

A Web Application for Filtering and Annotating Web Speech Data

David Lutz
Linguistics
Cornell University
del82@cornell.edu

Parry Cadwallader
Computer Science
Cornell University
wbc35@cornell.edu

Mats Rooth
Linguistics and CIS
Cornell University
mr249@cornell.edu

Abstract

A vast and growing amount of recorded speech is freely available on the web, including podcasts, radio broadcasts, and posts on media-sharing sites. However, finding specific words or phrases in online speech data remains a challenge for researchers, not least because transcripts of this data are often automatically-generated and imperfect. We have developed a web application, “ezra”, that addresses this challenge by allowing non-expert and potentially remote annotators to filter and annotate speech data collected from the web and produce large, high-quality data sets suitable for speech research. We have used this application to filter and annotate thousands of speech tokens. Ezra is freely available on GitHub¹, and development continues.

1 Introduction

A vast and growing amount of recorded speech is freely available on the web, including podcasts, radio broadcasts, and posts on media-sharing sites. Much of this speech is accompanied by automatically-generated transcripts, and content providers and hosts often provide the ability to search these transcripts— and therefore the audio— for tokens of specific words or phrases. While these search features are usually designed for users to find content on topics that interest them, their potential use as a source of speech data has not been lost on researchers. Howell and Rooth (2009) and Howell (2012) developed methods for automating data collection using these sorts of search engines using command line programs that interfaced with external search engines. This contribution continues the same research effort.

¹<https://github.com/del82/ezra>

Whether collected automatically or not, the hits search engines return do not yet represent data for the linguistic or speech researcher. There remains a significant amount of work to convert these hits into useful speech tokens. First, each search hit must be manually filtered to determine whether it represents an actual token or is a false positive resulting from an error in the transcript. When the token is present, it must be extracted from the longer audio file it appears in, often with some surrounding context. Manual annotation is also frequently called for, depending on the nature of the specific study. For example, it may be necessary to record information about the speaker, the context, or other semantic, pragmatic, or discourse factors that may affect the token’s acoustic properties or the way it was produced. The time required for researchers to filter and annotate hits in this way represents a major limiting factor in the efficiency of web speech data collection and therefore in the quantity and quality of web data that is available for speech research.

To address this efficiency challenge, we have developed a web application, which we call “ezra” that provides a simple but flexible interface for non-expert users to filter and annotate web-harvested speech data efficiently, and produce large and high-quality data sets suitable for speech research. Early versions of the application have been used within our own research group to process thousands of speech tokens. For one study, which examines the effects of semantic context on prosody, we collected tokens of the phrase *in my mind*. Published corpora were of limited utility for this study; transcripts of the Buckeye corpus (Pitt et al., 2007), for example, contain three tokens of this phrase. Using our application and freely-available audio from two radio stations,² we were able to collect and annotate

over 750 tokens of this phrase. By increasing the efficiency of filtering and annotating web audio, our application allows researchers to collect data to address very specific questions on a scale that had not previously been possible. The application is freely available.³

2 The application

Ezra fits into a workflow in which users collect speech data from the web containing potential tokens of the words, phrases, or other phenomena under investigation. Usually, the data are collected using the web harvest method detailed in (Howell and Rooth, 2009), where command line programs interface with a site that has indexed audio using automatic speech recognition (ASR). Once collected, data are imported into ezra, filtered, transcribed, and annotated based on the requirements of the specific research being undertaken. Once processing is complete, the speech tokens, transcript, and annotations are exported for analysis.

Targets and hits

Annotation is centered around a *target*, which is a collection of search hits for a single word or phrase, like “in my mind”, “some people”, or “South Korea”. Our specific research is concerned with targets that show focus prosody, or more generally have interesting patterns of prosody or variation in prosody. Each *hit* in a collection is a purported token of the target word or phrase. The goal of the application is to make as efficient as possible the process of *filtering* the hits, i.e. separating the genuine tokens of the phrase from the transcription and/or search errors, and *annotating* the genuine hits by correcting their transcripts and adding new information to their metadata.

Users

Users of the application are divided into two roles: supervisors and annotators. Supervisors are linguists who are working on a problem where a large sample of naturalistic uses of the target is expected

²The radio stations were WNYC (<http://www.wnyc.org>) and WEEL (<http://www.weel.com>) which use (or have used) media search tools from RAMP (RAMP, 2011). http://www.ramp.com/case_study/weelcentercom/ provides a case-study description of the RAMP audio search application at WEEL.

³Ezra is open-source and is available at <https://github.com/del182/ezra>. We welcome suggestions and contributions.

The screenshot shows the Ezra application interface. At the top, there are navigation buttons: 'Save', 'Previous', 'Next', 'Next Unconfirmed', and 'Cancel'. Below this, the main content area is divided into several sections. On the left, there's a search bar with 'any players' and a count of '6964'. Below the search bar are playback controls for a hit, including 'start 2s', 'selection', and 'last 2s'. A central player window shows a transcript: 'since the Bruins as you point out or or as you believe remain committed to Claude Julien uh do you envis do you see there being any players on the horizon anyone available via trade who could help the team'. Below the transcript are two more playback controls: 'start' (0:05:23, 825) and 'end' (0:05:38, 150). On the right side, there's a 'Features' section for 'Det N prosody' with a list of prosodic features like 'main stress', 'downstep', and 'distfluency'. At the bottom, there's a 'Notes:' field.

Figure 1: The result of annotating one hit. Boundaries have been marked, and the utterance transcribed. In the prosodic feature markup at the right, “=-” indicates a default prosody for “any players” where *player* is more prominent, but *any* still bears some stress. Buttons allow playing of the window, or of a shorter interval surrounding the target words. The Notes area at the bottom is used for free-form comments and interaction between users.

to provide evidence about theoretical issues, and to allow explicit models of the relation between acoustic form and linguistic levels (such as semantics and phonology) to be estimated.⁴ Supervisors identify targets, arrange for data to be collected, import data, and export it from the application after a target has been filtered and annotated. Supervisors also design *features*, which are annotation tasks that are carried out for each hit of a target. Finally, supervisors are able to view the activities of other users, helping to monitor the progress of annotators and to allocate effort to different targets.

In contrast, annotators may not create targets or features, import or export data, or view other users’ activities. Rather, annotators are focused on the filtering and annotation task. This division of roles, and attendant difference in privileges, allows annotators to focus on processing data and supervisors to concentrate on the tasks that precede and follow annotation.

⁴See (Howell, 2012) and (Howell et al., 2013) for an example of this research program.

18. some people

Total: 448
 Confirmed: 383
 Unconfirmed: 0
 Not Present: 56

Features

| Name | Created by | Number of targets | Instructions |
|---|------------|-------------------|--|
| focus (Edit) | 4 | 3 | Is the target word in the ngram (e.g. "South" in "South |
| Phonological phrase position (Edit) | 9 | 2 | PPhrase-initial should be marked if the phrase is at the beginning |

Hits

← Previous 1 2 3 4 5 6 7 8 9 ... 14 15 Next →

| Hit Number | Status | Flagged |
|----------------------|-------------|---------|
| 4303 | Not Present | |
| 4304 | Confirmed | |
| 4305 | Confirmed | |
| 4306 | Confirmed | |
| 4307 | Confirmed | |

Figure 2: Part of the summary page for the target *some people*. Counts are at the top: of 448 hits, 383 were confirmed as containing the target, and 56 hits did not contain the target. The remainder were marked as repeats, or flagged as having problems of other kinds. The Features area summarizes the feature design. The Hits area at the bottom displays confirmation status of individual hits. Through links at the left, the annotation page for an individual page can be accessed, or a sequential record of annotation steps for the hit displayed, including the user who made the annotation.

Features

In addition to filtering the hits and correcting the transcripts surrounding genuine hits, supervisors may also specify other annotation tasks to be carried out when each hit is processed. These tasks, called *features*, are created within the web interface by a supervisor, and then assigned to a target. They may require the user to select one or more properties from a list, or they may ask for some text response. Features may include questions like: is the target focused or not? Was the token uttered by a man, woman, or child? Was it uttered by a native or non-native speaker? Each target may have multiple associated features, and each feature may be associated with multiple targets. See Figures 1 and 2 for examples. When processing each hit of a target, the user responds to all features that are assigned to it. The feature values are saved with the hit and exported along with the audio and the rest of the annotations.

The inclusion of features in ezra is designed to allow supervisors to include arbitrary annotation tasks with the filtering and transcription tasks. This ensures that each hit need only receive attention from a human once; after a hit is processed,

its metadata will include all of the necessary information for the specific analysis for which it was produced.

3 Workflow

Once a target has been created in the system and hits have been imported, annotation begins. The annotator interacts with a hit through the web display seen in Figure 1. For each search hit, the annotator listens to the audio file around the time when it should, according to the ASR transcript, contain the target word or phrase. If the transcript is incorrect and the target is not present, the annotator notes that and moves on. Access to the ASR transcript, although it is imperfect, helps to orient the annotator and speeds up the filtering step. If the target is present in the audio, the annotator marks its exact location, and also marks the boundaries of a larger phrase or sentence in which the target appears, called the *window*. The annotator corrects the transcript of that window if necessary, and sets the values of each of the associated features for the token.

In our use of the application, for targets with no associated features but which required the annota-

| Id | Name | Total | Confirmed | Unconfirmed | Not Present | Flagged |
|----|------------------|-------|-----------|-------------|-------------|---------|
| 1 | Yankees | 482 | 292 (0) | 2 (2) | 132 (0) | 3 |
| 2 | that house | 227 | 67 (0) | 1 (1) | 140 (0) | 1 |
| 3 | New York Yankees | 75 | 4 (0) | 67 (0) | 1 (0) | 0 |
| 4 | that of | 554 | 114 (0) | 0 (0) | 394 (0) | 0 |
| 5 | south korea | 308 | 265 (1) | 1 (0) | 19 (0) | 1 |
| 6 | in my experience | 128 | 85 (0) | 0 (0) | 31 (0) | 0 |
| 7 | in my hand | 101 | 28 (0) | 0 (0) | 68 (0) | 0 |
| 8 | in my house | 164 | 66 (0) | 0 (0) | 69 (1) | 2 |
| 9 | really did | 441 | 251 (5) | 7 (7) | 119 (10) | 23 |
| 10 | really can | 179 | 75 (5) | 1 (1) | 79 (13) | 21 |
| 11 | really could | 160 | 91 (2) | 0 (0) | 49 (5) | 7 |
| 12 | really should | 227 | 140 (4) | 0 (0) | 50 (5) | 9 |

Figure 3: Targets page in ezra, giving summary counts for different targets.

tor to transcribe 10-15 seconds of audio surrounding the token, our annotators were able to filter and annotate about 60 hits per hour. Targets with more complex annotation requirements take longer; recent results suggest that about 45 hits per hour is achievable by an experienced annotator for these targets. Note that this rate depends on how many of the hits are false positives; a hit which does not contain a token is filtered in 20-30 seconds, but a hit which contains a token, and must therefore be annotated, may take three or four times that long to process. In our data, about 60% of the hits have been genuine tokens, while about 40% have been false positives or otherwise problematic.

When she creates each feature, the supervisor sets the possible values it may take for each token, and may include instructions for annotators, which serve as a ready reference while the annotator works. If an annotator is uncertain about how a particular token should be annotated, he may flag that token for further attention from the supervisor. Thus annotators can work without direct, immediate supervision without being forced to make decisions about which they are unsure, potentially introducing errors into the annotation.

Because ezra is a web application, annotators can work from anywhere, needing only a reason-

ably fast internet connection and a modern web browser. Robust user authentication and authorization allows the application to be deployed on the open web. Members of our group include researchers at three universities in two countries, and the application provides a shared environment in which to filter and annotate web speech data. Users may log in from anywhere, and only logged-in users may access the data. We found that the shared environment was important in coordinating our work. For instance, research leads for different targets can access ezra to check the progress of annotators, communicate about criteria for feature markup, and allocate annotation effort. Figures 2 and 3 show ezra pages that are used for examining summary results and the progress of annotation for a single target, and for all the targets together.

After the speech data has been filtered and annotated, the results are downloaded by the research lead for analysis. The download contains audio snippets, accurate transcripts, and the values of any features that were associated with the target. Each hit in the system is assigned a unique identification number on import, and retains this identifier through filtering, annotation, and export, allowing every audio clip to be uniquely identified and referenced in research and publications. In our

current workflow, after export we use the McGill ProsodyLab Aligner to create a phone-level alignment between the clip and a phonemic counterpart of the transcript. See (Gorman et al., 2011) and (Howell, 2012) for this methodology. Figure 4 shows a web presentation of the *any players* dataset that displays the alignment and allows audio to be played.⁵ The alignment for hit 6964 is accurate. We found that in getting a good alignment, it is crucial to have a transcript that includes disfluencies (such as *uh* in Figure 4) and repeated words (such as *or or* in Figure 1).

In addition to the standard workflow, we have experimented with a pre-filtering workflow, where the research lead filters the data and marks approximate temporal boundaries. Then the annotator creates the transcription and adjusts time boundaries to agree with word boundaries. This workflow has the advantage of allowing the lead to select on a theoretically-informed basis a window that includes the information which is relevant to what is going on in the discourse. For instance, for investigations of contrastive prosody, the preceding context may include an overt contrasting phrase that should be included in the window. While annotators working in the standard workflow also select a window which allows a listener to figure out what is happening in the discourse, the pre-filtering workflow allows the research lead to make the decision in a way that will allow specific hypotheses to be evaluated. The *any players* dataset seen in Figures 1 and 4 was created using the pre-filtering workflow.

Data collection and import

Before speech data can be processed in ezra, it must first be collected from the web. The specific type of data collected, and therefore its method of collection, depends on the goals of the research being undertaken. Ezra does not do the data collection, though we hope to add that capability through plugins in the future, as discussed in section 5.

In our work on prosody, our targets have been short two- or three-word phrases, and we have

⁵This web presentation, which is independent of ezra, is available at <http://compling.cis.cornell.edu/digging/>. In addition to the authors, Lauren Garfinkle contributed to the *anyplayer* dataset as annotator, and Kyle Gorman, Michael Wagner, and Jonathan Howell contributed in the implementation and tuning of ProsodyLab Aligner. Underlying audio data is property of WEEI. The graphical panel was produced with the Matlab code available at <http://www.cs.cornell.edu/~mats/matlab/>.

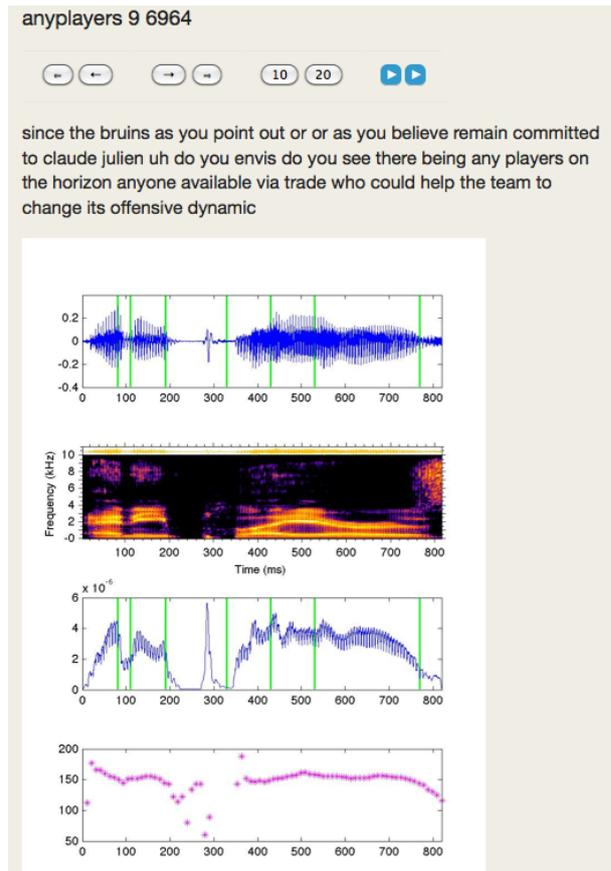


Figure 4: A web presentation of the *any players* dataset. The graphical panel displays an oscillogram, spectrogram, and plots of intensity and pitch. The green vertical lines indicate a temporal alignment for the seven phones of *any players* that was generated by using ProsodyLab Aligner.

collected our data from two radio stations in the northeastern United States using media search functionality available on the stations' websites. The searches of these websites were automated using tools similar to those reported in Howell and Rooth (2009) and Howell (2012), which are able to conduct searches for specific phrases automatically and retrieve the URL of the audio file, the location in the file where the token is purported to appear, and in some cases a transcript of the audio surrounding the purported hit. This information is loaded into our application.

Because our own data collection relies on the media search functionality of content creators, the data available to us is limited to that provided by this search functionality. However, there is nothing inherent in the application that limits users to these types of searches or this type of data. The minimum requirement for each hit imported

into the system is that it provide an audio file (local or remote), and a time in that audio file in which the target is purported to appear. Any method of searching or collecting media that produces this information can be used for data collection. For example, researchers may have a collection of audio files with transcripts containing speaker metadata, part-of-speech information, or syntactic or other structural information. Any time this information is automatically-generated, using e.g. automated speaker identification tools, part-of-speech taggers, or parsers, it is liable to contain errors. Ezra is designed to make the identification of false positives and the correction of errors as efficient as possible.

Put another way, ezra is intended to improve the *precision* of searches of audio data by allowing human users to filter and annotate the data quickly, removing false positives. It does not improve the *recall* of these searches, i.e. removing false negatives, as it has access to only those hits that the user imports.

In the future, we hope to develop partnerships with content providers that will allow us to address the challenge of limited recall, making the data our application produces more useful to language researchers for whom range of content, linguistic variety, and search recall are important considerations. Content providers might be willing to work with us to develop a way to search their audio which sacrifices precision in favor of recall—perhaps by providing the top three or top five most probable transcriptions generated by the speech-to-text system—which our human annotators could then correct, possibly helping them improve their transcripts and recognition system.

Despite these challenges in data collection, we view our application in its present state as one that can be of great benefit to researchers working with web speech data.

4 System Architecture

Ezra is written using the popular open-source Ruby on Rails⁶ web application framework. It comprises a browser-based user interface, an application layer running on a web server, and a relational database in which user and application data is stored. Users access the application via a web browser. The interface is built using standard web technologies, including HTML5 generated with

⁶<http://rubyonrails.org/>

Ruby's standard ERB⁷ template system, jQuery⁸, and Twitter's Bootstrap⁹ JS and CSS. The audio is played in an embedded player using SoundManager 2¹⁰, which uses HTML5 to play the audio in browsers that support it and falls back on Adobe Flash for browsers that don't. The audio player has been specifically designed for the filtering and annotation task, and provides controls for playing only the audio window, only the token, or only the first or last two seconds of the window. Our annotators have found that these controls help them set the audio window and annotate the hit as efficiently as possible.

The server with which the client communicates is a Ruby on Rails application, which handles requests, including user authentication and authorization, and interaction with the database. Rails is open-source, mature, popular, well-documented, and straightforward to install on modern operating systems. Because of its popularity it is well-supported by other web technologies, and free tools exist that make it straightforward to deploy on a production web server such as Apache or nginx. At present, deploying and administering ezra requires some knowledge of Ruby and Rails, but we hope to reduce or eliminate this requirement as development continues.

Database

Ezra stores its data in a SQLite¹¹ database. SQLite is a simple lightweight relational database management system that stores the database in a file on disk. Rails provides a simple and powerful Object-Relational mapper through which the database can be accessed, and which provides a layer of abstraction which makes it possible to use, and migrate between, more full-featured database systems like MySQL and PostgreSQL with minimal changes to the application code. While this flexibility allows users to deploy ezra with their existing database infrastructure, we do not anticipate that usage volume will ever reach a level where SQLite is not fully adequate for our needs.

In addition to user, configuration, and audit data, the database contains records for each target, hit, and feature in the system. To simplify

⁷<http://ruby-doc.org/stdlib-1.9.2/libdoc/erb/rdoc/ERB.html>

⁸<http://jquery.com/>

⁹<http://twitter.github.io/bootstrap/>

¹⁰<http://www.schillmania.com/projects/soundmanager2/>

¹¹<https://www.sqlite.org>

updating and auditing, no records are ever deleted from the database. Users who are no longer a part of the project can be disabled, invalidating their login credentials and preventing access to the private parts of the system. Their user records remain in the database, however, because they contain a record of the work the user has done.

Information contained in hit records includes the audio file the hit appears in and its location within that file, whether the hit has been confirmed to contain the token of interest (or found to be a false positive), and all annotations associated with that hit. The hit record also contains two annotator-selected time points within the file that together demarcate the audio window containing the hit. This window is usually between 8 and 20 seconds long, and its boundaries correspond to the transcript the annotator produces of the hit. That is, the audio window is that portion of the audio in which the words in the transcript are uttered.

The audio files themselves are not stored in the database, but on disk alongside the database file. Each hit record contains the filename of this audio file, so that the application can serve the audio file along with the rest of the hit data. Like all records, hit records are retained indefinitely, even when a human annotator indicates that the target token is not present in the audio (a false positive), or when a hit is found to be a duplicate of another hit in the database.

Database statistics

Our ezra deployment contains, at the time of this writing, 9908 hit records of 35 targets, of which 6307 have been processed. Of those, 3928 (about 62%) have been confirmed as genuine tokens of their respective targets and annotated, 1971 (about 31%) have been marked as false positives, and 408 (about 6%) have been found to be duplicate hits (wherein the audio token indicated in the hit is an exact copy of another hit) or otherwise problematic. These numbers continue to increase as more hits and targets are added to the database and as annotators process hits. The SQLite database file containing this information is about 6 megabytes on disk.

5 Future work

In addition to working to improve the quality and quantity of data available to import into ezra, we continue active development on the application it-

self, driven by the feedback and suggestions from the researchers and annotators who are using it. While our first priority is always to make the filtering and annotation process as efficient as possible, there are several features we hope to add or improve in the near future.

- More complete user auditing and statistics would make it easier for supervisors to interact with annotators. The system keeps track of every change made to a hit, including the user who made the change, but not including the specific changes that were made. We'd like to improve this auditing functionality, and also make useful statistics available about the filtering and annotation work being done.
- We would like to add plugin functionality to both the import and export ends of ezra. For importing, allowing users to add integration with existing search engines and other data sources would greatly improve the quantity and diversity of data available. For exporting, plugins could integrate with other tools, e.g. the ProsodyLab Aligner (Gorman et al., 2011), or other manipulation or analysis tools.
- Although access to the audio and annotation functionality requires authenticated users, the application also serves publicly-accessible pages, which can be used to post papers and descriptions of the research being conducted. We would like to expand this functionality, making it easier for members to update the public pages via the web interface, and also to make selected audio and annotations available via the public site.
- Occasionally, data collection will generate duplicate hits, where two hits indicate the exact same audio token. These are not always from the same audio file, so comparing metadata will not always prevent duplicates. We would like to develop a method of detecting duplicate hits from the audio, and flagging them for further human examination.
- Because ezra is a web application accessible from anywhere, annotation work could be crowdsourced using e.g. Amazon Mechanical Turk, which has been used successfully

for transcribing spoken language data (Marge et al., 2010) and gathering linguistic judgments (Sprouse, 2011), though its use is not uncontroversial (Fort et al., 2011). We'd like to explore how we could update authentication and authorization to allow researchers to crowdsource their annotation if they so choose.

Ezra is open-source software, and its development is hosted in a public GitHub repository at <https://github.com/del82/ezra/>. This repository hosts the code, a public issue tracker, and a wiki containing user documentation that is being developed concurrently with the application. We invite fellow speech researchers to use ezra and contribute to its development with code, issue reports, feature requests, and contributions to the wiki.

As it stands, our application makes web speech data more accessible to researchers by providing a browser-based interface for filtering and annotating search results based on imperfect transcripts. It emphasizes simplicity in its interface, efficiency in its use of human annotators' time, and flexibility in the definition of annotation tasks and in the location of researchers and annotators. Although development and the addition of new features continues, the application has already been used to filter and annotate thousands of speech tokens, and represents a meaningful step in making the vast quantities of speech data on the web much more accessible for speech research.

Acknowledgments

Neil Ashton wrote the documentation available on GitHub. Ross Kettleson, Michael Schramm, and Lauren Garfinkle have been very patient and helpful annotators as ezra has been developed; Lauren was the annotator for the *any player* and *some people* datasets. Anca Chereches is the supervisor for the *some people* dataset. Jonathan Howell and Michael Wagner assisted by providing requirements and supervising the annotation effort at McGill. Scott Schiller, the author of SoundManager 2, provided help with functionality for playing sound intervals. Our research was supported by grant NSF 1035151 RAPID: Harvesting Speech Datasets for Linguistic Research on the Web (Digging into Data Challenge). This work is a contribution to a collaborative project in which Wagner and Howell are partners. The work of the

McGill team was supported by SSHRC Digging into Data Challenge Grant 869-2009-0004.

References

- Karën Fort, Gilles Adda, and K Bretonnel Cohen. 2011. Amazon Mechanical Turk: Gold Mine or Coal Mine? *Computational Linguistics*, 37(2):413–420.
- Kyle Gorman, Jonathan Howell, and Michael Wagner. 2011. ProsodyLab-Aligner: A tool for forced alignment of laboratory speech. In *Proceedings of Acoustics Week in Canada*, pages 4–5, Quebec City.
- Jonathan Howell and Mats Rooth. 2009. Web Harvest of Minimal Intonational Pairs. In *Web as Corpus 5*.
- Jonathan Howell, Mats Rooth, and Michael Wagner. 2013. Acoustic Classification of Focus: On the Web and In the Lab. Ms. Brock University, Cornell University, and McGill University
- Jonathan Howell. 2012. *Meaning and Prosody: On the Web, in the Lab, and from the Theorist's Armchair*. Ph.D. thesis, Cornell University.
- Matthew Marge, Satanjeev Banerjee, and Alexander I Rudnicky. 2010. Using the Amazon Mechanical Turk for transcription of spoken language. In *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 5270–5273. IEEE.
- M.A. Pitt, L. Dilley, K. Johnson, S. Kiesling, W. Raymond, E. Hume, and E. Fosler-Lussier. 2007. Buckeye Corpus of Conversational Speech (2nd release).
- RAMP 2011. Universal Search Re-defined. White paper. <http://www.RAMP.com>
- Jon Sprouse. 2011. A validation of Amazon Mechanical Turk for the collection of acceptability judgments in linguistic theory. *Behavior research methods*, 43(1):155–67, March.