallel phoneme data, with the original phoneme sequence as the source data and the ASR-corrupted phoneme sequence as the target data. We adopt the phrase-based translation model for this purpose. Our baseline model will be the SPT model where the phrase table is restricted to a unigram mapping. We will explain why this is similar to Kurata's simple phone-to-phone FST [1]. Kurata's model is based on a distance measure of the representative phoneme Gaussians, which is a phoneme confusability based metric [2, 3, 4]. When we restrict the phrase-table length to the unigram, the SMT system is then finding the phone-to-phone transition probabilities $P(p_i|p_j)$ by essentially computing the counts when it sees the mapping and then normalizing the counts with some smoothing. This will be equivalent to learning the probability distribution from the confusion matrix of the ASR by count-and-divide, and the confusion matrix is closely correlated with the distance between the Gaussians corresponding to the phonemes, and arguably more accurate.

The main advantage of using the SMT is that we can vary this phrase table length, which we will from the unigram to the 7-gram. Another advantage of the SMT is that we will be able to smooth the transition probability with a phoneme language model built on the ASR corrupted phoneme sequences. This will allow us to prune the lattice, and also make use of context information to result in better mappings.

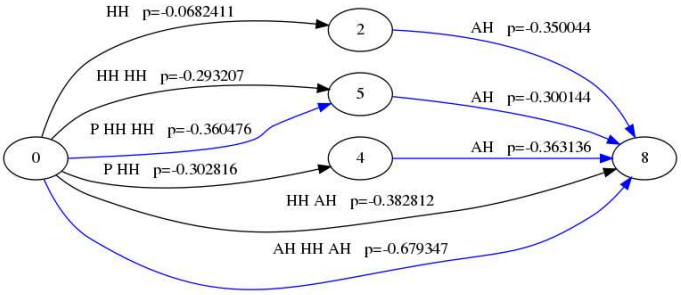

Figure 2: *The MOSES (7-gram phrase table, training set's target is phoneme output from ASR 1-best) output lattice for input phoneme "HH AH" for maximum of 5 hypotheses to each stack*

There are two ways of proceeding with the SMT system built. The first is that we can encode the phrase table and the phoneme language models into finite-state transducers and then compose a series of finite-state transducers to get the final system. However, such a long composition will not be efficient, thus we make use of the lattice output from the MOSES decoder. Figure 2 shows an example of such a lattice. The output of MOSES can be in the form of an N-best list or a lattice, which we can easily convert into a finite-state-acceptor (FSA) for use by the subsequent components.

By default lattice generation with MOSES results in extremely large lattices since in addition to containing all the possible paths, it also contains redundant paths resulting in same phoneme sequences generated from different phoneme partitions. A solution to this will be to use the cube pruning available in the MOSES system. We prune the lattice to have a maximum of 10 hypotheses at each stack, where each stack corresponds to each target phoneme symbol. This can be suitably adjusted according to the computing resources available.

For our SPT module we chose monotone decoding, where the distortion is 0. This is because the ASR's decoding does not allow for the interchanging of phonemes, thus the SMT has to model this aspect too.

The next component in our corruption model is the phoneme-word transducer. This is simply a finite state transducer which maps input phonemes to legitimate words in the dictionary. An issue with a simplistic transcription of phoneme to words is that there might be a phoneme lattice which does not contain the exact phoneme paths that maps to words in the dictionary. Thus we have to allow for deletions of phonemes in our transducer. However, to ensure that we do not delete phonemes arbitrarily, we introduce a penalty to the deletion. Thus, all the legal phoneme strings will map to words in the dictionary with probability 1, while deletions will happen with probability 10^{-10}.

The third component of our corruption model is the Language model (LM). The language model will score all the dif-

ferent word sequences and return the sequence which has the highest probability. The LM can be represented as an FST [8]. We first use SRILM to build our LM, and then converted it to OpenFST format [9]. One caveat is that SRILM language models make use of failure transitions which is essentially equivalent to an if-else condition. However, finite-state-machines are all probabilistic in nature. One solution outlined [8] is to replace the failure transitions with epsilon transitions, and to make sure that the backoff probabilities are smaller than the higher order N-gram probabilities by appropriate smoothing. In the tri-gram language model, this will be $P(z|x,y) > \alpha(x,y) * P(z|y)$ where $\alpha(x,y)$ is the back-off weight for the bigram (x,y). In practice, there will be some back-off weights where this particular condition is not satisfied, but its number is insignificant compared to the total number of back-off weights. Thus the epsilon-transition is a good approximation to the failure-transition model.

4. Experimental results

We train a phoneme recognition system on a portion of the Wall Street Journal (WSJ) corpus of 16.6 hours of data. This gave a phoneme error rate (PER) of 39.9%. We then decoded a set of 3361 phoneme sequences which have length between 1 and 70 in length to get the ASR corrupted phoneme output. The reason why we keep the phoneme sequences to be between 1 and 70 in length is because we will be using GIZA [10] to build our phoneme alignments, and keeping them within this length span will result in a faster runtime. We will use 361 of those sequences for our test set, and the rest for training (2500) and tuning (500).

We will be using the standard evaluation techniques for evaluating the goodness of our mapping schemes, namely BLEU score and Word Error Rate (WER). The BLEU score and WER will be computed with the real ASR's output as the reference, and the pseudo-ASR's output as the hypothesis.

Note that since we are dealing with vocabulary size of only 39 English phonemes, the coverage of the training set is sufficient for learning phoneme-to-phoneme mappings. Table 1 shows a 10-fold comparison of the n-gram coverage of 90% of the training data versus the full training data set.

The MOSES translation system has a tuning option which uses discriminative training to adapt weights like the LM weights, reordering weights etc. This discriminative training is optimized on the BLEU metric, and is not guaranteed to improve the PER/WER. For example, for the default phrase table length of 7-grams, we see that the BLEU score obtained is 42.83 and the PER is 36.1 without tuning, and the BLEU score obtained is 43.09 and the PER is 36.6 with tuning. Thus, we see that tuning leads to an improvement in the BLEU metric, but at the same time, damages the PER. Thus, for the evaluation purposes of this paper, we will be using the default weights that MOSES starts out with (flat initialization). Although in the case of the ASR channel model we are more interested in the phoneme and subsequently word error rates, for comparison we will be presenting figures of the BLEU metric as well. We use BLEU_P and BLEU_W to denote phoneme and word BLEU evaluations respectively.

Table 2 shows the results of our evaluation of the 1-best output of the SPT with respect to the 1-best output of the ASR at the phoneme level. We see that for increasing N-gram length of the phrase table, which means increasing phoneme context information, we see that the performance of the mappings improve. The BLEU_P score is the highest with the 7-gram model having a 11.17% relative performance improvement over the unigram baseline. Also, the PER falls from 37.3% to 35.1%.

Table 1: *10-fold comparison of the N-gram coverage of 90% of the training data versus the full training data set.*

N-gram order	Coverage
1-gram	1.00
2-gram	0.99
3-gram	0.95
4-gram	0.90
5-gram	0.87
6-gram	0.86
7-gram	0.85

Table 2: *Evaluation of our corruption model quality at the phoneme level trained on 1-best ASR output*

SPT order for phrase table and LM	BLEU_P Score	BLEU_P % gains over baseline	PER	PER % gains over baseline
1-gram	40.45	0.0000	37.3	0.00
2-gram	42.04	3.931	36.5	2.14
3-gram	42.49	5.043	36.3	2.68
4-gram	44.42	9.815	35.4	5.09
5-gram	44.54	10.11	35.2	5.63
6-gram	44.69	10.48	35.2	5.63
7-gram	44.97	11.17	35.1	5.90

The above SMT is trained on the 1-best output of the ASR. Since our goal is to model the corruption channel of the ASR, we would like to see if training on more than the 1-best could lead to a substantial improvement of our model quality. Thus, we also train the SMT on the 3-best list from the ASR output on the target phoneme side. For the source phonemes, we simply duplicate the phonemes 3 times.

Table 3: *Evaluation of our Corruption Model quality at the phoneme level trained on the 3-best ASR output*

SPT order for phrase table and LM	BLEU_P Score	BLEU_P % gains over baseline	PER	PER % gains over baseline
1-gram	40.45	0.0000	37.3	0.00
2-gram	42.10	4.079	36.5	2.15
3-gram	40.08	-0.9147	39.2	-5.09
4-gram	43.43	7.367	36.1	3.22
5-gram	43.33	7.120	36.1	3.22
6-gram	43.40	7.293	36.2	2.95
7-gram	43.49	7.515	36.1	3.22

Table 3 shows the evaluation results for the training set with the ASR 3-best output. Here, the best performance is achieved with the 7-gram with a BLEU_P score of 43.49 and a PER of 36.1%. We see that the BLEU_P and PER performances are in general lower than that trained from the 1-best output in Table 2. This is because since we are training on the 3-best output from the ASR, the SMT is learning more of the confusion mappings, which can result in a noisier phoneme output. The behavior of the results in this table is also more erratic, but generally shows the same overall increasing trend in performance as in Table 2.

In conclusion, we see that the optimal strategy to match the

output of the ASR with maximal $BLEU_P$ score and lowest PER at the phoneme level is to train the SMT on the ASR 1-best, with larger phrase table limits.

We will now evaluate the n-gram MOSES models trained on the ASR 1-best at the word level, by composing the SMT lattice generated with the phoneme-word transducer and word LM as outlined in Section 2. Here we will use $BLEU_W$ score and WER as our evaluation metric. For the reference text, we trained an ASR based on the WSJ corpus, with a word error rate (WER) of 23.3%, on 88 hours of data.

Table 4: *Evaluation of our Corruption Model (trained on the ASR 1-best output) quality at the word level from the complete system output for 1-gram (baseline) and higher order N-grams*

SPT order for phrase table and LM	$BLEU_W$ Score	$BLEU_W$ % gains over baseline	WER	WER % gains over baseline
1-gram	67.45	0.000	16.3	0.00
2-gram	67.51	0.0890	15.6	4.29
3-gram	68.18	1.082	14.7	9.82
4-gram	65.79	-2.461	16.9	-3.68
5-gram	64.91	-3.766	17.8	-9.20
6-gram	64.49	-4.388	18.1	-11.0
7-gram	63.76	-5.4707	18.9	-16.0

The runtimes for the decoding through the entire system increases with the number of N-grams. This is because the phoneme lattice gets more complicated and the composition of all the lattices is more resource intensive.

We see that from the results in Table 4, the 3-gram phrase table gives the highest $BLEU_W$ score of 68.18 and lowest WER of 14.7. We see that while increasing the phrase table length implies increased accuracy in phoneme mapping from Table 2, increasing the phrase table length does not imply a uniform increase in the accuracy on the system as a whole. However, we see context still helps the accuracy in this case. We see the 2, 3-gram models still outperforms the baseline result of 1-gram model (no context information). However, for the 4, 5, 6 and 7-gram models, the baseline gives a better result. The reason why the accuracy might decrease in the case of longer context is because the SMT might not have seen enough training data to adequately learn enough of the higher-gram mappings. This would result in a certain amount of overfitting, which will result a degraded performance in terms of $BLEU_W$ and WER. Thus, when we are building a corruption model, we will have to tune for the optimal number of N-grams to use in the phrase table based on the amount of training data we have.

5. Conclusion and Extensions

We demonstrate that phonemic context information is an important factor in improving the quality of our ASR corruption model. We also show that local optimization on the $BLEU_P$ score and PER of the phoneme translation/mapping does not correspond to an increased performance of the system as a whole, but however, context information still helps in improving our corruption model. We see that in our case, the 3-gram model gives the best performance in terms of $BLEU_W$ score and WER at the word-level output of the corruption model.

As discussed in the paper, being able to simulate output of the ASR channel is the first step in studying the channel properties of the ASR in greater detail.

For example, in applications where large amounts of ASR output data is needed, it is often impractical to collect such large amounts of transcribed audio. However, an ASR channel model allows generation of massive amounts of corrupted ASR-like output, only limited by the amount of text we have. Only a small collection of acoustic data for training the channel model system is necessary. One application is in training the machine translation component of a speech-to-speech (S2S) system, where we can incorporate the uncertainty of the source language in the translation model. Another application for this model is to use lattices generated from the channel model to train a discriminative Language Model system like in [1] and evaluate if the merits of having a more accurate ASR channel simulation would yield better Language Models.

6. Acknowledgements

This work was supported by the NSF, DARPA and US Army.

7. References

[1] G. Kurata, N. Itoh, and M. Nishimura, "Acoustically discriminative training for language models," in *ICASSP '09: Proceedings of the 2009 IEEE International Conference on Acoustics, Speech and Signal Processing*. Washington, DC, USA: IEEE Computer Society, 2009, pp. 4717–4720.

[2] J. Silva and S. S. Narayanan, "Average divergence distance as a statistical discrimination measure for hidden Markov models," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, no. 3, pp. 890–906, May 2006.

[3] J. R. Hershey and P. A. Olsen, "Approximating the Kullback Leibler divergence between Gaussian mixture models," *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on*, vol. 4, pp. IV–317–IV–320, 2007.

[4] ——, "Variational Bhattacharyya divergence for hidden Markov models," *Acoustics, Speech and Signal Processing, 2008. ICASSP 2008. IEEE International Conference on*, pp. 4557–4560, 2008.

[5] T. Rama, A. K. Singh, and S. Kolachina, "Modeling letter-to-phoneme conversion as a phrase based statistical machine translation problem with minimum error rate training," in *NAACL HLT 2009*, May 2009.

[6] P. E. Brown, V. J. Della Pietra, S. A. Della Pietra, and R. L. Mercer, "The mathematics of statistical machine translation: parameter estimation," *Computational Linguistics*, vol. 19, pp. 263–311, 1993.

[7] P. Koehn, H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, C. Dyer, O. Bojar, A. Constantin, and E. Herbst, "Moses: Open source toolkit for statistical machine translation," in *ACL '07: Proceedings of the 41st Annual Meeting on Association for Computational Linguistics, Demo Session*. Association for Computational Linguistics, 2007.

[8] C. Allauzen, M. Mohri, and B. Roark, "Generalized algorithms for constructing statistical language models," in *ACL '03: Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*. Morristown, NJ, USA: Association for Computational Linguistics, 2003, pp. 40–47.

[9] A. Cyril, R. Michael, S. Johan, S. Wojciech, and M. Mehryar, "Openfst: A general and efficient weighted finite-state transducer library," *Proceedings of the Ninth International Conference on Implementation and Application of Automata CIAA 2007*, vol. 4783, pp. 11–23, 2007.

[10] F. J. Och and H. Ney, "A systematic comparison of various statistical alignment models," *Computational Linguistics*, vol. 29, no. 1, pp. 19–51, 2003.