Growing a Spoken Language Interface on Amazon Mechanical Turk

Ian McGraw, James Glass, and Stephanie Seneff

MIT Computer Science and Artificial Intelligence Laboratory, Cambridge, MA 02139, USA

{imcgraw, glass, seneff}@mit.edu

Abstract

Typically data collection, transcription, language model generation, and deployment are separate phases of creating a spoken language interface. An unfortunate consequence of this is that the recognizer usually remains a static element of systems often deployed in dynamic environments. By providing an API for human intelligence, Amazon Mechanical Turk changes the way system developers can construct spoken language systems. In this work, we describe an architecture that automates and connects these four phases, effectively allowing the developer to *grow* a spoken language interface. In particular, we show that a *human-in-the-loop* programming paradigm, in which workers transcribe utterances behind the scenes, can alleviate the need for expert guidance in language model construction. We demonstrate the utility of these *organic* language models in a voice-search interface for photographs.

Index Terms: organic speech systems, language modeling

1. Introduction

Spoken language interface development is subject to a classic chicken and egg problem: training data are needed to build a system; however, to collect in-domain training data, one first needs the system. To combat this reality, researchers are forced to play a delicate game of seeding the speech recognizer with aligned speech or adapting language and acoustic models from one domain to another. It is our contention that crowd-sourcing platforms such as *Amazon Mechanical Turk* are poised to fundamentally change the development process of such systems by either eliminating or automating these awkward initial steps of building spoken language interfaces.

To the casual user Amazon Mechanical Turk (AMT) is a convenient mechanism for distributing tasks via the web to an anonymous crowd of non-expert workers who complete them in exchange for micropayments. Although at times cumbersome, the web interface is often sufficient for accomplishing simple text-based tasks. For more complicated work, arbitrary web pages can be inserted into the AMT interface and command line tools can be used to manage large quantities of Human Intelligence Tasks (HITs). With these tools, speech researchers have begun to crowd-source annotation, transcription and even elicitation of speech data [1].

The true power of Amazon Mechanical Turk, however, can only be harnessed by manipulating tasks programmatically through an API. Only then is it possible to construct and combine tasks on-the-fly, allowing developers to create true *human-in-the-loop* algorithms. While the latency of such systems is certainly a concern, for many tasks it is surprisingly low. In [2], Bernstein et. al. describe crowd-sourcing fairly complex word-processing tasks with wait times of less than 20 minutes. Latency may also be optimized in domain dependent ways.

This paper describes an architecture that employs *human-in-the-loop* programming to facilitate spoken language interface creation using automatically collected and transcribed utterances. In particular, we focus on the task of automatically growing a language model on-the-fly using in-domain speech data without expert guidance. For this work, we consider the relatively unconstrained domain of photo annotation and search. We show that photo query recall improves dramatically over the lifetime of a deployed system that builds its vocabulary and language model from scratch by automatically coordinating a crowd of AMT workers to provide training data.

2. Previous Work

Since 2006 the term *crowd-sourcing* has become commonplace in the vernacular of researchers of many data-driven sciences. The natural language processing community was an early adopter of the technology [3], and its use has propagated to a variety of fields including speech [4, 5, 6].

The types of crowd-sourcing tasks found in the literature are also growing in sophistication. An elegant extension of a basic HIT, is one in which the results are iteratively improved upon by one or more subsequent HITs. In [7], for instance, Little et. al. work with a task that iteratively improves upon the interpretation of sloppy hand-writing. More generally, feeding the results of one task into another can be used to construct complex *human in-the-loop* processes, such as writing a wikipedia entry [8], or even performing "impossible" database queries where humans answer otherwise incomputable sub-problems of the task [9].

In the speech community, companies such as CastingWords have long made use of Amazon Mechanical Turk for transcription. AMT allows CastingWords to offer services with various quality and turn-around time guarantees, made possible through AMT's qualification tests and worker assessment features. Academic researchers use Amazon Mechanical Turk to create high-quality data sets, where it becomes necessary to compare or combine work from multiple workers to assure quality [6]. Conversely, for *training* a recognizer, data verification has been viewed as an unnecessary and costly step [5].

Moving from speech annotation to elicitation is a daunting endeavor due to technological hurdles. Incorporating a speech component into a task is problematic due to the additional server-side complexity. Tools such as those described in [10] and [11], both of which have been used to collect read speech, alleviate some of the impediments to speech elicitation. Taking this to the next degree, we have begun to use the open source WAMI Toolkit to build fully interactive multimodal interfaces deployable to Amazon Mechanical Turk [4]. In the following section, we describe this toolkit in detail and explain how we have extended it to incorporate features which allow us to create and study dynamic spoken language systems.

```
<script
 src=http://wami.csail.mit.edu/portal/wami.js />

<script>
var myWamiApp = new Wami.App( ... );

var grammar = {
    grammar : "...";
    language : "en-us";
    type : "jsgf";
};

myWamiApp.setGrammar(grammar);
</script>
```

Figure 1: A piece of the WAMI Javascript API.

3. WAMI's Javascript API

In-domain training data are the bread and butter of speech interfaces. The larger the audience of a spoken language system, the easier it is to collect these data. Early successes of the wide deployment of sophisticated spoken language systems from academia include MIT's *Jupiter* [12] conversational weather information system, and CMU's *Let's Go* [13] transportation information system. Since improvements to these types of systems are made using the interactions collected, the ability to tap into a large user-base is essential to their continued development. It is with this in mind that we have built the Web-Accessible Multimodel Interfaces (WAMI) Toolkit [10].

WAMI is a complex bundle of server-side and client-side machinery based on an Apache Tomcat, PostgreSQL stack. WAMI's core functionality is to provide all the necessary plumbing to get audio from the client side, typically a web browser, to a recognizer running server-side. The recognition results must also find their way back to the client, of course, and we also add in logging code to capture events and audio from user interactions. WAMI scales gracefully with an arbitrary number of users, and care was taken in each component to ensure efficiency and thread-safety.

Unfortunately, implementing these relatively basic features of WAMI requires a myriad of technologies, presenting a large barrier-to-entry for anyone wishing to incorporate speech features into a website. For this reason we have not only open-sourced the WAMI project, but we also host a third-party-accessible version of WAMI[1] which exposes the easy-to-use Javascript API shown in Figure 1. This allows web developers with no knowledge of speech science to incorporate basic recognition into their sites in just a few lines of client-side code. In [14], we describe Quizlet.com, a flashcard website which has used WAMI to speech-enable two educational games.

Until recently, the only language models (LMs) available to third-party developers through WAMI were small context-free grammars based on the Java Speech Grammar Format[2] (JSGF). One advantage of these grammars is that embedded semantic tags can be parsed and decoded from the recognition results. Still, a feature commonly requested was the ability to loosen the language constraints of our recognition service.

In conjunction with the work presented here, we are releasing two new methods for defining WAMI's language model through the web interface. Now, a `corpus` argument is an accepted `type` in the `grammar` object passed through to the recognizer. The text associated with the grammar is then assumed to be a set of sentences in the domain. These sentences are

[1] http://wami.csail.mit.edu

[2] http://java.sun.com/products/java-media/speech/forDevelopers/JSGF/

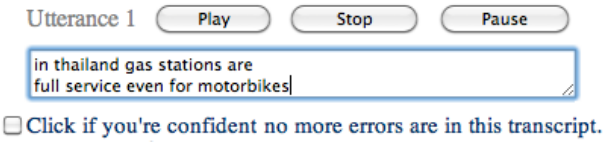

Figure 2: In the iterative transcription task above, batches of five utterances are combined into a 5¢ HIT. The first worker fills in a blank transcript which subsequent workers correct. A final worker approves the transcript if there are no more changes to be made.

then compiled into a trigram language model on-the-fly, which is then used in the recognizer for that session. Since it is not advisable to pass large amounts of data from the client, a `cached` type of grammar has also been implemented. In particular, we allow a third party developer to upload a corpus once, and provide them with an ID by which to reference it in their Javascript.

In this work, we extend the trigram compilation framework to add another language model type, `organic` language models, which, are *grown* via on-the-fly transcripts. Such a language model can be defined from multiple transcription sources, which are abstracted away into a table in the logging database. Currently, a background thread in WAMI is configured to check for updates to each language model defined in the database every ten minutes. If new transcripts are found they are seamlessly sent to the recognizer for LM recompilation. One could envision an adaptive language model whose complexity (e.g., n-gram size) is altered based on the amount of data or even test set performance. In this work, however, attention is restricted to the simple case where a set of transcribed utterances is compiled into a continually updating trigram language model.

While the framework is agnostic to the way in which the transcripts find their way into the database, Amazon Mechanical Turk is an obvious choice. Figure 2 shows part of a transcription HIT which can be deployed to AMT to gather utterances. Relying on a single worker is inadvisable when a certain level of accuracy is required of the transcripts. For this reason we have used AMT's Java API to implement an iterative HIT. The first worker is given a blank area in which to type the transcript. A subsequent worker will then determine if there are any changes to be made. The final worker checks a box indicating that the transcript is error-free. This continues to a maximum of five workers per utterance.

4. Photo Search: A Case Study

We decided on photo-annotation and search as the domain in which to perform our preliminary experiments on our generic framework for organic language models. In addition to spoken annotation, we allow the user to draw with the mouse. This gesture collection, however, is not the focus of this work.

The photo user interface described in the following subsections was written entirely in Javascript and required no domain-specific server-side code. On the back-end, the SUMMIT landmark-based recognizer is configured with a large hand-crafted lexicon and acoustic models trained on telephone speech. For words not in the lexicon, a letter to sound module generates pronunciations on-the-fly.

4.1. Photo Annotation

We devised a web interface where an individual can log into Picasa, Flickr, or Facebook to access their personal photos. For

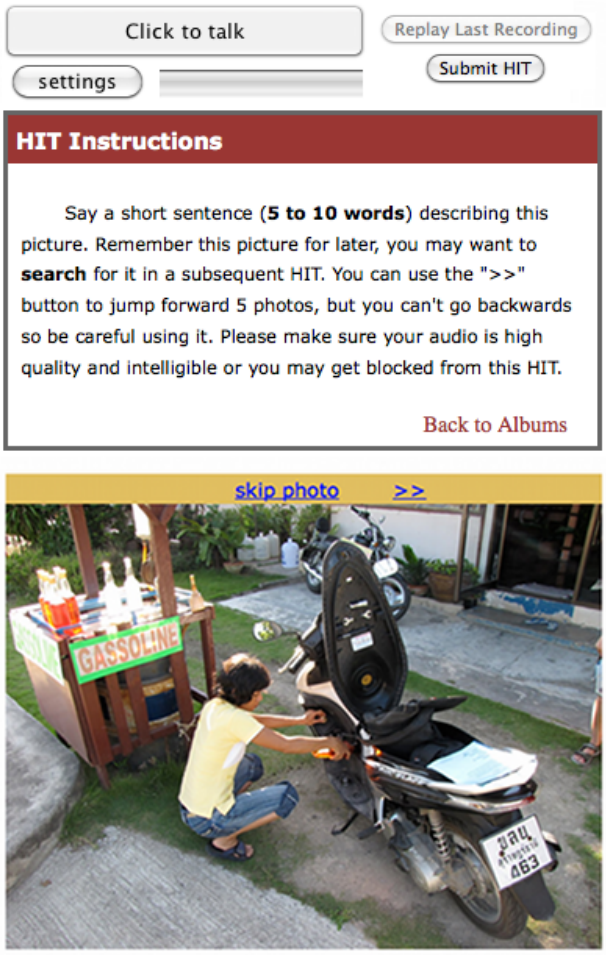

Figure 3: Photo annotation HIT. For every ten photos annotated with speech, a photo-search HIT would appear. The relative position of elements was altered to save space.

those with privacy concerns, we also created a version that uses random public images from Flickr. In either case, voice annotations are streamed to a recognizer configured from the client side to use the organic language model.

Since our photo annotation system is new, it has no cultivated user base, but we can exploit AMT to create a motivated user base through monetary award. To this end, we devised the photo-annotation HIT depicted in Figure 3, which encourages users to annotate photos for 10¢ a piece. This is relatively generous as AMT tasks go, but our reasoning was that a certain privacy barrier must be broken when people talk about their own photos. A similar task where workers could annotate public photos was also deployed for 5¢.

An organic language model, initially containing only a single word, "*nothing*", was grown over the life-time of the HIT by transcribing the collected speech behind-the-scenes via a separate but simultaneous iterative transcription HIT. As the language model matures, the hypotheses eventually become useful for photo search.

4.2. Photo Search

To measure the utility of recognition results from photo annotations, we designed a voice search component to our photo user interface. Since positing a novel speech-based photo-retrieval algorithm is beyond the scope of this initial prototype, we took a simple approach to the search problem. Instead of using the same recognizer configuration, a special context free grammar for photo search was constructed and compiled on-the-fly using code similar to the sample found in Figure 1. The recognition hypotheses from a set of voice annotations are stored to generate bags of words for each photo considered in the search: `<photo-i> = (word-1 | word-2 | ... | word-N)*` A few carrier phrases, such as `search for` and `find the`, were added to the grammar leading into these word-loops. Finally, semantic tags, e.g. `[photo=i]`, were embedded into each word-loop, to allow us to easily determine the query's hypothesized answer.

We inserted our search interface into the aforementioned annotation HIT in the following manner. After every set of ten photos, instructions were presented which asked the workers to think back over the photos that they had just annotated and create a voice search. A user might, for instance, say *"search for the photograph of my motorbike in thailand"* in hopes that the computer would display the photo shown in Figure 3. Upon displaying the hypothesized image, the computer asks the user to confirm whether this was the photo they were thinking of. We required the user to make three search queries before continuing with the annotation of more photos. Thus, provided the worker does not abandon the task early, each set of ten photo annotations is accompanied by three search queries.

	# workers	# utts.	# searches
personal	27	995	105
public	35	1099	117

Figure 4: Statistics for the two deployed photo annotation HITs.

4.3. Experimental Results

The two versions of our photo annotation HIT, running with either public or personal photos, were each deployed to Amazon Mechanical Turk for two days. We restricted the task to U.S. workers and required them to have an acceptance rating of over 75%. Still, since each user has their own microphone, speaking style, and subject matter, we decided to constrain the contributions of an individual worker over time, limiting each person to at most six searches an hour. Once a worker went over his or her limit, a warning appeared telling the user to come back after a couple hours if they wanted to do more. Lastly, we encouraged users to keep utterances relatively short, and explicitly warned them if the recognizer detected over 25 words.

Once we had finished testing our user-interface, we deployed our final HITs and set the organic language model in motion. Figure 4 displays some statistics regarding the two runs, each of which lasted 48 hours. It is interesting to see, for instance, that the additional 5¢ was enough to encourage workers to log into their personal accounts to complete the HIT. It is also clear that not everyone who starts working on the HIT makes it far enough to complete a search task.

The search error rate can be approximately determined by the workers' feedback on the correctness of the photo returned from the periodic search queries. While workers are not guaranteed to tell the truth, given that we restricted ourselves to the most trustworthy workers, and that they have no knowledge of our experimental interests, this assumption seems reasonable.

The average utterance length for the public photo annotation was 7.2 words, whereas for personal photos it was 6.6. The average delay between an utterance being logged in our system

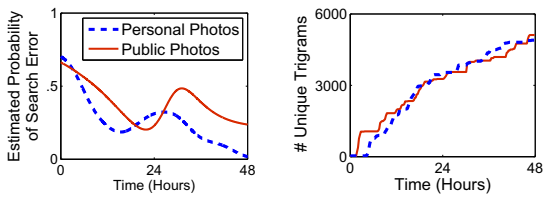

Figure 5: The plot on the left shows the probability of a search error occurring as a function of time as estimated by logistic regression. On the right, the number of unique trigrams in the organic language model is shown as it grows.

and a transcription being assigned to it via Amazon Mechanical Turk was 87 minutes. The high latency is due to a number of factors. Transcription HITs are bundled into groups of five, and so the first necessarily awaits the fifth before deployment. Furthermore, the background processes poll at fixed ten-minute intervals. This, along with the iterative nature of the HIT, requires a single utterance to go through a series of steps before being fully processed. Fortunately, the delay is still short enough to support studying system dynamics.

To determine the performance effects of the language model growth, the search error indicated by the worker is treated as a binary response variable dependent on time. Logistic regression was used to determine that the slope of the estimated probability of a search error over time is negative for both the public and personal photo tasks, with $p < .01$. We can also visualize the trend using kernel density estimation, as shown in Figure 5. The probability of a search error as a function of time is given by Parzen-Rosenblatt density estimation. On the right of Figure 5 is plotted the growth of the language model in terms of the number of unique trigrams. Despite similar growth, the search error rate trends downward, and is lower for personal photos. We believe that the personal photos were easier to remember, and the search queries were better matched to the annotations. Both plots exhibit considerable fluctuation, which we believe is due to variation across different users.

It is clear from the plots that the organic language models are operating as expected, improving the system behind the scenes with no expert input. The difference was dramatic enough that one worker emailed us with the comment: *"I just wanted to say that I have noticed that your system seems to be getting better at recalling specific pictures."*

5. Summary and Future Work

In this work, we have shown the feasibility of analyzing a dynamic spoken language system deployed on Amazon Mechanical Turk. We have explored growing trigram models using AMT for a spoken language interface, and shown that improvements can be achieved without expert guidance.

We hope this work opens the door to experimenting with other, more complex dynamic systems on Amazon Mechanical Turk. Obvious next steps for our framework include incorporating acoustic model training and perhaps even pronunciation learning. Beyond simple retraining, however, we envision systems that play an active role in their own learning, perhaps by choosing the data they wish to have transcribed a la [15].

AMT's API can empower programs to ask arbitrary questions about the world (e.g. "What is spoken in this audio clip?"). An important part of building these *human-in-the-loop* applications will be ensuring that they ask the right questions to cost-effectively improve system performance.

6. Acknowledgements

This work was funded in part by the T-party project, a joint research program between MIT and Quanta Computer Inc., Taiwan. We would also like to thank Chia-ying Lee for the transcription interface, Scott Cyphers for photo API work, Ming Zhu and James McGraw for assistance with data analysis, and all of our Mechanical Turk workers.

7. References

[1] C. Callison-Burch and M. Dredze, "Creating speech and language data with amazons mechanical turk," in *NAACL 2010 Workshop: Creating Speech and Language Data With Amazons Mechanical Turk*, 2010.

[2] M. S. Bernstein, G. Little, R. C. Miller, B. Hartmann, M. S. Ackerman, D. R. Karger, D. Crowell, and K. Panovich, "Soylent: a word processor with a crowd inside," in *Proceedings of the 23nd annual ACM symposium on User interface software and technology (UIST)*, 2010.

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

[4] I. McGraw, C. Lee, L. Hetherington, and J. Glass, "Collecting voices from the cloud," in *Proceedings of LREC*, May 2010.

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

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

[7] G. Little, L. B. Chilton, M. Goldman, and R. C. Miller, "TurKit: tools for iterative tasks on mechanical turk," in *HCOMP '09: Proceedings of the ACM SIGKDD Workshop on Human Computation*. ACM, 2009, pp. 29–30.

[8] A. Kittur, B. Smus, and R. E. Kraut, "CrowdForge: Crowdsourcing complex work," Human-Computer Interaction Institute, School of Computer Science, Carnegie Mellon University, Tech. Rep., Feb. 2011.

[9] M. Franklin, D. Kossmann, T. Kraska, S. Ramesh, and R. Xin, "CrowdDB: Answering Impossible Queries," in *SIGMOD*. ACM, 2011.

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

[11] I. Lane, A. Waibel, M. Eck, and K. Rottmann, "Tools for collecting speech corpora via Mechanical-Turk," in *NAACL Workshop on Creating Speech and Language Data With Amazons Mechanical Turk*, 2010.

[12] V. Zue, S. Seneff, J. Glass, J. Polifroni, C. Pao, T. Hazen, and L. Hetherington, "JUPITER: A telephone-based conversational interface for weather information," *IEEE Transactions on Speech and Audio Processing*, vol. 8(1), 2000.

[13] A. Raux, D. Bohus, B. Langner, A. Black, and M. Eskenazi, "Doing research on a deployed spoken dialogue system: One year of Let's Go! experience," in *Proceedings of INTERSPEECH-ICSLP*, 2006.

[14] A. Gruenstein, I. McGraw, and A. Sutherland, "A self-transcribing speech corpus: Collecting continuous speech with an online educational game," in *Proceedings of the Speech and Language Technology in Education (SLaTE) Workshop*, 2009.

[15] B. Varadarajan, D. Yu, L. Deng, and A. Acero, "Maximizing global entry reduction for active learning in speech recognition," in *Proceedings of ICASSP*, 2009.