Transcribing Continuous Speech Using Mismatched Crowdsourcing

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

Mismatched crowdsourcing derives speech transcriptions using crowd workers unfamiliar with the language being spoken. This approach has been demonstrated for isolated word transcription tasks, but never yet for continuous speech. In this work, we demonstrate mismatched crowdsourcing of continuous speech with a word error rate of under 45% in a large-vocabulary transcription task of short speech segments. In order to scale mismatched crowdsourcing to continuous speech, we propose a number of new WFST pruning techniques based on explicitly low-entropy models of the acoustic similarities among orthographic symbols as understood within a transcriber community. We also provide an information-theoretic analysis and estimate the amount of information lost in transcription by the mismatched crowd workers to be under 5 bits.

Index Terms: Speech transcriptions, Crowdsourcing, Noisy Channel Models, Information-theoretic Analysis

1. Introduction

Crowdsourcing has been demonstrated as a viable technique to collect transcriptions for large speech corpora [1, 2, 3]. However, this technique relies on the availability of native speakers online, of the language being transcribed. This constraint makes this approach applicable only to a small fraction of the world's languages [4].

Mismatched crowdsourcing [5] was introduced as a potential approach to circumvent the need for a crowd of native speakers, by using crowd workers unfamiliar with a language to transcribe speech in that language. Transcriptions resulting from native-language crowdsourcing suffer a labeling word error rate (LWER) of 20%, as compared to less than 5% WER for professional transcription [2]. Automatic speech recognition (ASR) trained using crowdsourced transcriptions is less accurate than ASR trained using professional transcriptions, but [2] found that the reduced cost (and consequent increased quantity) of crowdsourced transcriptions completely eliminated the difference: an ASR trained using 160K words of crowdsourced text performed as well as an ASR trained using 80K professionally transcribed words. Mismatched crowdsourcing results in LWER higher than that of native-language crowdsourcing, but the impact on ASR performance has not yet been measured: instead, this paper focuses on the problem of minimizing LWER.

The methods of mismatched crowdsourcing are similar to those of ASR. For example, since crowd workers know neither the phone set nor the vocabulary of the language they are transcribing, these knowledge sources must be provided by the experimenter. Consequently, recovering a native language transcription suffers the same computational problems as ASR. Accuracy of mismatched crowdsourcing is best when many crowd workers transcribe the same utterance, because each crowd worker's noisy transcription provides some information about the phonetic content of the utterance. Since each crowd worker makes mistakes, however, the entropy of the decoding graph increases linearly with the number of crowd workers; the size of a complete decoding graph is consequently exponential in both the number of crowd workers and the length of the utterance. Methods reported in [5] used a complete decoding graph, which is effective for mismatched crowdsourcing of isolated words, but does not scale to continuous speech.

This paper therefore considers the problem of optimally pruning the decoding graph for mismatched crowdsourcing, using a series of phonetically-motivated and graph-theoretic pruning steps designed to eliminate as much ambiguity as possible prior to composition with the language model. Results are analyzed both in terms of LWER, and by estimating the entropy of the composed decoding graph using an original extension to continuous speech of the information theoretic analysis of [5].[1]

Organization. Section 2 presents an initial architecture for decoding continuous speech using mismatched crowdsourcing. In Section 3, we introduce an additional step into the decoding process, called "data filtering." As seen in our experiments, detailed in Section 5, this step has a significant effect on improving decoding accuracy. We also include an information-theoretic analysis of the information lost in transcription by the mismatched crowd workers, in Section 4.

2. System Architecture

Mismatched transcription can be modeled as a noisy channel as shown in Figure 1. The input to the system is text in the foreign language, X which is encoded into speech A by a native speaker. This is fed as input to the "crowd channel" which outputs transcriptions in English orthography, Y. We use a simple repetition code illustrated in Figure 1, when the channel is invoked k times: once for each repetition of A. The outputs $y^{(1)}, \ldots, y^{(k)}$ are modeled as identically and independently distributed (i.i.d.) samples produced by the crowd channel. This is sent to the decoder which is considered to be successful if its output, $\tilde{X}$, matches X.

Ideally, we would like to use the maximum likelihood decoding rule:

$$\tilde{x} = \operatorname*{argmax}_x p(x|y) = \operatorname*{argmax}_x p(y|x) \cdot p(x)$$
$$= \operatorname*{argmax}_x p(y^{(1)} \ldots y^{(k)}|x) \cdot p(x) \quad (1)$$

[1]We use very short segments of continuous speech (up to 2 secs long), in order to make it easier for crowd workers to provide informative transcripts. However, as is pursued in ongoing work, the same approach (combined with additional techniques) is useful in transcribing longer segments of continuous speech, by segmenting the long utterance into (possibly overlapping) short segments.

However, this is not a scalable solution as the state space of the decoding algorithm grows exponentially with k. To ameliorate this, we propose using a merged representation of the transcripts, $\hat{y}$ which is a function of $y^{(1)}, \ldots, y^{(k)}$. For simplicity, if we assume there exists a single $\hat{y}$ from the merged transcripts, the decoding rule now becomes:

$$\tilde{x} = \operatorname*{argmax}_{x} p(x|\hat{y}) = \operatorname*{argmax}_{x} p(\hat{y}|x) \cdot p(x) \qquad (2)$$

We use weighted finite state transducers (WFSTs) (popularly used in speech recognition systems [6]) to implement this decoding rule.

2.1. Channel Mergers

As mentioned above, a scalable approach to decoding mismatched transcripts from a large number of crowd workers is to first obtain a merged representation of the transcripts, before decoding. The main tool we use for merging transcripts is the alignment module of ROVER [7]. Each row in this alignment corresponds to the transcript from a single crowd worker. The - symbols indicate deletions in the alignment. The transcripts were preprocessed in the following ways:

- After removing punctuation and converting all characters to lower case, words that appear in an English dictionary were rewritten phonetically (e.g., name would be rewritten as nYm, where Y is a new symbol standing for the English phone /ey/).
- Certain letter sequences of length two, which stand for a single phoneme and frequently occur in the transcripts (like sh, oo, aa), were replaced by a single symbol (like S, U and A, respectively). When appropriate, the new symbols used were the same as those used for representing English phones in the previous step (e.g., text strings ai, ay, ei and ey). After this, the word boundaries were erased to form a single string of symbols.

In order to obtain more meaningful alignments from ROVER's alignment module, the symbol set of the transcripts (after pre-processing) was partitioned into equivalence classes based on pronunciation: all the vowel symbols were grouped into a single class and the consonant symbols were grouped according to the place of articulation of the Hindi phones closest to them. For instance, the symbols k, g, K (for kh) and G (for gh) were grouped into a single class. Deletion is assigned to a class of its own. Before aligning, each symbol in the transcript was replaced by a class symbol, denoting its equivalence class; after the alignment, the class symbols were replaced by the original symbols. This results in an alignment as shown in Figure 2. Figure 2 illustrates the importance of the use of equivalence classes. If symbols were not replaced by equivalence classes prior to alignment, the highlighted column on the right would tend to be aligned as two columns, one with only d and - and another with only t and -. Then, when extracting transcripts from this merged channel, one could obtain transcripts with the sequence td or dt in lieu of a symbol from this column, increasing the number of decoding errors.

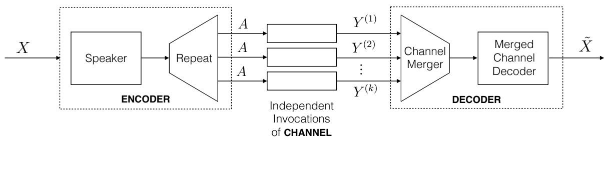

Figure 1: Basic system architecture

p	a	d	e	v	a	-	l	o	k	e	-	m	-	-	a	d	-	o	S	e
p	a	r	o	-	-	-	-	-	p	a	-	n	-	-	e	t	-	a	z	Y
p	-	r	i	v	a	-	r	o	k	Y	-	m	a	h	e	t	-	o	s	Y
p	a	r	i	v	a	d	-	o	k	Y	-	m	a	h	e	d	-	o	s	Y
p	A	r	E	v	a	h	r	U	k	e	m	m	a	h	e	t	-	o	s	Y
p	a	r	i	v	a	-	r	u	k	a	-	-	-	h	e	t	s	u	h	i
p	-	-	a	r	a	h	-	-	-	e	-	-	-	-	a	d	-	o	s	Y

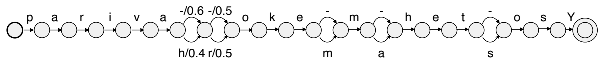

Figure 2: Example: transcripts, by 7 crowd workers who do not speak Hindi, of the utterance "parivāro ke mahatv se." Transcripts are aligned using ROVER, and merged to create a WFST.

In order to control complexity without frequently deleting the best path, the WFST confusion network is pruned prior to further analysis. Pruning leverages a small number of phonetic similarity heuristics to avoid pruning correct answers, as follows. Firstly, we retain the most frequent class symbols in each column (possibly more than one per column in case of ties). In addition, in every column in which deletion is the most frequent class symbol, we retain the second most frequent class symbol(s) as well. Then, each class symbol retained in a column is replaced by the most frequent symbol(s) in that class. This pruned alignment can be represented as a WFST, as shown in Figure 2. The weights on the arcs are only shown on the first few arcs (for visual clarity); these weights are derived from the frequencies of the symbols in each column.

2.2. Contextual Weighting

From the data, we observe that there are several instances when erroneous transcriptions occur in bursts. For instance, the highlighted rows in Figure 2 show two transcripts which match the others towards the beginning and the end, but are quite different in the middle. We model this kind of behavior by using a variant of the power iteration method [8] to assign a contextual weight to the symbols, before computing the most frequent symbols in each column.

Let m be the total number of columns in the alignment, and t be the total number of transcripts. Let σ_{ij} denote the class symbol in the j^{th} transcript, appearing in the i^{th} column. For a "window size" d, we define the d-contextual weight $\eta_{ij}^{(d)}$ as follows.

$$\lambda_{ij} = |\mathcal{K}| \qquad \text{where } \mathcal{K} = \{k \mid \sigma_{ik} = \sigma_{ij}\}$$
$$\eta_{ij}^{(d)} = \sum_{k:|k-i|\leq d} \lambda_{kj}$$

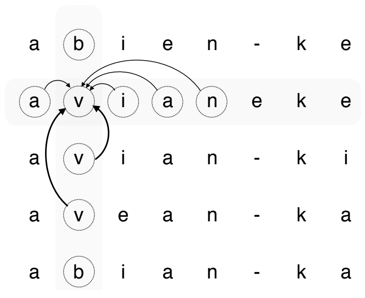

Figure 3: Illustration of contextual weighting corresponding to the Hindi utterance, abhiyān ke.

Here λ_{ij} is the number of occurrences of the class symbol σ_{ij} in the i^{th} column. $\eta_{ij}^{(d)}$ aggregates these weights from neighboring columns (up to a distance d) for each class symbol σ_{ij}. This is illustrated in Figure 3 with $d = 3$; the letter **v** receives weight $\eta_{ij}^{(3)} = 21$. The final weight of a class symbol σ in the i^{th} column is $\sum_{j:\sigma_{ij}=\sigma} \eta_{ij}^{(d)}$. We remark that if the window size $d \geq m - 1$ then $\eta_{ij}^{(d)} = \sum_{k=1}^{m} \lambda_{kj}$ (which is independent of i).

3. Improvements via Data Filtering

Mismatched crowdsourcing is most accurate when many crowd workers transcribe the same utterance. Each transcript contributes information, but each also contributes independent noise. Useful information tends to be represented as repeated orthography, i.e., clusters. We seek to identify an ordering of the transcripts that highly ranks the transcripts that are most similar to the others and gives low ranks to the outliers. We term this approach *data filtering*.

In order to obtain this ranking, each transcript is assigned a score which is defined as the sum of similarity scores between that transcript and all other transcripts for the same utterance. A natural choice for a similarity score would be based on edit distance. However, edit distance between shorter strings tends to be smaller than that between longer strings, leading to shorter strings appearing at the top of the ranking. To immunize against this length effect, we define the similarity score to be the difference between the sum of the lengths of the strings and the edit distance between the strings. We use the sum of the lengths instead of the maximum of the lengths while defining the similarity scores to give more prominence to the longer transcripts; we observe that the longer transcripts tend to capture more information about the speech signal than the shorter ones.

4. Entropy of the Decoding Graph

We use information-theoretic tools to estimate an upper bound of the conditional entropy of the inputs given the outputs of the mismatched channel used for continuous speech. For any probability distribution q, for data distributed as $p(x, y)$ we have

$$\begin{aligned} H(X|Y) &= \mathrm{E}_{(x,y)\sim X,Y}[-\log p(x|y)] \\ &\leq \mathrm{E}_{(x,y)\sim X,Y}[-\log q(x|y)] \end{aligned}$$

We shall define q based on an extension of our decoder from Section 2.1 that produces a multi-transcript alignment for a given y. This alignment consists of a sequence of lists $L_1(y), \ldots, L_n(y)$, where n is a sufficiently large upper bound on the number of words in X. Each $L_i(y)$ is a multi-set consisting of one or more words from the vocabulary X and optionally ϵ denoting an empty string. We will denote the multiplicity of a word w in a multi-set L by $\mu_w(L)$. A path in this alignment specifies a sequence of words $(w_1, \ldots, w_n)$, with $w_i \in L_i(y)$; the string obtained by concatenating these words (after omitting all occurrences of ϵ) is denoted by $< w_1, \ldots, w_n >$.

We define the conditional probabilities $q(x|y)$ using the alignment associated with y. First, we define probabilities $q_i(w|y)$:

$$q_i(w|y) = \begin{cases} \frac{\alpha \cdot \mu_w(L_i(y))}{|L_i(y)|}, & \text{if } w \in L_i(y) \\ \frac{1-\alpha}{|\mathcal{X} \setminus L_i(y)|}, & \text{if } w \notin L_i(y) \end{cases}$$

where α is a tunable parameter. Using these probabilities we define $q(x|y)$ to be the sum (computed using the sum-product algorithm) of probabilities of all paths corresponding to the string $x = \langle w_1, \ldots, w_n \rangle$, while $\hat{q}(x|y) \leq q(x|y)$ is the probability of the most likely such path (computed using the max-product algorithm). If $\hat{w}_i^{\ell}$ is the i^{th} token in the max-product path for $x^{\ell} = \langle w_1^{\ell}, \ldots, w_n^{\ell} \rangle$, then

$$\begin{aligned} H(X|Y) &\leq \sum_{i=1}^{n} \mathrm{E}_{(x,y)\sim X,Y}[-\log q_i(\hat{w}_i^{\ell}|y)] \\ &\approx \sum_{i=1}^{n} \frac{1}{N} \sum_{\ell=1}^{N} -\log q_i(\hat{w}_i^{\ell}|y_\ell) \\ &\approx \sum_{i=1}^{n} \frac{1}{N} \sum_{\ell \in N_0^i} \log |L_i(y_\ell)| + \sum_{\ell \in N_1^i} \log |\mathcal{X} \setminus L_i(y_\ell)| \end{aligned} \quad (3)$$

where N is the vocabulary size, $N_0^i = \ell \mid \hat{w}_i^{\ell} \in L_i(y_\ell)$ and $N_1^i = [N] \setminus N_0^i$. An estimate of the rate of conditional entropy can be derived by dividing the above estimate by the average number of words per utterance.

5. Experiments

Our crowdsourcing experiments were performed using Amazons Mechanical Turk (MTurk) [9]. A total of 278 crowd workers (Turkers) participated in our tasks. 143 of them were only familiar with English; Spanish, French, German and Japanese were the next most frequently listed languages.

We chose Hindi as our foreign language. We extracted Hindi speech from publicly available Special Broadcasting Service (SBS, Australia) radio podcasts described in [5]. This corpus comprised approximately 50 mins of speech from the podcasts, with roughly 10000 word tokens, which was then manually transcribed at the phonetic level by a Hindi speaker. We extracted short segments up to 2 seconds long from this corpus; keeping the segments short made the mismatched transcription task easier for the Turkers. The MTurk task was set up as described in [5]. The Turkers were presented short speech segments in Hindi, up to 2 seconds long and asked to provide English text (often in the form of nonsense syllables) that most closely matched what they heard. Any punctuations or word boundaries specified in the English text were discarded, as these markers had no agreement across workers. This is not very surprising as boundary markers would be very hard to perceive from continuous speech in a language unknown to the workers. Apart from the SBS data, we also extracted speech from Hindi news broadcasts on the All India Radio (AIR) website.[2] These broadcasts were accompanied by corresponding speech transcripts. A total of 200 utterances, approximately 1 second long, were extracted from the news podcasts. 100 of these utterances were randomly chosen as our evaluation data, totaling 391 word tokens; the rest were used as training data (in addition to the above-mentioned SBS data). Each of these 200 AIR utterances were transcribed by 15 distinct Turkers.

Our mismatched transcription system was set up to be a closed vocabulary task; our vocabulary comprised a total of 11419 words appearing in transcripts from the AIR-Hindi Bhopal news station between the months of December 2013 to July 2014. We built a trigram language model using these transcripts, along with additional text from the EMILLE monolingual written corpora [10]; we only extracted sentences that had no out-of-vocabulary words from the EMILLE corpus. As our evaluation set consisted of randomly spliced utterances that did

[2]http://www.sbs.com.au/podcasts/yourlanguage/hindi/

System	WER	PER
Best Individual Channel	77.2	61.5
5-Merged Channel	74.9	49.4
10-Merged Channel	65.7	46.9
15-Merged Channel	64.9	44.7
5-Merged Channel (CW)	71.1	48.6
15-Merged Channel (CW)	60.9	39.8
Combination	60.8	40.5

Table 1: WER/PER on the evaluation set using different systems. CW denotes merged channels with contextual weighting.

System	WER	PER
1-Filtered Channel	68.0	46.0
5-Filtered Merged Channel	54.7	35.7
15-Filtered Merged Channel	62.9	41.9
5-Filtered Merged Channel (CW)	57.3	37.5
15-Filtered Merged Channel (CW)	54.2	35.2
Combination with Filtering	**43.5**	**28.0**

Table 2: WER/PER on the evaluation set using different system configurations, with data filtering.

not necessarily coincide with the beginning or end of a sentence, we trained our trigram language model on N-gram counts without any explicit start-of-sentence or end-of-sentence markers.

The mismatched channel is modeled as a finite memory process using WFSTs. The channel model is trained using pairs of training instances, (x, y), where x is a phonetic sequence in Hindi and y is the corresponding English letter sequence produced by the crowd worker. This model probabilistically maps each Hindi phone in x to every English single-letter or two-letter sequence. The weights on the arcs of this FST model were learned using the EM algorithm [11] to maximize the likelihood of the observed training instances. This FST model mapping Hindi phones to English letters is then composed, after inversion, with a Hindi dictionary FST (mapping Hindi phones to Hindi words) and a Hindi language model acceptor (mentioned above) to build decoding graphs for our mismatched transcription task. The USC/ISI Carmel finite-state toolkit[3] was used for EM training of the WFST model and the OpenFst toolkit [12] was used for all finite-state operations.

Table 1 shows word error rates (WER) and phone error rates (PER) on the evaluation set using systems of varying configurations, without any data filtering. The first row shows the error rates using the best individual channel model. The next three rows show the improvements in error rate from using merged channels with 5, 10 and 15 transcripts, respectively. Increasing the number of Turkers from 5 to 15 significantly reduces the WER (at $p < 0.01$ as measured by the NIST MAPSSWE significance test [13]). The next two rows show the influence of using contextual weighting with the merged channels; this leads to further drops in error rates. The last row in Table 1 shows error rates derived from system combination. This system uses a hypothesis list compiled by accumulating the top-scoring hypotheses from each of the 15 individual channel models. This list is further augmented by the top-scoring hypotheses from all five merged channel systems in Table 1. This system is comparable in WER to the best system used in the combination.

Table 2 clearly demonstrates the importance of data filtering. All the error rates listed in Table 1 are significantly reduced

[3]http://www.isi.edu/licensed-sw/carmel/

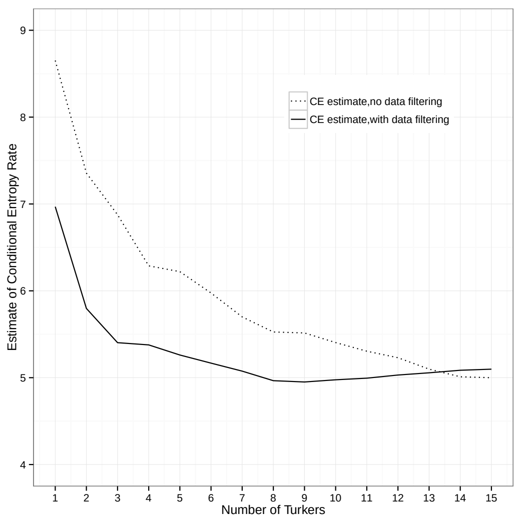

Figure 4: Estimates of conditional entropy (CE) of X given Y on the evaluation set for varying number of Turkers.

by using data filtering. With data filtering in place, a model trained and decoded using the top-5 Turker transcripts performs significantly better than a model that uses all 15 Turker transcripts (rows 2 and 3 in Table 2). This system (54.7%) performs comparably to a system with contextual weighting that uses all 15 Turker transcripts (54.2%). The last row in Table 2 corresponds to system combination using the top-scoring hypotheses from all the other systems shown in Table 2. Unlike in Table 1, the combined system with data filtering significantly improves over the best system used in the combination (at $p < 0.01$).

Figure 4 shows the rate of the conditional entropy estimate derived in Equation 3 plotted against individual channel models with varying number of Turkers. This plot further demonstrates the significance of data filtering; systems combined in the order specified by data filtering results in a tighter bound. The conditional entropy rate using all 15 Turkers is estimated to be under 5 bits per word and further drops to 4.6 bits when we include the merged channel models from Table 2.

6. Conclusions

This work establishes the possibility of deriving speech transcriptions for continuous speech using mismatched crowdsourcing. We propose several techniques capable of scaling effectively to continuous speech and obtain a labeling error rate under 45% for a large-vocabulary continuous speech task in Hindi using transcriptions in English orthography from crowd workers. We also provide a theoretical estimate of the conditional entropy rate of the mismatched channel that evaluates to less than 5 bits. Future work includes analyzing the impact of mismatched transcription on ASR performance and experimenting with a range of different languages, including tonal languages.

7. Acknowledgments

This research was supported in part by a Beckman Postdoctoral Fellowship for the first author. This work was performed under the auspices of the 2015 Frederick Jelinek Memorial Summer Workshop held at the University of Washington in Seattle, and sponsored by Johns Hopkins University, with partial support from NSF IIS Award No 1005411.

8. References

[1] C. Callison-Burch and M. Dredze, "Creating speech and language data with Amazon's Mechanical Turk," in *Proceedings of NAACL HLT Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, 2010.

[2] S. Novotney and C. Callison-Burch, "Cheap, fast and good enough: Automatic speech recognition with non-expert transcription," in *Proceedings of NAACL HLT*, 2010.

[3] M. Eskenazi, G.-A. Levow, H. Meng, G. Parent, and D. Suendermann, *Crowdsourcing for Speech Processing: Applications to Data Collection, Transcription and Assessment*. John Wiley & Sons, 2013.

[4] E. Pavlick, M. Post, A. Irvine, D. Kachaev, and C. Callison-Burch, "The language demographics of Amazon Mechanical Turk," *Transactions of the Association for Computational Linguistics*, vol. 2, pp. 79–92, 2014.

[5] P. Jyothi and M. Hasegawa-Johnson, "Acquiring speech transcriptions using mismatched crowdsourcing," in *Proceedings of AAAI*, 2015.

[6] M. Mohri, F. Pereira, and M. Riley, "Weighted finite-state transducers in speech recognition," *Computer Speech & Language*, vol. 16, no. 1, pp. 69–88, 2002.

[7] J. G. Fiscus, "A post-processing system to yield reduced word error rates: Recognizer output voting error reduction (ROVER)," in *Proceedings of ASRU*, 1997.

[8] G. H. Golub and C. F. Van Loan, *Matrix computations*. JHU Press, 2012.

[9] "Amazon Mechanical Turk," http://www.mturk.com.

[10] P. Baker, A. Hardie, T. McEnery, H. Cunningham, and R. J. Gaizauskas, "EMILLE, A 67-million word corpus of Indic languages: Data collection, mark-up and harmonisation." in *Proceedings of LREC*, 2002.

[11] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical Society, Series B*, vol. 39, no. 1, pp. 1–38, 1977.

[12] C. Allauzen, M. Riley, J. Schalkwyk, W. Skut, and M. Mohri, "OpenFst: A general and efficient weighted finite-state transducer library," in *Proceedings of the Ninth International Conference on Implementation and Application of Automata (CIAA)*, 2007.

[13] "The NIST Scoring Toolkit (SCTK)," Available from http://www.itl.nist.gov/iad/mig/tools/.