The AMIDA Automatic Content Linking Device: Just-in-Time Document Retrieval in Meetings

Andrei Popescu-Belis[1], Erik Boertjes[2], Jonathan Kilgour[3], Peter Poller[4], Sandro Castronovo[4], Theresa Wilson[3], Alejandro Jaimes[5,⋆], and Jean Carletta[3]

[1] Idiap Research Institute, P.O. Box 592
CH-1920 Martigny, Switzerland
andrei.popescu-belis@idiap.ch
[2] TNO ICT, Brassersplein 2
NL-2612 CT Delft, The Netherlands
erik.boertjes@tno.nl
[3] HCRC and CSTR, University of Edinburgh
2 Buccleuch Place, Edinburgh, EH8 9LW, UK
{jonathan,jeanc,twilson}@inf.ed.ac.uk
[4] DFKI GmbH, Stuhlsatzenhausweg 3
D-66123 Saarbruecken, Germany
{peter.poller,sandro.castronovo}@dfki.de
[5] Telefonica Research, C/Emilio Vargas 6
ES-28403 Madrid, Spain
ajaimes@tid.es

Abstract. The AMIDA Automatic Content Linking Device (ACLD) is a just-in-time document retrieval system for meeting environments. The ACLD listens to a meeting and displays information about the documents from the group's history that are most relevant to what is being said. Participants can view an outline or the entire content of the documents, if they feel that these documents are potentially useful at that moment of the meeting. The ACLD proof-of-concept prototype places meeting-related documents and segments of previously recorded meetings in a repository and indexes them. During a meeting, the ACLD continually retrieves the documents that are most relevant to keywords found automatically using the current meeting speech. The current prototype simulates the real-time speech recognition that will be available in the near future. The software components required to achieve these functions communicate using the Hub, a client/server architecture for annotation exchange and storage in real-time. Results and feedback for the first ACLD prototype are outlined, together with plans for its future development within the AMIDA EU integrated project. Potential users of the ACLD supported the overall concept, and provided feedback to improve the user interface and to access documents beyond the group's own history.

Keywords: just-in-time retrieval, meeting assistants, meeting processing, real-time document retrieval.

⋆ Work performed while at Idiap Research Institute.

A. Popescu-Belis and R. Stiefelhagen (Eds.): MLMI 2008, LNCS 5237, pp. 272–283, 2008.

1 Introduction

Participants in a meeting often mention documents containing facts that are currently discussed, but only few documents are at hand. Searches could be performed within a document management system for the right piece of information, but the participants in a meeting usually do not have the time to perform such operations frequently during the meeting. Moreover, even where they do have their documents available, few groups have access to recordings of their past meetings, much less an efficient device for searching them. And when browsing through the recordings of previous meetings, users do not have the time to search for additional information among the meeting documents.

Therefore, a system that would provide tailored access to potentially relevant documents or recorded meetings, based on ongoing discussions, could be very valuable in improving group decision-making. Such an *Automatic Content Linking Device (ACLD)* could be applied to at least two scenarios [1]: the device could be used online during a meeting to display potentially relevant documents in real time (*meeting assistant*), or it could be used offline to browse a past meeting that was recorded, enriching it with potentially relevant documents (*meeting browser*). These scenarios are broadly related to the following options observed in the literature. Conceptually, the content linking mechanism is the same in both cases, only the resources that are available and the constraints of producing results in real-time are different.

1. *Just-in-time retrieval* [2,3,4]: participants to a meeting are constantly given suggestions about documents (including excerpts of previous meetings) that are potentially relevant to the ongoing discussion. Participants are free to ignore them, or to start using them to enhance the discussion, e.g. with figures, precise facts, or decisions made in previous meetings.
2. *Document/speech alignment for meeting browsers* [5,6,7]: users of a meeting archive can view the recordings of previous meetings augmented with related documents, regardless of whether the participants to the meeting referred to them explicitly or not. This can be essential for meetings whose main purpose is to discuss a long document, e.g. a report, and might provide a quicker understanding of the meeting context.

The AMIDA Content Linking Device (ACLD) demonstrates the basic concept of tailored access to a group's history using a set of four meetings from one of the groups recorded in the AMI Meeting Corpus [8]. Although the primary use of such a device would be during live meetings, we need to be able to demonstrate the concept even when there is no meeting happening. Our demonstration replays the group's last meeting (ES2008d) to simulate a live meeting, treating segments from the three previous meetings (ES2008a-c) and associated documents as the group history to be linked. In the recordings, the group carries out a role-playing exercise in which they pretend to be a design team specifying a new kind of remote control. Each group member is given a unique role to play in the team and carries out individual work as well as taking part in the four meetings. Final design decisions are made in the last meeting, which is ES2008d, therefore a

number of project documents and fragments of previous meetings are relevant to the discussions in this meeting. The past documents available for linking include reports, emails, and presentations given during the first three meetings, plus segments derived from the first three meetings by dividing them into 200 second chunks.

The remainder of the paper is organized as follows. Section 2 outlines the concept, architecture, and components of the ACLD, which are described in detail in the various subsections of Section 3. Brief implementation notes for the proof-of-concept prototype appear in Section 4, while evaluation results and perspectives for future work are given in Sections 5 and 6 respectively.

2 Concept and Architecture

The Automatic Content Linking Device performs searches at regular intervals over a database of meeting-related documents and pseudo-documents. The search criterion is constructed based on the words that were recognized automatically from the meeting discussion, thanks to online or offline automatic speech recognition (ASR)[1]. The audio signal is captured in an instrumented meeting room [9] or elsewhere, but recording conditions have a strong influence on the recognition accuracy. If some pre-specified terms or keywords are recognized, then they receive greater weight in the subsequent query.

The results are presented as a list of document names ordered by relevance, which can be empty if no document matches enough the words that were recognized. A persistence (smoothing) mechanism ensures that documents which are often retrieved remain some time at the top of the list. A user interface offers the participants quick access to the content of the documents that are retrieved, if they need to search them for valuable information.

These functionalities are supported by a number of modules that communicate through a subscription-based client/server architecture called 'the Hub' [10]. The Hub allows the connection of heterogeneous software modules, which may operate remotely, and ensures that data exchange is extremely fast – a requirement for real-time processing of human interaction. Data circulating through the Hub is formatted as timed triples (time, object, attribute, value), and is also stored in a special-purpose database, which was designed to deal with large-scale, real-time annotation of audio and video recordings. 'Producers' of annotations send triples to the Hub, which are received by the 'consumers' that subscribed to the respective types; consumers can also query the Hub for past annotations and metadata about meetings.

The architecture of the ACLD is shown in Figure 1, while the main components are first outlined below and then described in the following subsections.

Document Bank Creator (DBC): Gathers documents that are of potential interest for an upcoming meeting. In the current implementation, this is

[1] An online ASR module was recently developed in the AMIDA project, and its connection to the ACLD is under work at the time of writing.

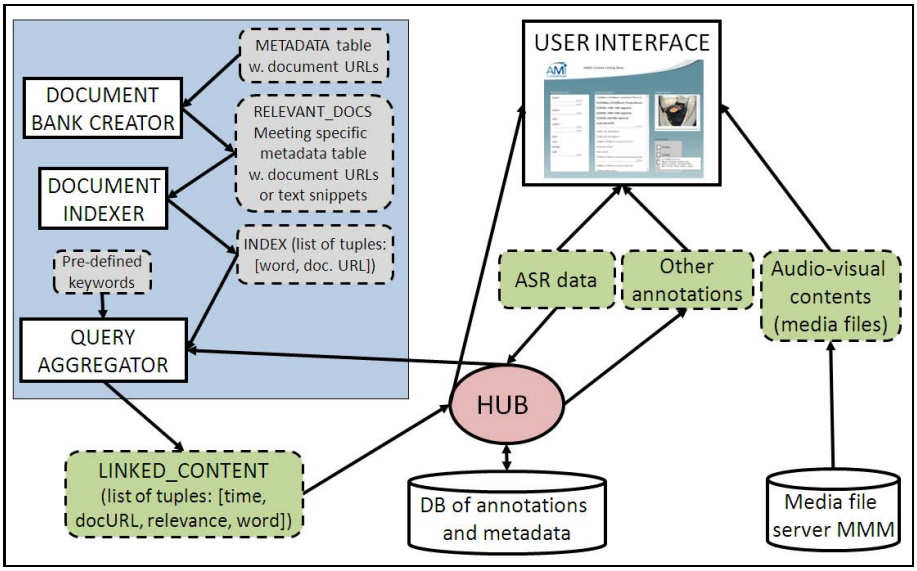

Fig. 1. Architecture of the AMIDA Automatic Content Linking Device

done semi-automatically from IDIAP's multimodal media file server (MMM, see `http://mmm.idiap.ch`), which gives access to the entire AMI Meeting Corpus, including media files, documents, metadata and annotations.

Document Indexer (DI): Creates an index over the document bank prepared by the DBC for the upcoming meeting.

Query Aggregator (QA): Performs document searches at regular time intervals, using words and terms that are recognized automatically from the meeting discussion, and produces a list of document names, ordered by relevance, based on the search results and on the persistence model explained below.

User Interface (UI): Displays results from the QA and offers quick access to text, HTML and source versions of documents, as well as to metadata and summaries for past meetings.

3 Components of the Automatic Content Linking Device

3.1 User Interface

We start the description of the ACLD with the User Interface, as this encompasses most of the functionalities of the system. In the online scenario of use, a connection must initially be established between the ASR device and a live meeting that is captured in a smart meeting room. In the offline scenario (or to demonstrate the online one from past recordings), the only information initially given to the UI is the identifier of a completed meeting to display. This allows the UI to retrieve via the Hub all the pointers to the related media, and to subscribe

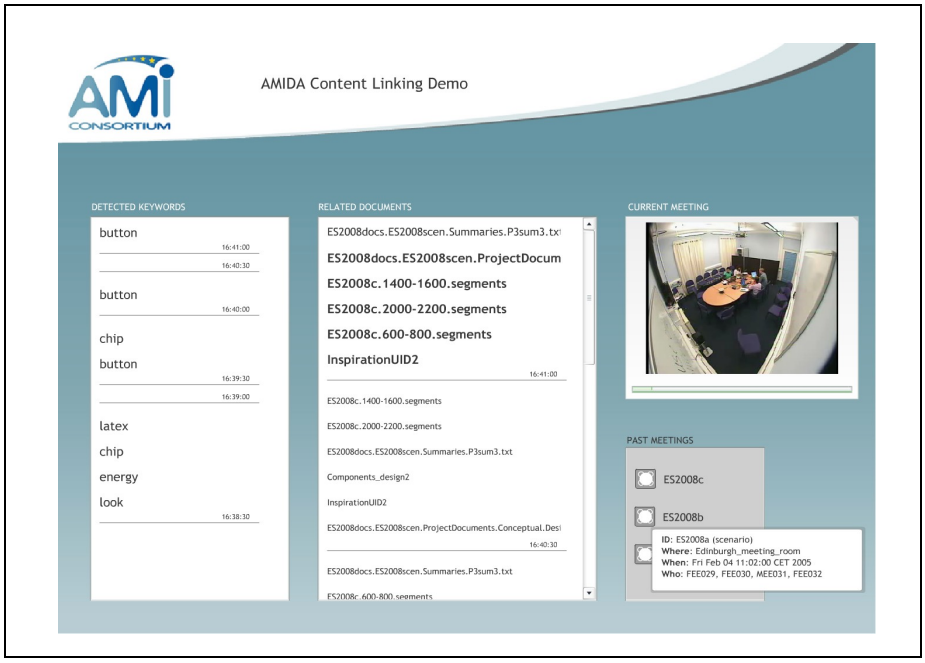

Fig. 2. Snapshot of AMIDA ACLD's user interface

to all the annotations that will be displayed, including the `Content_Linking` annotations produced by the Query Aggregator. For demonstrations and for meeting browsing, it is more convenient for repeatability reasons to use a completed meeting (ES2008d in the present version), hence, a number of metadata variables are hard-coded into the UI.

Figure 2 shows a snapshot of the UI over meeting ES2008d, three minutes from the beginning of the meeting. On the left, a list of keywords, referring to important concepts for the group's activity, reassures the user about the search terms being used, as they were recognized from the audio. Every 30 seconds, a newly recognized keyword set 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. At the bottom right there is a static display showing the three meetings in the history – giving access to their contents, metadata and summaries – and above that, the room-view video of the ongoing meeting (with the audio in the case of past meetings).

The UI displays at any given moment in a meeting at most N documents ordered by relevance, based on the data it receives from the QA, which contains information about the documents' URL, their type and relevance, meeting time and detected keywords. This list is constantly updated as the meeting proceeds. The interface offers the users several possibilities for interacting with documents,

depending on each document type. For a meeting fragment, hovering over its label displays its extractive summary (obtained on-the-fly by the UI from the Hub), while clicking on the label displays the ASR transcript. For documents, clicking on their label displays their content in a text window, from where a version formatted in HTML can also be obtained. This file 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 is quite slow for MS Office documents.

3.2 Document Bank Creator

The Document Bank Creator is run offline before a meeting to create the bank of documents and pseudo-documents that will be searched during the meeting. This is a preparation task, which copies documents in a separate folder, in preparation for the Document Indexer (alternatively, a metadata layer with pointers to documents could be generated to avoid copying). In less supervised scenario for the future, the DBC could determine automatically the documents that are potentially relevant, based on the project or series the meeting belongs to.

The DBC includes documents, fragments of previous meetings, slides, and emails. The fragments of past meetings are currently 200-second chunks of the ASR transcript, but a more logical segmentation based for instance on topics [11] is under study. The DBC accepts heterogeneous file formats, and extracts text from them using calls to the proprietary software that created the files. In the process, the module also generates HTML versions of each document, which are easier and quicker to visualize than the original MS Office versions.

3.3 Document Indexer

The Document Indexer uses the text version of the files associated to the current meeting by the DBC to construct an index, i.e. a data structure that optimizes word-based search over the document set, which can become quite large over time. The index can also be conceived of as a new annotation layer, represented logically as a list of tuples (meeting, keyword, doc_type, URL), where the URLs are used as unique identifiers of the documents. The DI uses a state-of-the-art system, Apache Lucene in its Perl implementation called Plucene, using all words as keywords and building an optimized index using word stemmers and the TF*IDF weighing scheme.

In the present implementation, the index is accessed directly by the Query Aggregator as a set of files in native Plucene format. However, as the index is a permanent layer of information concerning the documents related to a meeting, it could be stored in a declarative format in the Hub's main database, from where it could be retrieved at the beginning of the demo by the Query Aggregator, which is constantly using it.

3.4 Query Aggregator

The Query Aggregator periodically extracts from the speech of a given meeting a list of keywords that are mentioned, using the ASR, or even a manual transcript

for the offline scenario or for development purposes. The QA gets the words via the Hub and processes them in batches corresponding to time frames of fixed size, currently every 30 seconds. This size is a compromise between the need to gather enough words for search, and the need to refresh the search results reasonably often. Instead of the fixed time frame, information about audio segmentation into spurts or utterances could be used for a more natural segmentation of the ASR input.

The QA uses the words to build a query string for the Apache Plucene engine, which searches the index to retrieve relevant documents. These documents are sent as new `Linked_Content` annotations to the Hub, from where they can be used by the UI to display the document labels and give access to them via their URLs. This task has thus a similar goal as speech/document alignment [7,12], except that alignment is viewed here as the construction of sets of relevant documents for each meeting segment, and not only as finding the document that the segment "is about". The retrieval techniques that are employed are therefore quite different too, as speech/document alignment relies on precise matching between a referring expression and one of the elements of a document.

An offline version of the QA generates static XML and HTML views for completed meetings, which are used for debugging and for evaluation. The HTML view shown in Figure 3 displays on the left the ASR for the current meeting segment t_n, and on the right up to six most relevant documents (using their HTML version) with their relevance scores. The keywords are highlighted in red, both within the meeting transcript and in the documents. The words from the transcript are highlighted in blue, but only in the documents when they are

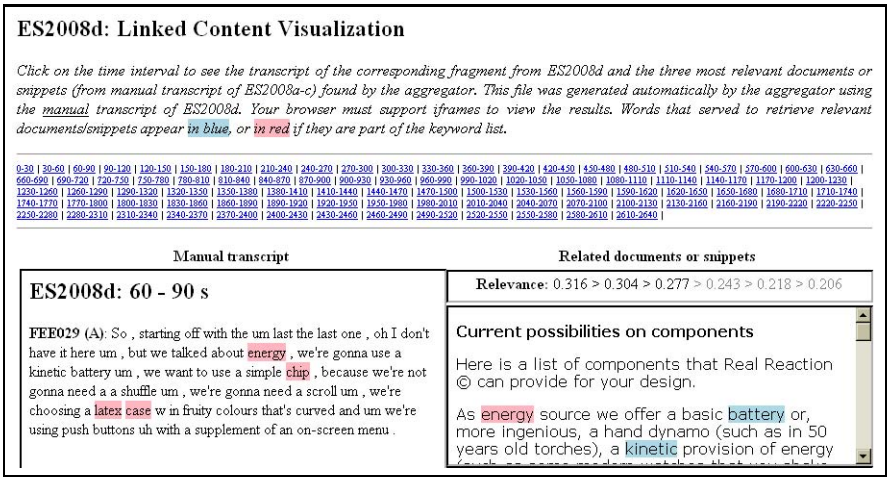

Fig. 3. HTML view of the offline output of the Query Aggregator – only the best document is shown (bottom right). Keywords are highlighted both in the transcript (left) and in the documents (right): e.g., 'energy', 'chip' or 'latex'. Words from the transcript that appear in the retrieved documents are highlighted only in the documents (right): e.g., 'kinetic' or 'battery'.

found – otherwise the entire meeting segment would be highlighted in blue, which is not very informative. The upper frame of the interface in Figure 3 allows the user to select a segment of the current meeting based on its timing in seconds.

Role of Pre-specified Keywords. Existing knowledge about the important terminology of a project can be used to increase the impact of specific words on search. A list of pre-specified keywords can be defined, and in case any of them is detected in the audio input from the meeting, their importance is increased when doing the search, using Plucene's boosting mechanism. The weight of the keyword boosting is currently set at five times the weight of non-boosted words. A specific list was defined by the user-study group for the meetings under study, and at present it contains words or expressions such as 'cost', 'energy', 'component', 'case', 'chip', 'interface', 'button', 'L_C_D', 'material', 'latex', 'wood', 'titanium', and so on, for a total of about 30 words. However, the QA works also without a list of boosted terms.

In addition, the words from the ASR or transcript are filtered for stopwords, so that mostly content words are used for search. Our list has about 80 words, including the most common function words, interjections and discourse markers.

The QA performs document search by matching the query words from the ASR with those from the index constructed by the DI and returns the most relevant set of documents for the respective time frame, more specifically a list of tuples such as (meeting, time, keyword, relevance, doc_type, pointer). It is useful to include in this annotation the keywords that were matched (i.e. the ones that helped to retrieve the specific document) as well as a relevance score produced by the search engine, to allow the interface to sort the relevant documents as needed. This annotation, of the `Linked_Content` type, is sent to the Hub (and also stored in the Hub's database), from where it is retrieved by consumers that have subscribed to `Linked_Content`, such as the user interface.

Persistence and Filtering Mechanisms. 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 *persistence (smoothing) mechanism* was defined. This mechanism was partly inspired by the notion of perceptual salience of entities, used for reference resolution, and more specifically from techniques that were implemented to compute salience in texts or in multimodal settings [7,13,14]. In the present case, the relevance of the documents amounts to a form of conceptual salience that evolves in time.

The persistence mechanism adjusts the current relevance scores for each document returned by the search engine, considering also the documents from the previous time frame and their own adjusted relevance scores. If t_n denotes the current time frame and t_{n-1} the previous one, and if $r(t_n, d_k)$ is the raw relevance of document d_k computed by the search engine after a query at time t_n, then the *adjusted relevance* $r'(t_n, d_k)$ computed using the persistence (smoothing) mechanism, is $r'(t_n, d_k) = r(t_n, d_k) + \alpha * r'(t_{n-1}, d_k)$, where α is a smoothing factor. Roughly, a larger value of α denotes a larger persistence – but α should be set below 1, because if $\alpha > 1$ then $r(t_n))$ keeps increasing even if the document is

no longer retrieved. In our experiments, a typical value of $\alpha = 0.8$ was used. The intuition behind the choice of this formula (as opposed to a more traditional $r'_n = \alpha * r_n + (1 - \alpha) * r'_{n-1}$) is a correction of the relevance score returned by the search engine, possibly increasing it if the document was already present, but without multiplying it from the start by an α factor.

Additionally, a filtering mechanism deletes the least relevant of the documents sent to the UI, returning at most N documents (currently $N = 6$), or fewer, depending on the following constraints. Given the list of all documents that were retrieved, sorted by decreasing relevance, the QA sends to the UI the documents that have an adjusted relevance above a certain threshold (currently 0.2), and the list of results is truncated where relevance decreases sharply, typically when $r'(t_n, d_{k+1}) \leq 0.5 * r'(t_n, d_k)$.

4 Implementation

The first version of the AMIDA Automatic Content Linking Device is now operational, and a second version is in preparation at the time of writing. Both the UI and the QA are implemented using two components: a Java front-end ensuring communication with the Hub – as a consumer for the UI or as a producer and consumer for the QA – and a separate piece of code in a different programming language – Flash for the UI and Perl for the QA.

The ACLD runs on a single Windows PC or over a network, and other operating systems will be considered in the future. The main software prerequisite is the Hub itself, which requires a MySQL database with one table for timed triples. To run the QA, Perl and the Plucene search and indexing modules are required. Compilation of all source files is centrally managed by a `build.xml` file in the top level directory of the repository, which requires the Apache Ant build tool. A number of variables can be set by modifying the initial lines of `build.xml`. The same `build.xml` file also executes the following groups of actions required to start the ACLD on meeting ES2008d, once all source code is compiled:

1. Start the Hub and roll back its database to the state that holds after meetings ES2008a-c and before ES2008d.
2. Start the QA and the UI, which subscribe to the Hub.
3. Stream the words obtained by the ASR for ES2008d to the Hub.

As both the QA and UI "listen" to the Hub, the words are sent to the QA, which sends back `Content_Linking` data, which is used by the UI to display the results.

5 Evaluation

The execution tests of the first prototype have been satisfactory: the communication between the modules using the Hub works smoothly, and the logs show that modules connect properly, and that annotation triples are correctly sent

and received. The documents that are retrieved contain the expected words and keywords, as we carefully checked using the static HTML representation of `Linked_Content` produced by the QA (Figure 3). The functionalities offered by the UI over these documents are available as described in Section 3.1. The nature of Perl scripting makes it easy to change many of the parameters of the QA, even while the system is running, which allows experimenting with various values of the persistence and filtering model, and with various lists of keywords and stopwords.

The performance evaluation of the ACLD is the topic of future work. One can test the performance of the retrieval system in terms of precision and recall, but this requires the definition of a ground truth document set for each time interval of a meeting, which is the main difficulty for such an evaluation. Three approaches to the ACLD evaluation problem are planned: (1) construct ground truth data using human annotators who associate documents to meeting segments; (2) evaluate the ACLD by judging the relevance of each document it returns; and (3) test the ACLD *in use* on the participants to an ongoing meeting, by measuring how often they consult the proposed documents.

The ACLD was demonstrated to potential industrial partners, namely about thirty representatives of companies that are active in the field of meeting technology. A series of sessions, lasting 30 minutes each, started with a presentation of the ACLD and continued with a discussion, during which notes were taken by the first author. The participants found that both online and offline application scenarios are promising, 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.

6 Future Work

The first implementation of the ACLD served as a demonstration or proof-of-concept, and enabled the authors to collect feedback indicating the most important developments that are required to turn it into a real-world application.

The **graphical layout of the interface** will be improved by allowing a larger part of the screen to be used for displaying the documents, using larger overviews of each document, and discarding past documents more quickly. This would also help to reduce the number of mouse clicks required to access the content of documents. Color-coding the document types and displaying their relations to the words from the ASR would also improve user experience.

Another line of suggestions concerns the **document repository**, which can be extended in various ways. The repository could include documents from larger sets, which are not entirely known to users, so that the interface brings new knowledge into a meeting. These sets could be private, personalized and better structured. A significant extension would be the connection to a Web search engine, which could be limited to a sub-domain to avoid potential noise in the results.

A number of **additional functionalities** were suggested. For instance, keeping a record of the documents that were consulted during a meeting might help

users who want to go back to them after the meeting. Detecting similarities with previous discussions would help alerting users that they already had this discussion before. Retrieval could be improved by including a relevance feedback mechanism for the returned documents, by representing keywords in a structured manner, e.g. using tag clouds, and by using word sense disambiguation to improve the precision of the retrieval.

Finally, the ACLD could be part of a **broader-scope meeting assistant**, which would not only help local participants with their documents, but would also improve the engagement of remote participants that attend a meeting using a mobile device [15]. In this case, document sharing would be one of the factors that improve the participants' engagement in a meeting.

Acknowledgments

This work was supported by the European Union's IST Programme, through the AMIDA Integrated Project FP6-0033812, Augmented Multiparty Interaction with Distance Access.

References

1. Nijholt, A., Rienks, R., Zwiers, J., Reidsma, D.: Online and off-line visualization of meeting information and meeting support. The Visual Computer 22(12), 965–976 (2006)
2. Hart, P.E., Graham, J.: Query-free information retrieval. IEEE Expert: Intelligent Systems and Their Applications 12(5), 32–37 (1997)
3. Budzik, J., Hammond, K.J.: 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)
4. Henziker, M., Chang, B.W., Milch, B., Brin, S.: Query-free news search. World Wide Web: Internet and Web Information Systems 8, 101–126 (2005)
5. Rhodes, B.J., Maes, P.: Just-in-time information retrieval agents. IBM Systems Journal 39(3-4), 685–704 (2000)
6. Franz, A., Milch, B.: Searching the Web by voice. In: Coling 2002 (19th International Conference on Computational Linguistics), Taipei, pp. 11–15 (2002)
7. Popescu-Belis, A., Lalanne, D.: Reference resolution over a restricted domain: References to documents. In: ACL 2004 Workshop on Reference Resolution and its Applications, Barcelona, pp. 71–78 (2004)
8. Carletta, J., Ashby, S., Bourban, S., Flynn, M., Guillemot, M., Hain, T., Kadlec, J., Karaiskos, V., Kraaij, W., Kronenthal, M., Lathoud, G., Lincoln, M., Lisowska, A., McCowan, I., Post, W., Reidsma, D., Wellner, P.: The AMI Meeting Corpus: A pre-announcement. In: Renals, S., Bengio, S. (eds.) MLMI 2005. LNCS, vol. 3869, pp. 28–39. Springer, Heidelberg (2006)
9. Moore, D.J.: The IDIAP Smart Meeting Room. Communication 02-07, IDIAP Research Institute (july 2002)
10. AMIDA: Commercial component definition. Deliverable 7.2, AMIDA Integrated Project (Augmented Multi-party Interaction with Distance Access) (November 2007)

11. Hsueh, P.Y., Moore, J.D.: Combining multiple knowledge sources for dialogue segmentation in multimedia archives. In: ACL 2007 (45th Annual Meeting of the Association of Computational Linguistics), Prague, pp. 1016–1023 (2007)
12. Mekhaldi, D., Lalanne, D., Ingold, R.: From searching to browsing through multimodal documents linking. In: ICDAR 2005 (8th International Conference on Document Analysis and Recognition), Seoul, pp. 924–928 (2005)
13. Huls, C., Claassen, W., Bos, E.: Automatic referent resolution of deictic and anaphoric expressions. Computational Linguistics 21(1), 59–79 (1995)
14. Kehler, A., Martin, J., Cheyer, A., Julia, L., Hobbs, J., Bear, J.: On representing salience and reference in multimodal human-computer interaction. In: AAAI 1998 workshop on Representations for Multi-modal Human-Computer Interaction, Madison, WI, pp. 33–39 (1998)
15. Matena, L., Jaimes, A., Popescu-Belis, A.: Graphical representation of meetings on mobile devices. In: MobileHCI 2008 Demonstrations (10th International Conference on Human-Computer Interaction with Mobile Devices and Services), Amsterdam (2008)