Training Hybrid Models on Noisy Transliterated Transcripts for Code-Switched Speech Recognition

JHU/GOVIVACE Challenge Submission

Matthew Wiesner, Mousmita Sarma[3], Ashish Arora[1], Desh Raj[1], Dongji Gao[1], Ruizhe Huang[1], Supreet Preet[3], Moris Johnson[3], Zikra Iqbal[3], Nagendra Goel[3], Jan Trmal [1] Paola García, Sanjeev Khudanpur

Preview

Training Hybrid Models on Noisy Transliterated Transcripts for Code-Switched Speech Recognition

Hybrid Models — Wide Residual Networks and BLSTMs

Noisy — Found data are noisy

Transliterated — Words are written in both Indic and Latin scripts.

Hybrid Models — nnet_pytorch

- Kaldi with pytorch-based neural network training using pychain
  - https://github.com/m-wiesner/nnet_pytorch/tree/conda_install
  - https://github.com/YiwenShaoStephen/pychain
- All minibatches are created randomly, on-the-fly, with SpecAugment-like perturbations and variable-width chunks.
- To support truly random mini-batching, numerator lattices are not used.
  - The single best pdf-id sequence is used as the target and the gradients are smoothed across time to mimic a lattice of multiple possible alignments
- Adam Optimizer
- Training and decoding otherwise mimics Kaldi-style training of neural networks

Hybrid Models

- BLSTM and WideResidual networks
  - All performed comparably. The BLSTM was slightly better
- Multilingual training and pretraining
  - Pretraining the BLSTM on 960h of Librispeech
  - Multilingual training on the Hindi and Bengali data starting from the pretrained Librispeech model
  - Pretraining seems to help slightly. Results from multilingual training experiments were inconclusive
- Final models were combinations of monolingual and multilingual models trained from scratch, and initialized with the Librispeech model.

Noisy Data

- Errors in Speaker labels
  - Speaker re-labeling
- Repeated transcripts in test data
  - Partition test-data into duplicate and non-duplicate sets for analysis
- Segmentation and transcription errors
  - Cleaning the transcripts is important!

Noisy Data — Speaker Relabeling

- Many lectures had sign-off statements in which speakers identify themselves
- The sign-off statements did not agree with the speaker labels
- We ran an x-vector based speaker identification system
  - Close to 100% agreement between the xvector-based system and sign-off statements
- Assuming the x-vector based system is correct, all speakers seen in training are also seen in both the test and blind test sets
- There are very few unique speakers
  - Closed-Speaker ASR task —> Models are prone to overfitting

Noisy Data — Speaker Relabeling

- Updated number of speakers
    - Note that all speakers are seen in the training set

	Train	Test	Blind	Total
# Spks Hindi	7	4	5	7
# Spks Bengali	10	7	8	10

Noisy Data — Transcript Deduplication

- Most lectures from which test set segments were drawn were also seen in the training set.
  - WER can be driven artificially low using bad models with an overfit language model, to the point where HMM-GMMs perform comparably to Deep-learning based ASR approaches.
  - Greatly reduces the importance of good acoustic modeling
  - For about 50% of the test set recordings more than 80% of all utterances were seen in the training data.

Noisy Data — Transcript Deduplication

- Created 2 new test set partitions for tuning to prevent overfitting
  - Recordings with >80% of utterances also present in training were assigned to a new test set called *Dup*
  - All other recordings were placed in a test set called *NoDup*
  - Tuning was always performed on *NoDup*
    - HMM-GMMs perform significantly worse than Deep learning approaches on the *NoDup* set, as expected.

Noisy Data — Cleanup

- Transcription and segmentation errors were significant
- Two approaches explored for cleaning transcripts
  - Resegmentation
  - Resegmentation and Data removal
- Resegmentation was challenging:
  - Long stretches of speech get erroneously mapped to <unk> and SIL which biases training to frequently produce no output
- Tossing segments that differed from reference significantly worked better than just resegmentation

System	**Split**		
	NoDup	**Whole**	**Dup**
WRN	23.7	17.2	7.8
WRN + Resegmented Cleanup	24.5	18.7	8.8
WRN + Toss Cleanup	**21.5**	**15.2**	**5.9**

Transliteration

- Many words, mostly technical, are written in both Indic and Latin Scripts

लिंक्ष — Linux

- Language model probability mass is spread over too many feasible alternatives
    - Boosts the relative scores of incorrect paths compared to the sum total of paths with valid alternative transcripts
- Pronunciation lexicons use disjoint phoneme sets for words written in both the Indic and Latin scripts
    - Redundant modeling units result in sparse training data for many triphonemes
    - Acoustic model probability mass is spread over too many feasible alternatives

Alternative orthographic forms and pronunciations should be merged!

Transliteration — Gathering Transliteration Pairs

- Hindi
  - All words written in the Devanagari script in test or occurring 10+ times in the training were paired with English words where applicable.
    - 968 word pairs
- Bengali
  - A semi-automated procedure based on acoustically confusable word-types produced candidate pairs for manual verification.
    - 236 word pairs

Transliteration — Transcript Normalization

- All transliterated pairs were mapped to their Latinate forms
- Language models were trained directly on the transliterated text
- We only use transliterated WER

Transliteration — Lexicon Normalization

- Phoneme sets are unified by using the IPA
- Lexicons are obtained via G2P
- A Phonetisaurus G2P model is trained on English and Hindi/Bengali Lexicons to produce all pronunciations
  - Seed-lexicon for Hindi and Bengali are obtained from Wikipron. For English, arpabet phonemes in the provided lexicon were remapped to the IPA
- Phonemes shared between English and Hindi/Bengali are “tagged” with a language marker
  - Enables further splitting when there is sufficient acoustic evidence
- All pronunciations, whether derived from the Indic and Latinate word-form, were kept after remapping transliteration pairs to their latinate forms

Transliteration — Accented pronunciation of English words

- Many retained pronunciations correspond to:
  - American or British pronunciations of English words
  - Erroneous pronunciations of Hindi/Bengali words
- We discover new, possibly Indian accented pronunciations for words by decoding the training data with a phoneme-level language model
  - Phoneme sequences are paired with time-aligned, word-level reference transcripts
- Erroneous pronunciations are pruned by retaining only the most likely alternative pronunciations according to a greedy selection strategy

Transliteration — Experiments

System	Split		
	NoDup	**Whole**	**Dup**
Baseline HMM-GMM	27.5	20.3	9.8
Phonetic	26.6	22.2	15.7
+ Transliteration Map	26.4	19.2	8.5
+ Learned Lexicon	25.5	18.0	7.2
WRN Learned Lexicon (1)	21.4	15.0	**5.5**
WRN Learned Lexicon (2)	**21.1**	**14.8**	5.6

- The unified phonetic lexicon improves performance on *NoDup* but ***hurts*** performance on the other test sets.
- Mapping transliteration pairs to their latinate forms for language modeling may help slightly
- The lexicon learning additionally improves performance.
- All combined, our approaches for dealing with transliterated text gave 10% relative improvement over the baseline system

Final Models

- Our best performing systems were BLSTMs pretrained on 960h of Librispeech
- Our approaches for dealing with transliterated speech worked well on Hindi, for which we had close to ground truth knowledge of transliteration pairs
  - Did not change performance in Bengali, for which we had many fewer pairs
- We used an expanded lexicon in decoding to which we added English words from CMU-dict as well as words scraped from technical web material in Hindi
- Our final systems rescored lattices with an RNNLM trained on the training transcript augmented with some web-scraped technical material in Hindi.
- The best performing systems in each language were combined via MBR decoding

Conclusion

- Good data-preparation is fundamental to training models!
- Transliteration pairs can be a valuable resource in handling codeswitched speech.

Thanks!