A COLLECTIVE DATA GENERATION METHOD FOR SPEECH LANGUAGE MODELS

Sean Liu, Stephanie Seneff, James Glass

MIT Computer Science & Artificial Intelligence Laboratory
32 Vassar Street, Cambridge, MA 02139
{seanyliu,seneff,glass}@csail.mit.edu

ABSTRACT

Recently we began using Amazon Mechanical Turk (AMT), an Internet marketplace, to deploy our spoken dialogue systems to large audiences for user testing and data collection purposes. This crowdsourcing method of collecting data contrasts with the time- and labor- intensive developer annotation methods. In this paper, we compare these data in various combinations with traditionally-collected corpora for training our speech recognizer's language model. Our results show that AMT text queries are effective for initial language model training for spoken dialogue systems, and that crowdsourced speech collection within the context of a spoken dialogue framework provides significant improvement.

Index Terms— Language models, crowdsourcing, Amazon Mechanical Turk

1. INTRODUCTION

Language models for speech recognizers require a substantial amount of data. Amassing quality data for spoken dialogue systems, however, is traditionally a laborious task. Developers must create a live system to record spoken user interactions, launch marketing campaigns to promote usage, and often manually transcribe the recorded audio. Recently we explored the use of Amazon Mechanical Turk (AMT) to, in a cost-effective manner, build large corpora in short spans of time [1]. Data collection is crowd-sourced by embedding the dialogue systems as a task for pay in the AMT interface, which then distributes the system to workers. Using this method, over ten thousand utterances can be collected in a few weeks [1].

This paper explores the usefulness of such data collected from AMT, in combination with previously collected data, for the purposes of training our speech recognizer's language model for the *City Browser* dialogue system. We compare the AMT corpora against those collected by traditional user testing, human-generated sample text queries, and synthetic sentences generated from templates. Since we are expanding the capabilities of a new mobile version of the system to handle arbitrary cities, we hope to identify which combinations of the legacy data can improve the system as we continue to collect more data. Our results show that crowd-based text collection of user queries is effective for seeding language models in a dialogue system, and that speech collected within the spoken dialogue context substantially improves system performance.

This paper proceeds as follows: Section 2 discusses background and related work; Section 3 introduces the architecture for the dialogue system used in our experiments; Section 4 describes enhancements to the system in preparation for crowdsourcing data collection; Section 5 presents the various data sets used for language model training; Section 6 compares various training corpora on their effectiveness in recognizer language model training; and Section 7 concludes with future work.

2. BACKGROUND & RELATED WORK

AMT is an online service which allows programmers to post Human Intelligence Tasks (HITs), tasks to be solved by workers for small monetary rewards generally ranging from $0.01 to $0.10 per task. The service has been increasingly used by the research community for its crowdsourcing ability. One such example is the use of AMT for user studies and feedback solicitation [2]. As long as tasks are verifiable and concrete, as opposed to subjective and qualitative, AMT has been shown to provide high quality user studies at a cheaper and faster rate than traditional in-lab studies.

Most relevant, however, is the use of AMT to efficiently construct and annotate large corpora, especially important for speech and natural language systems, as they rely on the training data for performance [3]. This is similar to other efforts, such as a custom-built online gaming platform which directs human computation [4], but instead focuses on monetary rewards. Corpora can be built at remarkable speeds using AMT. For instance, a corpus of over 100,000 read utterances was constructed in under two weeks [1]. These authors also collected data for a Flight Browser domain, and a novelty was that they asked AMT workers to provide plausible scenarios for the dialogue-based efforts. We adopt a similar strategy here. Similarly, researchers at Nokia were able to generate enough content from AMT to add new languages to their speech recognizer [5]. In this paper, we seek to verify that

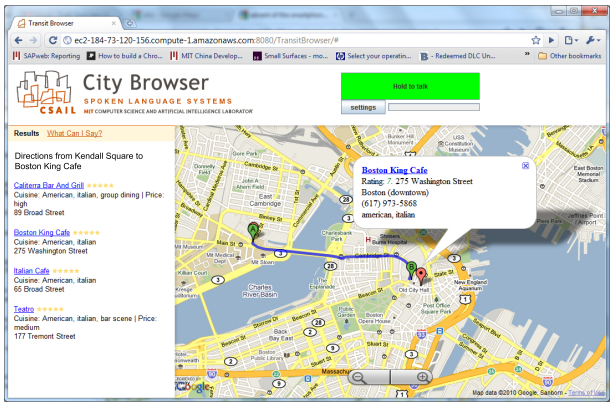

Fig. 1. A screenshot of the latest *City Browser* interface.

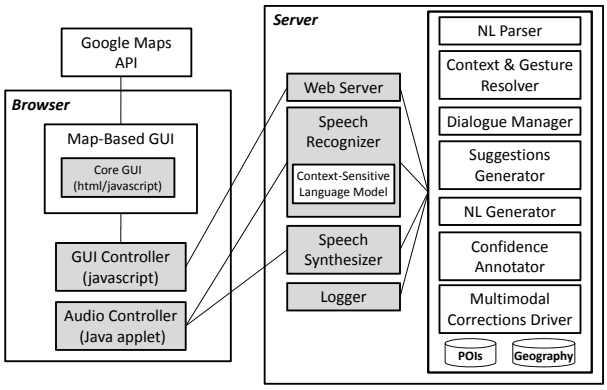

Fig. 2. *City Browser*'s Architecture (reproduction of Figure 5 in [7]). Shaded boxes represent components provided by the WAMI toolkit.

such AMT corpora are in fact useful as training data for the system's language models.

The experiments in this paper focus on the *City Browser* system, developed by the MIT Spoken Language Systems group [6]. *City Browser* is a web-based multimodal application designed for desktop and tablet use. A screenshot of the interface is shown in Figure 1. The system allows users to query for local points-of-interest (POIs) such as restaurants, hotels, and museums, using a combination of spoken language and manual interface controls. The users' spoken utterances can be fully conversational, and are handled by the system's speech recognizer and natural language parser.

3. SYSTEM ARCHITECTURE

City Browser is built with the Web-Accessible Multimodal Interfaces (WAMI) toolkit [6]. The architecture is shown in Figure 2. WAMI provides the skeleton, to which are attached a speech recognizer, a speech synthesizer, and the language processing component. The shaded, gray boxes represent components provided by the default WAMI toolkit. The language processing component consists of a number of specialized subcomponents, each specifically trained on the map-based city browsing domain.

The WAMI framework passes on spoken utterances from the client to the speech recognizer, SUMMIT [8]. SUMMIT has the ability to dynamically swap out vocabulary sets for proper nouns based on the user's current metro region [9]. The output is the text transcription of the utterance, with the proper nouns labeled by class. SUMMIT utilizes a class n-gram language model [8].

The speech recognizer output and GUI manipulations by the user are passed to the language understanding services. The natural language parser converts the text to a meaning representation, and context resolution and dialogue management are performed to resolve semantic understanding in the context of the preceding dialogue [7]. Once the user's intention is decoded, the system performs a database lookup. A language generation component prepares a well-formed verbal response. The relevant results are returned back to the user interface via the GUI and Audio controllers. The Suggestions Generator proposes context-sensitive sentences that could be spoken next, as a "Help" aid, and the Confidence Annotator decides whether the utterance should be rejected.

4. DYNAMIC DATABASE GENERATION

A challenge to the scalability of the *City Browser* system is the database content coverage. As a widely-deployed web application, *City Browser* receives queries from users requesting points of interest from a vast number of metro regions. Anticipating the locations, building the static databases, and maintaining up-to-date content is time-consuming for a large-scale system, and therefore impractical.

In preparing for a widely deployable system on AMT, usable beyond our system's previous Boston-only audience, we enhanced City Browser with the ability to generate database content on-demand. The goal was to supplement the static pre-generated database with dynamic on-demand sources for metro regions not in the static database. When users switch to new metro regions with no database coverage, *City Browser* expands the database on-the-fly with information pulled from other content providers, such as Yelp, an online restaurant information and review site. Through the content provider search APIs, relevant point-of-interest data are retrieved and converted to match the static database entries. Our enhancements are isolated within the database look-up function so the added functionality is transparent to the rest of the system.

5. LANGUAGE MODEL TRAINING CORPORA

Several different data collection methods were used to generate the language model training corpora.

5.1. Text-based Data

The text-based data consist of sample sentences generated directly as text. That is, either developers or paid workers produced text sentences that they believed could be spoken as inputs to our speech dialogue system.

5.1.1. Template-based Generation

To seed our language models when launching our dialogue systems, we use developer-engineered sentences. We have developed a flexible capability to generate sentences from context-free-rule-based templates [10], such as:

> [can_you] [tell_me_about] [named_restaurant_phrase]?

In which the rules in [ ] are expanded into alternative phrase instantiations to complete a sentence. An example output sentence for this rule might be:

> Could you give me more details about Cambridge Cafe?

An advantage of template-based sentences is that they do not need a widely-deployable system and can be hand-crafted by developers. But developers must take care to cover different usage patterns in statistically appropriate frequencies.

5.1.2. Crowd-sourced Text Data

A natural extension to developer-generated text sentences is to crowd-source the process. Like the template-based method, no functional system is required. We created an AMT task which requests workers to provide us with typed sample queries. Our task describes a hypothetical system and presents screenshots of the system interface. Additionally, we give the workers five reference sentences, representative of the queries understood by the system. A screenshot of this AMT task is shown in Figure 3.

Initially we paid workers $0.03 per sentence provided, for a total of 7800 sample inputs. We then experimented with decreasing the pay rate, collecting an additional 4200 inputs at $0.025 per sentence. We found that the rate of collection was essentially unchanged. Overall the entire collection process took approximately 3 days, and yielded 13,313 sentences.

In the process we encountered several difficulties. We found that the AMT workers were extremely biased by the example sentences we provided. In our initial iteration of the HIT, we emphasized the Boston metro region, only to find that a significant portion of the worker queries also referenced Boston. Therefore we updated the examples to reference cities across the nation. Similarly, the HIT also suffered from worker abuse. In one such example, a single worker submitted over 5,000 sample inputs consisting of a single phrase structure "show me `CUISINE` restaurants in `CITY`", in which each sentence differed from the others only by the cuisine type or city. These data are worse than automatic template-based data, and were therefore discarded. In our next iteration of the HIT, we made it clear that a variety of sentences were needed, and that simple swapping of proper nouns would result in a rejected submission.

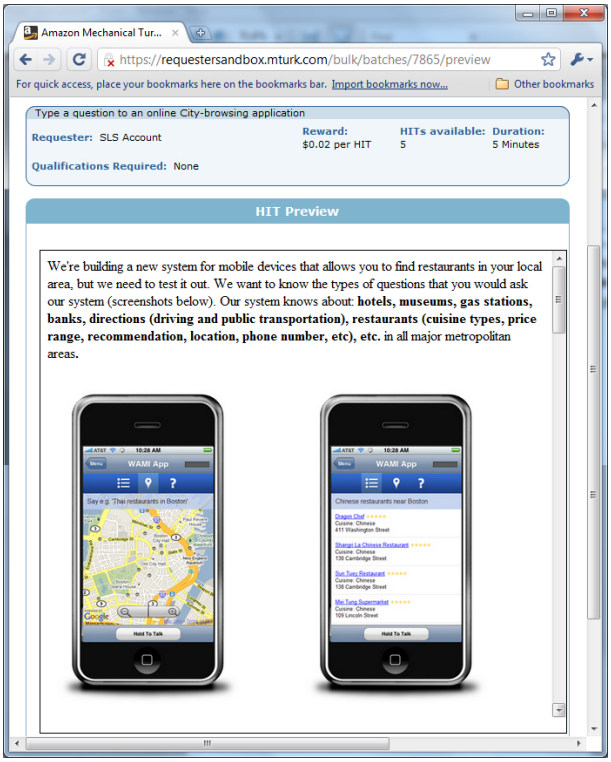

Fig. 3. The Mechanical Turk HIT used to collect queries.

In order to use the sample AMT inputs as training data, we needed to tag the POIs with appropriate word classes. Several well-known algorithms exist for this task, such as the contextual semantic tagger in [11]. For our purposes, we did not care if the contents within each tag were accurate – we only wanted to obtain representative patterns for the sentence-level constructs, and would replace each tagged entity with a tag label, ignoring the specific instantiation, before using these sentences in language model training. Our system utilizes eight distinct classes, namely, *city*, *hotel*, *landmark*, *museum*, *neighborhood*, *restaurant*, *station*, and *streetname*.

For this, we used a novel approach which is described more fully in a companion paper [12]. The natural language grammar is used to parse a large set of representative tagged utterances, and a set of strings encoding the sequence of preterminal categories in the parse trees are written out. A class trigram language model is constructed from these outputs, and is used to produce an N-best list of candidate preterminal sequences from each untagged utterance. Appropriate tag labels are inferred from the preterminal assignments. The parser then parses each untagged sentence while honoring the strong constraints (including both preterminal categories

and tag assignments) specified in each N-best preterminal sequence hypothesis. It finally chooses randomly among the top three parsable solutions. There were many erroneous assignments, for example calling a hotel a restaurant or calling a city a neighborhood, but we are hoping the general language patterns will be for the most part accurately represented and distributed reasonably over all the tag categories.

5.2. Speech-based Data

The speech-based data consists of transcriptions of actual spoken interactions with our dialogue system.

5.2.1. Traditional Data Collection

Our traditional method for collecting user interactions involved either bringing users into the lab, or recording user interactions with kiosks deployed around campus. More recently, the group has begun leveraging the potential of Internet-based applications, and deployed our dialogue systems online, offering substantial gift certificates ($10 - $20) and requesting users to solve scenarios [7]. We make use of a 2,163 sentence corpus of data collected via these methods.

5.2.2. Crowd-sourced Speech Data

AMT offers a more cost-effective means to collect data from a larger user base. The method follows along the lines of deploying online systems, but specifically embeds the online systems in the AMT Internet marketplace. First, following the ideas in [1], we used AMT to request workers to generate the potential interaction scenarios. We then prepared a customized version of the *City Browser* system for live interaction via AMT. This version mimicked the same dimensions and look and feel of the mobile version. However, we made a small number of changes, such as replacing audio synthesis with text system responses, presenting incremental recognition results, and constantly displaying the task description. These changes were guided by the AMT user testing framework developed in [1]. We offered AMT workers $0.10 per completed scenario.

A total of 69 scenarios were collected from workers, of which 36 were usable. We modified some of the scenarios to locate them in Boston, as the Boston database is enhanced and the system is more mature. However, several of our scenarios involved ten novel cities, such as Folsom California, Orlando Florida, and Plano Texas. Four example scenarios are shown in Figure 4. In total we collected 2360 utterances from 124 scenario-based tasks and 93 users. A sample interaction is shown in Figure 5. A screenshot of the AMT task is shown in Figure 6.

Since AMT workers have been previously shown to be cost effective for transcription [1, 13, 14], we decided to also use them for this task here. We first conducted an experiment on a subset of the spoken utterances, to assess whether they would be effective for our task. The transcription task presented workers with audio files and our speech recognizer's top transcription hypotheses. Workers were then asked to correct the recognition output. A screenshot of this task is shown in Figure 7. The HIT had three workers transcribing each utterance, and workers were paid $0.01 per transcription. It took approximately one day to obtain transcriptions in triplicate of the 579 utterance set, for a total cost of under $30.

An alignment script automatically merged the transcriptions based on a voting scheme using transcription similarities. Manual examination of 50 utterances showed that approximately 90% of the sentences were transcribed without errors. Encouraged by these results, we then used AMT to obtain transcriptions for the remainder of our audio corpus, an additional 1781 utterances.

You're in New York shopping on Canal Street, when your friend calls and tells you to meet her at the Gershwin Theatre. Find out how to get there using public transportation.

You are helping friends find their way to the Cheer's Bar in the Boston Commons. Help them find their way from State Street to the Bar.

You want to take your little sister to the water park and hotel about 1 hour north of Vancouver, WA but don't know the name. Find out the resort's name and phone number.

You are driving through Northern California to meet a long lost family member at the historic Cascade theater located in Redding, California, but your not sure what freeway exit to take or where the theater is located. Find the theater address and freeway exit information.

Fig. 4. Four example scenarios generated by AMT workers.

U_1:	Find a hotel near Harvard University.
S_2:	There are 10 hotels near Harvard University. All of them are in Cambridge. Most are in Harvard Square. [*results shown on map*]
U_3:	[*user clicks results and scans listing*] Find a hotel in Harvard Square.
S_4:	There are 8 hotels near Harvard Square. They are concentrated on Cambridge Street and Bennett Street. [*results shown on map*]
U_5:	Find a hotel on Cambridge Street.
S_6:	There are 2 hotels near Harvard Square on Cambridge Street. The A Friendly Inn and the A Bed and Breakfast in Cambridge. [*results highlighted on map*]
U_7:	What is the address for A Friendly Inn on Cambridge Street?
S_8:	A Friendly Inn is located at 1673 Cambridge Street in Harvard Square in Cambridge. [*A Friendly Inn is focused on map*]

Fig. 5. A sample interaction between a user and the *City Browser* system. U indicates the user, and S indicates the system.

6. COMPARING CORPORA BY PERFORMANCE

Our next step was to analyze the effectiveness of the Turker-collected data (both text and speech) for language model training for the speech recognizer. We use the recognizer's

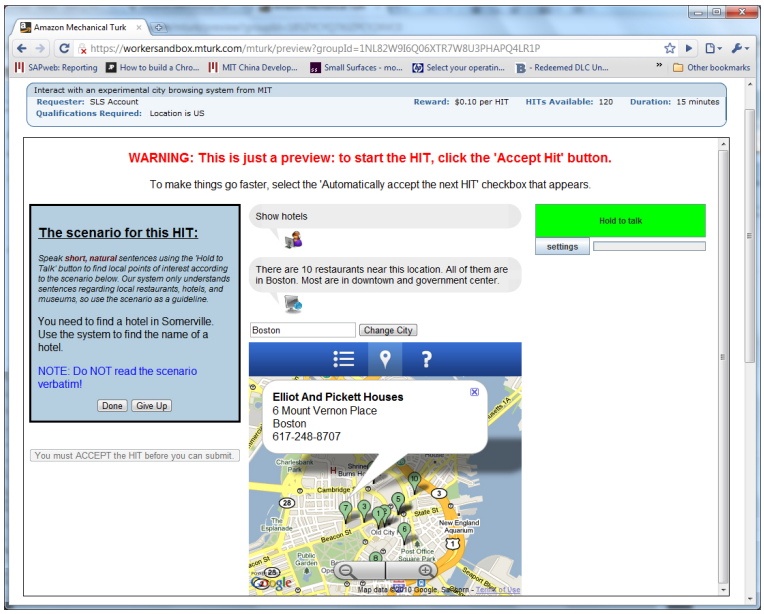

Fig. 6. A screenshot of the Mechanical Turk interface of *City Browser*.

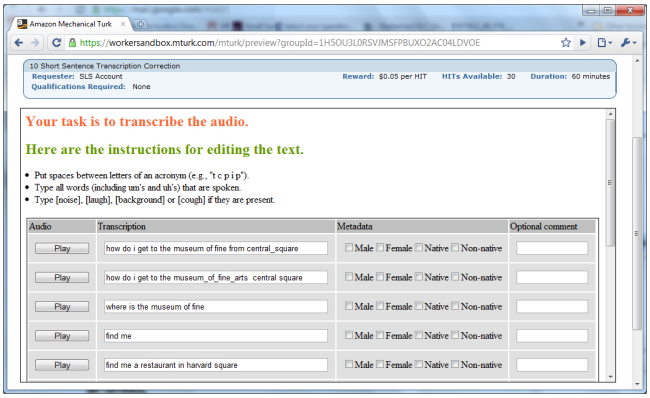

Fig. 7. The Mechanical Turk HIT for transcribing audio.

word error rate (WER) as the performance metric. Each of the experiments below uses a training corpus formed by configuring a different subset of the available corpora.

We set aside two corpora as test sets: a 289 utterance Boston-based corpus, and a 476 utterance non-Boston corpus. A set of 989 utterances from Boston-based scenarios formed our AMT speech training corpus.

It was at first unclear what to do about the proper nouns. In usage, the system is populated dynamically with proper nouns obtained from the database for the metropolitan region in dialogue context. However, this feature is difficult to implement in batch runs, and also difficult to interpret. So we decided to use the static full database of proper names for Boston, but to explicitly augment it with all the proper names that showed up in our test set. Thus, we are not evaluating the effect of out-of-vocabulary (OOV) proper names in these experiments. This seems appropriate, as what we are interested in is the degree to which our different data sets capture the sentence structure of the user queries.

	Language Model Training	BOS	Other
1.	corpus+templates (baseline)	25.6%	35.9%
2.	text	23.8%	35.0%
3.	speech	21.1%	30.7%
4.	text+speech	22.1%	30.2%
5.	corpus+speech	21.7%	30.4%
6.	corpus+text+speech	21.2%	30.9%
7.	corpus+templates+text+speech	22.9%	30.8%

Table 1. Language Modeling Experiments. The $CB\ corpus$ consists of 2,163 transcribed utterances from recorded user interactions with the previous *City Browser* system. The $templates$ are 20,000 sample sentences generated from manually created template files. The $AMT\ speech$ data consists of transcripts of 989 utterances drawn from AMT spoken interactions with Boston-based scenarios. The $AMT\ text$ represents 13,313 tagged text input queries. Results are reported as WER on two test sets: a 289 utterance Boston-based corpus (BOS), and a 476 utterance non-Boston corpus ($Other$).

We ran a number of experiments, evaluating language model performance using different combinations of training data. The complete results are shown in Table 1.

Experiments 2 - 4 demonstrate that crowd-based data collection can potentially replace conventional forms of data collection for spoken dialogue system development. Experiment

2 shows that text data collection alone (without any system involvement) performs favorably compared to our prior system's baseline data. Experiment 3 shows that even small amounts of Boston-based speech within the context of a spoken dialogue system is extremely effective. We believe the speech data were especially effective for Boston (yielding the best overall results) because the scenarios used for the Boston test data overlapped with the training scenarios. However, it is also interesting that the Boston-based speech data were highly effective for the non-Boston based test data. Experiment 4 gave the best results for the non-Boston scenarios overall.

Experiments 5-7 evaluated how well combinations of prior data and new AMT data could be used for language modeling. Although the WER results are not the lowest, we believe Experiment 6 would be more robust than the speech-only system (Experiment 3) because it is trained on a broader range of material including prior Boston-based speech data. It does appear however, that once data can be collected from users, the template-based data become less necessary. Using McNemar's test ($p<0.5$), we find the 17% relative reduction in WER between Experiments 1 and 6 on the Boston test data to be statistically significant.

7. CONCLUSION AND FUTURE WORK

Developing spoken dialogue systems is problematic because it is extremely difficult to acquire domain-dependent language model data. In this paper, we have examined the extent to which Amazon Mechanical Turk can be utilized to overcome some of the important hurdles to rapid system deployment. In particular, our results showed that AMT workers are useful for several aspects of system development, including (1) providing text corpora for system training even prior to the existence of the system, (2) creating scenarios to use in data collection experiments, (3) serving as subjects for dialogue-based data collection, and (4) transcribing the audio data collected through AMT interactions with the system.

We were especially interested in the question of the degree to which AMT data (both text and dialogue-based speech data) could be used to provide language model training for the speech recognizer. We were gratified to find that AMT text data (which could be collected prior to the existence of the system) were at least as good as the original templates and corpora we had previously collected in the laboratory for language model training. AMT speech training data provide further improvements in WER.

We also present here a new version of our system which exploits on-line databases to provide information for any city supported by Yelp. The model of linking directly to on-line sources alleviates much of the burden of maintaining up-to-date static databases.

As future work, we will continue to collect AMT dialogue interactions with the system, especially for cities other than Boston. We also want to investigate ways to use AMT to help with real-time vocabulary augmentation (both spelling and pronunciation).

8. ACKNOWLEDGEMENTS

Thanks to Ian McGraw, Chia-ying Lee, and Lee Hetherington for creating the framework to collect data using Amazon Mechanical Turk. We are grateful to Alex Gruenstein for creating the original *City Browser* dialogue system and guiding us in enhancing its capabilities.

9. REFERENCES

[1] I. McGraw, C. Lee, I. Hetherington, S. Seneff, and J. Glass, "Collecting voices from the cloud," *Language Resources Evaluation Conference*, 2010.

[2] A. Kittur, E. Chi, and B. Suh, "Crowdsourcing user studies with Mechanical Turk," *Conference on Human Factors in Computing Systems*, 2008.

[3] C. Callison-Burch and M. Dredze, "Creating speech and language data with Amazon's Mechanical Turk," *NAACL-2010 Workshop*, 2010.

[4] T. Paek, Y. Ju, and C. Meek, "People watcher: A game for eliciting human-transcribed data for automated directory assistance," *Proc. Interspeech*, 2007.

[5] J. Ledlie, B. Odero, E. Minkov, I. Kiss, and J. Polifroni, "Crowd translator: on building localized speech recognizers through micropayments," *ACM SIGOPS Operating Systems Review*, 2010.

[6] A. Gruenstein, *Toward Widely-Available and Usable Multimodal Conversational Interfaces*, Ph.D. thesis, MIT Department of Electrical Engineering and Computer Science, 2009.

[7] A. Gruenstein, I. McGraw, and I. Badr, "The WAMI toolkit for developing, deploying, and evaluating web-accessible multimodal interfaces," *ICMI*, 2008.

[8] J. Glass, "A probabilistic framework for segment-based speech recognition," *Computer Speech and Language*, vol. 17, pp. 137–152, 2003.

[9] I. Hetherington, "A multi-pass, dynamic-vocabulary approach to real-time, large-vocabulary speech recognition," *Proc. Interspeech*, pp. 545–548, 2005.

[10] G. Chung, S. Seneff, and C. Wang, "Automatic induction of language model data for a spoken dialogue system," *Proc. SIGDial*, 2005.

[11] A. Cucchiarelli, "Semantic tagging of unknown proper nouns," *Natural Language Engineering*, 1999.

[12] Y. Xu and S. Seneff, "Semantic understanding by combining extended cfg parser with hmm model," *Submitted to These Proceedings*, 2010.

[13] M. Marge, S. Banerjee, and A. Rudnicky, "Using the Amazon Mechanical Turk for transcription of spoken language," *ICASSP*, 2010.

[14] S. Novotney and C. Callison-Burch, "Cheap, fast and good enough: automatic speech recognition with non-expert transcription," *NAACL*, 2010.