Automating Crowd-supervised Learning for Spoken Language Systems

Ian McGraw, Scott Cyphers, Panupong Pasupat, Jingjing Liu, Jim Glass

MIT Computer Science & Artificial Intelligence Laboratory, Cambridge, MA 02139, U.S.A.

{imcgraw, cyphers, ppasupat, jingl, glass}@csail.mit.edu

Abstract

Spoken language systems often rely on static speech recognizers. When the underlying models are improved on-the-fly, training is usually performed using unsupervised methods. In this work, we explore an alternative approach that uses human computation to provide crowd-supervised training of a deployed system. Although the framework we describe is applicable to any stochastic model for which the training data can be generated by non-experts, we demonstrate its utility on the lexicon and language model of a speech recognizer in a cinema voice-search domain. We show how an initially shaky system can achieve over a 10% absolute improvement in word error rate (WER) – entirely without expert intervention. We then analyze how these gains were made.

1. Introduction

The speech research community has dreamt of organic spoken language systems that grow and improve without the expert-aid for many years [1]. The advent of crowdsourcing techniques for speech-related tasks provides a new way to overcome obstacles to this goal [2]. In addition to providing a new source of training data, micropayment-workforce platforms such as Amazon Mechanical Turk (mTurk) now enable developers to test spoken language systems cheaply with real users [3]. Moreover, we can now write human-in-the-loop algorithms to improve spoken language systems on-the-fly [4].

There are, however, a number of limitations which must be dealt with to fully realize the potential of crowdsourcing. First, any individual worker may provide noisy results. As the natural language processing (NLP) community showed, however, combining output from multiple workers can mitigate the effects of noise [5]. The second, perhaps more serious, limitation is that large-crowds are inevitably not experts in the domain at hand. This necessitates the decomposition of any crowd-supervised task into units which can be completed by non-experts.

In this work, we explore the use of human computation to retrain our new *Movie Browser* spoken language system. The system itself allows users to make speech queries to search the the Internet Movie Database (IMDB). Our training framework, however, is domain independent. With the help of TurKit [6], a toolkit for manipulating tasks on mTurk, we link together transcription, semantic tagging, and speech collection tasks in a fully automatic fashion. Using this assembly line of human intelligence tasks (HITs), we perform retraining operations, such as rebuilding the class-based language model that underpins the recognizer's search space.

The main focus of this work, however, is on the lexicon. As new movies come out and new actors become popular, it is easy to imagine that the lexical domain of the *Movie Browser* might shift. Due to the inherent difficulty with the pronunciation (and, consequently, recognition) of named entities, we may wish to improve upon the state-of-the-art letter-to-sound (L2S) models currently used to generate pronunciations of words where the spelling is known [7]. We show that this task can be reformulated for the mTurk crowd by collecting spoken examples and learning new words using a pronunciation mixture model [8].

In the remainder of this paper, we describe the *Movie Browser* system, and detail the manner in which we update its lexicon and language model using a fully-automated crowd-supervised framework. We perform experiments and show improvements of a metric based on movie search result relevance. We also examine WER improvements, and in doing so, we confirm that the pronunciation mixture model is robust to the noisy contributions of distributed workers in a continuous speech setting.

2. Movie Browser

The *Movie Browser* system provides the test-bed for our training framework. The system is intended to handle natural spoken queries such as *"What are some James Bond movies starring Sean Connery?"* The *Movie Browser* makes use of conditional random fields to parse the output of the speech recognizer into the set of semantic categories used during searches. The search index, handled by the Apache Lucene project, is initialized to contain over 12,000 movie titles from the IMDB database. More details about the text-based version of the *Movie Browser* are described in [9].

This work concentrates on the recognizer we have constructed for this domain. Decoding is handled by SUMMIT [10], a landmark-based speech recognizer that models its search space using a composition of weighted finite state transducers. We have configured the recognizer to make use of 112-dimensional MFCC-based feature vectors, which are whitened with PCA before reducing the dimensionality to 50. The acoustic models, which were trained on a corpus of telephone speech, consist of diagonal gaussians trained with up to 75 mixture components.

The language model (LM) of the recognizer uses a class n-gram to robustly model carrier phrases. Thus, the query above would appear to the LM as *"What are some CHARACTER movies starring ACTOR?"* Separate FSTs are built and dynamically inserted for each class. The ten manually chosen classes are: Actor, Genre, Director, Year, Rating (e.g. pg13), Character, Plot element, Title, Song, and Evaluation (e.g. "well rated").

To limit its size, our recognizer does not include the full list of over 200,000 actors, directors, and movie titles in our database . Instead, we initialize the classes with lists of *popular* actors, directors, titles, etc, that were scraped from the web. The lists contained almost 3,000 movie titles and over 1,000 popular actors. They were, however, a few years old, and thus somewhat stale with respect to the domain. We hoped the recognizer would learn missing lexical items on-the-fly via crowdsourcing.

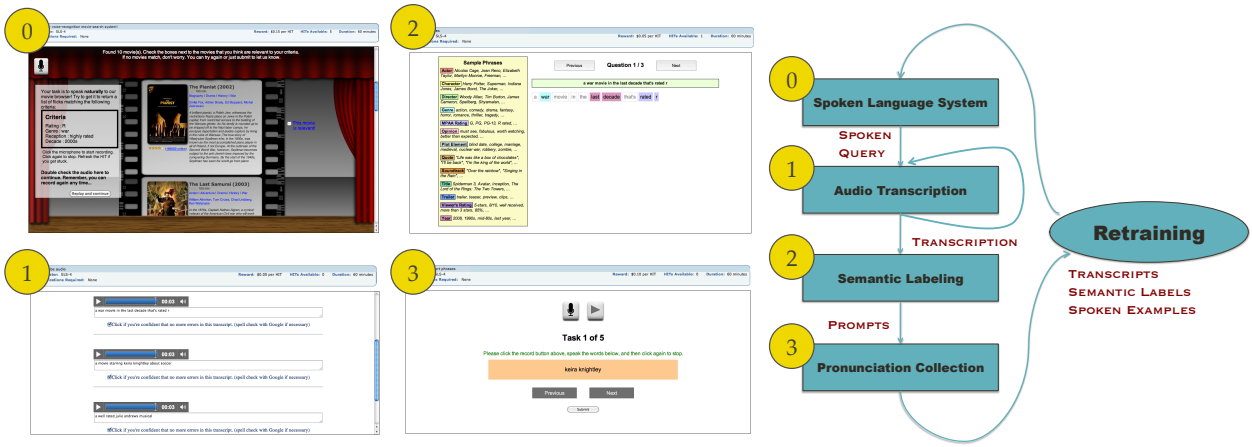

Figure 1: The system HIT (0) and three domain-independent HITs (1)-(3) that compose our crowd-supervised training architecture.

3. Learning the Lexicon

The pronunciation modeling techniques described in this section presume that a spelling is known but that a pronunciation is not or needs improvement. This assumption is reasonable in our movie search domain because the search index contains a large, but finite number of entries. Regardless of the domain, obtaining crowdsourced transcriptions also reduces the challenge to one of recovering a pronunciation given a spelling.

Our baseline L2S model is based on the work of Bisani and Ney [7], wherein a language model is learned over grapheme-to-phoneme units, called *graphones*. Though not inherently graphone specific, we have shown that a pronunciation mixture model (PMM) can expand upon the framework by providing a principled method of integrating acoustic information into the pronunciation generation process [8].

We provide an overview of the isolated word case and refer the reader to [8] for the adjustments used in this work to handle learning from continuous speech. The isolated word PMM presumes that we have M utterances $\{\mathbf{u}_i, \ldots, \mathbf{u}_M\}$ of a word $\mathbf{w}$, and that we have a model for the joint probability of this word and a candidate pronunciation $\mathbf{b}$. We parameterize the log likelihood of the data as follows:

$$L(\theta) = \sum_{i=1}^{M} \log\ p(\mathbf{u}_i, \mathbf{w}; \theta) = \sum_{i=1}^{M} \log \sum_{\mathbf{b} \in \mathcal{B}} p(\mathbf{u}_i | \mathbf{w}, \mathbf{b}) \cdot \theta_{\mathbf{w},\mathbf{b}}$$

The parameters, θ, are initialized to our graphone LM scores. We then run Expectation Maximization (EM) according to the following equations:

$$\text{E-step:} \qquad P(\mathbf{b}|\mathbf{u}_i, \mathbf{w}; \theta) = \frac{p(\mathbf{u}_i|\mathbf{b}, \mathbf{w}) \cdot \theta_{\mathbf{w},\mathbf{b}}}{\sum_{\mathbf{p}} p(\mathbf{u}_i|\mathbf{p}, \mathbf{w}) \cdot \theta_{\mathbf{w},\mathbf{p}}}$$

$$\text{M-step:} \qquad \theta^*_{\mathbf{w},\mathbf{b}} = \frac{1}{M} \sum_{i=1}^{M} P(\mathbf{b}|\mathbf{u}_i, \mathbf{w}; \theta)$$

Thus, one iteration of the PMM gives each utterance a probability mass of $\frac{1}{M}$ to distribute over the pronunciations under consideration according to the posterior probability of each baseform $\mathbf{b}$, given the acoustics and the spelling. EM can be used to iteratively optimize the log-likelihood of the data, but runs the risk of over-fitting when M is small. For this work, we run one iteration of EM using a 5-gram graphone LM over singular graphones. We normalize the learned parameters by word and use them in a stochastic lexicon. Pronunciations with probability less than 0.1 are excluded.

4. Crowd-supervised Training Architecture

In this section, we introduce a library of crowdsourcing tasks designed for processing spoken language data and describe how we connect these tasks together in order to fully automate the retraining of certain components of a spoken language system. The larger vision of this work is to substantially reduce, if not entirely eliminate, the need for the expert intervention and ad-hoc boot-strapping techniques typically employed in spoken language system development. To this end, we have been developing and fine-tuning a set of Human Intelligence Tasks deployable to mTurk. Three of these HITs, transcription, segment tagging, and prompted speech collection, are described here.

A transcription HIT has been shown to be a valuable tool for obtaining orthographies of audio. Combining multiple noisy transcriptions with techniques such as ROVER is known to significantly improve transcription quality [11]. Our transcription HIT, numbered (1) in Figure 1, deals with noisy data in a somewhat simpler manner. Each audio clip must be shown to two workers, and the job of the second worker is to improve upon or verify the transcript of the first worker. While perhaps not fully addressing the problem of noise, this procedure is sufficient for the training needs of this work and can be easily extended with more iterations of improvement or approval.

Semantic labeling, whereby phrases in a sentence are tagged with a category label, is another common NLP task for which mTurk has proven useful [12]. Our semantic labeling HIT, shown in (2) of Figure 1, uses a click-and-drag selection mechanism to allow workers to semantically tag multiple words with a label. In our case, the labels consist of the ten classes found in our recognizer. Although it is not difficult to imagine combining multiple outputs to reduce noise, for this HIT we took a domain-dependent approach described later.

The final domain-independent HIT is the prompted audio collection task depicted in Figure 1 (3). Here, a worker simply reads aloud the prompt on the page. While it is difficult to control the properties of the speech collected, we have found that forcing workers to listen to their own speech helps to address microphone problems. Restricting the country to which a HIT is deployed provides rough controls over accented-speech. Furthermore, in this work, we collect eight examples of each prompt for use in a pronunciation mixture model. Our hope is that a few noisy examples will not cause too many problems.

With our library complete, we now describe a domain-dependent HIT designed to collect data for the very system we

are training. Figure 1 depicts the *Movie Browser* system in a HIT labeled (0). This particular instantiation of the HIT provides the worker with a scenario which they must attempt to accomplish. Workers may be asked, for instance, to use the *Movie Browser* to find movies with ACTOR: Keanu Reeves and GENRE: Action. The search results are displayed along with checkboxes, which the worker uses to mark the movies relevant to the query. Upon submission, we collect the last utterance spoken and the list of movies marked relevant.

With the system HIT in place, we have all the pieces necessary to implement the crowd-supervised training of our system. To be fully hands-free, however, we must be able to use the output of one HIT as the input of the next. To do so, we rely on TurKit [6]. This open-source project allows us to implement simple JavaScript programs to manipulate HITs arbitrarily. The following code conveys the ease with which we can take audio collected from our system HIT and pass it along into a generic transcription task:

```
var out = systemHIT.outputs();
var transcribeHIT = new TranscriptionHIT();
for (var i = 0; i < out.length; ++i) {
    var id = transcribeHIT.add(out[i].audioUrl);
    // id can be used later to check for results
}
```

We extend this code to collect data with HIT (0) and pass it through HITs (1)-(3) according to the flow chart in Figure 1.

Noise control is relatively straightforward to implement using TurKit. The transcription HIT has a loop in the flowchart which indicates that once the audio is transcribed by the first worker, it is then passed through the HIT a second time to be verified by a second worker before moving on to the semantic labeling stage. When the labeling is finished, we compare the transcript to the recognition results in search of semantically important words that were misrecognized. Before we generate prompt HITs usingng these words, however, we employ a domain-specific form of noise control. In particular, each term is run through our movie search index to ensure its validity. Once the transcribed and semantically tagged term is validated, we begin to collect spoken examples of it.

TurKit relies heavily on the concept of *memoization* to ease the scripting of human computation. Memoization refers to the caching of the result of a function call based on its inputs. For example, the `transcribeHIT.add` function in the code snippet above is memoized internally. The first time it is called, a HIT is created and the HIT's `id` is returned. In subsequent calls to the function with the same URL input, no new HIT is created, but the same `id` is returned. To simplify the explanation, we have omitted the details of how we batch multiple utterances into the same transcription HIT, but a library for bundling inputs makes *batched* tasks similarly straightforward. Finally, we extend this approach to implement the series of HITs in the flowchart of Figure 1 and run the script iteratively. Each iteration sends a wave of updates across all of the active HITs.

To close the loop in the crowd-supervised training process, our system must be able to retrain and dynamically load new models on-the-fly. Given the spoken examples, we employ a pronunciation mixture model to learn new words and refine existing pronunciations in our lexicon. With the transcripts and semantic labels, we can train a class-based language model. Two additional obvious candidates for retraining not explored in this work are the acoustic model and the conditional random fields used for semantic interpretation.

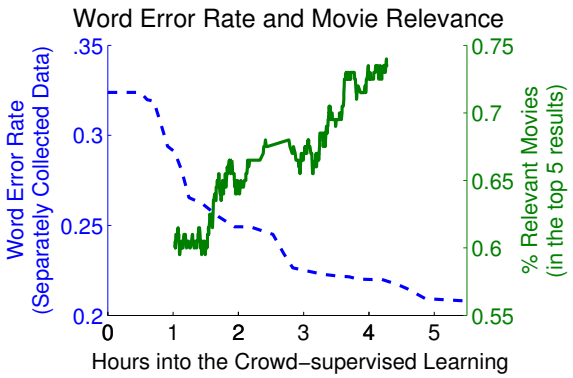

Figure 2: We show a windowed percentage indicating the fraction of the last 200 utterances that contain a relevant result in the top five movies returned. WER is computed on a separate dataset of 1001 utterances at various stages in the crowd-supervised learning process.

5. Experimental Results

To experiment with our crowd-supervised training framework, we decided to limit our domain to a set of scenarios. Rather than choosing our own scenarios, however, we ran a quick mTurk-task for scenario collection. This task consisted of a simple HTML form, for which a worker was asked to fill out two or three of the ten classes with reasonable values, e.g. DIRECTOR: Darren Aronofsky, RATING: R. These constraints then make up the set of scenarios used in our experiments. We collected 373 scenarios and filtered them through the movie search index to ensure their presence in our database. Forty scenarios were used for development to test our training framework, and 127 scenarios were used in the experiments described below.

We kicked off a small-scale crowd-supervised training experiment on a weekday evening and completed it roughly five-and-a-half hours later. All HITs were restricted to the US, and the prices of each HIT were set to optimize throughput rather than cost. Still, each utterance was collected, transcribed, and semantically labeled for less than a quarter. Over the course of deployment, 22 distinct workers contributed a total of 714 individual voice search queries to *Movie Browser* using the system HIT. Each worker was only allowed to complete a given scenario once. As workers completed the system HIT, their data progressed through the crowd-supervised training framework; about 45 minutes into the HIT, on-the-fly recognizer updates began to occur. Figure 2 shows that the on-the-fly training procedure has an appreciable effect on movie relevance, a metric computed from the worker's checkbox selections. We use a large window of 200 utterances to smooth out variation between users, and compute a moving average representing the fraction of the time that a relevant movie is in the top five search results. Since it is clear that the system improves, we now turn our attention to ascertaining how the gains are made.

To analyze the dynamics of the recognition, we took snapshots of the recognizer after each pass through the retraining phase. Rather than test on the same data, we decided to collect and transcribe a separate dataset with the same scenarios using only HITs (0) and (1) from Figure 1. We made no effort to prevent workers who performed the first HIT from working on the second, and indeed 11 workers of the 38 that performed this HIT had done at least one assignment during the crowd-supervised training. Also, for this collection, workers were allowed to perform each scenario twice. The test set collection task was left

Category	A	G	D	R	C	T	P	all
Spoken	666	486	275	263	193	99	96	1986
Learned	74	19	27	6	27	17	16	177
Missing	27	1	10	3	20	6	10	67
Acc. Before	77.0	79.8	77.4	77.5	35.7	51	42	70.2
Acc. L2S	94.1	80.0	93.0	81.4	81.3	77	90	86.5
Acc. After	**96.6**†	79.8	92.0	82.5	**87.6**∗	76	84	**87.5**∗

† PMM/L2S differences statistically significant with $p < 0.001$
∗ PMM/L2S differences statistically significant with $p < 0.05$

Table 1: Accuracies broken down by categories (abbreviated by their first letter.) Note that some categories overlap slightly, e.g. actors can be directors. Categories that were not learned or with fewer then 10 spoken examples are omitted.

running overnight, and by morning, we had collected and transcribed 1,179 utterances. Unlike the training phase, which was completely hands-free, some workers who contributed unintelligible audio were rejected. Using the final test set of 1,001 utterances, we compute a WER for each recognizer snapshot and plot it over time in Figure 2. Over the course of the crowd-supervised training, the WER of the test set drops from 32.3% down to 20.8%. The high error rates reflect both the difficulty of the domain and the noisy nature of crowd-sourced data.

There is an initial delay before the improvements become noticeable since it takes time for the data to propagate down through the semantic labeling HIT. After the semantic labeling HIT, the class-based language model retraining can begin. We determine the utility of retraining the language model via a post-processing step. We remove the crowd-supervised transcriptions from the LM training data while keeping the learned lexical items intact and recompute a WER on the test set. This causes the WER to bounce back up to 25.3%. Most of the gains, then, are due to the improvements to the lexicon.

Closing the lexical gap between the mTurk collected scenarios and the initial recognizer lexicon is the largest source of gains. Sam Worthington, for example, the star of Avatar, was not on the original outdated list of actors. When an utterance containing this name was transcribed and labeled, the name was inevitably determined to be a misrecognition. As such, it was first validated against the search index, and then added to the list of prompts to be collected. In the meantime, the L2S model was used to generate a pronunciation. Table 1 shows the number of terms learned and missing from each category, as well as the number of times they were spoken in the test set.

Since only 177 distinct words were learned, we examine the recognition accuracies of these words directly to get a picture of how the L2S and PMM affect recognition. Table 1 breaks accuracies down by category. We first compute the accuracies of each category *before* any training took place. They are relatively low due to the missing lexical items. The final row in the table shows the results of applying the entire crowd-supervised word-learning process.

The effects of the PMM can be inferred by first removing the pronunciations learned from spoken examples and replacing them with their L2S counterparts. As shown in Table 1, where the differences between the PMM and L2S are statistically significant under McNemar's test, the PMM consistently improves over the L2S approach. In particular, recognition of actors names was significantly improved despite having a high L2S baseline of 94.1%. For example, the last name of Cary Elwes, of Princess Bride fame, was given the pronunciation `eh l w eh s` by the L2S. The PMM's top pronunciation was more accurate: `eh l w ey z`. We were surprised that the workers knew the pronunciation of this name, but listening to their speech confirmed that they did. Even if they had not, learning mispronounciations can also be beneficial.

6. Conclusions and Future Work

This work has described an automatic crowd-supervised training framework and demonstrated its utility with respect to updating a recognizer's lexicon and language model in the *Movie Browser* system. While cinema's ever-shifting lexical domain highlights the need for systems that can grow and change to accommodate previously unknown words, we believe the framework itself may be generally useful to replace the manual configuration and bootstrapping that accompanies building a new spoken language system. Most of the tools used in the paper are open-source, enabling other researchers to test similar systems.

In addition to increasing the scale of our current training effort, we hope to explore a number of extensions to this work. First, the semantic labels can be used to retrain the conditional random field to learn semantics on-the-fly. Second, we would like to examine whether acoustic model adaptation might be performed based on mTurk-collected audio. Finally, we would like to integrate more sophisticated active learning techniques to let the system intelligently decide how to collect its own training data. Perhaps with these additions, we would come even closer to a truly *organic* spoken language system.

7. Acknowledgments

We would like to thank Carrie Cai for managing scenario collection, Will Walker for leading the user interface development, and Stephanie Seneff for helpful discussions. This research is funded by Quanta Computers, Inc.

8. References

[1] V. Zue, "On organic interfaces," in *Proc. INTERSPEECH*, 2007.

[2] G. Parent and M. Eskenazi, "Speaking to the crowd: looking at past achievements in using crowdsourcing for speech and predicting future challenges," in *Proc. INTERSPEECH*, 2011.

[3] F. Jurčíček, S. Keizer, M. Gašić, F. Mairesse, B. Thomson, K. Yu, and S. Young, "Real user evaluation of spoken dialogue systems using amazon mechanical turk," in *Proc. INTERSPEECH*, 2011.

[4] I. McGraw, J. R. Glass, and S. Seneff, "Growing a spoken language interface on amazon mechanical turk." in *Proc. INTERSPEECH*, 2011.

[5] R. Snow, B. O'Connor, D. Jurafsky, and A. Y. Ng, "Cheap and fast—but is it good?: evaluating non-expert annotations for natural language tasks," in *Proc. EMNLP*, 2008.

[6] G. Little, L. B. Chilton, M. Goldman, and R. C. Miller, "Turkit: tools for iterative tasks on mechanical turk," in *Proc. HCOMP*, 2009.

[7] M. Bisani and H. Ney, "Joint-sequence models for grapheme-to-phoneme conversion," *Speech Commun.*, vol. 50, May 2008.

[8] I. Badr, I. McGraw, and J. Glass, "Pronunciation learning from continuous speech," in *Proc. INTERSPEECH*, 2011.

[9] J. Liu, S. Cyphers, P. Pasupat, I. McGraw, and J. Glass, "A conversational movie search system based on conditional random fields," submitted to *INTERSPEECH*, 2012.

[10] J. R. Glass, "A probabilistic framework for segment-based speech recognition," *Computer Speech & Language*, vol. 17, 2003.

[11] M. Marge, S. Banerjee, and A. Rudnicky, "Using the amazon mechanical turk for transcription of spoken language," in *Proc. ICASSP*, 2010.

[12] F. Mairesse, M. Gašić, F. Jurčíček, S. Keizer, B. Thomson, K. Yu, and S. Young, "Phrase-based statistical language generation using graphical models and active learning," in *Proc. ACL*, 2010.