Vapor Engine: Demonstrating an Early Prototype of a Language-Independent Search Engine for Speech

Douglas W. Oard,
Rashmi Sankepally
University of Maryland
College Park, MD USA
{oard|rashmi}@umd.edu

Jerome White
New York University
Abu Dhabi, UAE
jerome.white@nyu.edu

Craig Harman
Johns Hopkins HLTCOE
Baltimore, MD USA
craig@craigharman.net

ABSTRACT

Typical search engines for spoken content begin with some form of language-specific audio processing such as phonetic word recognition. Many languages, however, lack the language tuned preprocessing tools that are needed to create indexing terms for speech. One approach in such cases is to rely on repetition, detected using acoustic features, to find terms that might be worth indexing. Experiments have shown that this approach yields term sets that might be sufficient for some applications in both spoken term detection and ranked retrieval experiments. Such approaches currently work only with spoken queries, however, and only when the searcher is able to speak in a manner similar to that of the speakers in the collection. This demonstration paper proposes Vapor Engine, a new tool for selectively transcribing repeated terms that can be automatically detected from spoken content in any language. These transcribed terms could then be matched to queries formulated using written terms. Vapor Engine is early in development: it currently supports only single-term queries and has not yet having been formally evaluated. This paper introduces the interface and summarizes the challenges it seeks to address.

1. INTRODUCTION

The best present approach to searching naturally occurring spoken content is to tune the vocabulary, the pronunciation model, and the language model of a Large-Vocabulary Continuous Speech Recognition (LVCSR) system to generate terms using Automatic Speech Recognition (ASR) [10]. By recognizing words in the spoken content to be indexed, typed text queries become possible. This is an expensive process however, requiring thousands of US dollars in some cases for languages in which state-of-the-art LVCSR systems already exist. For languages for which no LVCSR systems exist—several thousand, worldwide—that cost can balloon by an order of magnitude or more. Given the expense, alternatives involving phonetic recognition have been developed. Phonetic recognition is less language-specific than word recognition, but, because they operate with fewer constraints, phonetic recognizers suffer from low accuracy; and as the size of the phonetic inventory grows that accuracy degrades further. Going even further in the direction of language generality (although at some cost in the ability to generalize over speakers with dissimilar voices), a so-called zero-resource term discovery technique has been developed that is capable of using (approximate) acoustic repetition as a way of identifying indexing terms. Early work with this technique found roughly 50% term recall at a 10% false alarm rate. Recent work has found that, when sufficiently rich spoken queries are available, indexing terms generated in this way can also be useful as a basis for ranked retrieval. This paper describes an early prototype system for selectively transcribing some of those terms, which can then be used as single-term queries. This work is a first step toward the development of a fully functional system for interactive exploration of speech collections in languages for which no LVCSR systems are available.

We build on previously published work in which the "zero-resource spoken term detection" method for automatically detecting indexable terms based on acoustic repetition has been described [4]. In this paper, the focus is on the design of the collection exploration tool that we propose to demonstrate, which we call Vapor Engine. Vapor Engine is an interface for interactively examining, annotating, and using terms detected using zero-resource spoken term detection to explore a collection of recorded speech. The zero-resource spoken term detection runs as a batch process prior to indexing, building a mapping of terms to their respective recordings. Vapor Engine then presents the user with an interface for interactively browsing the term space and the recording (i.e., the "document") space, for replacing some term identifiers with readable strings, and for using those readable strings to facilitate selection of recordings that contain content that is of interest. The present version of Vapor Engine implements the following capabilities for interaction: a) a simple term cloud visualization of the terms detected in the collection as a whole, as well as in specific recordings; b) a facility to rapidly play a sequence of variants for a specific term and to manually annotate every occurrence of that term in the collection by making a single entry of a readable text representation for that term (which is similar in spirit to an approach that has been proposed to rapidly annotate clustered images [3]); and c) a primitive single-term query facility that displays a selectable list of recordings that contain any specified term.

CHIIR '16, March 13–17, 2016, Carrboro, NC, USA

ISBN 978-1-4503-3751-9/16/03…$15.00

DOI: http://dx.doi.org/10.1145/2854946.2854987

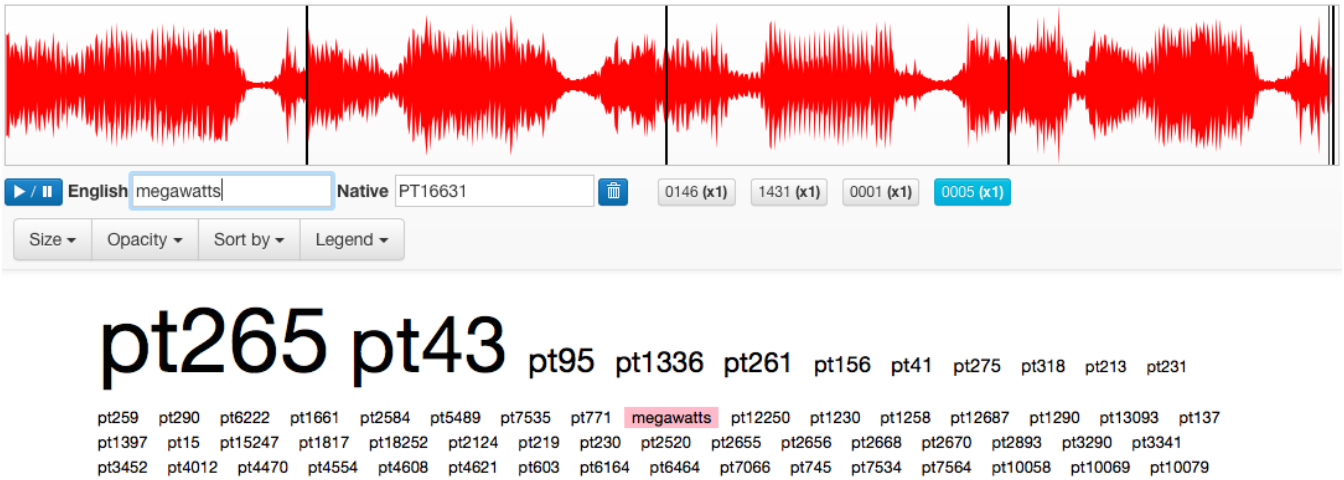

Figure 1: Vapor Engine word cloud view. Terms (indicated by the characters "PT" followed by a number), are presented at the bottom of the view. In this case, they are sized based on their frequency. Selecting a single term plays that term across all recordings in the corpus; the waveforms at the top of the view are each such instance. Terms can be renamed to something more linguistically familiar to the user. In this case, term "PT16631" was selected; there four instances of the term in the corpus. Playback revealed that it as the word "megawatt," so the term was renamed (English/Native below the waveform). In doing so, the term name is updated in every recording view where that term appears.

Vapor Engine is currently available on the Web.[1] That version indexes a freely redistributable collection of Enron phone calls in English [5]. We propose to demonstrate the system using that collection and two other licensed collections: a Gujarati collection from MediaEval for which ranked retrieval results have been reported [16], and the English Buckeye linguistics research collection from the Linguistic Data Consortium [14].

The remainder of this paper recaps how zero-resource spoken term detection works, reviews prior work on access to spoken content, describes the design of Vapor Engine and its principle use-case and outlines planned next steps.

2. ZERO RESOURCE TERM DISCOVERY

Zero-resource term discovery uses unsupervised learning to locate repeated units from digitized speech [6, 9, 13, 18]. Vapor Engine was designed, specifically, for the zero-resource spoken term discovery system developed by Dredze et al. [4]. Given a set of audio recordings, the system detects regions of similar speech patterns across those recordings. It assigns unique identifiers, or *terms*, to such sets of matching regions. The regions are identified by their acoustic patterns, as opposed to higher-level linguistic features, such as phonemes, syllables or words. Because of this, terms do not necessarily respect word boundaries; and their time occurrences can overlap. Indeed, it is not uncommon to have a single time point in a speech signal associated with dozens of terms. In an analogy with text, consider the terms 'periodicity,' 'period,' 'per,' 'city,' and 'it'; without spaces or other evidence, it might be wise to generate them all, which is essentially what zero-resource spoken term detection does.

Recorded speech can thus be represented as a set of terms that are found in this way. With appropriate attention to the temporal structure of term overlap, these terms can be indexed as a basis for retrieval [12, 16]. Vapor Engine, by contrast, simply views each recording as a bag of terms, which it conventionally label as "PT" followed by a unique numeric identifier.

[1] http://vapor.umiacs.umd.edu

3. RELATED WORK

This review of related work focuses on interactive exploration of spoken content, mostly with an eye toward illustrating the rather basic state of Vapor Engine's present development. SCAN [17] is an interface for audio navigation that addresses common user deficiencies in perusing audio data. Specifically, it allows users to search content based text transcriptions, as well as hear summaries and read text. One of the key underlying contributions of the work is the concept of, "what you see is almost what you hear," in which audio summaries are formatted in ways that are commonly used only in textual representations—varying text size, for example, which do not have a one-to-one correspondence to raw audio. Such features are incorporated into Vapor Engine based on statistical analysis of zero-resource output.

SpeechLogger [2] is a tool for searching and browsing transcripts from speech recognition. The interface allows users to search transcripts, and quickly playback audio from corresponding hits. SpeechLogger was designed to offer good search experience irrespective of transcription quality; thus, a major focus of the work was on integration and presentation of various retrieval systems to the end user.

Ranjan et al. [15] attempt to improve audio transcription through a specialized audio browser. The browser presents a transcription of the audio to the user; the user can then click on various text segments to hear the corresponding audio. Part of what the authors wanted to study was whether such an interface could aid in search tasks, even when transcription was erroneous.

Abdulhamid and Marshall [1] use a modified treemap to display ordered segments of audio. Each segment presents its underlying audio as a word cloud, with text presented based on its frequency within the segment. Users can play

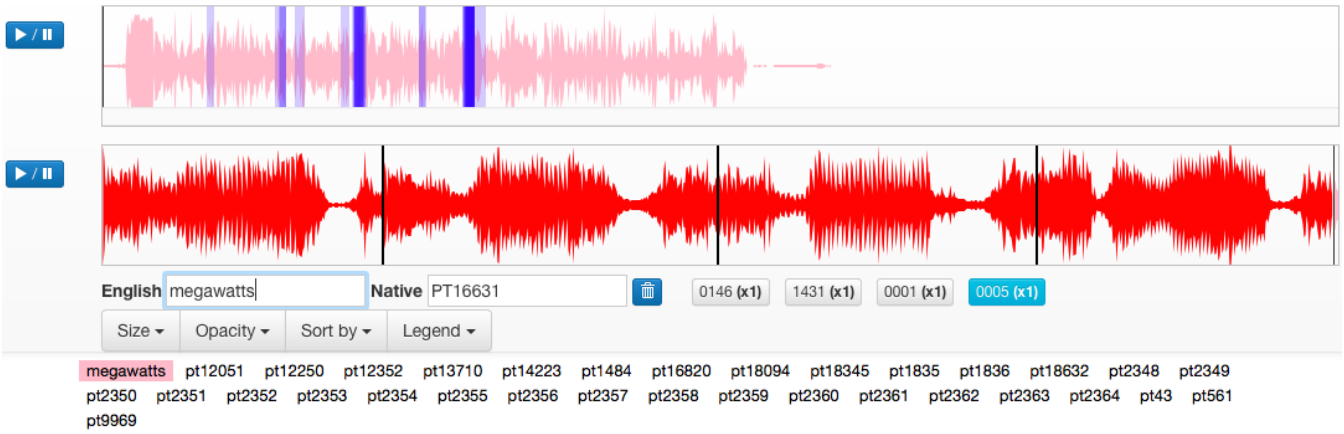

Figure 2: Vapor Engine recording view. The top-most waveform is the audio from the recording itself; the lower waveform is that of the selected term (in this case term "PT16631," renamed "megawatts."). Terms comprising the recording make up the lower portion of the screen, with the currently selected term highlighted.

audio from the segment while also seeing how that audio relates to the overall recording. The authors found that the use of word clouds as a presentation device aided in user topic selection; Vapor Engine takes a similar approach.

Each of the audio browsing methods relies on some transcription of underlying audio, a layer of processing that zero-resource systems do not provide. Thus, Vapor Engine can also be compared to interfaces specifically designed for audio transcription. Amazon Mechanical Turk has been a popular platform for transcription. Efforts over naturally occurring speech, such as what is studied in this work, include that of Novotney and Callison-Burch [11] for conversational speech, and Marge et al. [8] for meeting speech. Task organizers present segments of audio for workers to transcribe. This requires manual segmentation on the part of the organizer, along with a decision on how much audio to present versus the number of jobs to advertise. Presenting terms in this setting would likely be suboptimal due to the number produced, and their relationship to the underlying audio.

Luz et al. [7] turn the transcription task into a game. The primary focus of the game is to refine the output of speech recognition by allowing user corrections. Sentence-level transcription lattices are continuously presented to the user. The user's task is to select the correct word within portions of the lattice containing ambiguities. Because there are not always gold-standards for transcription, player scores are based on communal agreement. Thus, the game is similar to a crowdsourced task in which the incentive is non-monetary.

4. ARCHITECTURE

Vapor Engine is a term organization and display tool, capable of maintaining metadata about the terms themselves. It can be used by developers to enhance understanding of how zero-resource systems categorize matching regions of audio, or by searchers to explore a collection. Here we focus on that collection exploration use case. When started, Vapor Engine first builds a database of term information; most notably, each term's duration and recording association. Vapor Engine relies on having the original audio files analyzed by the zero-resource engine so that it can facilitate playback at arbitrary—term specific—points within each file. Vapor Engine also calculates document frequency for each term to build an understanding of how terms relate to each other.

5. PRESENTATION

At the top-level, Vapor Engine provides two corpus-level views: a word cloud, and a recording list. Each view allows a user to play a given term and alter the default name of the term. As will be discussed later, name alteration plays an important role in the utility of the system overall.

5.1 Collection View

The collection view presents a flat list of all terms; essentially a simple word cloud. By default, terms are initially listed alphabetically using a font size that is proportional to their collection frequency (i.e., the sum of the term frequency over all recordings); they can be re-sorted based on document frequency (i.e., the number of recordings in which a term occurs) if desired. Figure 1 shows the word cloud view sorted by document frequency.

Each term is presented as a link. By following the link, a subwindow is loaded with each occurrence of the term. Occurrences are displayed using their waveform, taken directly from the portion of the respective recording. In addition, each recording that contain the term appears as a link.

Users have the option to play all occurrences of a term in sequence. They can also annotate the label for a term to something more meaningful than the Vapor Engine default (which is PT followed by a unique identifier that is otherwise meaningless). An edit to the label for a term occurrence in any recording will be automatically reflected for the same term in all other recordings.

5.2 Recording View

The second view is the recording view (Figure 2). When the recording view is selected, Vapor Engine presents all recordings in the collection as a set of links. By following a link, users are taken to a page similar to that displayed in the collection view, but populated only with terms from the respective recording. The recording view has the same options for sorting and resizing terms as the collection view.

The recording view shows the entire recording as a light-red waveform with vertical indigo-colored lines that indicate the occurrence of terms. As the recording is played,

the waveform turns red. When the indigo colored lines are encountered, all term occurrences to which it pertains are highlighted yellow.

As was the case in the collection view, when a term is selected, all instances of that term are displayed as a waveform and can be played. Further, all recordings in which the term appears are listed as hyperlinks, thus essentially providing a single-term query functionality. This allows the user to move between recordings that share common terms.

6. USE CASE

One of the primary uses envisioned for Vapor Engine is corpus exploration. To explore the hundreds of hours of audio that our present zero-resource spoken term detection system can index without Vapor Engine, users would have to randomly select portions for listening. Even then, without any easy way of seeing how different portions of the audio collection are related, that understanding would be fragmented at best.

Zero-resource term detection simplifies this task by linking similar content in different recordings and by essentially allowing efficient note-taking—any time content in one recording is understood, its meaning is noted in all recordings. Through this process, it is expected that users will build out regions in the collection that are well enough summarized by the relatively few manually labeled terms that have been heard to provide a basis for navigation between recordings. Such a summary will allow users to recognize when other recordings share several of the same terms and potentially even topics. Moreover, as users gain experience with the system, there is the possibility that they will develop a sense for what cues—collection frequency, document frequency, term duration, etc.—are useful guides to prioritizing their labeling effort.

Ultimately, we seek to help users maximize their ability to characterize the contents of a collection from the specific perspective(s) that interest them, for any given level of effort. We seek to support that in three ways: a) minimizing the users listening requirement; b) minimizing the users transcription effort; and c) streamlining the interface so that, for example, many instances of the same term can be played while labeling that term. Through creative management of terms, Vapor Engine allows users to focus on more cognitively demanding activities and be more efficient with their retrieval efforts.

7. CONCLUSION AND FUTURE WORK

About half the world's population speak a language for which no LVCSR system currently exists; and approximately one fifth of that half can speak and hear, but cannot read or write. For those users—10% of the world's population—zero-resource spoken term detection offers the promise of some degree of information access. In their raw form, however, zero-resource systems require some amount of effort to navigate. Vapor Engine is a means to lower the barrier of entry, improving the technologies' accessibility. This work presents an early prototype of Vapor Engine based on an actual working system. Much work remains, but the design of a tool for exploring spoken content using zero-resource term detection is no longer a blank slate. Going forward, it is imperative to learn whether users find Vapor Engine simple, intuitive, and effective in its present form—a usability study is currently being conducting toward this end. The tool should also be extended to support multi-term queries using techniques that have been shown to be effective in batch experiments. Such multi-term queries require annotation of several terms, which in turn requires new ways of prioritizing terms for annotation; that is, methods beyond the sizing of terms by document frequency. Ultimately, that prioritization should be sensitive to factors such as overlap that could minimize duplicate effort, and to factors such as co-occurrence with already-annotated terms that could foster the construction of highly discriminating queries.

8. ACKNOWLEDGMENTS

This work has been supported in part by NSF 1218159.

References

[1] F. Abdulhamid and S. Marshall. Treemaps to visualise and navigate speech audio. In *OZCHI*, 2013.
[2] L. Begeja et al. A system for searching and browsing spoken communications. In *NAACL-HLT*, 2004.
[3] J. Cui et al. Easyalbum: an interactive photo annotation system based on face clustering and re-ranking. In *CHI*, 2007.
[4] M. Dredze et al. NLP on spoken documents without ASR. In *EMNLP*, 2010.
[5] J. Goldstein et al. Annotating subsets of the enron email corpus. In *CEAS*, 2006.
[6] A. Jansen and B. Van Durme. Efficient spoken term discovery using randomized algorithms. In *ASRU*, 2011.
[7] S. Luz et al. Supporting collaborative transcription of recorded speech with a 3D game interface. In *KES*, 2010.
[8] M. Marge et al. Using the Amazon Mechanical Turk to transcribe and annotate meeting speech for extractive summarization. In *NAACL-HLT*, 2010.
[9] A. Muscariello et al. Unsupervised motif acquisition in speech via seeded discovery and template matching combination. In *ICASSP*, 2012.
[10] K. Ng. *Subword-Based Approaches for Spoken Document Retrieval*. PhD thesis, MIT, 1990.
[11] S. Novotney and C. Callison-Burch. Cheap, fast and good enough: Automatic speech recognition with non-expert transcription. In *NAACL-HLT*, 2010.
[12] D. Oard et al. The FIRE 2013 question answering for the spoken web task. In *FIRE*, 2013.
[13] A. Park and J. Glass. Unsupervised pattern discovery in speech. In *ICASSP*, 2008.
[14] M. A. Pitt et al. The Buckeye corpus of conversational speech: Labeling conventions and a test of transcriber reliability. *Speech Communication*, 2005.
[15] A. Ranjan et al. Searching in audio: the utility of transcripts, dichotic presentation, and time-compression. In *CHI*, 2006.
[16] J. White et al. Using zero-resource spoken term discovery for ranked retrieval. In *NAACL-HLT*, 2015.
[17] S. Whittaker et al. SCAN: Designing and evaluating user interfaces to support retrieval from speech archives. In *SIGIR*, 1999.
[18] Y. Zhang and J. Glass. Towards multi-speaker unsupervised speech pattern discovery. In *ICASSP*, 2010.