A Study on LVCSR and Keyword Search for Tagalog

Korbinian Riedhammer1, Van Hai Do1,2, James Hieronymus1

1International Computer Science Institute, Berkeley, USA
2School of Computer Engineering, Nanyang Technological University, Singapore

{koried, haido, hieronym}@icsi.berkeley.edu

Abstract

We describe a state-of-the-art large vocabulary continuous speech recognition (LVCSR) and keyword search (KWS) system trained on roughly 70 hours of conversational telephone speech. Using the Kaldi speech recognition toolkit, we investigate several aspects: for the acoustic front-end, we analyze the use of mel-frequency cepstral coefficients (MFCC), pitch and probability-of-voicing (PoV), and deep neural network (DNN) bottleneck (BN) features, as well as their feature-level combination (“tandem”). For the acoustic-phonetic decision tree, we explore different hidden Markov model (HMM) topologies for the glottalization phoneme followed by a model typically short duration. For the acoustic model, we compare regular continuous HMM with a sort of multi-codebook subspace Gaussian mixture model (SGMM) that lead to an overall best word error rate (WER) of 58.7% and 56.3%, respectively. The KWS is implemented as a word lattice search, and is augmented by a syllable lattice back-up search to capture out-of-vocabulary keywords as well as misrecognized lexical surface forms due to ambiguous prefix and hyphenation rules.

Index Terms: speech recognition, keyword spotting

1. Introduction

Tagalog, a major register of Filipino, is the official language of the Philippines and is spoken by approximately 98% of the 103 million inhabitants as a first or second language. There are many Spanish borrow words which have been modified with Tagalog phonology due to 300 years of colonial rule. A large number of English words have also been incorporated with much less spelling and phonological modification due to the US influence. The result is an enlargement of the alphabet, phonetic inventory with more fricatives and affricates and changes in the syllable structure. Consonant clusters were previously restricted to the onset, but English borrow words have allowed coda consonant clusters. The resulting syllable structure is C-G-V-G-C, but a majority of syllables are C-V-C, V-C or C-V, where C stands for consonant, V for vowel, and G for glide. A special feature of Tagalog is that words beginning with a vowel and vowel-to-vowel transitions within a word are marked by glottalization, which constitutes a vowel boundary marker. This makes glottalization the second most common phoneme in Tagalog. Since glottalization is a rather short period of “creaky voice,” it is necessary to have special acoustic modeling for it. The phone set in Tagalog consists of 10 monophthong and four diphthong vowels and 21 consonants including borrow word additions of /ɛ/ and /ɛs/, for a total of 48 phones. Our acoustic modeling uses word position dependent phones (beginning, end, interior, and singleton) along with pseudo-phones for other acoustic events which leads to 227 word position dependent phones.

There is relatively little prior work on large vocabulary continuous speech recognition (LVCSR) for Tagalog, or more generally, the Filipino language. Sagum et al. used a hybrid (connectionist) approach to compute phoneme alignments on the Filipino Speech Corpus (FDC), a corpus of about six hours of read speech, with a small portion of spontaneous recordings, comprising about 1500 words [1, 2]. Ang et al. implemented a prototype for on-device ASR where the phone posterior is computed locally but the decoding is done remotely on the server; on a data set based on the FDC, they achieved a word error rate of about 44% [3]. Sakti et al. addressed the insufficient training data by a cross-language approach: the initial acoustic models were trained on Indonesian data and subsequently adapted to Filipino [4]. More recently, Ang et al. implemented a close-captioning system based on a Filipino news corpus consisting of about six hours of a daily news program; using a lexicon of about 3000 words and a Sphinx base recognizer (MFCCs, HMM-GMM), they showed an error rate of about 42% [5]. In further experiments on language model smoothing, the authors could reduce the error rate on newswire to about 20% using a vocabulary of about 16,000 words and a Kneser-Ney smoothed 5-gram language model [6].

In the following sections, we describe a large vocabulary speech recognition and keyword search system that is developed in the context of the IARPA Babel project using about 70 hours of spontaneous telephone speech in Tagalog. While most of the prior work, constrained by the lack of available training data, focused on implementing proofs of concept using well-established techniques, we compare and analyze different state-of-the-art features and acoustic models with respect to both automatic speech recognition and keyword spotting. The presented experiments are split three parts; for the acoustic front-end, we compare MFCC, pitch and probability of voicing, and bottleneck features, as well as their feature-space combination (“tandem”). For the glottalization phone (GP) /?/, we experiment with alternate HMM topologies including 1-state, 3-state left-to-right, and ordinary 3-state linear HMMs to allow for a better fit to the data and thus more reliable alignments. As Tagalog shows a high number of pre- in- and postfixes, we expect a fair amount of misses in the keyword search due to shortcomings in the language model. To compensate, we experiment with a syllable-based lattice decoding that shows stronger linguistic and acoustic constraints than a pure phone-based decoding, but is expected to be less error-prone to those deficiencies.

2. Data

We use the Tagalog data and transcriptions of the IARPA Babel base period. The full language pack consists of about 70 hours of spontaneous two-channel telephone recordings. The actual single-channel speech portion is about 43h for training.
and 9h for development. From those, we use 1116 conversations (about 64k utterances) for training, and 145 (10k) for development. The vocabulary consists of about 22k word forms, the transcriptions sum up to about 486k words. The keywords used to evaluate the search were provided by IARPA for the March 2013 project evaluation.

3. System Description

The Kaldi speech recognition toolkit [7], along with the TNet\textsuperscript{1} neural network trainer, were used for the experiments. Both are freely available under the Apache 2.0 open-source license and have recently been used in numerous research projects.

3.1. Features

3.1.1. Spectral

We use 13 MFCCs as primary spectral features, to which we apply cepstral mean subtraction. While we use deltas and second delta for early systems in the bootstrapping process, we use a variant of HLDA features for the final systems. We compute an LDA transformation that takes as input a context of 7 spliced static MFCC vectors and uses the context dependent states (from an initial alignment) as targets; we typically project the features down to a dimension of 40. Throughout the training iterations, this LDA matrix is composed with global MLLT matrices [8] as well as speaker dependent iMLLR matrices [9].

3.1.2. Pitch and Probability of Voicing

We extract pitch and probability-of-voicing (PoV) features using a sub-band autocorrelation classification, SAcC [10]. 24 constant-Q subbands are individually autocorrelated and reduced to 10 principal components per sub-band. These 24x10 dimensions form the input layer of an MLP classifier with 100 hidden units and 68 output units corresponding to quantized pitches in the range 60 to 400 Hz, plus one “no voice” bin. The MLP outputs are Viterbi-smoothed to give the final pitch track. The MLP is trained on clean and noisy data from the Keesle database [11] and from the DARPA RATS program, where the ground-truth pitch information is obtained via a standard pitch detect algorithm. We have recently been used in numerous research projects.

3.1.3. Bottleneck Features

Recently, bottleneck (BN) features have been used widely in speech recognition and provide a consistent improvement over conventional features such as MFCCs and perceptual linear predictive coefficients (PLP) [12]. BN features are generated using a neural network (NN) with several hidden layers where the size of one particular hidden layer (“bottleneck” layer) is very small. With this structure, we can choose an arbitrary feature size independently of the network input and output dimensions. In [13], a hierarchical NN structure was proposed to improve the posterior features. This idea is applied in [14] to build hierarchical bottleneck NNs that lead to a consistent improvement over the conventional BN network structure. Specifically, a hierarchical BN processing of NNs is a cascade of NNs, where the next NN uses BN features generated by the previous NN as input features.

\footnote{\url{http://speech.fit.vutbr.cz/software/neural-network-trainer-tnet}}

\footnote{\url{http://www1.icsi.berkeley.edu/Speech/faq/nn-train.html}}

Figure 1: Hierarchical bottleneck neural network architecture.

In this paper, we apply this hierarchical structure to generate BN features with several modifications. Fig. 1 shows the diagram of the BN-NN structure used for the experiments in this paper. The original features are 15-dimensional vectors which include 13 dimensional MFCCs, pitch and PoV. We consider a context of ±15 frames (31 frames total) and apply “row-wise” DCT to retrieve 16 coefficients each, including C0 (this is similar to the idea of critical bands). Finally, we have 15x16 = 240 coefficients to form the input for the first NN. To get a wider context and modest increase of the input size for the second NN, we sample 5 frames at the positions: -10, -5, 0, +5, +10 of the 60-dimensional BN features generated by the first NN to form the 300-dimensional input for the second NN. Finally, we obtain 30-dimensional BN feature generated by the second NN.

In this study, we use a deeper architecture for the first NN and a simpler architecture for the second NN. We believe that the first NN plays an important role to convert the raw features to BN features, while the second NN simply takes the broader context of the first NN features and compresses them. With such a deep structure in the first NN, the standard back propagation algorithm may be difficult to train with randomly initialized weights. To overcome this difficulty, we use Restricted Boltzmann Machine (RBM) training [15] to pre-train the first NN layer-by-layer. This unsupervised training provides better initial weights for the supervised back propagation step. Note that while using RBM pre-training in the first NN can improve the BN features consistently, no significant improvement is observed when using RBM pre-training for the second NN. This may be due to the fact that the conversion from the first NN features to the final BN features is a rather simple task and the back propagation algorithm can find a good optimum easily. To reduce the amount of training time, we use RBM pre-training only for the training of the first NN, while the weights of the second NN are initialized randomly.

The weights between the first two layers of the first NN are initialized by a RBM (Gaussian-Bernoulli), using a learning rate of 0.005 with 10 pre-training epochs. For the remaining RBMs (Bernoulli-Bernoulli), we use a learning rate of 0.05 with 5 pre-training epochs. Both NNs are fine-tuned using back-propagation algorithm. The “newbob”\textsuperscript{2} procedure is used with an initial learning rate of 0.008. We used the open-source toolkit TNet to implement the RBM pre-training and back propagation procedures. The targets at the output layer are the 51 context-independent phones (including non-speech models), obtained from a mono-phone alignment. The cross-validation accuracy after ten epochs was 63.69.

\footnote{\url{http://www1.icsi.berkeley.edu/Speech/faq/nn-train.html}}
3.2. Acoustic Model

3.2.1. Continuous Models

The most common way to model the emission probabilities of the context dependent states is to assign an individual GMM to each of the (clustered) states. We initialize the mixtures with a single component each, and subsequently allocate more components by splitting components at every training iteration until a pre-determined total number of components is reached. The final system has 5k context dependent states and 80k Gaussian components.

3.2.2. Subspace Gaussian Mixture Models

The idea of subspace Gaussian mixture models (SGMM) is to reduce the number of parameters by selecting the Gaussians from a subspace spanned by a universal background model (UBM) and state specific transformations. The SGMM emission pdfs can be computed as:

\[ p(x|j) = \sum_{i=1}^{N} c_{ji} \mathcal{N}(x; \mu_{ji}, \Sigma_{ji}) \]  

\[ \mu_{ji} = M_i v_j \]  

\[ c_{ji} = \frac{\exp w_{ji}^T v_j}{\sum_{j'} \exp w_{j'}^T v_j} \]

where the covariance matrices \( \Sigma_j \) are shared between all states \( j \). The weights \( w_{ji} \) and means \( \mu_{ji} \) are derived from \( v_j \) together with \( M_i \) and \( w_i \). A detailed description and derivation of the accumulation and update formulas can be found in [16], along with extensions to speaker dependent transformations.

Here, we use the SGMM2 extension within the Kaldi code and scripts, which extends the conventional SGMMs by two ideas. First, by a “symmetrization” which makes the speaker subspace and phonetic subspace behave the same way [17]. And second, an idea similar to state-clustered tied mixture; it involves sharing Gaussians among fairly small sets of context-dependent states, but applied at the SGMM sub-state level rather than the Gaussian level as in [18, 19]. The final system has 8k states and 50k sub-states derived from 700 Gaussians.

3.2.3. Tandem Features

For the tandem features, we paste spectral, pitch and bottleneck features together to form the tandem feature vector. Here, we modify the LDA transformation, so that it takes a context of 7 frames of MFCC, pitch and PoV, and projects it down to match the dimension of the bottleneck features. The later MLLT and fMLLR are applied to the combined feature stream. In contrast to the feature-level combination described in [20], this combination extracts spectral features from a wide temporal context, but uses the BN features as-is, as they have already been trained on a large temporal context. Furthermore, we expect the streams to be balanced which shall be beneficial for the full-covariance matrices in the SGMM framework.

3.3. Language Model

We estimate a basic 3-gram language model on the training transcripts, and apply Kneser-Ney smoothing and interpolated counts [21, 22]. However, for this work, we concentrate on the acoustic modeling as a foundation to application-specific language modeling later on.

4. Speech Recognition Experiments

We trained continuous and SGMM systems following the published Kaldi WSJ/s5 recipes, using speaker adaptive training and a multi-pass lattice decoding [23]; minor alterations were made to better accommodate the Babel data.

4.1. Features in Comparison

Tab. 1 shows the WER using different features and their feature-level combination. Adding pitch and PoV to the MFCC significantly improves the performance. The BN features independently show already a strong improvement; they benefit from the large temporal context and the deep NN structure. However, the feature-level combination of the MFCC, pitch and PoV, and BN features shows another significant improvement, which is particularly visible for the SGMM based system. Most likely, the full covariance matrices in the SGMM framework leverages the complementary information of the two feature streams.

4.2. Modeling the Glottalization Phone

Initial experiments where the glottalization phone (GP) was either deleted or merged with the subsequent phone lead to a significant increase in WER. This indicates that the one hand that the GP is a critical phone that requires modeling, but on the other hand should not be overspecified by attaching it to each vowel, thus leading to a artificially large phone inventory. An analysis of spectrograms indicated that the instances of a GP are a rather short period (10–40 ms) of “creaky voice,” thus a different topology of the GP HMM that accommodates shorter duration should lead to an improved fit of the models to the data. While the regular 3-state linear topology does not allow skips and enforces a minimum duration of three frames (here: 30 ms), we experiment with a 1-state and a 3-state topology, where the latter allows skipping from every state to each subsequent state (“left-to-right”).

Tab. 2 shows the WER for the three different topologies. The results confirm that a shorter GP HMM seems to fit the data better. Although the improvements appear only minor, they are significant and can be further analyzed. A particular problem during the training of the acoustic model is the forced alignment of the GP, especially when following silence, as both models will share the acoustic training data to some extent. Tab. 3 shows a comparison of the average phone duration (in frames) for the 3-state linear and 1-state topologies. The counts in columns /SIL/ and /?/ refer to the general phone alignments excluding the special co-occurrence /SIL ?/ which is
Table 3: Mean phone duration in frames for the SGMM system.

<table>
<thead>
<tr>
<th>topology</th>
<th>/?/</th>
<th>/SIL/</th>
<th>/?/</th>
<th>/SIL /?/</th>
</tr>
</thead>
<tbody>
<tr>
<td>3-state/linear</td>
<td>26.7</td>
<td>8.0</td>
<td>25.9</td>
<td>12.4</td>
</tr>
<tr>
<td>1-state</td>
<td>28.3</td>
<td>4.9</td>
<td>27.8</td>
<td>7.0</td>
</tr>
</tbody>
</table>

frequently found at the beginning of utterances and after short intermittent silences. The 1-state topology shows a by far more realistic phone duration: while the longer silence already indicates a possibly crisper detection of the actual phones, the on average 4.9 frames of a GP seem much more adequate than the previous average of 8.0 frames. For the special case of silence preceding the GP, the lengths of the individual phones have been re-distributed so that the silence portion is increased while the glottalization portion is reduced— which indicates that the GP model is still prone to confusion with the silence model. This more adequate GP model of the 1-state topology helps to reduce the number of utterances in which the leading silence was erroneously aligned to a GP by about 35%. Another fact that contributes to the better fit of the 1-state model is that possibly misaligned silence frames will not “take over” one or more of the HMM states during training, as it is most likely the case for the 3-state linear.

5. Keyword Search Experiments

5.1. Syllable Based Decoding

The KWS is implemented as a two-stage process, where the speech utterances are indexed first, and then searched for certain keywords without re-decoding the audio. To overcome the problem of future out-of-vocabulary searches, we implemented a syllable based lattice decoding. We assume that the syllable inventory of a language remains more or less constant, thus any unseen word can be mapped to the exact or phonetically most similar sequence of existing syllables. While those syllable lattices can still be expanded to phone lattices, the decoding is constrained to syllable-to-syllable transitions that can be estimated in a more robust way than just phone-to-phone transitions. The syllable dictionary is derived from the hand-labeled syllable boundaries in the language pack lexicon. The training data for the syllable language model is derived from the forced alignments of the training data; here we estimate a 5-gram syllable LM in the same way as the regular LM. The decoding graph is generated from the existing (word based) acoustic models and the syllable dictionary and LM.

5.2. Keyword Search Algorithm

Our keyword search algorithm is similar in spirit to what is described in [24]. We create a word (syllable) based index from the lattices, keeping track of all of the words that occur in the lattice, their start and end times, as well as their lattice posterior probabilities. For single word keywords, we return the list of all of the occurrences of the keyword, sorted by their posterior probabilities. For multi-word keywords, we retrieve the individual words from the index in the correct order with respect to their start and end times, but discard occurrences where the time gap between adjacent words is more than 0.5 seconds; the surviving occurrences are assigned an approximate probability equal to the minimum of the individual word probabilities. The detection threshold for each keyword is determined using an empirical estimate of its term weighted value.

3Due to the initial linear time alignment.

5.3. Results

Tab. 4 shows the results for searching the evaluation keywords on the development data, in terms of actual term-weighted value (ATWV) [25]. The word lattice based KWS results could be significantly improved using the 1-state model of the GP (the minimum program requirement for the current period is 0.30). The results using only syllable based KWS are rather disappointing (the drop of 0.015 for the different topologies is not significant). While we could show an improvement of ATWV using syllables in an earlier experiment using different keywords, we could not confirm this gain using the evaluation keywords. A possible reason for the poor syllable KWS results is that many syllables of the Tagalog language have up to four alternate pronunciations, however the most prevalent realization is most likely speaker dependent. The current implementation of the KWS considers only the first pronunciation as found in the lexicon, which is most likely sub-optimal.

6. Summary

The bottleneck features, trained on a large temporal context, show a particularly good performance for the continuous system. However, the spectral features benefit by far more from the SGMM architecture, pointing to a stronger covariance of the features. The feature-level combination introduces further covariance that can be leveraged by the SGMM based system, and shows the overall best WER. Using a 1-state HMM topology for the GP clearly improves the training alignments and leads further significant improvements. The overall high WER of the results can also be explained by the rather high perplexity of the 3-gram LM (309 on the training, 593 on development transcriptions). This underlines the difficulty of reliable language modeling due to the complex morphology. The initial KWS results are promising given the current program requirements. We believe that the syllable based KWS approach, once optimized for pronunciation alternatives, eventually leads to improvements on detecting out-of-vocabulary keywords.

7. Acknowledgements

The authors would like to thank Arlo Faria and Steven Wegmann for their support in generating the features and running the keyword search. This work is supported in part by the Intelligence Advanced Research Projects Activity (IARPA) via Department of Defense US Army Research Laboratory contract number W911NF-12-C-0014. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, DoD/ARL, or the U.S. Government. This work uses the IARPA Babel Program Tagalog language collection release babel106-v0.2g.
8. References


