ON-THE-FLY USER MODELING FOR COST-SENSITIVE CORRECTION OF SPEECH TRANSCRIPTS

Matthias Sperber[1], *Graham Neubig*[2], *Satoshi Nakamura*[2], *Alex Waibel*[1]

[1]Karlsruhe Institute of Technology, Institute of Anthropomatics and Robotics, Germany
[2]Nara Institute of Science and Technology, Augmented Human Communications Laboratory, Japan

ABSTRACT

We propose an on-the-fly updating framework for cost-sensitive manual correction of automatically recognized speech transcripts. This framework trains cost-models during the transcription process, and does not require the transcriber enrollment necessary in previous work. We use a baseline method that optimizes a segmentation into segments to supervise or not to supervise in a cost-sensitive fashion that minimizes human effort, and introduce a much faster algorithm for computing such a segmentation that can be used for on-the-fly updates. Besides removing the need to carry out enrollments, experiments show that our updating framework results in 28% higher human supervision efficiency than previous cost-sensitive approaches.

Index Terms— Speech transcription, error correction, cost-sensitive annotation, user modeling

1. INTRODUCTION

High-quality speech transcripts are required in a variety of tasks, ranging from training data for automatic speech recognition (ASR) to downstream tasks such as speech translation. Unfortunately, often automatically created transcripts contain too many errors to be useful in practice, and human transcribers must be employed to improve their quality. In this work, we examine how to make this supervision process as efficient as possible by choosing which segments should be supervised in a cost-sensitive manner.

Previous works on supervision for speech transcription showed that it is important to divide the speech into small segments that are convenient to transcribe [1], and that by supervising only low-confident segments of an ASR transcript, supervision time can be greatly reduced at a small loss in number of corrected errors [2, 3]. Studies on cost-sensitive annotation have shown that to maximize supervision efficiency, it is important to model human supervision effort when choosing which data to supervise [4, 5, 6]. Moreover, explicitly optimizing the location of segment boundaries helps focus supervision time on segments with a high proportion of errors, further increasing supervision efficiency [7].

Common to these previous works on cost-sensitive choice of data is a two step process. The first step consists of transcriber enrollment, where the transcriber first transcribes a certain amount of randomly selected data, which is then used to train predictive user models (usually a cost model that predicts supervision time). In the second step, the user models are used to determine which data should be supervised. However, enrollments are time-consuming, costly, and may be impractical, for example in a crowd sourcing situation.

In this paper, we make two contributions over previous methods, which allow us to perform efficient cost-sensitive transcription without need for enrollments. The first contribution (see Section 3) is a framework of updating user models and corresponding choice of supervised data on-the-fly (during the ongoing transcription process). Our second contribution (see Section 4) is a new, more efficient method for choosing which segments of the ASR transcript to supervise based on an iterative search process. This method explicitly optimizes locations and lengths of supervised segments to optimize supervision efficiency, as proposed in [7], which is a computationally difficult task. Computing such segmentations quickly is essential in our on-the-fly updating scenario, because the segmentation must be updated each time the user model is updated in order to incorporate the improved predictions of supervision time.

We use a partly simulated experimental framework that allows us to conduct detailed experiments. First, we demonstrate that our new segmentation algorithm drastically reduces computation time compared to a previous approach, without degrading quality. Next, we show that user models trained on-the-fly are as effective as when trained on balanced enrollment data, and remove the need to spend time on enrollment. We compare our approach to several baselines, showing that it outperforms both cost-insensitive baselines and cost-sensitive baselines without updates in terms of supervision efficiency (error corrections per time). An analysis shows that efficiency gains are attributed to (1) increasingly accurate cost models, (2) adjusting to the actual remaining time budget with each update, and (3) choosing a sensible (although crude) initial cost model that includes cognitive overhead. Finally, we show that our cost-sensitive approach can deal with noisy supervision times, as observed with novice transcribers.

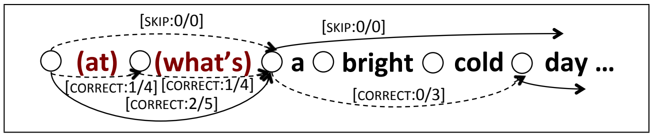

Fig. 1. A graph of different possible segmentations for the ASR transcript of the sentence fragment *"It was a bright cold day ..."*. Edges are segments (some edges are omitted for readability), and nodes potential segment boundaries. Segments are labeled with [supervision mode:predicted utility/predicted cost]. For optimal supervision efficiency, the solid edges might be preferable over the dashed ones.

2. BASELINE MODEL

As a baseline for our work, we use a state-of-the-art annotation framework, SESLA[1] (short for Segmentation for Efficient Supervised Language Annotation) [7]. Given an ASR transcript to be corrected by a human annotator, SESLA determines a segmentation of the transcript, and makes a decision whether or not to transcribe each segment, such that supervision efficiency is optimized. The idea is that we naturally want to concentrate on supervising parts that yield high utility, in terms of number of errors that are removed by supervising them. While choosing very small segments (e.g. single words) would allow us to really concentrate only on parts of maximum utility, longer segments are desirable from a cognitive point of view as they reduce cognitive overhead due to context switches.

Formally, SESLA searches for a segmentation of the N words of an ASR transcript w_1^N into $M{\le}N$ segments with boundaries at $s_1^{M+1}{=}(s_1{=}1, s_2, \dots, s_{M+1}{=}N{+}1)$. A segment boundary marker s_i is interpreted as placing a segment boundary before the s_i-th word (or the end-of-transcript marker for $s_{M+1}{=}N{+}1$). Each segment is further associated with a supervision mode $m_j \in K$, in this work either CORRECT (the segment should be corrected), or SKIP (the segment should not be corrected). This segmentation is selected based on predictive models to estimate supervision cost (here: correction time) and utility (here: number of corrected errors) for any particular segment. For each segment w_a^b and supervision mode k, we denote utility by $u_k(w_a^b)$ and cost by $c_k(w_a^b)$. Note that cost and utility for the SKIP mode are always 0. Figure 1 illustrates the segmentation problem.

We desire a segmentation that maximizes the total utility, while keeping the total cost within a cost budget C:

$$\max_{M;s_1^{M+1};m_1^M} \sum_{j=1}^{M} \left[u_{m_j} \left(w_{s_j}^{s_{j+1}} \right) \right]$$

$$\text{s.t.} \sum_{j=1}^{M} \left[c_{m_j} \left(w_{s_j}^{s_{j+1}} \right) \right] \le C$$

As a cost model, a regression model that predicts supervision time is learned from a transcriber enrollment. Gaussian Process (GP) regression is employed as a Bayesian technique that yields state-of-the-art regression accuracy. In addition, it allows convenient specification of a prior, a feature that we will exploit in this work. As regression features, segment length, duration, and average confidence are used. Utility, defined as the numbers of errors removed from the transcript, is estimated using scaled word confidence scores.

3. ON-THE-FLY UPDATING FRAMEWORK

The baseline method relies on a cost model that is trained via an enrollment, which is costly and takes up time that the transcriber might otherwise have invested in being productive. For instance, in [7] an enrollment of roughly 30 minutes is carried out, and in our simulations 50–100 minutes were required for the cost model to converge (Section 5.2). While investing enrollment time will improve supervision efficiency and pay off in the long run, for transcribers that supervise only a small amount of data, as is common in crowd-sourcing situations, this approach may not be economical at all.

Here, we improve upon this situation by allowing transcribers to be productive from the start. We create an initial segmentation with a crude initial cost model, and iteratively improve the cost model and consequently the segmentation as the transcription is underway. There are several advantages to this approach: (1) The transcriber enrollment is removed, instead the transcriber is productive from the start. (2) The segmentation will grow increasingly efficient due to the improving cost model, starting out rather crude in the beginning, and eventually reaching or surpassing the efficiency that would have been reached using a cost model trained via enrollment. (3) Each updated segmentation will take into account the actual remaining time budget, which may differ from what had been predicted as remaining when the transcriber would reach a certain position in the ASR transcript. This solves an issue pointed out in previous work [7], in which systematic errors in time predictions resulted in considerable over- or underspending of the given time budget. Re-segmenting ensures that the remaining time is used optimally, for example by skipping some of the less promising segments if the remaining time budget had been overestimated.

3.1. Approach to Updating Segmentations

We propose to update the segmentation periodically as in Algorithm 1. We initialize the cost model M_C such that it will rely solely on a prior. The transcription position j is initialized to the first word, and an initial segmentation of the complete ASR transcript is created. The transcriber starts transcribing the first batch of segments from the current position j, until B seconds have passed, at which time j is updated. M_C is updated using the observed supervision times. Finally,

[1] http://msperber.com/research/tacl-segmentation/

Algorithm 1 Supervision & Updating Framework

```
▷ given: $T$ (time budget), $w_1^N$ (ASR transcript), $B$ (batch size), $M_U$ (utility model, fixed)
$M_C \leftarrow M_C^{(0)}$                                ▷ initialize cost model
$j \leftarrow 1$                          ▷ set transcription pos. to first word
SEGMENT($w_1^N, M_C, M_U, T$)
while $t_{\text{remain}} > 0$ do
    SUPERVISEBATCH($w_j^N, \min(B, t_{\text{remain}})$)
    $j \leftarrow$ resulting transcription pos.
    UPDATE($M_C$, newly observed times)
    SEGMENT($w_j^N, M_C, M_U, t_{\text{remain}}$)
end while
```

Algorithm 2 Iterative Penalized Dynamic Programming

```
▷ Initialize lower-/upper-bound segmentations $S_L, S_U$ and penalties $\lambda_L, \lambda_U$
while $u(S_U)/u(S_L) > 1 + \epsilon$ do
    $\lambda' \leftarrow (\lambda_U + \lambda_L)/2$
    $S' \leftarrow$SEGMENTDP($\lambda'$)
    if $c(S_U) \geq c(S') \geq C$ then
        $\lambda_U, S_U \leftarrow \lambda', S'$
    else                                    ▷ i.e., $c(S_L) \leq c(S') \leq C$
        $\lambda_L, S_L \leftarrow \lambda', S'$
    end if
end while
```

the remainder of the transcript that has not yet been supervised is re-segmented using the updated cost model and the remaining time budget t_{remain}, and another batch of B seconds is transcribed. Transcription is stopped when the time budget is exhausted, or the end of the transcript has been reached.

3.2. User Model Priors

To create the initial segmentation, we need a reasonable initial cost model. Since transcribed data from similar users or tasks may not be available, we rely solely on a manually defined prior. We argue that to create a sensible initial segmentation, the cost model can be crude, but should capture two key observations, namely that longer segments take longer to supervise, and that the transcriber will need some time to process the context switch for each new segment. We therefore specify the prior cost as $2 + n$ seconds, where n is the segment length. Considering our transcript correction task, this prior underestimates the actual cost, but we deliberately avoid hand-tuning the prior to simulate a situation where little task expertise is available.

4. SEGMENTATION VIA ITERATIVE SEARCH

To make on-the-fly updates of the segmentation practical, new segmentations must be computed rapidly. Previous work transformed the segmentation problem into a constrained shortest path problem and used a general-purpose integer linear program (ILP) solver to find an approximate solution [7], but this approach is not sufficiently fast for updating segmentations during the transcription process. In this section, we introduce a new segmentation algorithm that we will demonstrate to be dramatically more time- and memory-efficient (s. Section 5.1). Because this segmentation problem is NP-hard, we focus on approximating the solution.

4.1. Description of Algorithm

Consider a penalized segment scoring function that is defined as a linear combination of a segment's utility and cost, $s_{\lambda;k}(i,j) := u_k(w_i^j) - \lambda c_k(w_i^j)$, with a cost-penalty $\lambda > 0$. For a given λ, it is easy to compute a segmentation that optimizes the total penalized score over all segments via dynamic programming (DP).[2] By varying the value of λ, we can obtain different *Pareto-optimal* segmentations with respect to utility and cost, i.e. we cannot increase their utility without increasing cost and vice versa. We desire to find the segmentation that has maximum utility while obeying our cost constraint, which corresponds to one of these Pareto-optimal solutions. Hence the search problem reduces to finding the optimal value for λ. Intuitively, if it turns out that by using the Pareto-optimal segmentation corresponding to a given λ we would be overspending our time budget (we say this segmentation is *infeasible*), we should increase λ, thus penalizing costly segments more strongly. Similarly, if the budget is underspent (the segmentation is *feasible*), λ should be decreased. We follow this simple intuition by iteratively first computing Pareto-optimal segmentations, and then adjusting λ. While finding the optimal value for λ is NP-hard, we can efficiently find an arbitrarily good approximation via a binary search.

According to Algorithm 2, we start by initializing upper and lower bound segmentations S_U, S_L and penalties λ_U, λ_L. We do this by first trying an arbitrary value for λ, and then repeatedly multiplying or dividing it by a constant factor (here: 10), until an (in-)feasible Pareto-optimal solution is found, depending on whether the first segmentation was feasible or not. Henceforth, the upper bound will refer to the lowest-scoring infeasible segmentation seen so far, and the lower bound is the highest-scoring feasible segmentation. We repeat until the gap between utilities associated with our upper and lower bounds is below a threshold ϵ: We consider a new penalty λ', halfway between the upper- and lower-bound values. We compute a corresponding Pareto-optimal segmentation S' via DP, and update the lower or upper bound, depending on whether the segmentation was feasible or not.

The DP algorithm that finds Pareto-optimal segmentations, given a penalty λ, computes the total penalized score a_j of the best segmentation of w_1^j, and keeps back pointers to

[2] Note that the original segmentation problem cannot be solved in polynomial time by dynamic programming due to the global constraint.

find an optimal segmentation as follows:

$$\begin{aligned} a_1 &= \max_{k \in K} s_{\lambda;k}(1,1) \\ a_j &= \max_{i<j}(a_i + \max_{k \in K} s_{\lambda;k}(i,j)) \end{aligned}$$

The DP has computational complexity $O(N^2)$. By limiting the segment size to be at most R (here: 20), the complexity reduces to $O(RN)$. Moreover, the number of iterations of the binary search for λ depends on the approximation threshold ϵ, but generally does not depend on N, so the overall algorithm's complexity is essentially linear in the length of the ASR transcript.

4.2. Computation vs. Supervision Time

According to Algorithm 1, the transcriber would wait idly while the cost model and segmentation are being updated. Depending on the amount of data for cost model training and segmentation, this may take several seconds or even longer, and waiting may be found undesirable. One solution would be to perform updates while the transcriber takes a break. Alternatively, updates might be performed in parallel to the transcription, and the transcriber would be switched to the new segmentation whenever a new segmentation is ready. Note also that training time of our GP regression user model is cubic in the number of training samples, and gets quite slow for more than about 1000 samples. A simple solution would be discarding older samples for long-term transcribers. For simplicity, we abstract from these factors in our analysis.

5. EXPERIMENTS

To evaluate our method, we considered a correction scenario where the transcriber is given an ASR transcription with the goal to remove as many errors as possible, given a 100 minute time budget. As transcription data, we used 10 TED talks[3] (short presentations by skilled speakers, the total length was 104 minutes or 17.8k words). The erroneous transcripts were created using the Janus speech recognition toolkit [8] with a simple TED-optimized setup. The word error rate on our test set was 22.3%, with a total of 3978 errors. The reference transcripts were carefully transcribed and of high accuracy. Our user models are set up similar to [7]: We employ scaled confidences as our utility model, and GP regression with a squared exponential kernel, predicting log times to avoid negative values.[4] Noise and kernel variances for the GP regressors were set to $\log(5\text{ sec})$. We empirically confirmed that these parameters are sensible, but did not fine-tune them because in a practical situation, parameter tuning data might not be available. We used the proposed segmentation algorithm to find a solution within 1% of the optimal solution.

To be able to compare a large number of different settings, we carried out partly simulated experiments. We asked a real, non-expert transcriber to transcribe from scratch 200 segments randomly chosen from TED data, balanced across different lengths and average confidence scores. We measured the time taken to transcribe each segment. Based on this data, we trained an oracle time model via GP regression, and used this model as the gold standard that the tested cost models will try to reproduce. The GP regressor was trained with an identical kernel as the user models, but with the prior set to the best linear fit in terms of least squares error. We then distorted its predictions with multiplicative gamma-distributed noise, to serve as our final oracle time model. The multiplicative noise had a mean of 1 and moderate variance of 0.01, unless otherwise noted. In our simulations, we assumed that the human transcriber behaves according to this oracle time model. We further assumed that the transcriber successfully transforms the ASR transcript of every supervised segment into the corresponding reference transcript, without making mistakes. The batch size B is set to 2.5 minutes, unless otherwise noted. All results are averaged over 10 simulation runs, with the order of talks and noise multipliers chosen at random.

5.1. Segmentation Algorithms

We first compared the computational efficiency of segmenting according to the algorithm proposed in Section 4, as opposed to using GUROBI as an off-the-shelve ILP solver as in the baseline method. GUROBI is highly optimized commercial software and among the fastest ILP solvers available,[5] while our Java implementation of the proposed algorithm is straightforward, with little code level optimization. We used the initial cost model to segment increasingly large subsets of our data. Both algorithms were run on a single processor, and stopped after producing a solution within 1% of their respective upper bounds. Figure 2 shows the results. It can be seen that the proposed method stays within a computation time of around 3 seconds, while the ILP solver needs more than two minutes to segment all data. Further, the proposed method's memory consumption grows only linearly in the number of words, needed to store the DP scores. In contrast, for segmenting the complete dataset, we observed memory consumption one or even two orders of magnitude higher for the ILP solver. Besides being faster, the proposed algorithm has the advantage of being easy to implement without relying on proprietary software. We therefore use the proposed algorithm in the following experiments.

5.2. User Model Convergence

First, we examine the convergence of the cost models in terms of prediction accuracy. A concern that arises when training a cost model on-the-fly rather than on a balanced enrollment

[3] www.ted.com

[4] GP regression was done using GPy: github.com/SheffieldML/GPy

[5] See benchmark on http://plato.asu.edu/ftp/milpc.html

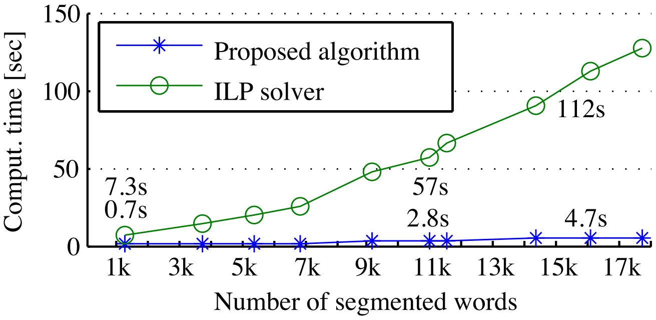

Fig. 2. Computation time needed to segment different amounts of data using an ILP solver or our proposed algorithm.

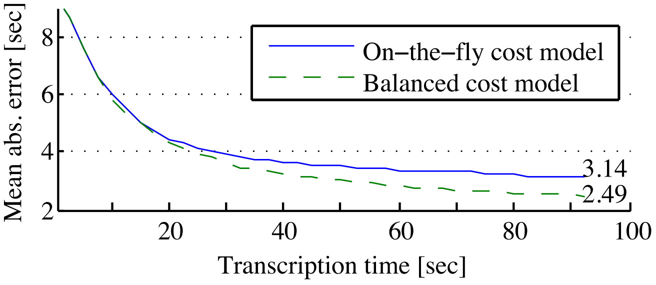

Fig. 3. Accuracy of time prediction models trained on-the-fly or using a balanced selection of segments. The latter converged slightly faster.

training set is that the model might suffer from the resulting data bias. We therefore compare convergence of a model trained on-the-fly to a second model that is trained on a set of segments uniformly sampled from all possible segments, resembling an enrollment on a balanced data set. Prediction accuracy was computed on all possible segments. Figure 3 shows that the user model generalizes well despite skewed training data, almost as well as when trained on balanced data. A detailed analysis revealed that the slight drop in accuracy is attributed to segments of high confidence and relatively long size, which were underrepresented among the segments chosen by the proposed method.

Next, we were interested in whether this drop would make a difference in terms of supervision efficiency. We therefore ran the full simulation with the proposed framework, and then ran it again but replaced the cost model with an enrollment-style cost model with balanced training data of equivalent supervision time at each update. However, we observed no significant difference in the final number of removed errors, presumably because the segments with the less accurate cost predictions were high-confident and long, thus having little chance of getting selected for supervision in the first place.

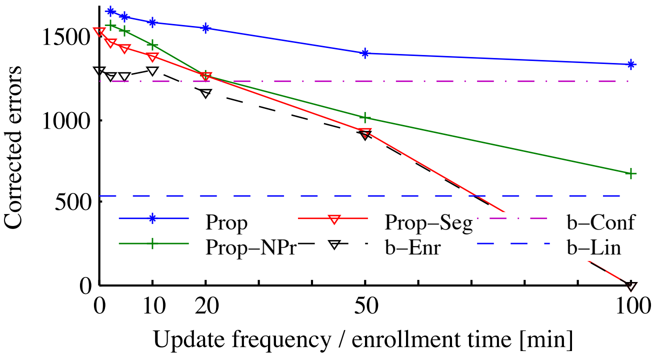

Fig. 4. Errors corrected for the proposed method (*Prop*) and several other settings, plotted against update frequency or enrollment time as described in the text.

5.3. Effectiveness of Updating Framework

In this section, we evaluate the end-to-end effectiveness of the proposed updating framework, compared to several baseline approaches, in terms of the final number of errors removed after 100 minutes of transcription time. As Figure 4 indicates, with the proposed setup *(Prop)* 1655 errors are removed when performing updates every 2.5 minutes, with the utility decreasing for fewer updates, until only 1328 errors are removed when performing no updates to the initial segmentation. For an update interval of 2.5 minutes, we observed the efficiency (corrections/second) to start out around 0.2 at the beginning of the transcription process, and converging halfway through transcription at about 0.3, which illustrates increasing quality of segmentations due to on-the-fly updates.

Figure 4 also shows several other experimental settings: *Cost-insensitive baselines (b-Conf, b-Lin).* Without a cost model, SESLA cannot be used to optimize segmentations, so in this experiment we consider predefined, fixed segmentations. Previous research found that transcribing (sub-) sentences is preferable over transcribing individual words [1, 3], so we divide our transcript into segments of 10 words. In *b-Lin*, the transcriber corrects segments in linear order from the start until the time budget is exhausted, while *b-Conf* exploits our utility model and chooses the segments with highest utility. The first baseline is much inferior, while the second baseline still slightly underperforms the proposed approach, even without updates and only the prior cost model. This indicates that even using our crude prior cost model is better than not using a cost model at all.
User models from enrollment (b-Enr). This is a cost-sensitive baseline similar to [7]. Segmentations are optimized using SESLA, but not updated on-the-fly, and the cost-model is trained via enrollment. In this scenario, we assume that supervision time can be divided between enrollment and productive error correction. In the graph, we vary over how much time is reserved for enrollment, and find optimum utility for

enrollments of 0 or 10 minutes, only slightly better than the confidence baseline, and 28% worse than the best proposed setup. Note that in a practical situation, the optimal division between spending time for enrollment and productive transcription is unknown.

Updating segmentations, but not user models (Prop-seg). To analyze the impact of frequently updating segmentations to reflect the actual remaining time budget at different times, here we update segmentations as proposed, but use fixed user models trained via enrollment as the previous baseline. Compared to *b-Enr*, this brings considerable gains, especially when setting enrollment time to 0, in which case 1540 errors are corrected (7% worse than with the best proposed setup). This indicates that adjusting to the actual remaining time budget is a crucial factor in our framework.

Full updating framework with naive prior (Prop-NPr). To examine the importance of using the proposed prior, we run the full proposed updating framework, but use a naive cost model prior that simply assumes transcription time to take up $1s$ per word, dropping the segment overhead. This leads to considerable losses, especially when running updates infrequently, indicating that modeling segment overhead in the prior is another significant factor.

Finally, we tested how well our approach generalizes to other users. We created a second oracle time model via enrollment of a second transcriber, doing post-editing (unlike the first person, who transcribed from scratch). Transcription was about 4% faster than the first transcriber. Next, we increased the multiplicative noise variance from moderate 0.01 to a high value of 1. High variance in supervision time can be expected with novice transcribers, or when the transcription task is inherently difficult. Consistently for both transcribers, the gain in end-to-end utility using the *(Prop)* instead of *(b-Conf)* dropped from 34% to 11%. This indicates that even when cost is difficult to model, doing so is worthwhile and beats cost-insensitive approaches, though at somewhat diminished gains.

6. CONCLUSION

We proposed a new on-the-fly updating framework for cost-sensitive correction of speech transcripts, along with a new, faster algorithm that determines which segments to correct. Besides being more convenient to use than previous cost-sensitive methods that required transcriber enrollment, we show that our approach yields higher supervision efficiency. Efficiency gains are attributed to updating cost models, updating segmentations, and a sensible initial cost model.

Cost models trained on-the-fly are useful for other tasks, such as cost-efficient computer-assisted translation [9], predicting post-editing time [10], and anytime active learning [6]. Future work will consider such additional tasks, investigate in how to efficiently train large cost models for long-term transcribers, and conduct real user studies.

Acknowledgments

The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement n 287658 Bridges Across the Language Divide (EU-BRIDGE).

7. REFERENCES

[1] Brandon C. Roy and Deb Roy, "Fast transcription of unstructured audio recordings," in *Interspeech*, 2009.

[2] Isaias Sanchez-Cortina, Nicolas Serrano, Alberto Sanchis, and Alfons Juan, "A prototype for Interactive Speech Transcription Balancing Error and Supervision Effort," in *Int'l Conf. on Intelligent User Interfaces (IUI)*, 2012.

[3] Matthias Sperber, Graham Neubig, Christian Fügen, Satoshi Nakamura, and Alex Waibel, "Efficient Speech Transcription Through Respeaking," in *Interspeech*, 2013.

[4] Burr Settles, Mark Craven, and Lewis Friedland, "Active Learning with Real Annotation Costs," in *NIPS Workshop on Cost-Sensitive Learning*, 2008.

[5] Katrin Tomanek and Udo Hahn, "A Comparison of Models for Cost-Sensitive Active Learning," in *Int'l Conf. on Computational Linguistics (COLING)*, 2010.

[6] Maria E. Ramirez-Loaiza, Aron Culotta, and Mustafa Bilgic, "Anytime Active Learning," in *Conf. on Artificial Intelligence (AAAI)*, 2014.

[7] Matthias Sperber, Mirjam Simantzik, Graham Neubig, Satoshi Nakamura, and Alex Waibel, "Segmentation for Efficient Supervised Language Annotation with an Explicit Cost-Utility Tradeoff," *Transactions of the Assoc. for Computational Linguistics (TACL)*, April 2014.

[8] Hagen Soltau, Florian Metze, Christian Fügen, and Alex Waibel, "A One-Pass Decoder Based on Polymorphic Linguistic Context Assignment," in *Automatic Speech Recognition and Understanding Workshop (ASRU)*, 2001.

[9] Jesús González-Rubio and Francisco Casacuberta, "Cost-sensitive active learning for computer-assisted translation," *Pattern Recognition Letters*, vol. 37, Feb. 2014.

[10] Trevor Cohn and Lucia Specia, "Modelling Annotator Bias with Multi-task Gaussian Processes: An Application to Machine Translation Quality Estimation," in *Assoc. for Computational Linguistics (ACL)*, 2013.