Improved Hindi Broadcast ASR by Adapting the Language Model and Pronunciation Model Using *A Priori* Syntactic and Morphophonemic Knowledge

Preethi Jyothi[1], *Mark Hasegawa-Johnson*[1,2]

[1]Beckman Institute, University of Illinois at Urbana-Champaign, USA
[2] Department of ECE, University of Illinois at Urbana-Champaign, USA
jyothi@illinois.edu, jhasegaw@illinois.edu

Abstract

In this work, we present a new large-vocabulary, broadcast news ASR system for Hindi. Since Hindi has a largely phonemic orthography, the pronunciation model was automatically generated from text. We experiment with several variants of this model and study the effect of incorporating word boundary information with these models. We also experiment with knowledge-based adaptations to the language model in Hindi, derived in an unsupervised manner, that lead to small improvements in word error rate (WER). Our experiments were conducted on a new corpus assembled from publicly-available Hindi news broadcasts. We evaluate our techniques on an open-vocabulary task and obtain competitive WERs on an unseen test set.

Index Terms: Hindi LVCSR system, Broadcast news ASR, Grapheme and phoneme-based models, Knowledge-based language-model adaptation

1. Introduction

Hindi is the most widespread language in India, with roughly 200 million native speakers and many more who speak it as a second language. Due to its large number of speakers, there is immense potential for automatic speech recognition (ASR) applications in Hindi. Unfortunately, there has not been much progress in ASR for Hindi, especially in building large vocabulary systems. The most comprehensive LVCSR system in Hindi was developed by IBM-India in 2004 [1] where they utilized existing English acoustic models to bootstrap their recognizer. Other more recent attempts include an LVCSR system to recognize speech in the travel domain developed by CDAC, India [2, 3]. Many other recent attempts at Hindi ASR cater to either small-vocabulary tasks or isolated word tasks ([4, 5, 6, 7], to name a few).

In this work, we develop a new large-vocabulary ASR system to recognize broadcast news in Hindi. We use publicly available All India Radio (AIR) broadcasts to build our corpus and utilize an open-source toolkit (Kaldi [8]) to configure our ASR system. In Section 4, we investigate the use of both graphemic and phonemic models in our ASR system. We also introduce two techniques in Section 5 that are used to build knowledge-based adaptations to the language model and study its effect on ASR performance.

2. Hindi Broadcast News Corpus

We used publicly available news broadcasts from the All India Radio (AIR) website [9] to build our corpus. Many of these broadcasts are also accompanied by corresponding speech transcripts. For each major language, there are typically multiple regional news bulletins hosted from various cities in India, where the language is predominantly spoken. For our corpus, we restricted ourselves to bulletins from AIR-Bhopal (a city in the Hindi-speaking region of India). These broadcasts included news reports read by different speakers, with varying speaking styles but a fairly standard Hindi accent, in a recording environment pre-determined by the broadcast station.

We collected a set of broadcasts totaling ≈5.5 hours. Our corpus comprised news broadcasts from Bhopal between the months of April 2014 to July 2014.[1] The transcripts corresponding to these audio files were rendered in a legacy Hindi font, Krutidev (commonly used for Hindi), which we converted into a Unicode encoding. We lightly edited the transcripts to match the newsreader's speech where there were noticeable differences between the speech and the transcript text (mostly due to word substitutions or paraphrases).

Each individual news broadcast was typically 5–11 mins long and the transcripts were provided for the entire broadcast. In order to derive sentence-level transcripts from the latter to train our acoustic models, we adopted a semi-automatic technique using the Praat open-source toolkit [10]. The "To TextGrid (silences)" command in Praat marks speech intervals as silence or non-silence sounds using tunable parameters including a silence duration threshold and a silence intensity threshold (set to 0.25 ms and -25 dB in our experiments, respectively). The output from this program is a set of time-aligned intervals that are labeled as either silence or non-silence. This procedure labels almost all sentence boundaries as silences. However, some pauses within a sentence are also labeled as silences. We manually removed these intra-sentence silences; this can be achieved with a real time factor close to 1. This technique was employed since we did not have access to a baseline Hindi ASR system that could provide a set of alignments automatically. We note that the ASR systems built in this work can be subsequently used to automatically align larger Hindi datasets (as in [11, 12]).

3. Data Preparation

We split our 5.5 hour corpus into disjoint training, development and evaluation sets, comprising approximately 4 hours, 0.5 hours and 1 hour of speech, respectively. Table 1 shows

[1]The audio files and transcripts were downloaded separately and manually matched with each other. This process was not amenable to automation, due to how the data files are organized on the AIR website. This was the main bottleneck in collecting a larger corpus, and will be addressed in future work.

	Train	Dev	Eval
Number of sentences	2242	223	547
Number of word tokens	43173	4290	10245
Number of word types	4749	1254	2064
Size of dataset (in mins)	≈252	≈23	≈62
Speaker composition	7M,3F	1M,1F	1M,1F

Table 1: Statistics of the data sets used for training, development and evaluation, respectively. M and F refer to male and female speakers, respectively.

detailed statistics for our chosen data splits. Keeping with standard practice, the speakers in the development and evaluation sets were kept disjoint from the speakers in the training set.

We set up an open vocabulary broadcast news task. In order to build the vocabulary, we used written Hindi corpora from the EMILLE monolingual Hindi text corpus [13]. The text was suitably pre-processed and normalized. For example, HTML tags and special characters were deleted, Hindi/Roman numerals were written as Hindi words, consistently occurring Unicode errors were fixed, etc. Words with very small counts in the text corpus (< 5) were discarded to obtain a final list of 29924 words. With this word list, the out-of-vocabulary (OOV) rates on the development set and evaluation sets were 1.8% and 2.28%, respectively.

We extracted text from the EMILLE corpus, restricted to the dictionary words, to train our language model. We also included transcripts from our 4-hour training set totaling over 6 million word tokens in the language model training text.

4. Graphemic and Phonemic Models

Hindi has a largely phonetic orthography i.e., words in Hindi are mostly pronounced exactly as they are written. However, there are a few exceptions to this rule: one of them is due to the schwa deletion phenomenon in Hindi which also occurs in many other Indo-Aryan languages. Consonants in written Hindi are associated with an implicit schwa ([ə]), but implicit schwas are sometimes deleted in pronunciation. The orthography does not provide any indicators of where the schwas might be dropped. For example, the word badal ([bədəl]) retains the schwas after the first two consonants; however, in the word badali ([bədli:]), the inherent schwa after [d] is not pronounced. There are many rules that govern when schwas are not pronounced [14].

We experiment with both graphemic and phonemic models to learn about their effect on ASR performance. We start with a purely graphemic system (G_0) that deterministically maps Hindi Unicode characters in a word to a sequence of graphemes. We used a total of 46 graphemes: 11 for the non-schwa vowels, 2 for the consonantal diacritics (anusvara and visarga) and 33 for the consonant letters. The next variant considers graphemic models with word boundary context information (G_1) [15]. Each grapheme now has an accompanying marker that specifies whether it appeared at the beginning (_B), inside (_I) or at the end (_E) of a word. The transcription for the word, / h a m / for example, changes to / h_B a_I m_E /. The idea here is that distinguishing the grapheme with information about its position in a word might allow the acoustic models to implicitly learn phonetic rules.

Next, we build two phoneme-based systems, P_0 and P_1. We implemented a rule-based schwa deletion algorithm as described in [16] and apply it to grapheme sequences for words to derive word pronunciations. As a preprocessing step, this algorithm also handles the pronunciation of nasalization diacritics (such as "anusvara" and "visarga"). We note that our implemention does not include any morphological decomposition; thus, compound words and multi-morphemic words might end up having erroneous pronunciations.

An anusvara diacritic appearing at the end of a word results in the nasalization of the preceding vowel. In P_0, these nasalized vowels are distinguished from their non-nasalized counterparts and represented as separate phonemes. In P_1, we replace all nasalized vowels with the vowel phoneme followed by a single nasalization phoneme (a "meta phoneme", of sorts, representing nasalization). Considerable research proves that, in many languages, all nasalized vowels should be distinctly modeled, e.g., as phonemes in Portuguese [17] and as finals in Mandarin [18], but conversely, orthographic conventions suggest that nasalization is not as tightly bound to the vowel as fronting and height (e.g., Devanagari transcribes vowel nasalization using a diacritic, Portuguese using a tilde accent, and Pinyin using a coda nasal consonant). Because of the loose association between nasalization and the vowel, we form the hypothesis that a small training dataset might adequately train a separate coda-nasal acoustic model, even when it is inadequate to train acoustic models for every phonologically distinct nasalized vowel in the language; experimental comparison of models P_0 and P_1 tests this hypothesis.

5. Knowledge-based Language Model Adaptations

Hindi is a morphologically rich language. We explore two unsupervised techniques to discover some of the morphological patterns in the language. These learned patterns are then incorporated into the language model of our ASR systems.

5.1. Using Word Segmentation

Identifying stems and suffixes in Hindi words has been explored in prior work (see e.g. [19, 20] and references therein). Our approach is most similar to that of [20] who also employ a statistical analysis to discover stems. However, since our focus is not on obtaining a linguistically precise stemmer, we use a simpler approach devoid of various heuristics used to reduce erroneous segmentations.

Our approach uses an unsupervised algorithm that automatically discovers stems and suffixes from a corpus vocabulary. The pseudocode of this iterative algorithm is described in Algorithm 1; it requires a corpus vocabulary, a stem frequency threshold (θ_P) and a suffix frequency threshold (θ_S) as inputs. Step 1 corresponds to accumulating stems and suffixes by splitting all the words in the vocabulary, at every character (according to the Unicode encoding). We define a bi-partite graph that represents all the stems and suffixes as its partite sets with edges between vertices corresponding to a valid word in the given vocabulary. Then we restrict this graph to the maximal set of stems and suffixes such that each stem has at least θ_S different suffixes and each suffix has at least θ_P different stems. This maximal set is found using a subroutine Prune which iteratively removes the stems and suffixes corresponding to every vertex whose degree is below the corresponding threshold.

Now, a word could be split in more than one way. For example, in Step 1, the word uṭhātā is split at every character as, u-

Algorithm 1 Data-Driven Word Segmentation

Require: List of words, V and thresholds θ_P, θ_S

1: Let $P := \{x \mid \exists y, xy \in V\}$ and $S := \{y \mid \exists x, xy \in V\}$ /* all stems and suffixes */

2: Let G be the bipartite graph with partite sets P and S, and edge set $E := \{(x, y) \mid xy \in V\}$.

3: $E' = \text{Prune}(E)$

4: $\forall x \in P$, let $\mu(x) := \text{length}(x) \cdot \sum_{y:(x,y)\in E'} \deg(y)$

5: $E'' = \{(x, y) | (x, y) \in E', (x, y) = \underset{(x',y'):x'y'=xy}{\text{argmax}} \mu(x')\}$

6: **return** $\text{Prune}(E'')$

FUNCTION Prune(E)
$t := 0, E_0 := E$
repeat
  $t := t + 1$
  $E_t := \{(x, y) \mid (x, y) \in E_{t-1}, \deg_{t-1}(x) \geq \theta_P, \deg_{t-1}(y) \geq \theta_S\}$
  where $\deg_{t-1}$ is the degree in the graph defined by E_{t-1}
until $E_t = E_{t-1}$
return E_t
END FUNCTION

-ṭhātā, uṭh- -ātā, uṭhā- -tā and uṭhāt- -ā.[2] After Step 3, only the splits uṭh- -ātā and uṭhā- -tā are retained. However, we need to identify at most one way to segment each word. Among the multiple segmentations retained after Step 3, we choose the one that maximizes the weight of the stem, where the weight function μ is defined in Step 4.[3] To compute the weight $\mu(x)$ of a stem x, we accumulate the degrees of all the suffixes connected to x in the pruned graph; this quantity is further scaled by the length of x (number of characters) to account for the fact that a longer stem is more desirable, in the sense that it conveys more information. In Step 5, we only retain edges corresponding to stems with the maximum μ among all its possible suffixes (ties are broken arbitrarily). At this point some of the stems and suffixes may have their degrees drop below the thresholds θ_P and θ_S, respectively. So we make a second and final call to Prune in Step 6 to obtain our final set of edges, which specifies a unique way to segment each word (if at all) into a stem and a suffix. E.g., the word uṭhātā is split as uṭh- -ātā.

The language model training corpus can be rewritten using the above word segmentation and an N-gram language model (encoded as an FST, L_{WS}) can then be trained on this text.

5.2. Using Inflectional Agreement

In Hindi, adjectives are often inflected to agree with the case, number and gender of the nouns they modify [21]. Further, an inflected adjective is often directly followed by the noun that it modifies. Since this pattern is very frequent, we devise a targeted model to capture it.[4]

We consider three types of inflected words, denoted as ā, ī and e (reflecting the ending of the inflected word). Morphologically, not all words ending in these suffixes are inflected adjectives. We consider a word to be inflected only if it occurs with each of the three endings more than a threshold θ_I times in the corpus. Further, we shall use the corpus to identify which type of inflected words can precede a word and which ones cannot, as described below.

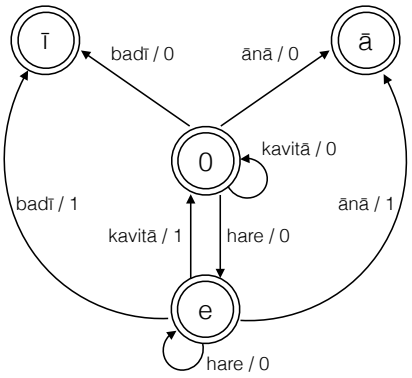

Figure 1: Structure of the "inflectional agreement" FST. All four states and a few example arcs with labels and weights are shown.

For each type $\tau \in \{\text{ā}, \text{ī}, \text{e}\}$ and each word $w \in V$, we define $\text{count}(\tau, w)$ as the number of instances in the corpus when the word w was preceded by an inflected word of type τ. Let $\text{count}^*(w) = \sum_{\tau' \in \{\text{ā},\text{ī},\text{e}\}} \text{count}(\tau', w)$ denote the number of times when w is preceded by any inflected word. Also, let $\text{count}(w)$ denote the total number of occurrences of w in the corpus.

We say that a word w is *incompatible* with type τ if

$$\frac{\text{count}(\tau, w)}{\text{count}^*(w)} < \alpha \qquad \text{and} \qquad \frac{\text{count}^*(w)}{\text{count}(w)} > \beta, \qquad (1)$$

where $0 < \alpha, \beta < 1$ are appropriately chosen parameters. That is, w is incompatible with τ if it is often preceded by an inflected word, but not often by an inflected word of type τ.

We build an FST, L_{IA}, to encode inflectional agreement, as discovered above. The purpose of this FST, which accepts any string of words, is to assign a penalty whenever an inflected word is followed by an incompatible word. It consists of four states to remember the type of the last word seen (ā, ī, e or null). Figure 1 shows a few example arcs, leaving from the states 0 and e on words ānā, baḍī, hare and kavitā. These words are of types ā, ī, e and null respectively, which determines the states to which the corresponding arcs lead to; kavitā, baḍi and ānā are incompatible with type e, resulting in penalties on the corresponding arcs leaving from that state.

6. Experiments

6.1. Experimental Setup

Our ASR systems were configured using the Kaldi toolkit for ASR [8]. As the acoustic features, we used standard Mel-frequency cepstral coefficients (MFCCs) with delta and double delta coefficients; linear discriminant analysis (LDA) and maximum likelihood linear transform (MLLT) feature-space transformations were applied for feature-space reduction. These acoustic features served as the observations to our Gaussian

[2] ṭhā consists of two characters, the consonant ṭh and the symbol corresponding to the vowel ā. Note that in the split ṭh- -ā, -ā stands for the vowel symbol, and not the full vowel character ā.

[3] $\deg(y)$ in Step 4 of Algorithm 1 refers to the degree of the vertex y in the pruned graph with edge set E'.

[4] This pattern also holds for an inflected verb, which is often directly followed by a form of honā (to be). Both the inflected verb and the form of honā agree with the subject of the verb (and hence with each other).

Graphemic/phonemic model	Language Model (LM)	Dev WER	Eval WER
G_0	LM_0	14.36	14.22
G_1		12.91	13.36
P_0		12.61	13.55
P_1		12.56	13.44
P_1	LM_0 + IA	12.49	13.40
	LM_0 + WS	12.45	13.32
	LM_0 + IA + WS	**12.38**	**13.24**

Table 2: WERs on the development/evaluation sets using different systems.

mixture model (GMM) based acoustic models. We built speaker-adapted GMMs using feature-space maximum likelihood linear regression (f-MLLR) for tied-state triphones. These GMMs were subsequently used to build subspace Gaussian mixture models (SGMMs) that are implemented in Kaldi as described in [22]. Our Hindi ASR system, along with a training recipe, has been hosted online[5] for public access.

6.2. Experiments with Graphemic and Phonemic models

The first four rows of Table 2 shows WERs on the development and evaluation sets for different grapheme and phoneme-based SGMM models with a trigram language model trained on the EMILLE text (denoted by LM_0). Using grapheme-based models with word-boundary information (G_1) gives a significant reduction in WER on both the development and evaluation sets, over graphemic models without the word-boundary context (G_0). Hindi exhibits phonetic variations that are associated with word endings. The reduction in WER suggests that such rules might be implicitly learned by incorporating word boundary information. This kind of enhancement of graphemic systems with implicit phonetic information has been employed in prior work on Arabic LVCSR [15].

The phonemic models P_0 and P_1 show small WER improvements compared to the graphemic model G_1 on the development set (and none on the evaluation set). We conjecture that for languages with orthographies of high grapheme-to-phoneme correspondence and sufficient amounts of training data, it might be sufficient to use graphemic based acoustic models with word-boundary context that implicitly captures phonetic information.

The ASR system using nasalized vowels (P_0) performs comparably to the system using a single meta-phone representing vowel nasalization at the end of words (P_1). This suggests we can adequately train a separate coda-nasal acoustic model when the training data might be insufficient to reliably estimate the nasalized vowel units.

6.3. Experiments with Language Model Adaptations

The last three rows of Table 2 show the effect of incorporating the language model adaptations described in Section 5, namely inflectional agreement (IA) and word segmentation (WS). The lattices from the base system were rescored using the two FSTs, L_{WS} and L_{IA} from Section 5. The language model scaling factor was tuned on the development set. In constructing L_{WS}, we set $\theta_P = 4$ and $\theta_S = 30$ in Algorithm 1 and trained a 4-gram language model.[6] θ_P, θ_S, α and β were tuned using a coarse grid search such that the resulting stem-suffix splits (for L_{WS}) and inflectional agreement among word pairs (for L_{IA}) were most meaningful; this was determined by a manual inspection. In constructing L_{IA} in Equation 1, we set $\alpha = \beta = 0.1$.

[5] https://sites.google.com/site/hindiasr/

[6] Since the stems and suffixes are units of different length compared to the full words, we used a higher-order N-gram model.

LM of System 1 using DNN	LM of System 2 using SGMM	Dev WER	Eval WER
LM_0	-	11.98	12.66
LM_1	LM_0	11.31	11.61
	LM_0+IA+WS	**11.12**	**11.50**

Table 3: Final WERs using lattice interpolation with DNN-based models.

We see modest WER improvements using both language model adaptations. It is informative to examine the kinds of errors that were corrected by these incremental models. For example, the base system recognized part of an utterance as kā yog instead of kā yug as they are acoustically confusable. However, yog being a feminine noun, cannot be preceded by a masculine postposition kā. If the word yog appears frequently enough (in appropriate contexts), we will infer its gender and incorporate a penalty for kā yog in L_{IA}. On the other hand, an N-gram language model would favor kā yug over kā yog only if the former word bi-gram appeared frequently.

Lattice interpolation with DNN models: We used the Deep Neural Network (DNN) training recipe in Kaldi (with RBM pre-training and frame-level cross-entropy training [23]) to build DNN-based acoustic models. The first row in Table 3 shows the significant WER improvements compared to the SGMM systems in Table 2. The next two rows show the results from combining two different ASR systems using lattice interpolation. The best results are obtained by interpolating lattices from a DNN-based system and an SGMM-based system. Further, using a different language model LM_1 for the former led to improved results. LM_1 was built using a small text corpus of about 200,000 word tokens, obtained from transcripts from AIR-Bhopal (disjoint from the development and evaluation data). As shown in Table 3, we see small but consistent WER improvements by using our language model adaptations.[7]

7. Conclusions

This work describes a new large-vocabulary broadcast news ASR system in Hindi. We study various graphemic and phonemic acoustic models; graphemic models with word-boundary markers perform almost as well as our phonemic models. We develop knowledge-based language model adaptations for Hindi, derived from a large text corpus and demonstrate small improvements in ASR performance.

This work provides a competitive baseline system as a starting point and suggests many possible directions for future work. Our adaptations could be further improved by incorporating more morphophonemic constraints (e.g., schwa deletions in compound words). The models in this work could also be potentially used to bootstrap ASR systems for other Indian languages that are similar to Hindi in morphology (such as Marathi, Gujarati, Punjabi, etc.).

[7] We note that L_{WS} uses a 4-gram language model. To rule out the possibility that the improvement is solely due to having a higher order language model in the system, we considered rescoring the language model LM_0 in System 2 in the second row of Table 3 with a 4-gram language model. This gives WERs of 11.24 and 11.57 on the development and evaluation sets, respectively.

8. References

[1] M. Kumar, N. Rajput, and A. Verma, "A large-vocabulary continuous speech recognition system for Hindi," *IBM journal of research and development*, vol. 48, no. 5.6, pp. 703–715, 2004.

[2] M. Rajat, Babita, and K. Abhishek, "Domain specific speaker independent continuous speech recognition using Julius," in *Proc. of ASCNT*, 2010.

[3] K. A. S. Arora, B. Saxena and S. S. Agarwal, "Hindi ASR for travel domain," in *Proc. of COCOSDA*, 2010.

[4] G. Sivaraman and K. Samudravijaya, "Hindi speech recognition and online speaker adaptation," in *Proceedings of ICTSM*, 2011.

[5] K. Kumar, R. K. Aggarwal, and A. Jain, "A Hindi speech recognition system for connected words using HTK," *International Journal of Computational Systems Engineering*, vol. 1, no. 1, pp. 25–32, 2012.

[6] K. Bali, S. Sitaram, S. Cuendet, and I. Medhi, "A Hindi speech recognizer for an agricultural video search application," in *ACM Symposium on Computing for Development*, 2013.

[7] A. Mohan, R. C. Rose, S. H. Ghalehjegh, and S. Umesh, "Acoustic modeling for speech recognition in Indian languages in an agricultural commodities task domain," *Speech Communication*, vol. 56, pp. 167–180, 2014.

[8] D. Povey, A. Ghoshal *et al.*, "The Kaldi speech recognition toolkit," *Proc. of ASRU*, 2011.

[9] "All India Radio News," Available from www.newsonair.com.

[10] P. Boersma and D. Weenink, "Praat: doing phonetics by computer [computer program]." Version 5.3.51, Retrieved on 2 June 2013 from http://www.praat.org/, 2013.

[11] T. J. Hazen, "Automatic alignment and error correction of human generated transcripts for long speech recordings." in *Proc. of Interspeech*, 2006.

[12] M. Elmahdy, M. Hasegawa-Johnson, and E. Mustafawi, "Automatic long audio alignment and confidence scoring for conversational Arabic speech," in *Proc. of LREC*, 2014.

[13] P. Baker, A. Hardie *et al.*, "EMILLE, A 67-Million Word Corpus of Indic Languages: Data Collection, Mark-up and Harmonisation." in *Proc. of LREC*, 2002.

[14] M. Ohala, *Aspects of Hindi phonology*. Motilal Banarsidass Publisher, 1983.

[15] F. Diehl, M. J. F. Gales, X. Liu, M. Tomalin, and P. C. Woodland, "Word boundary modelling and full covariance gaussians for Arabic speech-to-text systems." in *Proc. of Interspeech*, 2011.

[16] M. Choudhury, "Rule-based grapheme to phoneme mapping for Hindi speech synthesis," in *90th Indian Science Congress of the International Speech Communication Association (ISCA), Bangalore, India*, 2003.

[17] H. Meinedo, D. Caseiro, J. Neto, and I. Trancoso, "Audimus. media: a broadcast news speech recognition system for the European Portuguese language," in *Proc. of PROPOR*, 2003.

[18] C. Huang, Y. Shi *et al.*, "Segmental tonal modeling for phone set design in Mandarin LVCSR," in *Proc. of ICASSP*, 2004.

[19] A. Ramanathan and D. D. Rao, "A lightweight stemmer for Hindi," in *Proceedings of EACL*, 2003.

[20] A. K. Pandey and T. J. Siddiqui, "An unsupervised Hindi stemmer with heuristic improvements," in *Proceedings of the second workshop on Analytics for noisy unstructured text data*, 2008.

[21] M. C. Shapiro, *A primer of modern standard Hindi*. Motilal Banarsidass Publ., 1989.

[22] D. Povey, L. Burget *et al.*, "The subspace Gaussian mixture model—A structured model for speech recognition," *Computer Speech & Language*, vol. 25, no. 2, pp. 404–439, 2011.

[23] Veselỳ, Karel and Ghoshal, Arnab and Burget, Lukás and Povey, Daniel, "Sequence-discriminative training of deep neural networks." in *Proc. of Interspeech*, 2013.