USER INTERFACE DESIGN IN A JUST-IN-TIME RETRIEVAL SYSTEM FOR MEETINGS

Andrei Popescu-Belis Peter Poller
Jonathan Kilgour Mike Flynn Sebastian Germesin
Alexandre Nanchen Majid Yazdani

Idiap-RR-38-2009

DECEMBER 2009

User Interface Design in a Just-in-time Retrieval System for Meetings

Andrei Popescu-Belis
Idiap Research Institute
andrei.popescu-belis@idiap.ch

Peter Poller
DFKI GmbH
peter.poller@dfki.de

Jonathan Kilgour
HCRC, Univ. of Edinburgh
jonathan@inf.ed.ac.uk

Mike Flynn
Idiap Research Institute
mike.flynn@idiap.ch

Sebastian Germesin
DFKI GmbH
sebastian.germesin@dfki.de

Alexandre Nanchen
Idiap Research Institute
alexandre.nanchen@idiap.ch

Majid Yazdani
Idiap Research Institute
majid.yazdani@idiap.ch

ABSTRACT

The Automatic Content Linking Device (ACLD) is a just-in-time multimedia retrieval system that monitors and supports the conversation among a small group of people within a meeting. The ACLD retrieves from a repository, at regular intervals, information that might be relevant to the group's activity, and presents it through a graphical user interface (GUI). The repository contains documents from past meetings such as slides or reports along with processed meeting recordings; in parallel, Web searches are run as well. The acceptance by users of such a system depends considerably on the GUI, along with the performance of retrieval. The trade-off between informativeness and unobtrusiveness is studied here through the design of a series of GUIs. The requirements and feedback collected while demonstrating the successive versions show that users vary considerably in their preferences for a given style of interface. After studying two extreme options, a widget vs. a wide-screen UI, we conclude that a modular UI, which can be flexibly structured and resized by users, is the most sensible design for a just-in-time multimedia retrieval system.

Author Keywords

just-in-time retrieval, user interfaces, multimodal meeting recordings, meeting assistants

INTRODUCTION

This paper explores the trade-off between informativeness and unobtrusiveness in the design of a just-in-time multimedia retrieval system. The principle of such a system, in the present case the Automatic Content Linking Device (henceforth, ACLD), is to monitor the activities of one or more users, especially their verbal output, and to retrieve from a given repository, from time to time, documents that might be relevant to them. Several repositories can be searched, including, past meeting recordings, related documents, slides, minutes, etc., as well as websites.

The ACLD is a meeting support application that provides just-in-time or query-free access to potentially relevant documents or past recorded meetings, based on speech from ongoing discussions. As participants in meetings often mention such documents, which contain facts that are currently discussed, but do not usually have the time to search for them, the ACLD aims at retrieving and presenting the documents automatically, hence the idea of *content linking* between documents and discussions.

The requirements and feedback that were collected while developing and demonstrating the successive versions of the ACLD show that the presentation of results and of the information related to them has a crucial importance for the overall acceptability of the concept, beyond the retrieval quality per se. However, user preferences for a given style of presentation interface vary considerably. In this paper, we will analyze the series of interfaces that were implemented for the ACLD, from the initial design (Section) to the dilemma between an unobtrusive "Widget UI" vs. an informative "Wide screen UI" (Section), and its proposed solution under the form of a modular resizable interface (Section). The paper starts by summarizing our view of user requirements (Section), reviews previous attempts at designing such systems (Section), and then by describing the main components and functionalities of the ACLD (Section), before proceeding to discuss the user interfaces.

SCENARIOS OF USE AND USER REQUIREMENTS

Many organizations currently possess the technological competencies to record their most important meetings, and potentially all meetings held in an equipped meeting room, and store the results in a multimedia archive. The possibility of searching efficiently through such domain-specific archives conditions then the intrinsic utility of the archives. Although there are many times at which one might wish to search the archive, a particular need for efficient search occurs during meetings themselves. Often people have a feeling during meetings that they need some piece of information, but they can't lay their hands on it, at least not during the meeting itself, because search would require more time than participants can afford to spend during the discussion. And yet, producing the right piece of information at the right time can change the course of a meeting.

The solution proposed here is to have the room not just record meetings that happen in it, but also "listen" to them and search quietly in the background for the most relevant documents and meeting segments from a multimedia repository, or from the Web, and have them ready for whenever someone in the meeting feels the need to consult them. Participants thus only need to decide if they want to explore any further, and possibly introduce in the current discussion, the meeting fragments or documents retrieved automatically for them – in the case of past meetings by using a meeting browser.

Two main distinctions can be made regarding the use of such a *content linking* system. The system can be used privately by each participant, with personalized repositories and search criteria, or the results of a joint system could be shown to the entire group on a separate projection screen. Also, the system can be used online, as a "meeting assistant" with the role just described, or it can be used offline to browse a past discussion or presentation enriched with potentially relevant documents, as a "meeting browser".

These two distinctions correspond to four combinations, and we focus in what follows on the {'online', 'private'} mode, i.e. an individual meeting assistant that provides content linking in real-time during a real meeting. For development and demonstration purposes, this can also be demonstrated with a pre-recorded meeting, and possibly other pre-recorded results of processing modules such as automatic speech recognition.

To test the validity of the above user requirements, the scenarios were described – accompanied by more or less functional graphical interfaces – to potential industrial and academic partners, about twenty of each, such as representatives of companies that are active in the field of meeting technology. A series of sessions at workshops and other events, lasting 15-30 minutes each, started with a presentation of the ACLD and continued with a discussion, during which notes were taken by one of the ACLD authors. People found that both online and offline application scenarios were of interest to them, as well as both individual and group uses. The ACLD received very positive verbal evaluation, as well as useful feedback and suggestions for future work which are presented in Section below.

JUST-IN-TIME RETRIEVAL SYSTEMS

The idea of content linking could be traced back as far as the *Memex* tool imagined by Vannevar Bush [3], which would have allowed its users to browse the contents of a library (microfilmed for convenience) by jumping from one document or sequence to another one by using topic-based and interest-based links created by the same user, or by other, professional users.

The first "content linking" applications were implemented however only after the advent of the personal computer, under the names of either *query-free search*, in the Fixit system, or *just-in-time retrieval*, in the Remembrance Agent. Fixit [8] is an assistant to an expert diagnostic system for a given line of products, which monitors the state of a user's interaction with the diagnostic system, in terms of positions in a belief network, and runs searches on a repository of maintenance manuals to provide additional support information, based on the position in the network. The results of the searches are in fact pre-computed for each node of the belief network, in order to speed up the process at run time.

The Remembrance Agent [12, 13], which is closer to the ACLD, is integrated to the Emacs text editor, and runs searches at regular time intervals (every few seconds) using a query that is based on the last words typed by the user (e.g. using a buffer of 20–500 words). Results from a repository of emails or text notes are displayed in a separate frame, and can be opened within Emacs as well. In both cases, there is no need for users to formulate explicit queries, and results are updated regularly so that users receive them, ideally, exactly when they need them.

The creators of the Remembrance Agent have also designed Jimminy, a wearable assistant that helps users to take notes and to access information when they cannot use a standard computer keyboard [11]. Jimminy uses a number of contextual capture devices, in particular a positioning device and a device for identifying the user's interlocutors using badges. However, the use of speech was not implemented in Jimminy, and the detection of the subject of conversation was only simulated – by entering this topic as real-time notes – and the focus was mostly on the note-taking function.

The Watson system [2] monitors the user's operations in a text editor, but proposes a more complex mechanism than the Remembrance Agent for selecting terms for queries, which are directed to a web search engine. Besides automatic queries, Watson also allows users to formulate queries and disambiguates them using the terms that were selected automatically. Another query-free system was designed for enriching television news with articles from the Web [9]. The system annotates TV broadcast news, using queries derived from closed captioning text, with links to potentially relevant news wire from the Web. More recently, many speech-based search engines (e.g. [5]) and multimedia information retrieval systems [10] have been proposed, and inspiration from their technology – which is not *per se* query free – can also be used for just-in-time retrieval systems.

STRUCTURE OF THE AUTOMATIC CONTENT LINKING DEVICE

While borrowing ideas from the other applications described above, the ACLD is a just-in-time retrieval application that gives access to multimodal processed data, in a real-time, autonomous fashion. The ACLD is also the first system that is fully implemented in a multimodal interaction context, giving access to indexed multimedia recordings, documents and websites, based on automatic speech recognition and keyword spotting in an ongoing conversation.

Overview

In a nutshell, the Automatic Content Linking Device performs searches at regular intervals over a database of documents and meeting recordings with a search criterion that is constructed based on the words that are recognized automatically from an ongoing discussion. The ACLD is intended to be used during meetings, but the system can also be demonstrated or tested even when there is no meeting happening, by replaying a group's meeting from an archive such as the AMI Meeting Corpus [1, 4] as a live meeting, and building a repository from the group's previous meetings and associated documents. The past meetings are divided into fragments before inclusion in the repository, and the past documents include reports, emails, and presentations given during the past meetings.

The architecture of the ACLD is represented in Figure 1, and includes the following modules and communication infrastructure:

1. *Document Bank Creator and Indexer*: prepares, and updates before each meeting, a database of documents and of snippets of previous meetings in which media and annotations are aligned.
2. *Query Aggregator*: at regular and frequent intervals during a meeting (e.g. every 30 seconds or upon the user's demand formulated by clicking a button), prepares a query to this database, derived from the ongoing conversation through automatic speech recognition (ASR) or keyword spotting (KWS), optionally emphasizing certain keywords. The module then executes the query, retrieves the results, and integrates them with previous ones, so that their variation is smoothed in time.
3. *User Interface*: displays the current search results, as clickable document links, and provides through these links access to past documents and meeting recordings, but also to web pages.
4. *Meeting processing modules:* automatic speech recognition (ASR, e.g. [6]) in the meeting room, keyword spotting (KWS, e.g. [14]), and segmentation into continuous spurts, possibly separated by speaker, or using speaker location from a microphone array.
5. *Support infrastructure*: the *Hub* architecture is represented as an area connecting several modules in Figure 1, as it allows communication between modules through a subscription-based client/server protocol. Annotation data circulating through the Hub is formatted as timed triples – i.e. tuples of (time, object, attribute, value) – and is stored in a relational database. Producers of annotations send them to the Hub, which forwards them automatically to the consumers that are subscribed to the respective types. For development and demonstration, real-time is simulated using data streamers that send data to the Hub from the files containing the annotations of recorded meetings. Complementing the Hub, the *HMI Media Server* can be used to broadcast the appropriate audio and video, in a synhronized fashion, to the various media consumers.

Document Bank Creator and Indexer

The *Document Bank Creator* (DBC) is run offline before a meeting, to create or update the repository of documents and pseudo-documents that will be searched during the meeting. Text versions of documents, for indexing, are generated from heterogeneous file formats (HTML and MS Office). In the current implementation, documents are gathered semi-automatically from the distribution server of the corpus [1], which gives access to the entire AMI Meeting Corpus, including media files, documents, metadata and annotations.

Past meetings are cut into snippets, which are currently one minute long fragments[1]. Snippets are prepared from the ASR transcript [7] of the past meetings, which is available with the AMI Corpus; in a standalone scenario, ASR can be obtained with acceptable quality (for human readers) a couple of hours after the meeting. Real-time ASR, used by the Query Aggregator, has a significantly lower accuracy. The trade-off in choosing the length of these snippets is between the need to have enough content words in each snippet, and the need for precision when snippets are opened in a meeting browser: if they are too large, then the user cannot easily find the region of interest.

The text version of all files is indexed by a *Document Indexer* using the Apache Lucene open-source software[2], creating indexes for each meeting in the present demonstration version. Indexing optimizes word-based search over large document sets. Here, all words are used as keywords, and the index is optimized using word stemmers and the TF*IDF weighting scheme.

Query Aggregator: Query Preparation and Document Retrieval

The *Query Aggregator* (QA) performs document searches at regular time intervals, using words and terms that are recognized automatically from the meeting discussion. Real-time large vocabulary speech recognition is still a challenging objective for the ASR community, but such systems have recently become easily available[3]. The words from the ASR are filtered for stopwords (our list has about 80 stopwords), so that only content words are used for search.

[1]Approximately, as we avoid interrupting a speech segment.

[2]From http://lucene.apache.org.

[3]For development purposes, the offline ASR provided with the AMI Corpus [7] is also used for simulations.

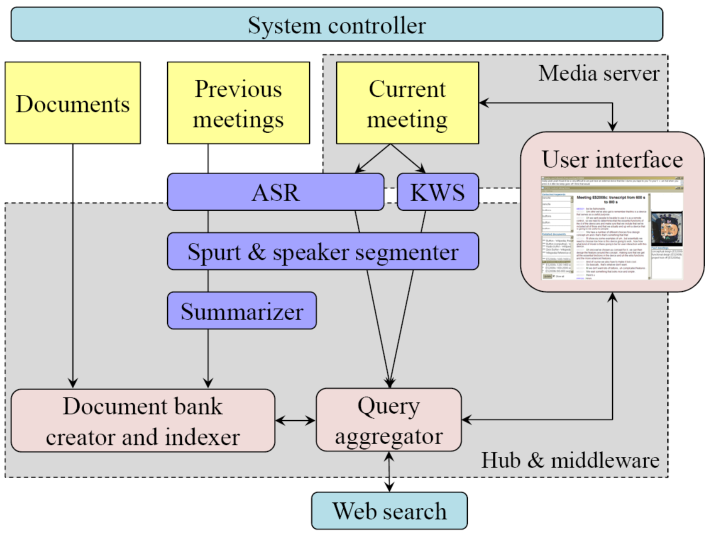


Figure 1. Main components of the ACLD. The two areas labeled as 'Hub & middleware' and 'Media server' represent the connection between all modules covering the respective areas. The Hub ensures real-time annotation exchange, while the Media server enables playback of audio and video from files or capture devices. ASR stands for automatic speech recognition, and KWS for keyword spotting.

A list of pre-specified keywords was defined, and if any of them is recognized, either by the Keyword Spotting device, or by matching in the ASR output, then they can be specifically marked in the query so that their importance is increased when doing the search (using Lucene's keyword boosting mechanism). As the development meeting corpus (AMI Corpus) is mostly made up of meetings where participants are working within a group to produce a new television remote control, the list of keywords for this domain contains words or expressions such as 'component', 'chip', 'interface', 'button', 'L_C_D', 'material', 'latex', 'wood', 'titanium', and so on, for a total of about 30 words. This list can be modified online by the users of the ACLD, by adding or removing words. If no list of keywords is available, then all words simply receive equal importance in the query.

The words or keywords that were recognized are processed in batches corresponding to time frames of fixed size, e.g. every 20-30 seconds – this can be modified when running the application, and queries can be launched also on demand when the user presses a button. Queries are addressed to the Apache Lucene information retrieval engine, over the index of documents previously created. The QA can also access the Google search engine via its public API, and it manages separately a list of the top hits retrieved from a web domain that is specified by the user (e.g., http://en.wikipedia.org). Indeed, selecting a specific domain for search ensures that results are more focused and less subject to noise.

The results returned by the Lucene engine are the meeting fragments and documents that most closely match the query – in information retrieval terms – in the respective time frame, accompanied by relevance scores. To avoid inconsistent results from one time frame to another, due to the fact that word choice varies considerably in such small samples (and therefore search results vary as well), a model of relevance persistence (described in a previous publication) ensures that items that are often retrieved tend to remain for a time at the top of the list.

Information Presentation: User Interfaces

The information produced by the modules described above, and naturally the main results produced by the QA, must be displayed to the user(s) in an unobtrusive, yet conspicuous manner. The designs and experiments that were carried out with the successive versions of the ACLD will be described in the following sections. Here, we make first some general observations.

The main requirement for the user interface (henceforth, UI) is to display results from the QA in an informative way, so that users can easily grasp their content, and then to offer upon demand quick access to the full content of past meetings, documents, and websites. The UI should also inform the user about the "perception" of the ongoing conversation by the system, so that the users understand the origin of the results and the potential sources of error. Finally, the UI should give access to ACLD settings, and, when used in a demonstration context with a replayed recording of a meeting, render audio and video from the recording. Yet other functionalities can be thought of, bringing the ACLD closer to the concept of a general meeting assistant that provides information about other users, documents, and so on. However, in what follows, we will avoid stretching the concept too much.

The properties of the UI have a large influence on the user's perception of the whole system, and it appeared gradually, through the versions described below, that they could also have a much stronger influence on overall evaluation scores than the core search-and-retrieval functionality. Therefore, instead of concentrating on retrieval quality (which amounts to evaluating ASR + Lucene), we invested significant effort in refining the UI based on feedback from users.

DESIGN OF THE USER INTERFACE: FIRST VERSION AND FEEDBACK

Following several initial trials, a first version of the UI intended for demonstration was released, which is represented in Figure 2 above. As development and demonstration of the system typically used a pre-recorded meeting in lieu of an ongoing one – which is harder to setup at will and not repeatable in constant conditions – this version devotes some space to show a video of the "current" (simulated) meeting.

Outline of the UI

The snapshot in Figure 2 shows the first version of the UI over meeting ES2008d of the AMI Corpus, three minutes from the beginning of the meeting. On the left, the list of keywords actually detected in the conversation reassures the user about the main search terms being used, as they were recognized from the audio. Every 30 seconds (in this version), a set of newly recognized keywords is added at the top, with the timestamp shown as a horizontal line. The central column, which scrolls in the same way as the keywords, shows the six most relevant documents for that time in the meeting, with font size chosen to reflect the hypothesized degree of relevance. This list is constantly updated as the meeting proceeds. At the bottom right there is a static display showing the three previous meetings in the history, giving access to their contents, metadata and summaries when hovering the mouse over their name, as for ES2008a in Figure 2. Above that, the room-view video of the ongoing meeting is displayed, with media controls as well.

This UI offers the users several possibilities for interacting with the resulting documents from the central frame, depending on each document type. For a meeting fragment, hovering over its label displays its extractive summary, which is obtained on-the-fly by the UI from the Hub, while clicking on the fragment's label displays its ASR transcript in an HTML browser. For documents, clicking on their label displays their text content in a new window, from where a version formatted in HTML can also be obtained. The HTML format was selected as it preserves a significant part of the original document's formatting, and is much quicker to visualize than opening the source document with its dedicated program, which becomes quite slow for MS Office documents as computers grow older.

Received Feedback

The ACLD was submitted, at various gatherings, to about thirty representatives of companies active in the field of meeting technology, which received the concept very warmly, and provided feedback. The presentation sessions lasted 30 min. each: they started with a presentation of the ACLD, which was followed by questions and feedback from the audience.

Many of the comments concerned the graphical layout of the UI, and suggested that screen real estate should be used more rationally, for instance by increasing, in proportion, the size of the frame displaying the results. Users suggested that access to document contents could be facilitated and that ideally a quick overview of each document could be provided. The relation between documents and meetings could be made clearer, e.g. using colour coding. It was also suggested to include a parallel facility for web search (which was mentioned above), and to represent keywords in a more structured way, for instance as a tag-cloud with emphasis varying with their relevance.

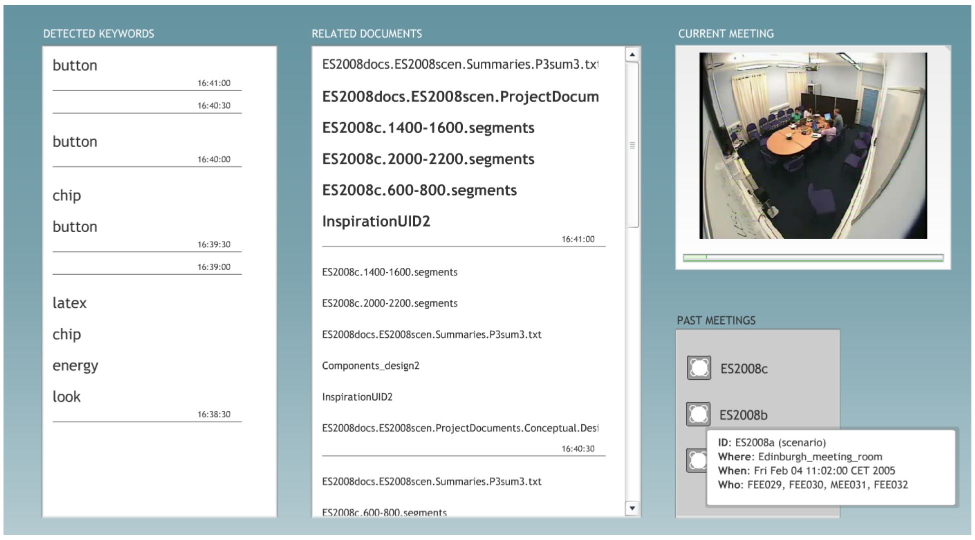
As the ACLD application grew more complex, following the initial feedback, it was decided to move the ACLD control functions, and in particular the repository management commands, into a System Controller that is displayed as a separate window, shown in Figure 3 above. In addition to document conversion and indexing, the controller also allows the selection of the running mode or "scenario" (live meeting or demonstration), the selection of the meeting to replay for a demonstration, and commands for starting and resetting the various components, and displaying logs.

INFORMATIVENESS VS. UNOBTRUSIVENESS: TWO ALTERNATIVE UI'S

User feedback as well as our own intuitions appeared to send contradictory messages about the screen's real estate: on the one hand, several types of information are considered useful to show, in addition to the results of the QA, mainly to justify or to enrich these results. On the other hand, as users might perform other operations on their computers during a meeting, the ACLD UI should occupy as little space as possible, and its results should be viewable in a large window only if they are considered of interest by the user, and if enough attention is available for them.

Therefore, a *Wide-screen UI* (shown in Figure 4) was developed from the initial UI, to display simultaneously, in several frames, all the information related to content linking. This was also accompanied by a major change in software technology, from Flash to Java. At the same time a *Widget UI* (shown in Figures 5 and 6) was designed to minimize the use of the screen's real estate through the use of tabs in the UI, with documents and meeting snippets opening in separate viewers, upon request only. The display of document names was also nuanced, to help users decide whether a document is worth opening or not.

Figure 4 shows a snapshot of the Wide-screen UI over meeting ES2008d. On the top left, the same list of keywords recognized from the audio is displayed. Every 30 seconds or on demand (by pressing the 'update' button at the bottom left), a newly recognized keyword set is added in the top left frame; this also triggers an immediate query in the QA, and the update of results (documents/snippets, and websites). The bottom left frame, which scrolls in the same way as the keywords, shows the five most relevant document names for that time in the meeting, as well the five most relevant web pages, ordered by relevance. Document relevance, available from the Lucene engine and modulated by the persistence model, is coded using asterisks ('*') in front of the document names. The frame can also display all past results as well, appended to the current one, if the user wishes so (by checking the 'show all' checkbox at the bottom left).


DETECTED KEYWORDS

button
16:41:00
16:40:30
button
16:40:00
chip
button
16:39:30
16:39:00
latex
chip
energy
look
16:38:30

RELATED DOCUMENTS

ES2008docs.ES2008scen.Summaries.P3sum3.txt
ES2008docs.ES2008scen.ProjectDocum
ES2008c.1400-1600.segments
ES2008c.2000-2200.segments
ES2008c.600-800.segments
InspirationUID2
16:41:00
ES2008c.1400-1600.segments
ES2008c.2000-2200.segments
ES2008docs.ES2008scen.Summaries.P3sum3.txt
Components_design2
InspirationUID2
ES2008docs.ES2008scen.ProjectDocuments.Conceptual.Desi
16:40:30
ES2008docs.ES2008scen.Summaries.P3sum3.txt
ES2008c.600-800.segments

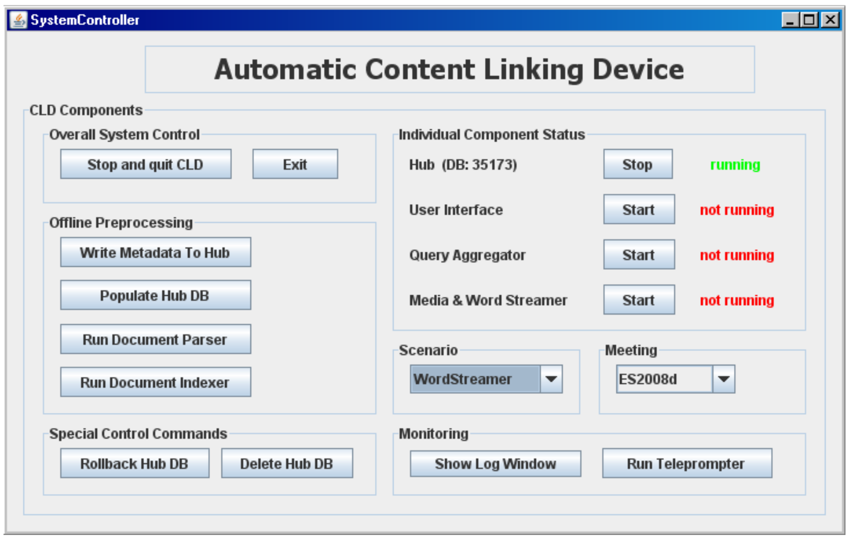
CURRENT MEETING

PAST MEETINGS

ES2008c
ES2008b

ID: ES2008a (scenario)
Where: Edinburgh_meeting_room
When: Fri Feb 04 11:02:00 CET 2005
Who: FEE029, FEE030, MEE031, FEE032

Figure 2. Snapshot of the first version of the user interface.


SystemController

Automatic Content Linking Device

CLD Components

Overall System Control
Stop and quit CLD | Exit

Offline Preprocessing
Write Metadata To Hub
Populate Hub DB
Run Document Parser
Run Document Indexer

Special Control Commands
Rollback Hub DB | Delete Hub DB

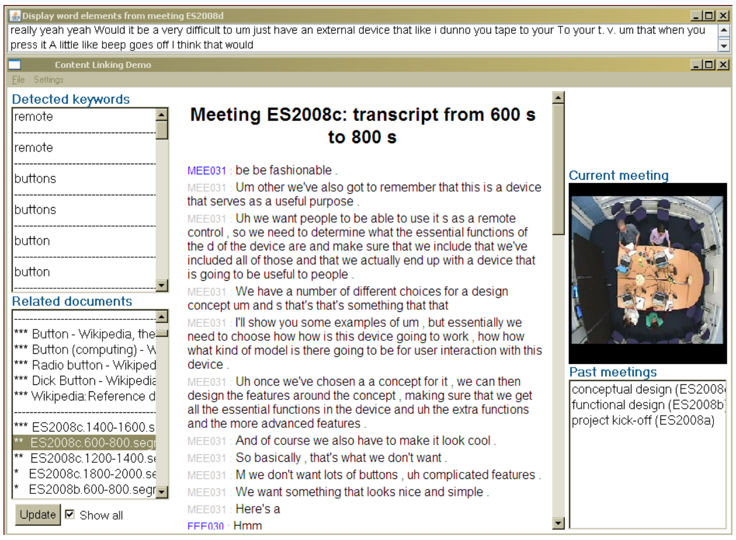
Individual Component Status

Hub (DB: 35173)	Stop	running
User Interface	Start	not running
Query Aggregator	Start	not running
Media & Word Streamer	Start	not running

Scenario: WordStreamer
Meeting: ES2008d

Monitoring
Show Log Window | Run Teleprompter

Figure 3. Snapshot of the System Controller, separated from the main UI.


Display word elements from meeting ES2008d

really yeah yeah Would it be a very difficult to um just have an external device that like i dunno you tape to your To your t. v. um that when you press it A little like beep goes off I think that would

Content Linking Demo

File Settings

Detected keywords

remote
remote
buttons
buttons
button
button

Related documents

*** Button - Wikipedia, the
*** Button (computing) - W
*** Radio button - Wikiped
*** Dick Button - Wikipedia
*** Wikipedia:Reference d

*** ES2008c.1400-1600.s
** ES2008c.600-800.segr
** ES2008c.1200-1400.se
* ES2008c.1800-2000.se
* ES2008b.600-800.segr

Update ☑ Show all

Meeting ES2008c: transcript from 600 s to 800 s

MEE031 : be be fashionable .

MEE031 : Um other we've also got to remember that this is a device that serves as a useful purpose .

MEE031 : Uh we want people to be able to use it s as a remote control , so we need to determine what the essential functions of the d of the device are and make sure that we include that we've included all of those and that we actually end up with a device that is going to be useful to people .

MEE031 : We have a number of different choices for a design concept um and s that's that's something that that

MEE031 : I'll show you some examples of um , but essentially we need to choose how how is this device going to work , how how what kind of model is there going to be for user interaction with this device .

MEE031 : Uh once we've chosen a a concept for it , we can then design the features around the concept , making sure that we get all the essential functions in the device and uh the extra functions and the more advanced features .

MEE031 : And of course we also have to make it look cool .

MEE031 : So basically , that's what we don't want .

MEE031 : M we don't want lots of buttons , uh complicated features .

MEE031 : We want something that looks nice and simple .

MEE031 : Here's a

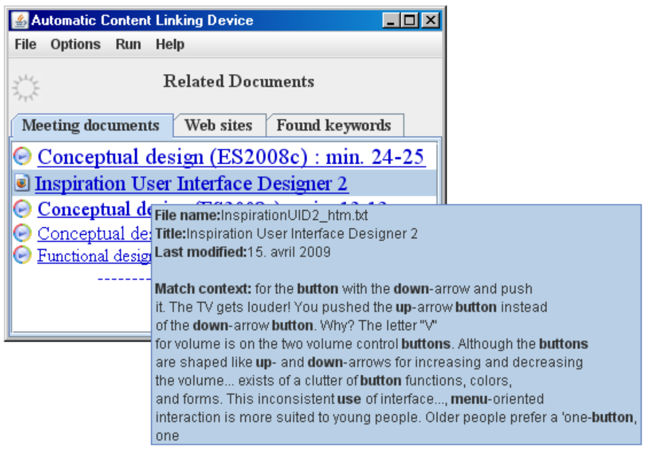
FEE030 : Hmm

Current meeting

Past meetings

conceptual design (ES2008
functional design (ES2008b
project kick-off (ES2008a)

Figure 4. Second version of the user interface: Wide-screen UI.


Automatic Content Linking Device

File Options Run Help

Related Documents

Meeting documents | Web sites | Found keywords

Conceptual design (ES2008c) : min. 24-25
Inspiration User Interface Designer 2
Conceptual de
Conceptual de
Functional desig

File name: InspirationUID2_htm.txt
Title: Inspiration User Interface Designer 2
Last modified:15. avril 2009

Match context: for the **button** with the **down**-arrow and push
it. The TV gets louder! You pushed the **up**-arrow **button** instead
of the **down**-arrow **button**. Why? The letter "V"
for volume is on the two volume control **buttons**. Although the **buttons**
are shaped like **up**- and **down**-arrows for increasing and decreasing
the volume... exists of a clutter of **button** functions, colors,
and forms. This inconsistent **use** of interface..., **menu**-oriented
interaction is more suited to young people. Older people prefer a 'one-**button**,
one

Figure 5. The Widget UI, showing the tab with the list of relevant documents at a given moment, with explicit labels. Hovering over a label displays the metadata associated with the document, as well as excerpts where the keywords were found. The transcript tab is closed in this snapshot.

At the bottom right of the Wide-screen UI is again a static display showing the past meetings in the group's history, giving access to their contents, metadata and summaries. Above that, the interface displays a room-view video of the ongoing meeting, with the audio in the case of past meetings. The UI displays in the large central frame a text or HTML version of the selected document, or the selected web page.

At the opposite side of the UI-dimension spectrum, Figure 5 shows a snapshot of the Widget UI, taken at some time during replay of meeting ES2008d. This version of the UI is a deliberately simplified window frame reduced to show exclusively content that is actually delivered by the ACLD at runtime, split into four optional tabs, which contain respectively:

1. Labels of the relevant documents and past meeting snippets found in the meeting index, preceded by an appropriate icon corresponding to the document type, to support faster identification by the user.

2. Relevant web links found within the pre-specified web domain.

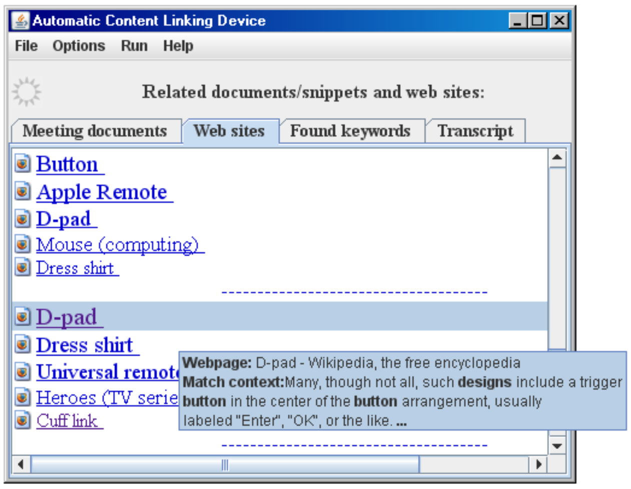
3. Keywords recognized in the respective time interval.

4. All words recognized by ASR with highlighted keywords (this tab is only activated in Figure 6).

The rank of a label in the document result list, as well as its font size, indicate its relevance within the query result; however, for web search results, the only indicator of relevance is the rank. Hovering over a result link (document, meeting snippet, web link) provides metadata about it in a pop-up window, including most importantly the match context shown in Figure 5. The match contexts shows excerpts of the document that match keywords and words detected from speech, with surrounding words. Clicking on a label (link) opens the respective document using an appropriate viewing program – respectively, a native editor, a meeting browser (such as JFerret, a successor to Ferret [15]), or a web browser. Similarly, the website tab shown in Figure 6 displays the results of the Google search over a given domain.

EXPERIMENTS WITH TASK-BASED EVALUATION

Two pilot experiments were conducted with the Widget UI, following a task-based scenario: four subjects were given the task to complete the design of a remote control that was started in a series of three past meetings (ES2008a through c from the AMI Corpus), and their success is evaluated in terms of satisfied constraints, overall efficiency and satisfaction. The two experiments have led to the following observations.



Webpage: D-pad - Wikipedia, the free encyclopedia
Match context: Many, though not all, such **designs** include a trigger **button** in the center of the **button** arrangement, usually labeled "Enter", "OK", or the like. ...

Figure 6. Widget UI with the website results, hovering over one result.

Firstly, in spite of the small size of the Widget UI, some users still minimized it, and then ignored its results. After improvements were made to the interface and to the pre-meeting training procedure, the second experiment showed improved levels of UI use over the first, but there remain questions about the appropriateness of the ACLD technology in this particular setting – viz., four-person meetings where all the participants are co-present. There are social and practical reasons why participants might feel too rushed or uncomfortable to open the linked content displayed in the UI very often. For example, participants did express appreciation of the full meeting browser technology for meeting preparation (by exploring the past three meetings), but during a meeting this kind of UI is generally too complex to use. Even in the case of linked web pages and documents, it is only during certain parts of a meeting that these were appreciated: some participants suggested that content should only be linked on-demand rather than constantly during the meeting.

These initial observations have led to consideration of remote participation scenarios, and also e-learning areas as potentially more promising application area for this technology. There are also arguments that in larger, laptop-open style meetings, users could more freely examine the linked content presented than in four-person meetings.

Further participant comments concern the visualisation of results, with some participants even suggesting replacing document names with the keywords they contain. This would offer a direct view to the relevance of documents with no need of mouse actions. In a similar vein, it was suggested that some context for each linked document should be presented without the need for mouse-over, so that less user action is required to allow judgement of relevance. Using tag-clouds of keywords with related documents could be an even more useful visualization, and these are all approaches to be invesigated.

It was considered important to demonstrate the link between the meeting and the results presented, but nonetheless some users thought that showing the full transcript as recognized by the ASR could be a bit confusing, especially when word error rates are quite high.

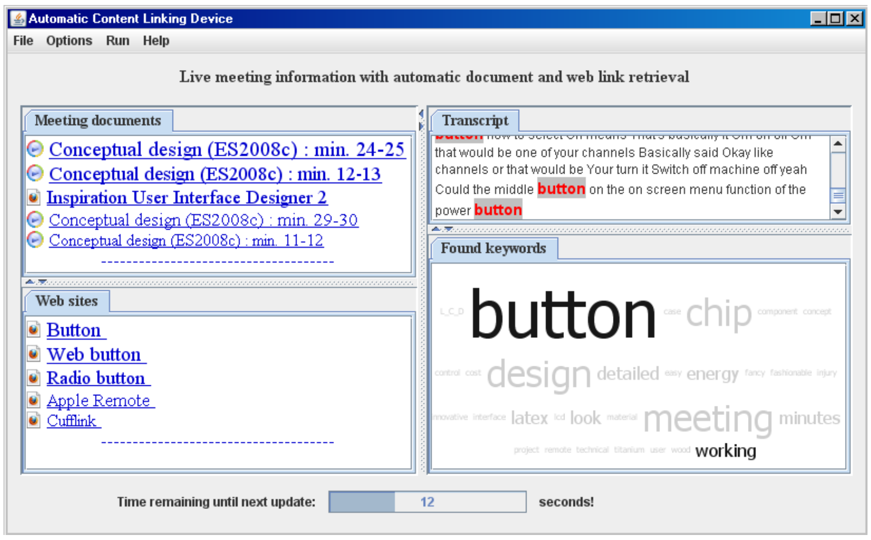
These comments echo those received from professionals to which the ACLD was demonstrated at various events. Some persons prefer a small, unobtrusive device, while others seem to consider a UI occupying their entire laptop screen. The synthesis of the received feedback reveals a number of directions for improvement along the following themes: make UI more user-friendly; make the document repository more customizable by the user; and allow users to play a more active role in search when they wish (and have time to do so), e.g. to run a specific search, or to send relevance feedback to the Query Aggregator in order to refine the results.

TOWARDS A MODULAR UI

As an intermediate conclusion of the evaluations and experiences made with the versions above we are now moving towards a user configurable and modifiable version following a modular approach. The core idea behind that is to provide the user maximal flexibility by offering a complete repertoire of UI "widgets" that may be configured at runtime according to individual user preferences. This would allow users to select the widgets they would like to see, to scale and position them at will, and resize the entire window as well. Furthermore, the approach is using standard UI elements such as option menus to select visible widgets, as well as mouse drag&drop facilities for realizing UI modifications interactively.

The list of foreseen UI widgets includs first those mentioned in previous sections, i.e. for showing transcript words (with keyword highlighting), matching keywords, document browsing, snippet visualisation, and web browsing. In addition, appropriate widgets for visual representation of timeline, meeting participants, past meetings as a whole, and video signals are currently being explored. Orthogonally to the contents of the widgets, we also experiment with alternative content visualizations such as list presentations as opposed to tag-clouds or speaker-segmented ASR.

Figure 7 shows the current implementation of the Modular UI. It contains the same elements as the Widget UI (see 6), but now each of them is realized as a separate tabbed pane. The 'options' menu allows to interactively activate or deactivate each pane and the panes themselves can be arranged as desired simply by dragging and dropping them with the mouse. For example, dragging one tab on top of a specific pane adds this tab to the pane and inversely dragging a tab out of a pane opens a new pane containg this tab only. In this way, the Widget UI and the Modular UI can even be smoothly transformed into each other or every intermediate representation could be chosen at runtime. This includes of course changing the size of the individual panes and also the UI frame itself, which may now range from a small window running as a secondary meeting assistance tool (like the Widget UI) to a wide screen UI running as the primary application (like the Wide-screen UI).


Automatic Content Linking Device
File
Options
Run
Help
Live meeting information with automatic document and web link retrieval
Meeting documents
Transcript
Web sites
Found keywords
Time remaining until next update:
12
seconds!

Conceptual design (ES2008c) : min. 24-25
Conceptual design (ES2008c) : min. 12-13
Inspiration User Interface Designer 2
Conceptual design (ES2008c) : min. 29-30
Conceptual design (ES2008c) : min. 11-12

Button
Web button
Radio button
Apple Remote
Cufflink

that would be one of your channels Basically said Okay like channels or that would be Your turn it Switch off machine off yeah Could the middle button on the on screen menu function of the power button

L_C_D button case chip component concept control cost design detailed easy energy fancy fashionable injury innovative interface latex lcd look material meeting minutes project remote technical titanium user wood working

Figure 7. First implemented version of the Modular UI.

CONCLUSION

This paper has put forward the diversity of user preferences concerning the design of a user interface for a multimedia just-in-time retrieval system, the ACLD. As suggested by the pilot studies that were conducted, a whole range of data visualisations seem to be supported at least by subsets of user preferences. Therefore, adopting a modular approach is potentially the best way to increase user satisfaction with the UI, and hence user uptake of the ACLD. Such an approach allows us to simply plug in and study new visualisations as alternatives, which is especially useful if new modes of usage are explored, for instance with remote meeting participants as suggested by some pilot experiments.

ADDITIONAL AUTHORS

REFERENCES

1. AMI Consortium. The AMI Meeting Corpus, http://corpus.amiproject.org, accessed 15 September 2009.

2. J. Budzik and K. J. Hammond. User interactions with everyday applications as context for just-in-time information access. In *IUI 2000 (5th International Conference on Intelligent User Interfaces)*, New Orleans, LA, 2000.

3. V. Bush. As we may think. *The Atlantic Monthly*, July, 1945.

4. J. Carletta, S. Ashby, S. Bourban, M. Flynn, M. Guillemot, T. Hain, J. Kadlec, V. Karaiskos, W. Kraaij, M. Kronenthal, G. Lathoud, M. Lincoln, A. Lisowska, I. McCowan, W. Post, D. Reidsma, and P. Wellner. The AMI Meeting Corpus: A pre-announcement. In S. Renals and S. Bengio, editors, *Machine Learning for Multimodal Interaction II*, LNCS 3869, pages 28–39. Springer-Verlag, Berlin/Heidelberg, 2006.

5. A. Franz and B. Milch. Searching the Web by voice. In *Coling 2002 (19th International Conference on Computational Linguistics)*, pages 11–15, Taipei, 2002.

6. P. N. Garner, J. Dines, T. Hain, A. El Hannani, M. Karafiat, D. Korchagin, M. Lincoln, V. Wan, and L. Zhang. Real-time ASR from meetings. In *in press*, 2009.

7. T. Hain, L. Burget, J. Dines, G. Garau, M. Karafiat, M. Lincoln, J. Vepa, and V. Wan. The AMI system for the transcription of speech in meetings. In *ICASSP 2007 (32nd International Conference on Acoustics, Speech, and Signal Processing)*, pages 357–360, Honolulu, 2007.

8. P. E. Hart and J. Graham. Query-free information retrieval. *IEEE Expert: Intelligent Systems and Their Applications*, 12(5):32–37, 1997.

9. M. Henziker, B.-W. Chang, B. Milch, and S. Brin. Query-free news search. *World Wide Web: Internet and Web Information Systems*, 8:101–126, 2005.

10. M. Lew, N. Sebe, C. Djeraba, and R. Jain. Content-based multimedia information retrieval: State of the art and challenges. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMCCAP)*, 2(1):1–19, 2006.

11. B. J. Rhodes. The Wearable Remembrance Agent: A system for augmented memory. *Personal Technologies: Special Issue on Wearable Computing*, 1:218–224, 1997.

12. B. J. Rhodes and P. Maes. Just-in-time information retrieval agents. *IBM Systems Journal*, 39(3-4):685–704, 2000.

13. B. J. Rhodes and T. Starner. The Remembrance Agent: A continuously running information retrieval system. In

PAAM 1996 (1st International Conference on Practical Applications of Intelligent Agents and Multi-Agent Technology), pages 486–495, London, 1996.

14. I. Szoke, P. Schwarz, P. Matejka, L. Burget, M. Karafiat, M. Fapso, and J. Cernocky. Comparison of keyword spotting approaches for informal continuous speech. In *Eurospeech 2005 (9th European Conference on Speech Communication and Technology)*, pages 633–636, Lisbon, 2005.

15. P. Wellner, M. Flynn, and M. Guillemot. Browsing recorded meetings with Ferret. In S. Bengio and H. Bourlard, editors, *Machine Learning for Multimodal Interaction I*, LNCS 3361, pages 12–21. Springer-Verlag, Berlin/Heidelberg, 2004.