An Extensible Web Tool for the Collection, Sharing, and Annotation of Audio Documents

by

Yang Yang

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2008

Author ....
Department of Electrical Engineering and Computer Science
September 2, 2008

Certified by .....
Dale Joachim
Visiting Professor
Thesis Supervisor

Accepted by .....
Arthur C. Smith
Chairman, Department Committee on Graduate Theses

An Extensible Web Tool for the Collection, Sharing, and Annotation of Audio Documents

by

Yang Yang

Submitted to the Department of Electrical Engineering and Computer Science
on September 2, 2008, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

The growth of the World Wide Web has been accompanied by a proliferation of rich, time-based media such as audio and video documents. However, the ability to categorize and index these documents has not improved comparably, a limitation for humans and machines alike to browsing, retrieving, and making sense of them.

Motivated by a need to collect and annote natural sounds, this thesis proposes a web-based audio annotation system and describes its design and implementation. Users upload audio using a browser or via cell phone stream, and add timed annotations in the form of comments, tags, or metadata. Plug-ins allow custom scripts to parse metadata and add functionality to the user interface.

Finally, applications in the fields of environmental monitoring, oral histories, and music performance and analysis are described. In particular, the flexibility of cell phones and adaptability of plug-ins makes the system applicable to other disciplines and is the main contribution of this work.

Thesis Supervisor: Dale Joachim
Title: Visiting Professor

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant No. 0634690. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the author and do not necessarily reflect the views of the National Science Foundation.

Contents

List of Figures

Glossary of acronyms and terms

AJAX Asynchronous JavaScript and XML – the name of a group of techniques used to create interactive web applications, coined by Jesse James Garrett [1].

DOM Document Object Model – an object model specified by the W3C for representing and rendering webpages. Many programming languages implement DOM, including JavaScript, and thus are able to modify webpage layouts dynamically.

ECMA European Computer Manufacturers Association – the non-profit standards organization that publishes language specifications for ECMAScript, of which JavaScript is a dialect.

HTTP Hypertext Transfer Protocol – a network protocol used for transmitting webpage documents over the Web.

JavaScript – a programming language (properly, a dialect of ECMAScript) that is commonly used for client-side processing and interaction with web applications in a browser.

MIME Multipurpose Internet Mail Extensions – an Internet standard used to describe the content type of data transmitted via e-mail and other networked applications.

NCSA National Center for Supercomputing Applications – a research center at the University of Illinois at Urbana-Champaign[1], where Marc Andreessen and Eric Bina wrote the Mosaic web browser.

RDF Resource Description Framework – a group of specifications published by the W3C to describe and structure machine-readable metadata.

[1]Website URL: http://ncsa.uiuc.edu

VoIP Voice over Internet Protocol – a network protocol used for the transmission of voice audio through the Internet and implementation of digital telephony.

W3C World Wide Web Consortium – an organization, founded in 1994 by World Wide Web creator Tim Berners-Lee, that creates standards and publishes specifications for the Web[2].

XMLHttpRequest – a JavaScript object that provides support for asynchronous HTTP requests, thus allowing a script to communicate with a server without interrupting the user's interaction with a web application.

[2]Website URL: http://w3.org

Chapter 1

Introduction

Since its humble beginning in 1989 as a pet project of Tim Berners-Lee at CERN[1], the web has in recent years exploded in size, diversity, and visibility [2]. People connect to it to read popular news items, listen to internet radio, and share photographs of their families and videos of their eccentricities. They buy books, learn about obscure topics, tell their friends, who in turn tell their other friends. They connect with one another.

In addition to web traffic, connectivity and bandwidth have also increased, resulting in an increase in the amount and variety of web media to which people are exposed. Gone are the days when plain-text and HTML documents were average-sized downloads sprinkled with patience-demanding images; today, time-based media like audio and video take on those roles, their download times made bearable by the possibility of streaming, or playing while loading.

Search engines have honed their skill at indexing and searching for text documents and, to a lesser extent, images. However, audio and video documents are generally not as well-treated. While a significant body of work has been done in the area of using speech and language processing techniques to index audio documents [3, 4], generally the best heuristics of machine processing are comprised of techniques such as examining a speaker's vocabulary to estimate the content of speech [5]. Additionally,

[1]Originally the *Conseil Européen pour la Recherche Nucléaire*, CERN has been renamed the European Organization for Nuclear Research

these techniques are difficult to apply to audio documents that do not contain speech.

Thus, there is little support for searching general audio and video documents based on higher-level metadata, and audio and video sharing services are content to make note of the media's author, title, and perhaps a short description, a depth comparable to the first few lines of a webpage's HTML code[2]. Given this lack of metadata, how can we make sense of the rich audio and video content now on the web?

Granted, it is much easier for a search engine to index an HTML page than to index an audio or video document. The latter would require a substantial effort in signal processing and image recognization. Even if a machine knows what it is listening to or watching, the task still remains of describing it in a way that is understandable by humans. These are very difficult tasks, and may remain so for a long time.

1.1 Goal and motivation

The solution, at least in the interim until machines become "smarter," is to enlist the help of humans in annotating the media. Annotations can be seen as a form of metadata, or supplementary information that helps describe the primary data, in this case audio and video. Metadata also provides a way to attach semantics to media, so that meaning (possibly subjective meaning) that is not inherent to the media can be interpreted and used by humans and computers alike [6].

Returning to the context of the web, having metadata attached to an audio or video document increases the ability and likelihood of it being categorized and indexed effectively. The benefit is that these media will then be just as browsable and retrievable as text documents. Consider the possibilities of browsing a set of musical pieces by instrumentation, or searching through a set of home videos by the date they were shot, not the date they were created or uploaded to a website.

The original motivation for this work was to provide a convenient and flexible way for the Owl Project, to be described in further detail in section 5.1, to collect

[2]The music service Pandora (http://pandora.com) does seem to be aware of certain musical properties such as mood or instrumentation. However, this information is used solely as approximate metrics for recommendation, not categorization or retrieval.

and annotate recordings of natural sounds. Participants in the Owl Project must be able to make recordings in forests where computer access may not be available. In additional, the project seeks to involve an online community of bird enthusiasts to contribute their knowledge in making sense of the collected recordings. As a result, a number of features in this work were designed with the intent of being leveraged by the Owl Project.

In this context, recordings of owl vocalizations are treated as audio documents containing natural stories gathered from the environment, just as other audio documents may contain stories originating from humans. Examples of the latter include oral histories and musical performances. However, the contribution of this work is not limited to its existing applications, but rather the flexible and extensible way in which users collect, annotate, and explore audio documents. As will be seen in chapter 5, the generality with which the collection and annotation of various kinds of stories are performed makes this work applicable to many disciplines.

1.2 Scope and overview

In order to address these shortcomings, the objective of this thesis is to design and build a web-based shared annotation system for audio documents. Note that this objective explicitly does *not* attempt to deal with video documents. The rational for this decision is twofold. Due to the fact that video almost always implies an audio track, audio annotation can be considered a stripped-down version of video annotation, for the sake of tackling a simpler problem first. Furthermore, many of the considerations and insight related to building a system for audio can be directly applied to the video counterpart. In the context of the web, it is their time-based nature and opaqueness to machine interpreters that are key, after all.

The remainder of this thesis is organized as follows:

- Chapter 2 is a brief survey of annotations systems, noting their advantages and the aspects that need to change in order to accommodate audio annotations.

- Chapter 3 integrates the findings of the background survey and provides a set of design requirements for the annotation system.
- Chapter 4 describes Krik Krak, a prototype system that addresses the previously stated design requirements.
- Chapter 5 explores a number of possible applications for an audio annotation system, emphasizing aspects of the design that may be especially suitable to the area of the application.
- Chapter 6 concludes the thesis with a summary, an evaluation of the system for future work, and a brief perspective on the project as a whole.

Chapter 2

Background

2.1 An overview of web annotation systems

Before examining the task of annotating time-based media, this section will first navigate a brief history of general annotation systems. The concept of a knowledge-sharing "memex" in which users could collaboratively build "trails" of ideas linked to documents was famously envisioned by Vannevar Bush in 1945 [7]. It was the World Wide Web that supplied the networking capacity to implement Bush's concept. Due to the bandwidth available during the early days of the Web, it was text documents and images that first started being annotated.

2.1.1 Mosaic web browser

Web-based annotation systems have existed since 1993, when Andreessen and Bina wrote Mosaic, the browser that popularized the World Wide Web [8, 9]. Developed at the National Center for Supercomputing Applications (NCSA), the initial version of Mosaic was designed with asynchronous collaboration functionality in mind, in the form of the Mosaic group annotations [10]. This approach consisted of attaching text and voice annotations to documents located on the web, and included an access control scheme that allowed for private, group, or public annotations.

Mosaic's default implementation provided only one centralized annotation server,

which did not allow users to customize the view to the annotations or filter them based on search queries. The annotations interface, being integrated into the Mosaic client, was not available to users of other web browsers, and as such its usage ceased by the time NCSA discontinued development and support for the browser in 1997.

2.1.2 ComMentor

Shortly after the release of Mosaic, Röscheisen et al. [11] leveraged the work done by NCSA to build the ComMentor system at Stanford University in 1994. The ComMentor system was built on NCSA's HTTP server and Mosaic browser as an implementation of a general architecture for shared "meta-information" on the web. While the architecture is designed to be browser-agnostic, it does involve the transferring of non-standard meta-information to the browser and relies on the browser's ability to recognize the meta-information; the ComMentor implementation uses a custom MIME type to transport meta-information and requires augmenting the basic Mosaic browser to handle it.

Nevertheless, ComMentor introduced a useful concept that improves on Mosaic's built-in model of group annotations. Rather than merely attaching annotations to the document in question, ComMentor users are able to place annotations at any position in the document. Furthermore, these positions may be shared between users as "landmark" reference positions. As a result, this model represents each document not as an indivisible quantity but a continuum, a metaphor well-suited for time-based documents.

2.1.3 CoNote

The CoNote annotation system, developed in 1995 by Davis and Huttenlocher [12] at Cornell University, provides a similar mechanism for the placement of annotations at specific points of a document, with the restriction that the allowable points must be chosen ahead of time by the document's author or submitter. In contrast to the browser-centric implementations of Mosaic and ComMentor, however, CoNote's

architecture involves inserting the annotations directly into the HTML stream. This approach is supported by all web browsers, thus decoupling the interface from the browser.

2.1.4 GrAnT

It was the Open Software Foundation's Group Annotation Transducer ("GrAnT"), however, that addressed many of the limitations of Mosaic, ComMentor, and CoNote [13]. Developed by Schickler et al. in 1996, GrAnT synthesizes the advantages of many of these approaches while introducing a number of distinguishing features. As with the CoNote system, GrAnT provides a browser-independent interface by merging annotations into the document stream and avoiding the need for special browser support.

Moreover, annotations in GrAnT are attachable not only to the document as a whole or arbitrary points in the document, but also specific selections of text. The ability to annotate sections of the document builds upon the metaphor of the document as a continuous stream that can be arbitrarily segmented.

GrAnT also introduces the model of annotation sets, in which each annotation is associated with an annotation set, and each set resides on an annotation server. Administrators are expected to define sets to represent topics, and annotators are expected to add annotations to the relevant set, though neither practice is required by design. In conjunction with annotation sets, GrAnT allows users to query for annotations according to inclusion or exclusion from a particular set, thus providing a basic mechanism by which users can create filtered views of annotations.

2.1.5 Annotea

As a step toward building a Semantic Web [15], the more modern Annotea system was created in 2001 by Kahan et al. at the W3C [14]. The aim of Annotea was to leverage the Resource Description Framework (RDF) infrastructure in order to formalize the information stored in annotations. Annotations themselves are typed,

allowing users to classify the annotations as well as the documents.

Annotea uses RDF schema to describe annotation types, and moreover users may create custom annotation types according to the RDF specification. Thus, a typed annotation is treated not merely as text; it may be classified, for example, as a typographical correction, an editorial response, or even a more specialized statement about a poem's rhyme scheme.

In addition to classification, the semantic-richness of annotations are useful for customizing client-side views of annotations. The client prototype of Annotea, built on the W3C test-bed browser Amaya[1], is equipped with a filter that allows users to show or hide annotations based on the author, type, or server location. The client also allows users to hide annotations manually, providing a flexible way to show only annotations relevant to the user.

While so far only a handful of shared web annotation systems have been considered out of the many that exist [16], it is sufficient to note the advantages and drawbacks of each in order to draw a better picture of what can make an annotation system effective and helpful.

Generally speaking, the flexibility of being able to annotate certain points or long passages in a document is important, and even more so when dealing with time-based media such as audio. In shared web annotation it is moreover crucial for a user to be able to filter out non-relevant annotations and show only those of interest. And in particular the requirement of Annotea that annotations be typed appears to increase the semantic value of annotating.

2.2 Annotation of non-textual media

So far text documents have been the focus of shared web annotation. While most of the insights pertinent to the annotation of text can be applied directly to the anno-

[1]Website URL: http://w3.org/Amaya

tation of non-textual media, the latter does require some additional consideration.

2.2.1 Image annotation

The group annotations system built into NCSA's Mosaic browser in fact allowed both text documents and images to be annotated. Of course, there was no feature for positioning annotations within the target document, as users could only attach them to the document as a whole. Even as later annotation systems such as ComMentor and CoNote added this feature by treating the document as a continuum, either support for image annotation was nonexistent altogether or images continued to be treated as single units.

However, it is clear that an image exists as a spatial continuum, and that for an image annotation system to be on par with the more mature text annotation systems, spatial posititoning at a more granular level must be possible. Consider, for example, the task of using annotations to label an image that is a geographical map; certainly attaching the visible cities to the image as a whole is not nearly as useful as pinpointing their location on the map.

2.2.2 AKTive Media

As an effort to expand the task of annotation beyond text, in 2006 the AKTive Media tool was built by Chakravarthy et al. at the University of Sheffield to support annotation of cross-media documents [17]. In this case, HTML pages containing both text and images may be annotated. In the image annotation modality, AKTive Media users identify a portion of the image by dragging a rectangular area using the mouse. A minor drawback of this user interface is that only rectangular and elliptical areas may be selected, not points, lines, or other more complex shapes.

In conjunction with allowing annotations on parts of images, AKTive Media allows its annotations to be either highly structured and typed, as with the RDF-defined annotations of Annotea, or informal as with the free text method of earlier systems. Information in structured form, including resource metadata such as document author

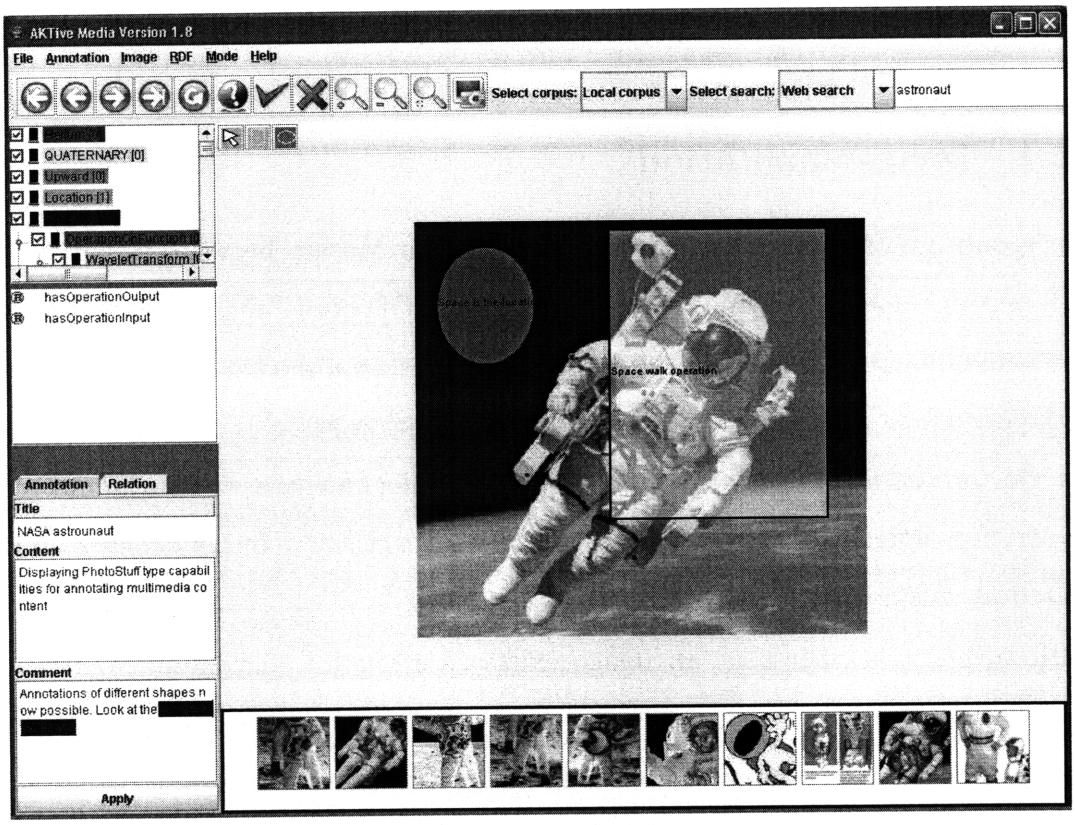

Figure 2-1: Image annotation in AKTive Media. ©2005 Ajay Chakravarthy.

or creation time, can be treated as an instance of typed annotations. On the other hand, the free text mode may be used to store comments about the document, as basic text annotations have previously been used.

The authors of AKTive Media submit, however, that free text annotations may also be used to implement folksonomies. As will be explored in the following section, folksonomies are closely related to the use of non-hierarchical keywords, called "tags," a practice which is largely exclusive of longer, full-sentence comments.

2.2.3 Tagging and folksonomy

The term *folksonomy* was likely coined in 2004 by information architect Thomas Vander Wal as a reference to the social phenomenon that occurs when many people collaboratively use tags to classify documents [18]. Generally, folksonomies exist as a large collection of taggable resources, equipped with navigation mechanisms that display, for example, the most commonly used tags. The social bookmarking web-

sites Delicious[2] (formerly "del.icio.us") and Furl[3] are salient examples of folksonomic systems where users submit and tag content, in this case bookmarked URLs.

Flickr

Tagging also plays a central role in Flickr[4], a photograph sharing service and repository. In addition, photographs on Flickr can be annotated with author descriptions, commented upon by other users (at both the whole-image and mouse-selected sub-image granularities), and organized into groups of photographs called "sets." As photographs can belong to more than one set, in reality set membership is a form of categorical metadata which is a more specialized instance of tagging, with the cursory distinction that in practice sets tend to be organized by theme rather than keyword.

There are a number of drawbacks to a classification system that relies purely on tagging, generally stemming from the lack of a controlled vocabulary [19]. As users are free to submit whichever tags they choose, they may use different tags to mean the same thing, or the same tags to mean different things. Synonyms such as "sea" and "ocean" or plurals such as "tree" and "trees" are often interchangeable, and a naive attempt to include all foreseeable variations leads to over-tagging. Conversely, tags that denote different things, either through vagueness or having multiple meanings, can incorrectly conflate disparate ideas.

While there is debate over whether folksonomy, even with a controlled vocabulary, is superior to hierarchical taxonomy in many situations [20, 21], it is clear that the former offers valuable flexibility and ease of adaptability in areas where a rigid hierarchy is insufficient. Furthermore, the low barrier to entry for individuals untrained in classification and the social aspect inherent in sharing documents and tags gives many users the opportunity and incentive to contribute. As a result, folksonomies that include some sort of social networking aspect, such as Flickr, are able to leverage the significant power of a large group of people.

[2]Website URL: http://delicious.com
[3]Website URL: http://furl.net
[4]Website URL: http://flickr.com

2.2.4 Folksonomy and "synnotation"

The term "Synnotations" was introduced in 2008 by Wald et al. [22] at the University of Southampton to describe folksonomic annotations that are synchronized to time-based media, that is, annotations that are given a temporal position within an audio or video stream. Much as sub-image annotations in AKTive Media or Flickr provide a significant degree of control over location in an image's spatial continuum, synchronized annotations allow for the same precision of location in a time-based continuum. A couple applications of synnotations include providing a table of contents accompanying a long recording or subtitles for individuals with a hearing impairment.

At the time of writing, an implementation to be called Synote is under development, with the goal of combining existing speech recognition software, which outputs synchronized text captions, with the ability for users to add annotations by inputting their own tags and comments. Thus Synote is essentially an audio annotation tool equipped with speech recognition software playing the role of a first-pass machine annotator.

In this chapter, a number of historical and landmark shared web annotation systems, primarily dealing with text annotations, were discussed. An emphasis was placed on features that would be useful and drawbacks to be addressed when building a web-based audio annotation system.

A few non-web-based systems geared toward non-text annotations were also explored, as well as the general concept of folksonomy, a relatively new mode of annotation that relies on the power of tagging and social collaboration. With this background material it is now possible to formulate the criteria for the design of a web-based audio annotation tool.

Chapter 3

Design requirements

In this chapter the design requirements for a web-based audio annotation tool are described. A few basic conditions on the usability of the interface are stated first, followed by the actual features of annotation tool and the implications thereof on the organization of data.

3.1 Interface accessibility

3.1.1 Browser deployment

In order to minimize barriers to entry and encourage user involvement, the annotation application should be deployed within a web browser. This means that the possibly lengthy process of downloading and installing a desktop application is avoided. Consequently, deployment on multiple machines does not necessitate multiple installations; as a web service, the annotation tool can be used from any computer.

3.1.2 Browser and platform independence

The annotation interface should be accessible to any web browser, so that it is available to as wide a population of users as possible. Likewise, dependence on custom browser modifications, as with ComMentor and Annotea, or non-standard browser extensions is also to be avoided.

Thus, only an internet connection should be strictly necessary for access to the annotation system. This appears to be reasonable given the ubiquity of internet access in industrialized countries and, arguably, the United Nations' current push toward universal internet access [24].

As an added benefit, this design requirement implies that the annotation tool will be accessible to future web browsers as well (as long as the basic client-side web application technologies such as HTML and CSS remain, of course) – contrast this with the Mosaic group annotations system, which necessarily ended when NCSA ceased support for the Mosaic browser.

3.1.3 Real-time collaboration

In addition to being accessible to users of a wide range of browsers and platforms, the annotation system should be usable by groups of individuals, who wish to submit and annotate shared audio resources simultaneously. This requirement allows multiple users to collaborate in real-time.

Collaboration should be free to occur either between annotators adding responses to each other, or even between annotators and submitters of audio content. In the latter case, imagine an interview being conducted and uploaded to the annotation server in real-time, with listeners immediately commenting and possibly steering the course of the conversation.

3.2 Annotation features

The general aim of the following requirements is to make annotations expressive and semantic-rich while preserving the flexibility of allowing users to input informal free text.

3.2.1 Synchronized annotations

It should be possible for users to attach an annotation to a particular temporal position in an audio document. As described in section 2.2.4, synchronized annotations allow users to annotate time-based media with a high level of precision. In particular, the system should support annotations at a single point in time, over a duration of time, or on the document as a whole.

3.2.2 Tagging

The annotation system should support categorization by tagging. As discussed extensively in section 2.2.3, tagging can be used to implement folksonomies, a form of social collaboration that categorizes data non-hierarchically. The possibility of tagging is already inherent in any text annotation system, as tags are just short annotations. The system should also be able to distinguish tags from other annotations, and provide aggregation mechanisms such as viewing the most popular tags or the frequency of two related tags being used together.

3.2.3 Structured metadata

Annotations have generally been used in previous web annotation systems to add remarks to an existing document (refer to section 2.1). Even the design of the Annotea system, which supports typed annotations, treats them as an instance of metadata. Thus the type of an annotation may classify it as commentary or errata, but typically not a machine-parsable statement about the author or creation time of a document, for example.

In this system, annotations should be flexible enough to allow the inputting of both structured metadata and informal text; that is, structured metadata is merely a type of annotation. The system should be able to parse the metadata where relevant and ignore the rest, much as a compiler would ignore documentary comments in a piece of code.

A consequence of this design is that the annotation system may not be able to

perform as many validation checks on the user-submitted metadata. For instance, a document may semantically have multiple authors but not multiple creation times. Because the metadata for creation time is now a generic annotation, however, both cases would be acceptable. The design places this responsibility in the hands of the user or users annotating the document.

3.2.4 Extensible metadata types

The Annotea system requires that annotations be typed, and allows users to define their own types [14]. The restriction on this is that the semantics of all annotation types must be described by an accompanying RDF schema, much as an XML schema is used to describe the allowable structure of an XML document.

Unfortunately, Annotea's requirement places a significant burden upon the user who is defining a custom annotation type, especially if the user happens not to be familiar with RDF schemata. This design therefore eschews the schema requirement in favor of a more flexible framework for specifying metadata types.

As with Annotea, metadata types should be extensible, but the system should not require the user to provide any description of custom types. Instead, the semantics of new metadata types is interpreted by plug-ins to the web application, which try to parse the metadata and may modify the behavior of the user interface accordingly. If no plug-in recognizes the metadata type, the system falls back silently and ignores the annotation. The following example will illustrate this design:

Suppose that a user adds a custom metadata type to describe the musical genre of an audio document if applicable, and annotates a number of music files using this method. Without any plug-ins that recognize this metadata type, system treats these annotations the same way it treats informal text annotations containing the string "Classical" – by displaying the text. One plug-in might filter the annotations by displaying only files marked as Baroque or Classical, while another plug-in might at the same time suggest possible composer names.

Thus, this system is somewhat analogous to the concept of dynamic typing in programming languages. It is not necessary to define explicitly how a type is im-

plemented before instantiating data of that type. Rather, it is up to the system to determine at run-time whether the data is usable or not. Similarly, it is the user's responsibility to make sure that this flexibility does not result in metadata being misinterpreted. The hypothetical genre annotator may need to be careful with a plug-in for labeling readings of Classical literature.

3.3 Data organization

In addition to the use of tags for categorization purposes, machine-interpretable metadata presents many possibilities for indexing audio files for retrieval or organizing them for browsing and exploring. A few of these properties, to be described in the following sections, are essential and universal enough to be included in the annotation system as built-in features.

3.3.1 Ownership and permissions

The annotation system should support a framework for users to register for accounts and to submit audio. All such audio will be owned by the user who submitted it, and will by default not be visible to other users browsing the system. However, it should be possible for the owner of an audio file to share it via structured metadata with other users or groups of users – including the universal group of all users. Thus, annotations with this metadata type serve as a basic system of access control lists.

3.3.2 Date and Location

Nearly all audio documents have a meaningful date[1] of creation. A significant portion of them (certainly those which are real-life recordings) also have a meaningful location of creation. Given this, the annotation system should support structured metadata types which describes the audio's date and location of origin.

[1]Note: here the term "date" may be arbitrarily precise here; it is not limited to the granularity of calendar dates.

Date and location may well be so meaningful because the properties of time and space are so universal and naturally understood. Calendars and maps, of course, in some form have been ubiquitous representations of time and space, respectively, long before the popularization of computers. The usefulness of date and location as both an indices for retrieval and continuums for browsing should not be underestimated. Thus, in addition to having metadata types that represent these dimensions, the annotation tool should also provide interfaces to navigate through time and space.

The design requirements for a web-based annotation system were detailed in this chapter, with an emphasis on maximizing the browser accessibility of the web interface as well as the expressiveness and flexibility of the annotations tool itself. The description of the actual design and implementation of the system architecture follows.

Chapter 4

Implementation

Krik Krak[1] is an implementation of a web-based audio annotation tool which was developed to satisfy the design requirements detailed in chapter 3. The goal was to provide a web application with a rich interface through which communities of individuals and groups could share and annotate audio documents.

The following section provides an overview of the system and briefly outlines how Krik Krak fulfills the previously given design requirements. The remaining sections of this chapter describe each of the components of the system in detail.

4.1 Overview

Krik Krak is a rich web application developed in the "Web 2.0" paradigm. That is, it was designed to facilitate social participation, and implemented with web technologies such as AJAX to provide a rich user interface [26, 27].

The client-side interface consists of an HTML document that uses a combination of cascading stylesheets (CSS) and JavaScript to interact with the layout of the webpage in response to the user's actions. The JavaScript code uses the document object model (DOM) to affect the appearance of the page dynamically. These web technologies are

[1]The name "Krik Krak" is borrowed from the Haitian storyteller's custom of starting a story by asking the audience "Krik?" The audience responds with "Krak!" to acknowledge the beginning of the story [25].

specified by web standards[2], and while not all browsers implement them consistently, they are widely-supported on the web.

In addition, the JavaScript-based XMLHttpRequest object, an embedded Java applet, and an embedded Flash object provide methods by which the application can communicate with the web server. The ability to send and receive data allows the interface to update itself when new audio documents or annotations are submitted by other users, thereby facilitating real-time collaboration.

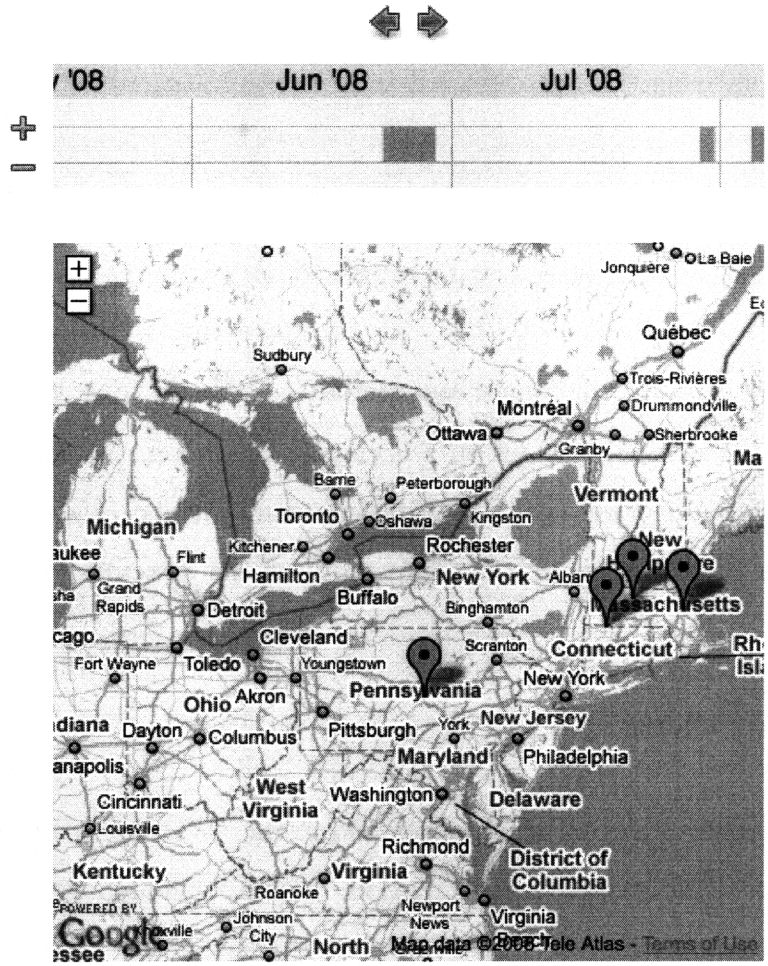

Figure 4-1: Screenshot of the Google Maps object and timeline.

The user interface itself consists of a Google Maps[3] object and a timeline, with which the user can filter and browse audio documents by location and time, respec-

[2]The World Wide Web Consortium (W3C) at http://w3.org provides specifications for HTML, CSS, and DOM.
JavaScript is a dialect of ECMAScript, which is specified by ECMA [28].

[3]Website URL: http://maps.google.com

tively. The timeline, like the map, can be scrolled and zoomed to change the visible range of time. Audio documents in the current location and time view are marked on both the map and timeline, and displayed in a list. The user can select a particular document to retrieve its annotations, play the audio from any point in time, and add annotations to the document as a whole or synchronized to a specific part.

Annotations are presented to the user in the form of text fields. The text representation of structured metadata contains a prefix identifying the metadata type surrounded by colons (:). On the other hand, annotations without a prefix are either tags or comments. An annotation that contains a space is treated as a comment. Otherwise, it is treated as a tag, and must use an underscore (_) instead of a space if there are multiple words. This system of encoding annotations as either metadata, tags, or comments allows everything to be displayed and edited as plain text, and differentiates between the three formats using simple and intuitive criteria.

The implentation details will be discussed in further depth in the following sections.

4.2 Browsing interface

When a user browses the audio repository, the list of documents visible to and selectable by the user for inspection is determined by the current view. Conceptually, a view is essentially a variable number of filters, or boolean-valued functions, which determine whether a particular audio document is visible and filter out those that aren't. If the document satisfies the conditions of all active filters, it is marked on both the map and timeline, and present on the list of visible documents.

There is always at least one implicit filter in effect when browsing: the current user's permission to access a document. This filter prevents the user from seeing any document owned by another user which the latter has not shared, or which does not contain the browsing user on its access control list (see section 3.3.1 for specifics about using metadata to define access control lists).

4.2.1 Filtering by space and time

In addition, the map and timeline controls impose view filters on the user's browsing by hiding all audio documents that are not within the window of visibility. Specifically, the map imposes maximum and minimum values on the latitude and longitude of a document's geolocation. These four filters also allow documents with no geolocation to be shown; the map provides a checkbox with which the user can include a filter that rejects un-located documents.

Likewise, the timeline imposes two view filters that determine whether any part of an audio document's duration lies between the minimum and maximum values of the timeline. The filter rejects audio that exists completely before the timeline's visible range (i.e. the document's ending time precedes the minimum time) or completely after the range (i.e. the maximum time precedes the document's starting time).

Thus, the view filtering performed by the map and timeline, combined with their controls that allow the user to scroll around or zoom in and out, provide an intuitive interface by which the user can narrow down the list of visible documents or explore documents according to temporal or spatial proximity.

4.2.2 Filtering by tagging

While the map and timeline windows provide a loosely hierarchical way for the user to retrieve audio documents, tagging allows users to filter the view non-hierarchically. The interface features a form with which the user to define a list of tag filters, where each tag corresponds to a filter that accepts only documents with that tag. The list of tags defines a view where the visible documents contain all the tags on the list. The more tags the user adds to the list, the more specific the view.

As a convenience, every instance where a tag is displayed in the interface – in the list of annotations attached to a document, for example – is mouse-clickable and responds by toggling that tag's presence in the tag filter; it is added if not already present, and removed if otherwise. This allows the user to modify the browsing parameters with ease.

So far this section has described a number of built-in filters that narrow down the set of visible documents based on user ownership, spatial and temporal position, and tagging. Every time the user changes the view by modifying the list of active filters, an XMLHttpRequest is made to the server to retrieve the new visible set. Note, however, that the number of filters is not limited to the few provided here; plug-ins which parse and interpret structured metadata, to be discussed in more detail later, can also add their own filters with more complex acceptance tests.

4.3 Content contribution

The Krik Krak application is designed so that the user can easily contribute content to the repository. A number of interfaces allow the user to submit audio documents to the system, while annotations can be directly added or deleted via the AJAX webpage.

4.3.1 Contributing audio

There are three different interfaces by which audio can be submitted to the repository of audio documents. Uploading an audio file is the first method and the one that is most straight-forward to use. Above the list of visible audio documents is a file upload form, similar to that by which webmail clients upload attachments.

Contributing audio via network access

After the user selects a file from the client machine, Krik Krak calls the JavaScript and Flash library SWFUpload[4] to make an HTTP POST request to the server machine. Flash is used here for the reason that the asynchronous XMLHttpRequest object is not allowed to attach files; an alternate method of uploading is necessary in order to avoid reloading the client page. Once the file has been uploaded to the server, the

[4]Available at http://swfupload.org under the MIT License.

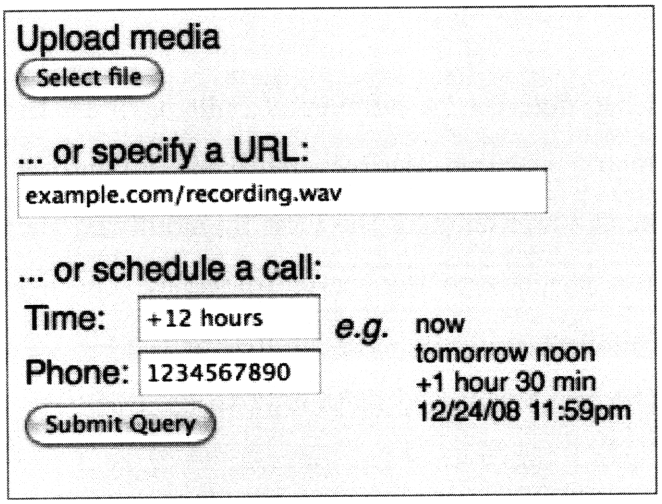

Figure 4-2: Interface for contributing audio via HTTP upload, server download, or telephone.

user only needs to browse for the audio document with the newest creation date and no geolocation.

A similar route for submitting audio to the repository involves downloading the file from another web server instead of uploading it via the client's browser. In this method, a text field next to the upload form is used to tell the server a third-party URL pointing to the file in question. The file transfer is then performed by a server-side script that downloads it directly from the third-party web server. While the result is nearly the same, the method is better suited to larger files as the restrictions on the HTTP POST request, such as file size, no longer apply.

In both of these cases, the submitted audio document is automatically annotated with a creation date equal to the time of upload, and a null geolocation to indicate that it has not been placed on the map.

Contributing audio via telephone

The third way by which audio can be added to the repository is more involved and geared toward streaming live recordings to the server. In order for this feature to be accessible to a large population of users, the audio is streamed to the server via telephone. Rather than requiring an audio streaming server, to which users typically do

not have access, the phone abstraction allows the use of many technologies, including landline and mobile cellular phones, and computers that can make VoIP calls.

To stream audio to the server via a phone, either the phone must call the server to initiate a transfer, or vice versa. For the latter case, the Krik Krak interface contains a form for scheduling phone calls; the user submits a phone number for the server to call and a time at which to call. As the audio is transmitted, an audio document is added to the repository at the start of the call, and lengthened in real-time (during which it can be played back or annotated) until the call terminates.

While the phone interface has been designed with the goals of convenience and ease of use, the server architecture underlying the streaming functionality is somewhat more complex, and will be discussed later, in section 4.4.

Scheduled phone calls

When the user schedules the phone server to call a particular phone number, the call does not necessary occur immediately. Instead, the user has the option to specify a later date and time at which the call will be made. Functionally, the phone server merely behaves as if the web interface had waited for the appropriate duration of time before submitting, during which the user may have left the computer.

The ability to schedule phone calls ahead of time may be useful for a number of purposes. A user who carries a personal cell phone will not find much benefit in a server that can only schedule immediate calls; the cell phone is essentially tethered to the computer from which the call is scheduled, thus defeating the purpose of a mobile device. Even if this were not the case, scheduling non-immediate calls may be convenient if the call time were late at night, for example. This is often relevant to the area of environmental monitoring, which will be discussed in section 5.1.

4.3.2 Adding annotations

Once audio documents have been entered into the repository, users can browse them via the web application interface and add annotations. This section describes the

generic text input method for the three kinds of annotations, how they differ in format, and some applications of structured metadata.

When the user selects an audio document from the list of visible documents, the annotations associated with it are displayed alongside it, each with a button to delete it. A text input field allows the user to add new annotations associated with the entire document. For synchronized annotation, the user clicks a temporal position on the timeline to create an instantaneous annotation, or clicks and drags to select a duration. Upon releasing the mouse, a similar input field appears for the user to type in and submit text. In all three cases, the JavaScript XMLHttpRequest object is used to inform the server asynchronously so that the user can continue to use the web interface.

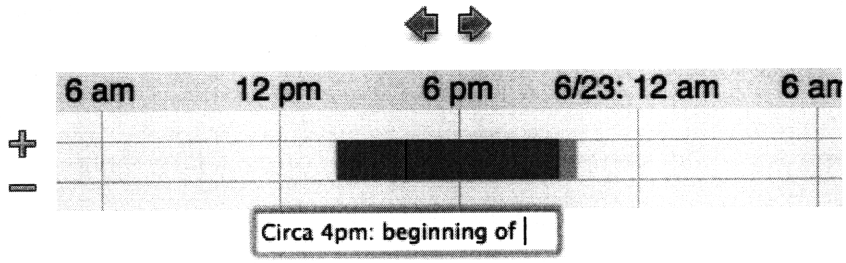

Figure 4-3: Example of adding an annotation synchronized to a point in time.

4.3.3 Child annotations

Additionally, annotations can be attached not only to audio documents, but to other annotations as well. When a user creates an annotation, a parent annotation may be specified, making the new annotation a child of the parent. If no parent is specified, the new annotation is merely a child of the audio document. Once an annotation has been created, it has no child annotations by default, until the user in turn attaches additional child annotations to it.

This mechanism essentially allows the user to create annotations of annotations. So far, it has been possible to annotate audio documents as a whole or in part. Child annotations allow the user to do the same for existing annotations. One use of child annotations is to facilitate adding supplementary information about but not directly

related to an annotation – for example, a statement of whether a particular annotation was created by a human or a machine process, or about its confidence level.

The design of child annotations also allows the user to create a group of annotations in a hierarchical tree structure; the terminology of "parent" and "child" annotations was chosen to reflect this structure. A commentator on a speech may choose to denote large sections based on topic using parent-less annotations, and attach more specific remarks to each section using child annotations. Alternately, multiple commentators may use child annotations to respond to one another, as in a discussion forum. Moreover, if the discussion does not branch, particularly if only two users participate, it can be thought of as a conversation thread, and the associated data structure is the linked list.

It is important to note that, despite being attached to another annotation, a child annotation is ultimately still about the audio document. This relation is evident when one traces a child annotation's parent (and possibly ancestor) annotations back to the audio to which it is rooted.

4.3.4 Annotation types

Every annotation has a text representation, which is the format that is used when annotations are created via a text input field on the web application. Despite sharing a text interface, annotations can be categorized into comments, tags, and structured metadata (with the latter allowing for even more specific categorization). The annotation system uses the format of an annotation's text representation to decide which type it is and how to treat it.

Comments are the most straight-forward type of annotation, as they take the form of a phrase, sentence, or paragraph that users typically write when making remarks about a document. The annotation system assumes that comments contain multiple words separated by spaces, and uses the presence of the space character to decide whether an annotation is a comment or not. To exclude metadata annotations which contain spaces, a comment also cannot begin with a colon. This requirement should not be significantly limiting, however, as sentences do not begin with colons. The web

interface displays comments in paragraph form below the document being annotated, and does not perform any additional processing.

In contrast to comments, tags are typically single-word keywords or indices used for categorization purposes rather than rhetorical description. Therefore, the annotation system interprets annotations with no spaces (and, again, not beginning with a colon) as tags. Rare tags consisting of multiple words can use underscores in place of spaces, as in `alto_sax`, in order to differentiate from comments. As described in section 4.2.2, the web interface makes tags clickable, and upon being clicked they are added or removed from the active view filter as appropriate.

Finally, metadata allows users to create annotations in structured formats that allow for plug-ins to parse and interpret them. The text representation of such an annotation consists of two parts: the name of the metadata type and the data's text representation. Typically, the type name is itself a tag that categorizes what kind of metadata it describes. The type name is delimited by colons, and therefore cannot contain any colons.

The text representation is then appended to the delimited type name, forming the entire annotation's text representation. For example, an annotation with the text representation `:temp:78F` has a type name of `temp` and its data part is `78F`, which may be interpreted by a suitable metadata plug-in to mean that the temperature was 78 °F when the document in question was recorded.

While comments, tags, and metadata perform different roles in the annotation of an audio document, they are alike in that they all have text representations and share the same text field input method on the Krik Krak web interface. The primary objectives of the text format requirements that differentiate the three were intuitiveness and readability.

In the case of comments and tags, the user can write a few sentences in response to a document or add a few tags to label, and the annotation system will recognize the user's intention and treat the annotations accordingly. The user may even see the metadata annotation `:temp:78F` and decide to change it to `:temp:91F` without needing to know how a particular metadata plug-in uses that annotation. As a result

the annotation framework and the user are both able to recognize and work with the different types of annotations.

4.3.5 Built-in metadata

A few of the structured metadata types are important enough to the usage of the annotation system that the web interface provides built-in mechanisms to parse and interpret them.

As previously discussed in section 4.2.1, the browsing interface interprets the location and date metadata types and provides view filters using windows for latitude, longitude, and date. Since location and date are such essential indices for the browsing and retrieval of audio documents, the interface additionally prevents the user from deleting them or creating multiple instances of them with conflicting values.

Although newly submitted audio documents are automatically annotated with the current creation date, the system has no way of reliably determining their location. Users are encouraged to enter in the most meaningful location metadata for the benefit of organization – even the location of the client computer from which the document was uploaded can be a helpful browsing index. The user can set the document's location by dragging its Google Maps marker (an upside-down teardrop) from its initial position off the map to a point on the map. This drag-and-drop interface also allows the user to edit the location metadata later.

On the other hand, the web application offers no drag-and-drop interface for the user to change the date of an audio document. The user can instead click on the date field to edit it in place. The date field can accept a number of textual date representations as specified by the GNU date input formats[5], such as `Aug 8 2008 8:08:08pm`, `last Monday`, and `2 hours ago`.

Of course, machine interpretation of metadata is not limited to these built-in examples. The annotation system allows users to contribute their own JavaScript plug-ins, which can create view filters or provide more human-friendly ways to display and edit structured metadata. The plug-in framework will be explored in the following

[5]See http://gnu.org/software/tar/manual/html_node/Date-input-formats.html

section.

4.4 System architecture

While the previous sections in this chapter have focused primarily on the client-side web interface to a fairly simple backend, there are some less visible components of the annotation system whose complex architecture is worth describing. This section first discusses the metadata plug-in framework, which gives users the ability to extend the built-in system and provide custom interpretations of structured metadata. In addition, the main components of the server architecture will be explored.

4.4.1 Metadata plug-ins

The built-in abilities of the annotation system to parse and interpret structured metadata allow the user to perform basic filtering based on location, time, and tagging. The map and timeline provide intuitive interfaces to visualize an audio document's position in space. The plug-in framework lets the user customize how the web application responds to certain metadata types by registering a collection of JavaScript functions that are called when certain metadata annotations are displayed.

A plug-in is essentially a JavaScript handler for one or more events that are fired by the application when the user performs certain actions. Specifically, the handler is initialized when it is registered to the application; it is later notified by the application after the list of visible documents is updated, after the user selects a document to inspect its annotations, and after the user deselects a document. Additional arguments are passed to the handler where appropriate: view change events are accompanied by the list of documents, selection events are accompanied by the document and its list of annotations, and deselection events are accompanied only by the document.

Once a plug-in contributer has prepared a JavaScript file containing the handler, inserting the plug-in to the annotation interface is as simple as providing the file's location on the web via a text form. Once the webpage is reloaded, the provided JavaScript will be incorporated into the page and its handler will be called at the

appropriate time. In its current state, Krik Krak does not keep track of loaded plug-ins when the user exits the page, and so the user must repeat this procedure after starting a new browsing session. A future implementation may allow the user to specify a preference for certain plug-ins to be loaded whenever that user starts browsing.

View filtering

Although the plug-in API is quite sparse, the JavaScript handler can alter any data structure or DOM object it needs in order to modify the behavior of the web interface. Typically, the view change handler performs additional filtering on the list of visible documents. Every time the view is modified, the new view parameters are asynchronously sent to the server, which applies the built-in filters and replies with the new document list.

When the new list is in turn passed to the handler, the handler can iterate through it, parse the annotations whose type it recognizes, and reject documents by removing them from the list. Thus, the handler adds its own view filter which is implemented on the client-side. Taking the previous example of the `:temp:` metadata type, a temperature-aware plug-in might register a handler that removes documents with metadata indicating a temperature lower than 90 °F, leaving only a list of documents recorded on hot days. Of course, it is also up to the handler how to deal with documents with no recognizable metadata.

4.4.2 Plug-in applications and examples

Custom metadata interfaces

A plug-in that handles document selection events may provide custom interfaces with which the user can view and edit the metadata. When the user selects a document, the annotations associated with the document are displayed, although for metadata types other than location and time, only the text representation is used. In the handler, a plug-in may replace the text representation in the layout of the webpage with a more

intuitive interface. Furthermore, the handler may attach JavaScript event handlers to the custom interface so that it will be notified when the user interacts with it using the mouse or keyboard.

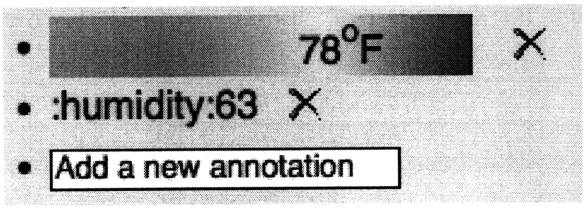

Figure 4-4: Example of a plug-in displaying temperature metadata, indicating mildly warm weather. Below is an example of humidity metadata in its text form, which the plug-in did not interpret.

Consider once again the example of a `:temp:` metadata type that signifies temperature. Instead of using the default text representation, a plug-in can replace it in the DOM with an image of a thermometer that color-codes the temperature. It can also request to be notified when the user mouse-drags the temperature marking up or down and interprets this action by redrawing the thermometer. When the user releases the mouse, the plug-in's handler is notified again and can send an XMLHttpRequest to update the server with the new the temperature value for this annotation.

Tag aggregation and visualization

One feature of folksonomies is the aggregation of the collective "wisdom" of a large number of users and tags, and using this data for visualizations or additional tools for tag manipulation. For example, a text field for adding tags can read what the user has already typed to suggest possible completions, based on the number of existing tags that start with that string. A more common tool is the *tag cloud*, a visualization of a folksonomy's popular tags and their relative frequencies. Tag clouds on Flickr and Delicious allow users to see what tags others commonly use. In these two instances, the popularity of a tag is reflected in its font size or color, so that the most popular tags are naturally displayed more prominently.

As a tool for navigation and browsing, tag clouds allow the user to see (and filter by) not only the most popular tags in a folksonomy, but also the most popular tags

within a particular subset of content. For example, within the view filtered by the tag `concerto`, the tag `piano` may be popular. This reveals that these two tags are strongly related due to the frequency with which both of them are used to tag a document.

The plug-in framework can be used to implement a client-side tag cloud. The plug-in provides a JavaScript handler that is called whenever an annotation is received from the server. Instead of replacing the text representation of the annotation with a graphical interface, however, the handler merely checks whether the annotation is a tag, and if so adds it to a counter of that tag's frequency. It also constructs a tag cloud using HTML, updating the font of each tag accordingly whenever that tag is passed to the handler.

Figure 4-5: Sample cloud generated by a plug-in, with tag frequency represented by font size.

There are a number of benefits associated with a client-side tag cloud plug-in. First, the nature of a tool that modifies the tag cloud while processing annotations means that the cloud is updated as soon as the tags change. For example, when the user adds a tag to a document, the handler is called immediately and updates the cloud immediately. More importantly, however, is the extensibility of such a plug-in. The user can easily modify the handler to count a tag two or more times depending on the document's date, thus producing a cloud biased toward tags describing more recent documents.

Using JavaScript handlers, plug-ins can create custom view filters, metadata interfaces, and aggregation tools to enhance the user's interaction with the web application. With this feature, browsing is not limited to filtering by location, time, and tagging; similarly, viewing and editing metadata annotations is not limited to reading and typing in accordance to a rigid text format. The plug-in framework allows for a much richer user experience with structured metadata and tags.

4.4.3 Server architecture

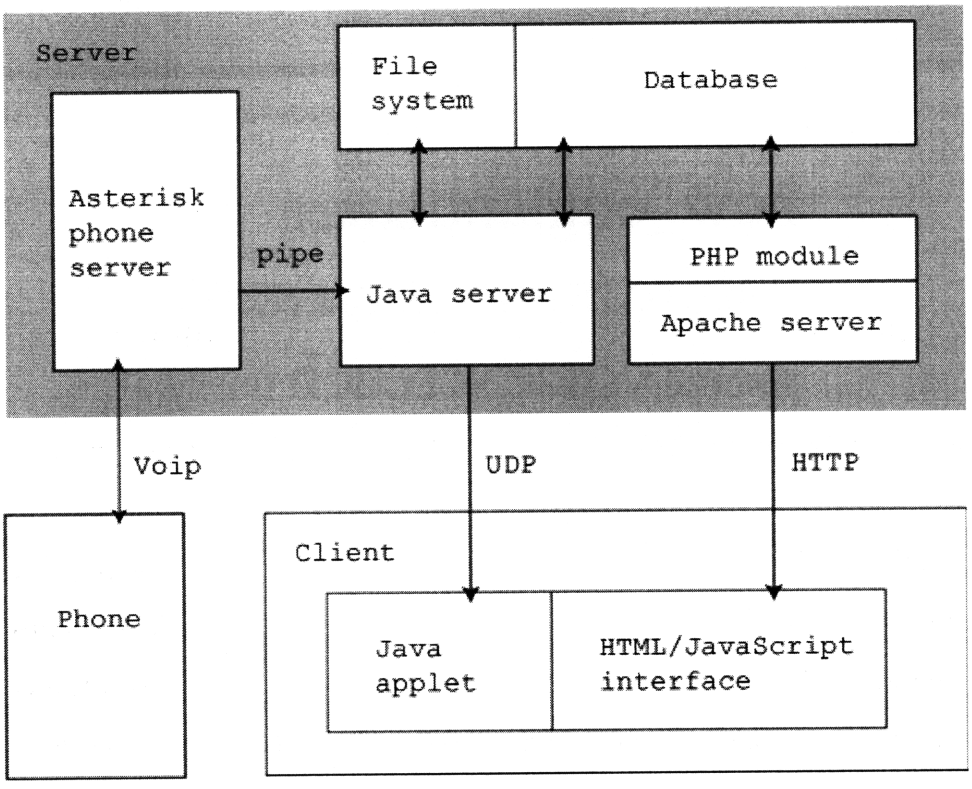

Figure 4-6: Overview of system architecture with client, server, and telephone modules.

Krik Krak is built on a client-server architecture, where the user interacts with the web application on the client machine with a web browser, which in turn frequently makes requests to the server whenever it needs to retrieve more data. The chapter has so far focused on the web application side of the system; this section turns to

look at the server side.

The annotation server runs on one host machine, and consists of three daemon processes that run in the background and respond to client requests: the Apache HTTP Server, which handles HTTP requests, the Java server, which handles connections from Java applets, and the Asterisk[6] server, which makes or takes phone calls. In conjunction with these transient services, the file system and MySQL database store all the permanent data in the annotation system, namely, the audio documents and annotations themselves.

The MySQL database uses a very simplistic and straight-forward design to store structured information pertaining to audio and annotations. Two tables are used: the audio table allocates one row per audio document and contains all relevant information not related to annotations (including the document's file system path); the annotations table conversely allocates one row per annotation and contains all information relevant to annotations.

Accessing audio and annotations

The MySQL database is most frequently accessed by the Apache HTTP Server. Apache accepts HTTP connections created by the client application's XMLHttpRequest object and uses a PHP module to retrieve lists of documents and annotations from the database based on view parameters passed in the HTTP request. The PHP module also handles new files sent by the client's Flash SWFUpload library or downloaded from third-party web servers and inserts them into the database.

The database is also accessed by the Java server, which provides the capability to stream audio to the client in real-time. When the user plays an audio document, the client-side Java applet opens a socket connection[7] to the Java server, requesting audio. In turn, the Java server consults the database for the document's file system path, reads the file into memory, and streams the audio content back to the client

[6]Website URL: http://asterisk.org

[7]In practice, the stateless, connectionless protocol UDP is used instead of TCP for the sake of streaming performance. However, important messages such as starting and stopping the stream are sent reliably, so the socket behaves, where necessary, as if it were connected.

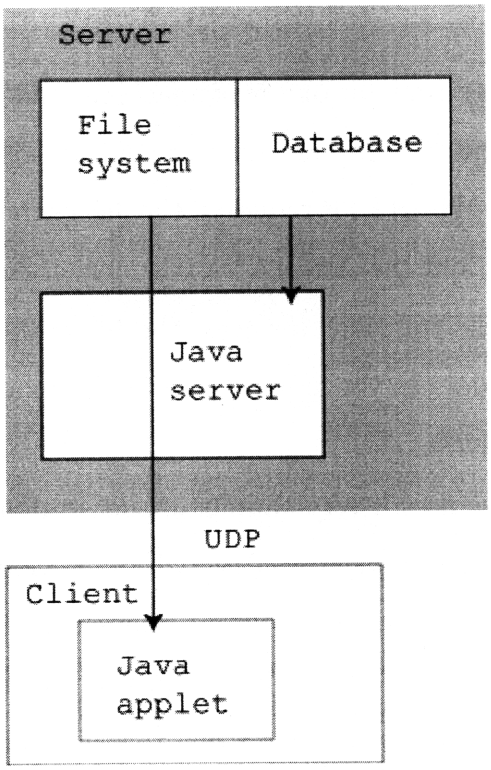

Figure 4-7: Flow of data in stream-from-file mode.

applet until the user decides to stop playback.

Live streaming

When streaming a live phone call, however, the audio document has not yet been written to the file system; it is processed by the Asterisk phone server running in a separate process and, by default, discarded. In order to capture the audio, an Asterisk script creates a named pipe on the file system, provides the Java server with its location – just as the database would, with a real file – and subsequently redirects the audio to the pipe. Thus, the Java server reads the audio into memory as if it were reading a file, and simultaneously writes it to a real file known to the database while streaming it to the Java applet.

Consider, in summary of this section, the process by which a user plays and annotates an audio document while it is being recorded live and streamed in real-time. At the start of the phone call, the Asterisk server notifies the Java server and begins piping the audio stream to Java. The Java server inserts an entry into

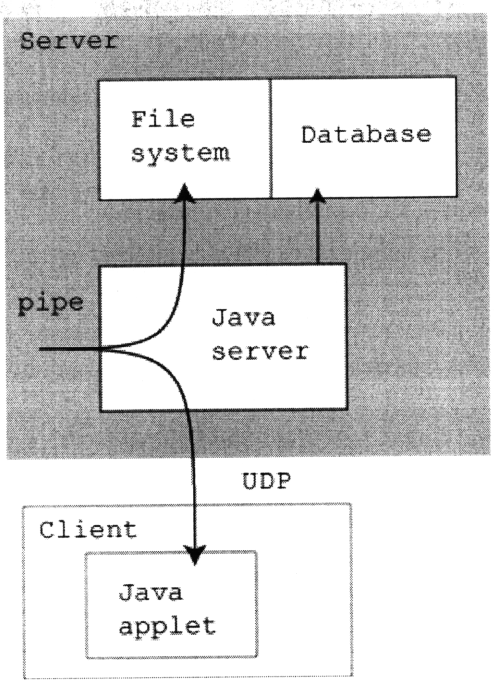

Figure 4-8: Flow of data in live streaming mode.

the database, which is in turn visible to the Apache PHP module that lists visible documents.

The web application user (who may be different from the phone caller) notices, while browsing, that a new document has been added and that it is being recorded live. With piqued interest, the user plays the ongoing recording, prompting a handshake between the user's Java applet and the Java server. The Java server replies to the applet with a copy of the incoming stream, which the user hears and annotates. The Apache PHP module accesses the database to insert the new annotation, which immediately becomes available to the user creating the ongoing recording, closing the loop.

Chapter 5

Applications

Up to now, the design requirements and implementation of Krik Krak have been discussed abstractly, with few examples of how some of the features might be used. As a result, the collective benefits of such an annotation tool may not be immediately clear. This chapter discusses some of the applications of the annotation tool and presents them as a series case studies.

5.1 The Owl Project

The Owl Project is, at the time of writing, an ongoing project at the MIT Media Lab seeking to make use of the ubiquitous technology of mobile devices for owl censuses and other environmental sensing and monitoring [29, 34]. The Owl Project's need for a convenient way to make recordings in a forest, and for a web-based interface for browsing and annotating recordings, was the motivation for this thesis.

The following sections discuss the role of volunteer scientists in the field of ornithology, and the role of Krik Krak in leveraging their involvement in the collection of natural sounds.

5.1.1 Citizen science

Recently, certain scientific communities have begun to enlist the help of large groups of non-scientists to accomplish particular tasks that may be otherwise difficult to accomplish [30]. This emerging trend has come to be called "citizen science" and may involve individuals of varying experience and training, from other scientific communities to amateur scientists and even hobbyists and recreationists.

One area where citizen science has become popular is the field of ornithology, the study of birds, which can often benefit significantly from the large and spread-out army of observant eyes afforded by a citizen scientist population. The partnership between ornithology and citizen science has a history dating back to 1900 with the first annual Christmas Bird Count, held by the Audubon Society [31]. More recently, the prominent Cornell Laboratory of Ornithology has created a lab program dedicated to citizen science[1] [32].

5.1.2 Environmental monitoring with cell phones

The Owl Project evolved out of a desire to explore the possibilities of using mobile technology, in particular cellular phones, for audio interaction with animals and environmental sensing and monitoring. The viability of using cell phones to perform and record bird vocalizations, one of the tasks popularly used in owl census-taking procedure [33], has been shown to be sufficiently comparable to that of using the conventional playback technology [34].

Typically, broadcast surveys of owl populations involve scientists and volunteer citizen scientists who play pre-recorded samples of owl vocalizations and record responses at survey points in the species' habitat. The cell phone methodology of the Owl Project is able to automate much of this process when used in conjunction with the Krik Krak annotation system's phone scheduler. Furthermore, it is possible to place multiple phones along the survey route to be called simultaneously, thus augmenting the capabilities of human-conducted experiments.

[1]Website URL: http://birds.cornell.edu/citscitoolkit

5.1.3 Annotation of owl responses

Already it can be seen that the annotation system is a tool that can streamline the process of performing broadcast surveys in a number of ways. The leveraging of ubiquitous cell phones as live recording devices coincides with the somewhat serendipitous suitability of phones as vocalization players and recorders. This frees potential census-conductors from the need to bring CD players and audio recorders to surveys, and thus significantly increases the scalability of such citizen science programs.

Another advantage of using cell phones over traditional players and recorders is that the cell network can automatically gather the recorded audio in a centralized repository, the annotation server, avoiding the task of manually assembling the collectively recorded audio off of many separate recording devices. The audio documents' presence on the server, in turn, opens up many opportunities for annotation by both human and machine.

By the time an audio recording has been submitted to the server, it is already annotated with the correct date and time metadata. A cell phone with access to a GPS service and the internet (no longer a lofty requisite!) can also immediately add the correct geolocation metadata. The collaborative nature of the annotation system lets users on the web observe and annotate the experiment in progress, and even provide live feedback to the scientist. After the experiment finishes, scientists and untrained web users alike can browse through the recordings.

With an Owl Project plug-in, the user can decide and annotate the species of an owl based on its recorded vocalization. Owl Project audio documents can also be annotated with various metadata pertaining to environmental conditions, such as temperature or humidity, and recording state, such as the directionality of a vocalization. Once these metadata are added to the system, the plug-in can replace the text representations with more intuitive visualizations, such as a compass with directionality that is updated as the corresponding recording is played back.

A drawback of a number of citizen science programs, including The Birdhouse Network at Cornell [35], is that volunteers' participation, from enrollment to research to debriefing, is often done with minimal interaction with other volunteers or even with the organizing scientists. Making an annotation website the focal point of research-doing facilitates both types of interaction and opens the project to an even wider population of curious web surfers.

5.2 Oral histories and storytelling

While the Owl Project calls and records owl vocalizations to document the owl population and environment, another practice in a disparate multi-disciplinary field performs a similar task. Oral histories record the experiences of people via interview to document society and culture, and are used by researchers of various disciplines including historians, anthropologists, and linguists.

Although the modern tradition of gathering oral histories dates back to long before the popularization of the internet, the internet has given rise to a new wave of digital oral histories, salient examples of which include StoryCorps [36], the UN Intellectual History Project [37], and Our Stories[2], a web-based collection founded by UNICEF, One Laptop per Child (OLPC), and Google.

Regardless of its use in diverse settings and disciplines, the practice of collecting oral histories is fairly consistent across disciplines: the collector interviews the subject, records the conversation, and may later convert the conversation into a text transcript [38]. The recording, along with the text transcript if available, is typically then added to a catalogue of oral histories for future reference.

[2]Website URL: http://ourstories.org

5.2.1 Transcribing and annotating oral histories

However, it appears that oral histories are, in general, sparsely annotated, and as a result they can only be browsed using very basic indices or parameters such as the interviewee's name or location. The online interface to Our Stories provides little information alongside its interview videos. The Books with Voices interface, developed at the University of California, Berkeley [39], recognizes these shortcomings and attempts to address the usability issues both of recordings and of transcripts by augmenting transcripts with random access to the corresponding video, essentially synchronizing the transcript with the recording.

A richer browsing interface for recording and transcript alike is nevertheless left to be desired. Consider, as follows, the transcription process suggested by excerpts quoted from the interview production guides of Telling Their Stories[3], an oral histories project presented by the Urban School of San Francisco focusing on Bay-area Holocaust Survivors and other individuals who lived during World War II era.

- Use "Question:" to tag all interviewer questions and comments.
- Put in parentheses (...) all unintelligible names and terms, unknown spelling, etc. This [is] our signal to make sure to revisit to clarify.
- Record in parentheses (...) the TIME in "(minutes:seconds)" for every well-spaced question – this need not be exact – it just serves as a useful reference.
- *Italicize* obvious non-English words (e.g. "Shabbat"). These will often also be put in parentheses (...) if you do not know the correct spelling.

The similarities between the task of creating plain-text transcripts of recordings and the process of annotating audio documents with structured metadata and tags are substantial. Questions and comments by the interviewer, as well as parenthesized names and terms, can easily be given tags. Timing information is already inherent in a system supporting synchronized annotations; finally, non-English words can, again,

[3]Website URL: http://tellingstories.org/about

be tagged as appropriate, or even be given a metadata annotation such as `:lang:heb` to signify Hebrew words.

The benefits of an annotation system for oral histories are not limited to an easier and more streamlined transcription process. Moving the tags and metadata from stylistic conventions for text documents to an annotation system means that the transcript is decoupled from the annotations. Consequently, this allows for future users to retrieve all occurrences of Hebrew words, as an example, or to read just the transcript, absent of editorial notes and shorthand added later.

It is worth repeating that the web-based aspect invites a much larger community of web users to view the oral histories. The task of transcribing an interview is also easily open to multiple users. And, as suggested in section 3.1.3, the capability to stream an interview opens up the possibility of onlooking web users commenting on and influencing the course of the interview, adding a new dimension to the concept of conversation in oral histories.

5.2.2 Oral histories in the context of natural stories

A generalized way of viewing oral histories may consider them to be subset of natural stories, which in turn include the vocalizations studied by the Owl Project. After all, as humans are classified as a specie, a number of overlaps between the analyses of oral and natural stories can be seen: speech recognition can parse a recording, and the field of linguistics seeks to analyze its grammar. Conversely, a recording of an owl's vocalizations can be considered as a story, which contains information about the environment that bird specialists must interpret. As discussed in the following section, a musical performance can be subjected to the same analysis and thus treated as another instance of natural stories.

5.3 Music performance and analysis

5.3.1 Live performance

Of course, musical performances are another prominent circumstance suggesting the use of unobtrusive recording devices which are capable of streaming real-time audio. A concert organizer can decide to broadcast a live stream of a performance, possibly for increased exposure. Alternately, a concert attender can just decide to share the performance with a group of friends, if permitted. Either way, web users elsewhere, who for some reason may not have been able to attend, can experience the performance vicariously and comment on it – although in this case the real-time comments are not likely to influence the performer.

A musician who does not require a live stream can also, of course, just record the concert with a microphone and upload the digital audio later. This approach does allow for better sound quality; after all, cell phone audio quality, while sufficient for carrying believable signals of owl vocalizations, is not likely to replace professional microphones for capturing high-fidelity sound. In the end, the musician still has an easy method of listening to and annotating the live performance, which can often be artistically enlightening feedback loop.

5.3.2 Musical analysis

The real benefit of an annotation system for music, however, is its applicability for students and researchers of Western classical music history and theory. In contrast to text documents, which be simultaneously read and annotated by highlighting on paper, musical documents suffer a disconnect between reading scores on paper and hearing a recording; additionally, scores do not express important temporal relationships precisely.

On the annotation timeline, spatial distance between two points corresponds proportionally to temporal distance. Furthermore, reading and hearing are linked; if the parts of a sonata movement are annotated, it is trivial to start listening from, for

instance, the recapitulation's second theme, without having to seek to it in an opaque CD track. Musical annotations are not limited to bookmark-like labels, either. With an appropriate plug-in, structured metadata can allow a student to track complex harmonic progressions or subtle thematic transformations.

Given the timeline interface, scholars may appreciate the visual evidence of composers' use of Fibonacci numbers and the golden ratio [40]; whether or not these claims are valid, it is difficult to ignore repeated proportions and geometric series when they are presented visually. Analysis of the fugue, a highly complex type of piece involving multiple voices, can also be elucidated with annotations: once the occurrences of a fugue's theme are labeled on the timeline, the relationships between the voices and the order in which they occur are immediately clear. This may aid, for example, in completion of Bach's unfinished Art of Fugue [41, 42].

This chapter will conclude by describing a plug-in designed to be a study tool for music history students, who are often asked to identify the composer and name of a piece from its historically-significant stylistic features. After a set of musical pieces have been annotated, the plug-in enters a practice-test mode by temporarily replacing all metadata annotations of type `:composer:` or `:name:` with question marks.

When a document is selected, the plug-in handler automatically starts playing it from a random point within the piece and creates two blank fields in which the student is to supply the composer and name. As the piece plays, the plug-in shows annotations of type `:style:` as hints until the student is able to answer correctly. At this time, the plug-in moves ahead to the next piece on the practice test. Of course, the ability to filter the pool of testable pieces down to the student's two known weak points, `:period:Baroque` and `:form:concerto`, is merely further testament to the system's flexibility and applicability.

Chapter 6

Conclusion

6.1 Summary

This thesis began by examining the current state of the World Wide Web with regard to the ability of people and machines to make sense of audio and video documents on the web. As a result, it proposed to design and implement a web-based audio annotation system, the basis for this thesis.

In chapter 2, a survey of existing web annotation systems was given, starting from 1993 with the group annotation server of Mosaic, the first popular web browser. A number of similar annotation systems were discussed, noting particular advances such as the ability to select excerpts in a document or to filter the view of annotations. In addition, a number of image annotation systems were discussed, including AKTive Media, which introduced strongly-typed metadata, and Flickr, which popularized tagging.

As a response to the advantages and disadvantages of previous annotations systems, the design requirements for a web-based audio annotation tool were described in chapter 3. The basic deployment restriction that the annotation tool be platform-independent ensures the application's accessibility. A number of annotation features, namely synchronicity, tagging, and extensible metadata, are intended to maximize the system's expressiveness and flexibility. Finally, a number of built-in metadata types and interpretations were suggested to improve the basic usability of the system.

Chapter 4 described the implementation of an annotation system, Krik Krak, that satisfies the aforementioned design requirements. The browsing interface, including a map, timeline, and tag filter which define the annotation view, comprises an important component of the system. The three ways of submitting audio (by HTTP upload, server-side download, or phone streaming) and the three kinds of annotation (comments, tags, and structured metadata) were discussed, with a focus on how the common text representation of annotations allows for flexible, polymorphic treatment of them. Lastly, the architectures of client-side plug-ins and of server-side daemons were discussed.

Finally, a number of possible applications for the annotation system were given in chapter 5, with an emphasis on the framework's flexibility and adaptability to other fields. The Owl Project is one instance that makes extensive use of the ability to schedule phone calls and stream audio live and in real-time to the server for the purpose of environmental sensing and monitoring. The collection, transcription, and annotation of oral histories was given as another field with needs closely matched by the capabilities of the annotation system. The chapter also described a hypothetical tool for the storage and analysis of musical pieces and cumulated in a plug-in extension that could administer customizable practice tests to music history students.

6.2 Future work

Although the annotation framework was designed with usability and flexibility in mind, there are a number of issues or shortcomings suggesting directions for further work. These will be discussed in the following sections.

6.2.1 Scalability

For the sake of accessibility from any computer connected to the internet, the architecture of the annotation system places the burden of storing and serving all data on a single server machine. Clearly, as more users access the website and add more audio and annotations, performance will degrade for all users in terms of server re-

sponsiveness and ability to handle many simultaneous requests.

One solution is to modify the design so that multiple server machines can serve the same data in order to distribute the load. The issue then becomes one of consistency: how does one user see another's newly created annotation if they connect to different servers? Since database reads are far more common than writes (due to the fact that browsing constantly generates reads, but only annotation creation, modification, or deletion generate writes), it may be possible to implement a system in which servers usually read from data caches which are periodically invalidated and resynchronized.

6.2.2 Extending live streaming

Even though the ability to stream live audio to the server with ubiquitous cell phone technology while away from a computer has proved to be extremely useful and adaptable, there are cases in which the audio contributor wants to stream higher-quality audio. Currently, there is no interface by which a digital microphone can access the phone server.

To address this issue, the Java server needs to be extended to accept incoming audio streams from a network socket, similar to the Asterisk phone server. As the audio is being sent directly to the Java server, there is no need for a file system pipe; the Java server merely needs to insert it into the database and stream it to interested clients. Note that this revised architecture also allows users to submit audio using yet another method: their computer's built-in microphone. Only the Java applet needs to be modified in order to capture the audio from the microphone and send it to the server.

6.2.3 Video annotations

Perhaps the feature that is most conspicuously missing from Krik Krak is the support for video documents and annotations, a feature that was intentionally omitted from the goals of this project. Including it would have implied a number of additional requirements with little to no overlap with the current system: video playback in

the client browser, sub-image selection for spatially positioned annotations, and the ability to upload a live video stream in real-time to the server, probably from a webcam or another video capture device.

On the other hand, a framework supporting video playback and annotation appears to be well within reach, if only because all of the aforementioned requirements are natural extensions of their counterpart in audio annotation. There are, however, a number of technological hurdles to overcome that are unique to video. For instance, the bandwidth requirements of streaming video are vastly greater than those of streaming audio, which may exclude many users from uploading video over slower internet connections. Much more work, and possibly even architecture re-designing, may be necessary before web-based video annotation can become a reality.

6.3 Contribution and perspective

In this thesis, a web-based audio annotation tool supporting synchronized annotations, folksonomic tagging, and extensible structured metadata was described. While the initial implementation of such a system was necessarily experimental and prototypical in nature and ultimately, in many parts, not yet suitable for large-scale deployment, a number of insights were gained in designing and building such a system, particularly resulting from discovering its applicability to a diverse variety of disciplines and tasks.

While the goal of the project was to design an audio annotation tool, it may be argued that two main factors contributed to its ability to adapt to multiple usage scenerios, neither of which was the niche-filling choice of audio as the medium. Firstly, the plug-in framework must be given credit for opening the web interface to user-generated functionality. Not only are plug-ins able to parse and interpret custom metadata types, but the JavaScript handler mechanism allows plug-ins to alter the entire web interface as they see fit – with no threat to an annotation server protected by a well-designed HTTP abstraction.

Secondly, it can be seen that the principle of accessibility, in its broadest sense,

invites potential communities of web users built around sharing and collaborating on annotating as well as an entire population of cell phone users. If the former is confirmation that the Web 2.0 paradigm empowers crowds to act in concurrence, the latter suggests that liberating users from their computers magnifies their versatility enormously.

I believe there is no other annotation system with a design similar to that of Krik Krak, at the time of writing. Certainly, all of the components have been explored and exploited elsewhere, but this combination and arrangement of them is unique, as suggested by the scarcity of tools that have applications specific to ornithology, oral histories, and music history.

I hope that in time, the combination suggests and lends itself to many more applications, and that it may lead to a better understanding of how we, as individuals connected by a digital web, share, collaborate, and make sense of the information surrounding us.

Bibliography

[1] Garrett, Jesse James. Ajax: A New Approach to Web Applications. Adaptive Path, February 18, 2005.
http://adaptivepath.com/ideas/essays/archives/000385.php

[2] Abbate, Jane. *Inventing the Internet.* Cambridge: MIT Press, 1999.

[3] Wilcox, L., Smith, I., & Bush, M. Wordspotting for Voice Editing and Audio Indexing. *SIGCHI Conference on Human Factors in Computing Systems*, May 1992.

[4] Van Thong, J.M., Goddeau, D., Litvinova, A., Logan, B., Moreno, P., & Swain, M. SpeechBot: a Speech Recognition based Audio Indexing System for the Web. *6th RIAO Conference*, 2000.

[5] Makhoul, J.K., Kubala, F., Leek, T., Liu, D., Nguyen, L., Schwartz, R., & Srivastava, A. Speech and Language Technologies for Audio Indexing and Retrieval. *Proceedings of the IEEE*, Volume 88, no. 8, pp. 1338-1353, August 2000.

[6] Dorai, C. & Venkatesh, S. Computational Media Aesthetics: Finding Meaning Beautiful. *IEEE Multimedia*, Volume 8, no. 4, pp. 10-12, 2001.

[7] Bush, Vannevar. As we may think. *The Atlantic Monthly*, July 1945.

[8] Reid, Robert H. *Architects of the Web: 1000 Days That Built the Future of Business.* New York: John Wiley and Sons Inc., 1997.

[9] Wolfe, Gary. The (Second Phase of the) Revolution Has Begun. *Wired Magazine*, 2:10, October 1994.

[10] Andreessen, Marc. NCSA Mosaic Technical Summary. *National Center for Supercomputing Applications*, February 20, 1993.

[11] Röscheisen, M., Mogensen, C., & Winograd, T. Shared Web Annotations as a Platform for Third-Party Value-Added Information Providers: Architecture, Protocols, and Usage Examples. Stanford University, 1994. http://dlib2.stanford.edi/diglib/pub/reports/commentor/commentor.ps

[12] Davis, J.R. & Huttenlocher, D.P. The CoNote System for Shared Annotations. Cornell University, Dept. of Computer Science, 1995. http://www.cs.cornell.edu/ dph/annotation/annotations.html

[13] Schickler, M.A., Mazer, M.S., & Brooks, C. Pan-Browser support for annotations and other meta-information on the World Wide Web. *Computer Networks and ISDN Systems*, vol. 28, May 1996 pp1063-1074.

[14] Kahan, J., Koivunen, M.R., Prud'Hommeaux, E., & Swick, R.R. Annotea: an open RDF infrastructure for shared Web annotations. *Computer Networks*, vol. 39, issue 5, August 5, 2002, pp 589-608. (c) 2002 Elsevier Science B.V.

[15] Berners-Lee, T., Hendler, J., & Lassila, O. The Semantic Web. *Scientific American*, May 17, 2001. http://www.sciam.com/article.cfm?id=the-semantic-web

[16] Heck, R.M., Luebke, S.M., & Obermark, C.H. A Survey of Web Annotation Systems. Dept. of Mathematics and Computer Science, Grinnell College, 1999. http://math.grin.edu/~rebelsky/Blazers/ Annotations/Summer1999/Papers/survey_paper.html

[17] Chakravarthy, A., Ciravegna, F., & Lanfranchi, V. Cross-media document annotation and enrichment. *1st Semantic Authoring and Annotation Workshop*, November 2006.

[18] Smith, Gene. Folksonomy: social classification. August 3, 2004. http://atomiq.org/archives/2004/08/folksonomy_social_classification.html

[19] Mathes, Adam. Folksonomies – Cooperative Classification and Communication Through Shared Metadata. Graduate School of Library and Information Science, University of Illinois Urbana-Champaign, December 2004. http://adammathes.com/academic/computer-mediated-communication/folksonomies.html

[20] Shirky, Clay. Ontology is Overrated: Categories, Links, and Tags. *Clay Shirky's Writings About the Internet*, 2005. http://www.shirky.com/writings/ontology_overrated.html

[21] Smith, Gene. Market Populism In The Folksonomies Debate. April 20, 2005. http://atomiq.org/archives/2005/04/market_populism_in_the_folksonomies_debate.html

[22] Wald, M., Wills, G., Millard, D., Gilbert, L., Khoja, S., Kajaba, J. & Butt, P. Multimedia Annotation and Community Folksonomy Building. *World Conference on Educational Multimedia, Hypermedia and Telecommunications*, 2008. pp. 2213-2220.

[23] Fields, Kenneth. Ontologies, categories, folksonomies: an organized language of sound. Organised Sound (2007), 12:101-111 Cambridge University Press. ©2007 Cambridge University Press.

[24] Best, Michael L. Can the Internet be a Human Right? *Human Rights & Human Welfare*, vol. 4, issue 1. 2004.

[25] Danticat, Edwidge. *Krik? Krak!*. New York: Soho Press, 1995.

[26] O'Reilly, Tim. What is Web 2.0: Design Patterns and Business Models for the Next Generation of Software. *Communications & Strategies*, no. 65, 1st Quarter 2007.

[27] Paulson, Linda D. Building Rich Web Applications with Ajax. *Computer*, vol. 38, no. 10, pp. 14-17, October 2005.

[28] ECMAScript Language Specification, Standard ECMA-262. ECMA Standardizing Information and Communication Systems, 3^{rd} ed., December 1999.

[29] Fizz, Robyn. Cell Phone Project Counts Owls Through Call and Response. *MIT Information Services and Technology News*, Volume 23, no. 1. September-October 2007.

[30] Trumbull, D.J., Bonney, R., Bascom, D., & Cabral, A. Thinking Scientifically during Participation in a Citizen-Science Project. *Science Education*, vol. 84, no. 2, pp. 265-275. ©2000 John Wiley & Sons, Inc.

[31] Root, Terry L. *Atlas of Wintering North American Birds: An Analysis of Christmas Bird Count Data.* Chicago: University of Chicago Press, 1988.

[32] Bhattacharjee, Yudhijit. Ornithology: Citizen Scientists Supplement Work of Cornell Researchers. *Science*, vol. 308, no. 5727, pp. 1402-1403. June 3 2005.

[33] Takats, D.L., Francis, C.M., Holroyd, G.L., Duncan, J.R., Mazur, K.M., Cannings, R.J., Harris, W., & Holt, D. Guidelines for Nocturnal Owl Monitoring in North America. Beaverhill Bird Observatory and Bird Studies Canada, Edmonton, Alberta.

[34] Joachim, D. & Goodale, E. On the use of cellular telephony for audio interaction with animals. *Biology Letters.* ©2007 The Royal Society.

[35] Brossard, D., Lewenstein, B., & Bonney, R. Scientific knowledge and attitude change: The impact of a citizen science project. *International Journal of Science Education*, vol. 27, no. 9, July 15 2005, pp. 1099-1121. ©2005 Taylor & Francis Group Ltd.

[36] Lamothe, P. & Horowitz, A. StoryCorps. *The Journal of American History*, Volume 93, no. 1, pp. 171-174. June 2006.

[37] Jolly, Richard. *The Power of UN Ideas: Lessons from the First 60 Years.* United Nations Intellectual History Project, January 2005.

[38] Grele, Ronald J. *Envelopes of Sound: The Art of Oral History.* Greenwood Publishing Group, 1991.

[39] Klemmer, S.R., Graham, J., Wolff, G.J., & Landay, J.A. Books with Voices: Paper Transcripts as a Tangible Interface to Oral Histories. *SIGCHI Conference on Human Factors in Computing Systems*, April 2003.

[40] Rothwell, James A. The Phi Factor: Mathematical Proportions in Musical Forms. Kansas City: University of Missouri, 1977.

[41] Göncz, Zoltán. The Permutation Matrix in J. S. Bach's Art of Fugue. *Studia Musicologica*, Volume 33, pp. 109-119, 1991.

[42] Hughes, Indra N.M. Accident or Design? New Theories on the Unfinished Contrapunctus 14 in J. S. Bach's The Art of Fugue BWV 1080. University of Auckland, 2006.