Improving document retrieval using special characteristics of lecture recording documents

Christoph Hermann
Institute for Computer Science
University of Freiburg
Georges-Koehler-Allee 51, 79115 Freiburg
Germany
hermann@informatik.uni-freiburg.de

ABSTRACT

With the increasing use of lecture recordings, content providers are facing the challenge to make electronic lecture materials both easily accessible and searchable. Therefore powerful search engines need to be implemented that allow users to easily retrieve documents fulfilling their information needs.

While there has been a lot of research in the domain of text search, special characteristics of lecture recording documents have not yet gotten much attention. Lecture recording documents differ from text documents such as papers, scripts or web pages because they usually do not contain running texts but rather listings and enumerations. Additionally, lecture recording documents contain time-based data such as an audio or video stream of the lecturer as well as handwritten annotations. Analyzing these additional data streams leads to an improvement of the search process.

Our novel approach to analyze annotations of lecture recording documents improves document relevance estimation during the search in lecture materials. Hence, technology had been developed to make the contents and special properties of lecture recordings accessible and searchable. We describe key issues encountered during these developments and present experimental results of our search engine which takes into account the special characteristics of lecture recording documents during the indexing process. Searching our archive of over 15,000 files only takes a few milliseconds and enables us to offer a search-as-you-type user interface, query auto-completion and visual browsing.

Categories and Subject Descriptors

H.3.1 [**Content Analysis and Indexing**]: Indexing methods; H.3.3 [**Information Search and Retrieval**]: Search process; H.3.7 [**Digital Libraries**]: Systems issues

Keywords

search, lecture recording documents, annotations, gestures

SoICT 2012, August 23-24, 2012, Ha Long, Vietnam.

1. INTRODUCTION

With the increasing use of computer-based lecture recordings (they have been used for over 25 years[1] and are nowadays widely accepted), content providers (such as universities) are facing the challenge to make those contents easily accessible to their users (students). The vast amounts of data collected over years at universities arises the need to implement powerful search engines that allow users to easily retrieve documents, fulfilling their information needs.

While text search has been in the focus of research so far, special characteristics of lecture recording documents have not yet gotten much attention.

Lecture recording documents differ from text documents such as papers, scripts or web pages because the contained text usually does not consist of running texts but rather listings and enumerations. Their text structure differs a lot from conventional documents [9]. Additionally, lecture recording documents are not plain page-based media, but consist of continuous time-based data such as the audio and the video stream of the person giving the presentation, as well as handwritten annotations that were added by the lecturer. Therefore new methods have to be implemented to improve the search for lecture recording documents.

Research in the field of searching lecture recording documents is primarily focused on the text structure, the audio stream or the text contents. Our intent is to additionally analyze annotations, as well as the layout information of lecture recording documents, in order to improve document relevance estimation during the search in lecture materials.

To make such an analysis possible, new technologies had to be developed so the contents and special properties of lecture recording documents become more accessible and searchable.

Our main contribution and technical developments described in the following sections consists of the following parts:

a) Using a newly developed tool, we are able to access closed-source lecture recording document formats such as *LECTURNITY* or *Camtasia* video to make these contents accessible for search.

b) Analyzing the annotations by using gesture recognition allows the boosting of contents to improve document relevance estimation.

[1]First experiments with lecture recordings have already been conducted in 1986 [13].

c) Combining this new approach with previously developed tools allows the creation of a very fast and powerful search engine including auto-completion, search-as-you-type and visual hit highlighting.

2. RELATED WORK

Only few publications concentrate on annotations in lecture recording documents.

In 2004 Richard Anderson, who also led the development of classroom presenter (one of the first lecture recording tools), analyzed lecture recording documents regarding the interaction between the different media streams as well as usage patterns of those documents [1, 2].

2.1 Use of annotations in lecture recording documents

Anderson et al. mentioned, that a detailed analysis of their archive of lecture recording documents revealed usage patterns occurring in the annotations [2]. The following three aspects were described:

- the use of annotations in analogy of physical gestures (called *attentional marks*)
- the difference between the ephemeral display of annotations during the lecture and the static representation after it has been recorded
- the spare use of the annotation tools of the recording software by lecturers.

The authors named *attentional marks* as handwritten annotations, which are used in analogy of physical gestures of a lecturer. They are used to create a relation between the spoken text and the slides presented during the talk. These marks consist of arrows, circles, underlines, check boxes, check marks, parentheses, dots and other symbols. Using these annotations, the lecturer can combine or isolate elements on the slides or highlight certain parts of it [2].

Pointing at research by McNeill, they further commented that physical gestures are only used when the lecturer is talking [15]. Other properties of physical gestures identified by McNeill seem to apply in an almost identical manner to digital annotations and gestures.

Anderson et al. also explored the effect of annotations on the students attending a course where lectures were recorded [2]. The results of those investigations showed that 414 of the 479 surveyed students (over 86%) direct their attention to the highlighted parts of the slides which therefore exert a direct influence on the learning process.

Those results are particularly interesting, since they expose a direct relation between the slides, the gestures and the users. Therefore special attention needs to be paid to those annotations during the process of relevance estimation when searching lecture recording documents.

Additionally, Anderson et al. mentioned that there are annotations that tell the listeners that some of the contents are not relevant or less relevant than others. Together with strike through annotations these should significantly diminish the relevance of the related parts of the lecture.

Anderson et al. further described a detailed analysis of annotations in lecture recording documents [1]. They mentioned, that annotations used during lecture recordings can be clustered in three main categories: textual, diagrammatic and attentional marks.

Diagrammatic annotations are sketches and scribbles drawn onto the presented slides.

Textual annotations are basically handwritten notes that are added to rectify, complement or develop contents like formulas, hypothesis or proofs. The authors analyzed those handwritten annotations using handwritten text recognition software.

They state that attentional marks make up more than 50% of the total annotations in lecture recording documents and are mainly used to link the speech of the lecturer to the presented slides. Those marks are used to draw the listeners attention to the current topic, to emphasize the relevance of certain parts or to link contents together.

The authors analyzed these three kinds of annotations with the goal to be able to cluster annotations in one of the three categories. Mohamed solved this problem by analyzing time features of annotations (*lead* and *lag times*) [16].

Anderson et al. reported that most of the annotations only consist of a few strokes: circles, underlines and such [1]. Similarly to Mohamed, they described a temporal and geometrical proximity of gestures consisting of more than one stroke.

Anderson et al. and our approach can be regarded as similar, since one of our goals is to classify attentional marks, identify the related text and boost the relevance of the highlighted text regarding the type of annotation used. All of this can be used to improve the precision in the document retrieval process while searching large databases of lecture materials.

2.2 Annotations supporting searching in documents

Literature research is one of the most common cases of annotating documents. During the process of a literature research certain documents can be downloaded, printed and read by users. Usually parts of these printed documents are then highlighted using a pen. If relevant text parts are found, users often search for similar contents in online databases.

Golovchinsky et al. note that annotations can represent a readers interest in certain parts of text documents [5]. They explored possibilities to support users during the document retrieval by using annotations in text documents to find other similar documents. Instead of printing and highlighting on paper the authors wanted to avoid this media disruption by offering a software (*Xlibris*) that enables the users to read, highlight and search documents from within one environment. Within *Xlibris*, annotations are used to derive new search queries from parts of documents that are annotated by users.

Obviously different annotations within a document lead to different weighting of the text snippets during the search. If text has multiple annotations, such as a mark annotation on the border as well as encircled text, only the most precise annotation is considered important for search. The authors stated that compared to a simple boolean relevance feedback mechanism, this kind of relevance feedback improves the search experience [5].

Our work considers multiple annotations without disregarding any information. In the example given above, the relevance of the encircled text would be even more emphasized compared to the whole paragraph.

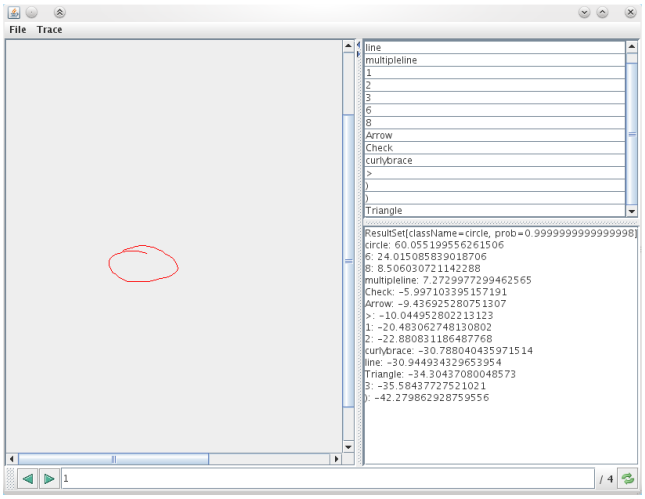

Figure 1: Software for the analysis of freehand annotations

3. ALGORITHMIC METHODS

To improve the relevance weighting in the document retrieval process using annotations, 143,176 annotations from lecture recording documents were gathered, which were retrieved from lecture recording documents from the *Electures-Portal* of the University of Freiburg [7, 8]. These annotations were analyzed by using a software especially developed for this purpose (see Fig. 1).

During this process eight classes of annotations could be identified (see Fig. 2) which can be used to gain information about the relevance of text in lecture recording documents:

- *ellipses* (circles)
- *rectangles* (boxes)
- *arrows*
- *curly braces*
- *wavy lines*
- *multiple lines*
- *lines*
- *check marks*

Ellipses clusters all kinds of circles and round annotations. *Rectangles* comprise handwritten boxes as well as precise rectangles created using the authoring tools. *Arrows* can point in different directions where the end of the arrow is usually the end of the gesture. *Curly braces* can be drawn horizontally or vertically and are used to embrace multiple elements (for example enumeration items) or to underline text (usually followed by a [handwritten] explanation). *Wavy lines* as well as *multiple lines* and *lines* are used to underline or strike through texts. They are also used to create diagrammatic annotations. *Check marks* are often employed to check enumeration items, to highlight single elements or in general to draw users attention.

As a next step, the annotations occurring in the lecture recording documents have to be classified to one of the eight identified gesture classes.

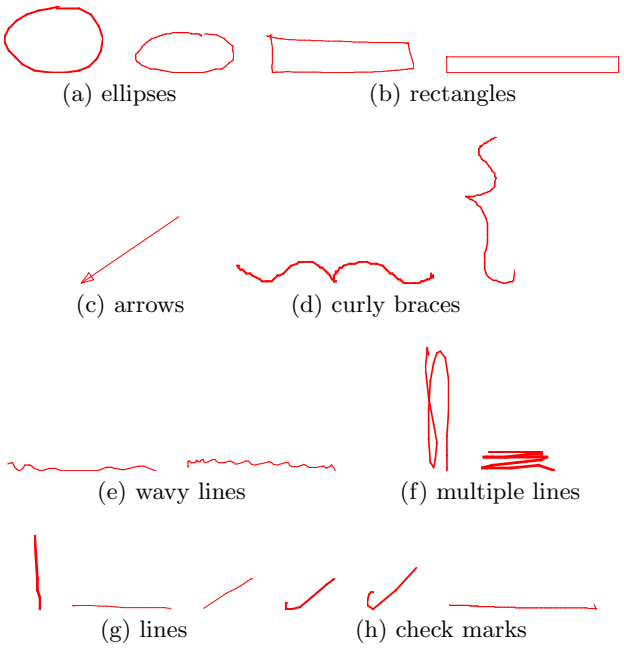

Figure 2: Some of the annotations belonging to the eight relevant gesture classes

3.1 Gesture classification

There are several algorithms which can be employed for gesture classification. Gestures are usually defined as a set of triples where each triple belongs to one point of the gesture:

$$F_{\text{Gesture}} = \{< x_1, y_1, ts_1 >, \ldots, < x_n, y_n, ts_n >\}$$

with the following values for one point i:

x_i x-coordinate

y_i y-coordinate

ts_i time-stamp

Using those triples gesture classification algorithms can be used to cluster the annotations into the above mentioned classes.

Most gesture classification algorithms fit into the following two categories:

1. Algorithms that only take into account the geometric x and y-coordinates, but not the timestamps of the gesture points.
2. Algorithms that are based on feature extraction.

On this basis, several experiments were run on our data, using different gesture recognition algorithms like grid-based algorithms, the SiGer algorithm or the Rubine algorithm.

Grid-based algorithms.

Grid-based algorithms project the gestures on a predefined grid (after scaling and rotation if necessary) (see Fig. 3). Based on the intersected areas, the algorithm decides whether a gesture belongs to a certain class or not. Worth described how such an algorithm was implemented in the open source application *xstroke* to implement a gesture recognition on the whole surface of a computer screen [19].

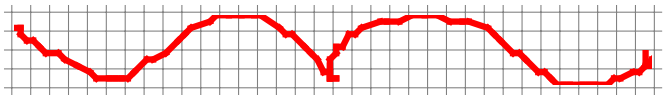

Figure 3: Schematic representation of grid-based gesture recognition

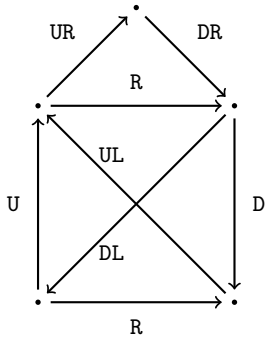

Figure 4: Gesture explaining the SiGer algorithm

SiGer algorithm.

Another very simple algorithm is Swigarts SiGer algorithm [18]. This algorithm uses a vector representation of the gesture which represents the orientation of the gesture at every point of the strokes of a gesture. For every orientation there is a corresponding string representation (e.g. left=L, right=R, up=U, down=D). More complex gestures like the the one in Fig. 4 can be described as a combination of these: `U,UR,DR,D,UL,R,DL,R`. The gestures are then recognized, using a combination of this string representation, the start and end coordinates and the distances between single points of a gesture.

Other algorithms additionally use the timestamps of the points or other features like the maximum or the average speed of the gestures to improve the quality of the classification. One of the most well known algorithms in this area will be briefly described in the following section.

The Rubine algorithm.

The Rubine algorithm named after its author Dean Rubine uses feature extraction to classify gestures [17].

As a first step in the classification process the following thirteen features $F_1, ..., F_{13}$ are extracted from a gesture:

F_1 cosine of the initial angle

F_2 sine of the initial angle

F_3 length of the bounding box diagonal

F_4 angle of the bounding box diagonal

F_5 distance between the first and left point

F_6 cosine of the angle between the first and the last point

F_7 sine of the angle between the first and the last point

F_8 total length of the gesture

F_9 total angle

F_{10} absolute value of the total angle traversed

F_{11} squared sum of the value of those angles

F_{12} maximum speed

F_{13} duration it took to draw the gesture

Once provided with the set of features extracted from a given gesture, it is classified using a linear discriminator that decides to which of the gesture classes the current gesture belongs. On top, there are two measures (a minimum similarity measure, as well as the Mahalanobis distance [12]) that are calculated to reject a gesture if the classification is ambiguous.

One of the main advantages of the Rubine algorithm is that it is scale and rotation invariant if correctly implemented. A major drawback of Rubines algorithm is that the Features F_{12} and F_{13} can only be computed if timestamps ts_i are available for all gesture points i. Rubine states that if this information is not available, those two features can simply be omitted.

Our research shows that omitting those two feature heavily decreases the quality of the recognition. Therefore it is necessary to extract the time information of the gestures from the lecture recording documents.

Some file formats such as LECTURNITY lecture recordings do not contain the data necessary for computing those values. Using the aforementioned tools, the missing time information of the annotations can be reconstructed using other objects display data.

During our experiments with different algorithms it crystallized that Rubines algorithm best fits our purpose if all thirteen features are considered during the recognition process. Author-specific training of our gesture data additionally improves the gesture recognition process.

In the following sections, we examine how gesture recognition can be used to improve searching in lecture recording documents.

3.2 Search and indexing process

Indexing is often described as the process of collecting, parsing and storing data, in order to facilitate fast and accurate information retrieval (search). Fig 5 shows a diagram of the search and indexing process. Every step is shortly depicted followed by an explanation how the results of our gesture analysis were integrated into the search process using the open source search engine Apache Lucene [14].

Acquisition.

Documents first have to be converted into a readable format to make the raw data (lecture materials) accessible for search. File types such as binary document formats of Microsoft Office or *LECTURNITY* have to be converted to be able to extract text, meta-data and special properties of these documents.

In our case the result of the acquisition process is a common file format for all different kinds of lecture materials which is called Extended HTML Format. This document format (which is an XML file based on some sort of HTML notation, therefore the name) includes the text information of the raw data, additional meta-data and properties extracted from the original documents. Listing 1 shows an example of such an Extended HTML document. The attribute `concatenatedString` labels text parts that were recombined during the text extraction process (see section 3.3).

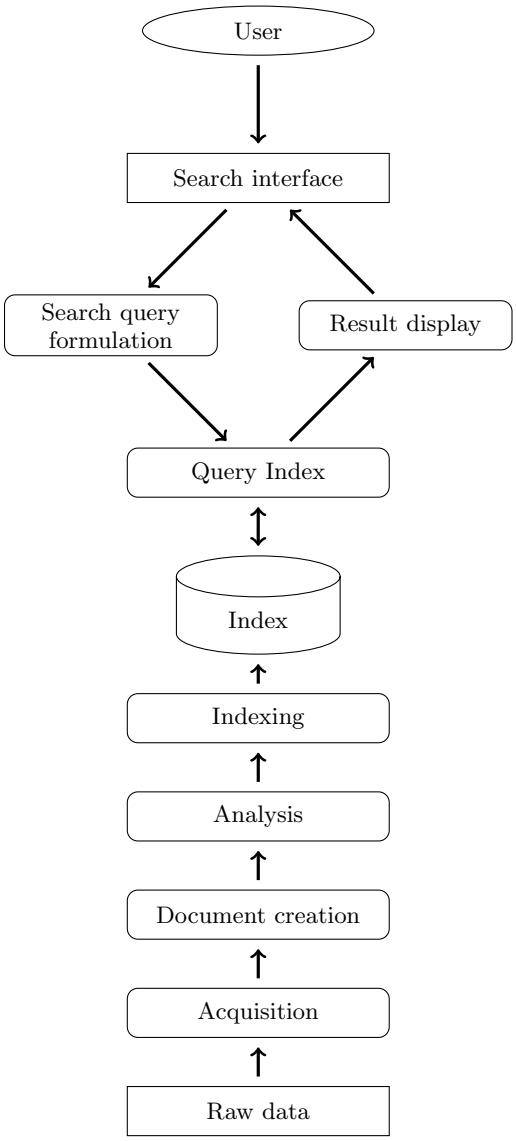

Figure 5: Search and indexing process

Listing 1: Excerpt of a Extended HTML document with the most important additional attributes

```
1 <div class="page" pageNumber="1">
2 <div class="article" articleNumber="1">
3 ...
4 <span class="writeString" xCoord="311.0"
    yCoord="732.0" width="272.0" height="
    16.0" fontSize="7.9701"
    concatenatedString="true"> research
    interests include Algorithms and Data
    Structures, Computational Geometry,
    Multimedia Systems and the use of
    computers for</span>
```

Document creation.
During the next step, these contents have to be transformed into a format indexable by the search engine. For the open source search engine *Apache Lucene* this format is called `Document`. `Documents` contain structured data which can be indexed by the search engine. They comprise several fields that may contain meta-data or the text contents of the original files.

Analysis.
The `Documents` are processed by an analyzer. An analyzer builds streams of tokens. It represents a policy for extracting index terms from text. This step is often referred to as tokenization, too.

Indexing.
The indexing step adds the result of the analysis to the index in order to make it searchable.

Search interface.
The search interface is the part of the search engine that the users get to see. Often web interfaces are used to enable the user to input search queries. The search results are then displayed in the same interface.

Search query formulation.
The search query entered by the user is parsed into a special format used to query the search index. This enables the search in different fields of the search index, stop word filtering, language detection or other processes that are similarly applied in the analysis, like stemming for example.

Query Index.
The rewritten search query is then applied to the index to retrieve the most relevant documents matching the search query.

Result display.
This step is used to reformat the results to be displayed in a user-friendly way. The search results can be filtered or complemented by additional data which can be retrieved from additional databases.

3.3 Data acquisition from lecture recording documents

When trying to extract data from lecture materials, inevitably a few problems arise:

a) First, not all lecture materials are available in document formats that are easily accessible. For example the *LECTURNITY* File format is a binary encrypted format, that is not publicly documented.

b) Other file formats like PDF Documents are built using word processors like *LaTeX*, *LibreOffice Writer* or *Microsoft Word* which split up words at the end of the line. This makes searching for these words impossible unless those split words are recombined.

c) Last but not least the necessary information (for example timestamps of the gestures) is not directly available in all lecture recording documents (e.g. *LECTURNITY*) or has to be extracted from the video data (e.g. with *Camtasia* recordings).

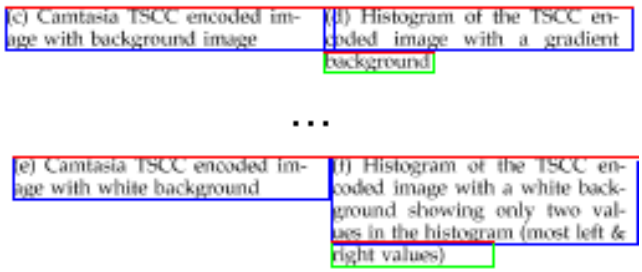

Figure 6: Screen capture of the result of the analysis of PDF documents

To address issue a) those document formats were thoroughly reverse-engineered to be able to access the documents contents. Using the *LECTURNITY* player, written in the Java programming language, two ways to access the required information were found.

One technically difficult approach was to closely monitor and debug the execution of the *LECTURNITY* player while using it to replay the recordings. Using this technique enabled us to decipher the binary file format up to a point where the internal file contents could be extracted using a tool especially written for this purpose.

Another opportunity, was to reuse the *LECTURNITY* player as a library using the Java Reflection API. This approach has the advantage that whenever the file format of the binary files change, it is not necessary to start the reverse engineering process over again, since the player still is able to play the files.

The unencrypted contents of the binary *LECTURNITY* files are very similar to the original file structure of the AOF File format documented in [3, 4]. Changes in the file format mainly consist of things like utf8 support, support for different fonts and additional formatting styles.

To address problem b), another new tool was implemented which allows the extraction of detailed text information from PDF documents, including words split during the typesetting process.

To achieve the recombination a set of rules was defined for the two main languages German and English, allowing to recombine the split words into the original words. One of the rules is, for example, that a hyphen that has been introduced by a word processor is usually not followed by capital letters (such as abbreviations "email" in German which is spelled "E-Mail"). Another special case are combined words ("editor-in-chief") that can be detected by using language dictionaries. By considering the following lines of text, most of the splits can be detected and recombined using these rules.

Fig. 6 shows a screen capture of the results of the analysis of one PDF document. Lines that end with words which were split are recombined in the same box with the following lines. Additionally, this tool allows us to extract both the exact x and y coordinates, and the length and with of the bounding boxes. This can be used later to implement an advanced hit highlighting in the search results.

The results of this analysis are then included in the Extended HTML format as shown in listing 2.

The problem mentioned in c) was addressed by combining the playback time information with the represented objects

Listing 2: Excerpt of a document after conversion to the Extended HTML format

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <html xmlns="http://www.w3.org/1999/xhtml">
3  <head>
4  <title>Electures-Wiki - Engaging Students
       to Work Actively with Lecture
       Recordings</title>
5  </head>
6  <body>
7  <div class="page" pageNumber="1">
8  <div class="article" articleNumber="1">
9  <span class="writeString" xCoord="70.971985
       " yCoord="78.72998" width="362.45624"
       height="23.623379" fontSize="23.9103"
       concatenatedString="false"> Electures-
       Wiki - Engaging Students to Work</span>
10 <span class="writeString" xCoord="133.55598
       " yCoord="106.62598" width="319.02917"
       height="23.623379" fontSize="23.9103"
       concatenatedString="false"> Actively
       with Lecture Recordings</span>
11  ...
12 <span class="writeString" xCoord="311.0"
       yCoord="732.0" width="272.0" height="
       16.0" fontSize="7.9701"
       concatenatedString="true"> research
       interests include Algorithms and Data
       Structures, Computational Geometry,
       Multimedia Systems and the use of
       computers for</span>
13  ...
14 <span class="writeString" xCoord="311.97296
       " yCoord="742.773" width="39.101288"
       height="7.8744597" fontSize="7.9701"
       concatenatedString="false"> educational
        purposes.</span>
15 </div>
16 </div>
17 </body>
18 </html>
```

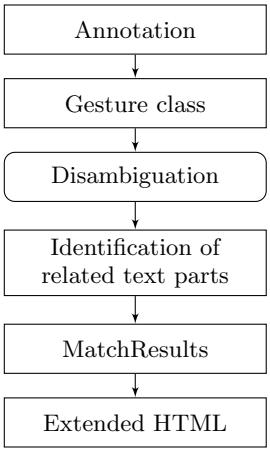

Figure 7: Flowchart of the integration of the gesture recognition results into the Extended HTML format

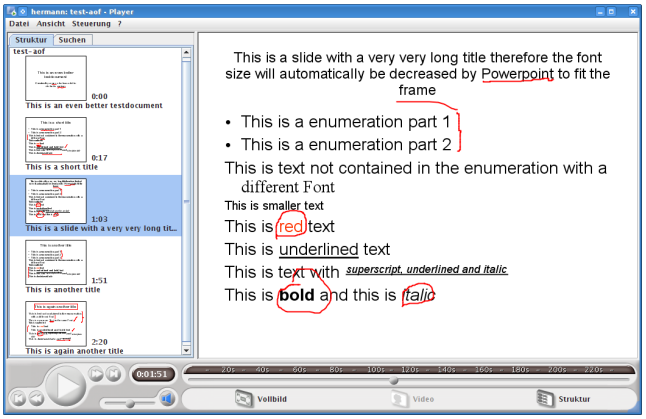

Figure 8: Screen capture of a lecture recording document where parts of the text are highlighted by annotations

during that time, allowing a reconstruction of the time information necessary to apply Rubines algorithm to the strokes and gestures contained in *LECTURNITY* recordings.

3.4 Integration of gesture recognition data in the Extended HTML format

As a next step the results of the gesture recognition have to be integrated into the Extended HTML format during the acquisition process.

Fig. 7 shows a flowchart of the integration.

After the gestures have been successfully classified using the Rubine algorithm an additional problem arises. Some of the gestures have no direct relation to a specific part of the text and could possibly be related to different parts of the presented slide. Fig. 2d,2f and 2g are examples of such gestures. Therefore a disambiguation had to be implemented to identify which part of the text is related to the currently processed gesture.

This ambiguity comes mainly from the fact that Rubines algorithm is rotation invariant. Of course this could be fixed by adding additional features in the recognition process to resolve this ambiguity as it has been done by Mohamed [16]. Every feature added to the set of extracted features adds to the complexity of the algorithm which is not necessary in our case. Using the coordinates, as well as the bounding box of the gesture the related text can be reliably identified.

The identified gestures then have to be analyzed in order to find out which parts of the related text are affected. This identification can be quite difficult. Fig. 8 shows a screen capture of a test lecture recording document, where this problem can be found.

It is quite clear that words like *Powerpoint*, *frame* or *red* are highlighted, but how about the word "*and*" in the last line? For an automated analysis this can be particularly problematic.

Therefore a three step approach was developed to encounter this problem:

1. First, all affected text parts are identified.
2. Then, relevant and affected words are detected.
3. In the last step the correct matches are created and the text relevance is recalculated using the gestures weight.

Step 1) can be easily achieved using the coordinates of the gesture, its bounding box and the coordinates and bounding boxes of the text elements.

Starting from there, the relevant words from 2) are detected, using rules specific to the language of the contents. For European languages like German and English the complete words can be easily identified by spaces and punctuation marks. Then, depending on the length of the affected characters compared to the length of the whole word, our system decides if the word still needs to be processed. Other measures that can be used additionally are stop word detection and a co-occurrence analysis.

The last step integrates the results of the analysis into the Extended HTML format by adding boost values (see listing 3), taking into account multiple annotations as described in section 2.2.

3.5 Data analysis and search

The contents of the Extended HTML format are then interpreted during the indexing process. The calculated boost values and additional meta-data such as the x and y coordinates of the text are stored within a so-called payload in the index. The page numbers or the time-stamps where the current text occurs in the lecture recording document are also stored. By adding these values as a payload we are able to retrieve and use them during the evaluation of a search query and when displaying the search results to the users.

Documents containing search terms which are underlined or encircled by annotations are preferred during the search compared to documents containing no annotations, but having the same relevance regarding the text contents. This is achieved by using the boost values to raise or decrease the relevance of the annotated text parts.

Therefore, when comparing three different kind of lecture materials, for example a PDF version, a *PowerPoint* version and a lecture recording of the same slides, our search engine will return the lecture recording document first if the terms of the search query are highlighted in the lecture recording document.

Listing 3: Integration of multiple annotations into the Extended HTML format

```
1 <p class="paragraph" width="516" height="34
    " xCoord="43" yCoord="239">
2 <span class="writeString" xCoord="43"
    yCoord="239" fontSize="32">
3 <span class="boost" boostScore="1.25"
    boostText="This␣is␣even␣more␣Text␣in␣
    the␣same␣Font" boostType="de.freiburg.
    iif.gesture.annotationfamily.
    MarkAnnotation">This is even more </
    span>
4 <span class="boost" boostScore="1.25"
    boostText="This␣is␣even␣more␣Text␣in␣
    the␣same␣Font" boostType="de.freiburg.
    iif.gesture.annotationfamily.
    MarkAnnotation">
5 <span class="boost" boostScore="1.5"
    boostText="Text" boostType="de.freiburg
    .iif.gesture.annotationfamily.
    UnderlineAnnotation">Text</span> in the
     same Font</span>
6 </span>
7 </p>
```

4. EXPERIMENTAL RESULTS

Experimental results using our search engine implementation shows that we are able to achieve better results than a default implementation that does not take the gesture analysis into account.

By using standardized test data sets, the quality of the search results can be identified. In the domain of information retrieval benchmarks the two measures *precision* and *recall* are used.

For the assessment of our search engine, we are using the standardized method used at TREC[2] which is based on ground truth files. Two main files are used, a topics file and a query relevance file (qrel).

The topics file contains information about the search queries given to the search engine. In addition to the query itself, it includes a detailed description of the query in natural language.

The query relevance file comprehends ground truth information about the data stored in the index in relation to the query in the topics file. The format of this file is very simple. Besides an `ID` pointing to a specific entry in the topic file, it contains file names of files in the index as well as a boolean relevance score.

An automated evaluation of the search engine is performed using these two files. For every query in the topics file this search query is fed to the search engine and its results are compared to the ground truth information from the query relevance file. Files from the index supplying relevant results for the current query are denoted by a boolean *one*, irrelevant results are marked with *zero*.

Creating such a query relevance file is a tedious process and has to be done very carefully. The quality of the evaluation is directly influenced by the quality of the query relevance file. Therefore our ground truth files contain manually crafted relevance information.

[2]http://trec.nist.gov/

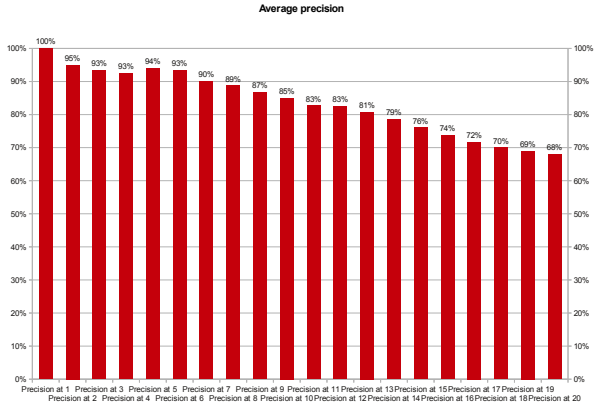

Figure 9: Average values of the precision (precision at one until precision at 20) of our search engine implementation.

For our experiments 235 lecture materials from the computer science domain were selected from our archive (the *Electures-Portal*) and loaded into a search index. Then our implementation of the search engine was assessed, using the method described above.

We calculated the precision at position one of the search results until position 20. Fig. 9 shows the averages of these values of our experiments.

At best, the precision and recall should always be 1.00. It shows that starting from the sixth position, the precision of our implementation decreases more and more. In average a precision of 0.839 and a recall of 0.901 can be achieved. In the value series shown here, the precision at one is always 100%, starting at position two (95.5%), the precision decreases down to around 68% at position 20. The calculated average recall is very high (over 90%).

5. IMPROVEMENTS TO THE USER INTERFACE

To improve the user experience when using our search engine, previously developed technologies such as *aofconvert* [6] were integrated in order to improve the display of the search results. Hürst mentioned that when using a search engine, the browsing through the search results and especially the visual appearance has a very high significance [10].

Using the software *aofconvert* enables us to create screen captures from almost every file listed in our *Electures-Portal*. These screen captures can then be used to improve the illustration of the search results to enable visual browsing in the search results. Fig 10 shows a screen capture of the results of a search.

In addition to highlighting the search terms in the search results we are also able to implement a direct replay of the lecture recording documents. Using the same technology as in our *Electures-Wiki* (see [6]), it is possible to visually reference the documents and directly start the replay in time at the precise moment where the search terms were found

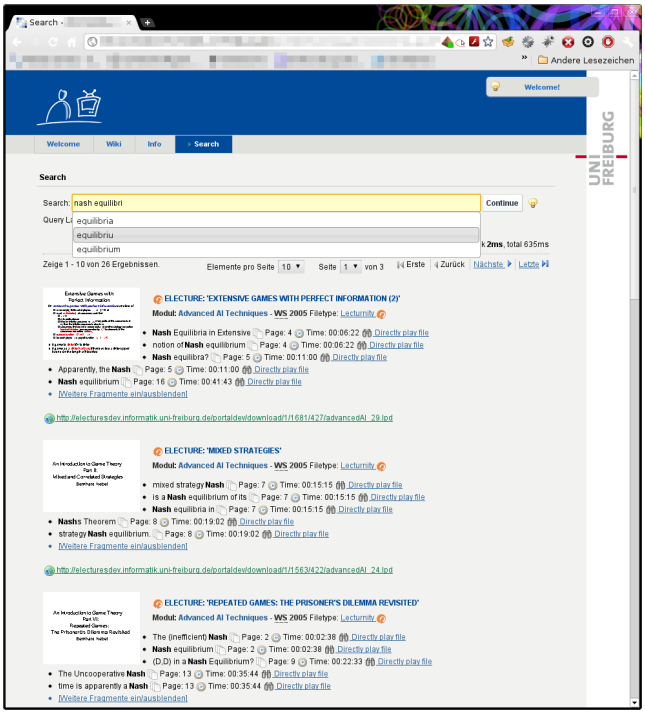

Figure 10: Displaying the results of the search query "nash equilibria" processed by our search engine implemented in the next generation prototype of the *Electures-Portal*

using the payloads stored in our index.

Processing a search query only takes a few milliseconds. Therefore it is possible to start the search query already during the input while displaying the search results almost in real-time to the user. This kind of search is often referred to as "search-as-you-type". Using AJAX calls, the displayed web page is constantly updated with new search results without having to reload the whole new page.

Additionally, the user is supported in his query formulation by suggestion of search terms, matching the prefixes of the word, he is currently typing. Fig. 10 shows the auto-completion and the visual display of the search results.

6. CONCLUSIONS

By using reverse engineering and the Java Reflection API, we were able to make the contents of *LECTURNITY* lecture recording documents accessible for search. Additional meta-data and special properties of lecture recording documents were extracted.

Eight gesture classes were considered to be relevant for improving the relevance estimation during the indexing process of lecture recording documents. The introduction of the Extended HTML Format allows to include the results of the gesture classification and additionally extracted properties into the index. In this way data can be reused during the search process as well as improving the information and visual appearance of the search results.

An efficient implementation of our search which leads to very short search times of only a few milliseconds, enables the implementation of search-as-you-type and auto-completion even during the search query formulation.

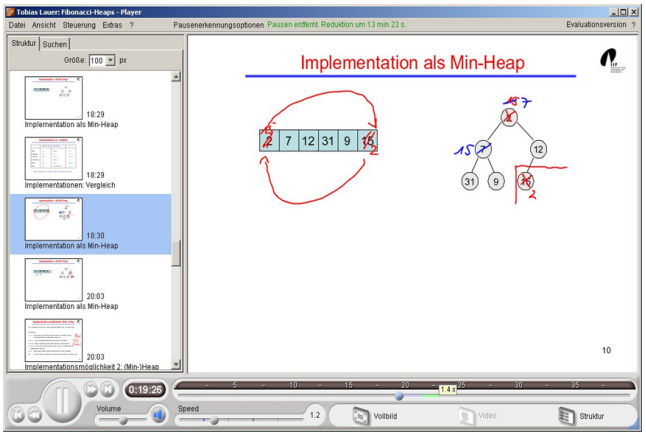

Figure 11: Screen capture of a *LECTURNITY* prototype with pause-detection and the possibility of a faster replay of the lecture recording documents

The integration of previously developed technologies like *aofconvert* allows us to implement the direct replay of the lecture recording documents in time, namely at the precise moment where the search term appears in the recordings.

7. OUTLOOK AND FUTURE WORK

Searching lecture recording documents could be further improved by making additional screengrabbing-based videos (e.g. *Camtasia* recordings) accessible for search. Ziewer describes how an automated analysis of screen recorded data (mainly using optical character recognition (OCR)) allows to access the text contents of these videos [20]. Since we already implemented a *Techsmith Camtasia* decoder which is able to extract images of the slides from the *Camtasia* recordings [6], only OCR needs to be integrated into our implementation.

Search in screengrabbing-based videos could be changed for the better by applying our gesture analysis to these videos. To make this possible, annotations need to be extracted from the captured image data. This could be implemented by detecting small color changes between different video frames and retracing the path of this changes to recreate the original gestures. To achieve this, a similar technique as in [6] could be used, where a histogram analysis is used to detect slide transitions.

Since there is a direct relation between the explanations of the lecturer and the annotations, investigations should be done to find out if this relation can be used to automatically adapt the replay speed of lecture recording documents. Ideas on how to implement a variable replay speed for lecture recording documents were already presented in a publication by **?** [**?**].

Fig. 11 shows a screen capture of a *LECTURNITY* player capable of changing the replay speed. By further analyzing the contents of the lectures, the replay speed could be automatically adapted. Using such a player could improve the learners understanding of the contents by slowing down more difficult sections or speeding up less important parts of the lecture recording documents.

References

[1] R. Anderson, C. Hoyer, C. Prince, J. Su, F. Videon, and S. Wolfman. Speech, ink, and slides: the interaction of content channels. In *Proceedings of the 12th Annual ACM International Conference on Multimedia*, MULTIMEDIA '04, pages 796–803, New York, NY, USA, 2004. ACM. ISBN 1-58113-893-8. doi: 10.1145/1027527.1027713. URL `http://doi.acm.org/10.1145/1027527.1027713`.

[2] R. J. Anderson, C. Hoyer, S. A. Wolfman, and R. Anderson. A study of digital ink in lecture presentation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '04, pages 567–574, New York, NY, USA, 2004. ACM. ISBN 1-58113-702-8. doi: 10.1145/985692.985764. URL `http://doi.acm.org/10.1145/985692.985764`.

[3] F. Augenstein, T. Ottmann, and J. Schöning. Ein typ- und regelgesteuertes autorensystem. In *Hypertext/Hypermedia, Tagung der GI, SI und OCG*, pages 25–33, London, UK, 1991. Springer-Verlag. ISBN 3-540-54145-4. URL `http://portal.acm.org/citation.cfm?id=647947.742201`.

[4] C. Bacher. *Das Authoring on the Fly System*. PhD thesis, Albert-Ludwigs-UniversitÃd't Freiburg, Mai 2000.

[5] G. Golovchinsky, M. N. Price, and B. N. Schilit. From reading to retrieval: freeform ink annotations as queries. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '99, pages 19–25, New York, NY, USA, 1999. ACM. ISBN 1-58113-096-1. doi: 10.1145/312624.312637. URL `http://doi.acm.org/10.1145/312624.312637`.

[6] C. Hermann and T. Ottmann. Electures-wiki – toward engaging students to actively work with lecture recordings. *IEEE Transactions on Learning Technologies*, 4:315–326, Oktober 2011. ISSN 1939-1382. doi: 10.1109/TLT.2011.18. URL `http://doi.ieeecomputersociety.org/10.1109/TLT.2011.18`.

[7] C. Hermann, W. Hürst, and M. Welte. The eLecture-Portal: An advanced archive for lecture recordings. In *Informatics Education Europe*, IEE, Oktober 2006.

[8] C. Hermann, W. Hürst, and M. Welte. Das eLecture-Portal der Universität Freiburg. In *Workshop Effiziente Erstellung von E-Learning Content, DeLFI 2006*, LNI, September 2006.

[9] W. Hürst. Suche in aufgezeichneten Vorträgen und Vorlesungen. *Tagungsband der 1. e-Learning Fachtagung Informatik*, 37:27–36, September 2003. ISSN 3-88579-366-0.

[10] W. Hürst. *Multimediale Informationssuche in Vortrags- und Vorlesungsaufzeichnungen*. PhD thesis, Albert-Ludwigs-UniversitÃd't Freiburg, Februar 2005.

[11] W. Hürst. Audio-Browsing mit elastischen Schiebereglern im E-Learning. *DeLFI 2006, Tagungsband der 4. e-Learning Fachtagung Informatik*, 87:279–290, September 2006.

[12] P. C. Mahalanobis. On the generalised distance in statistics. In *Proceedings National Institute of Science, India*, volume 2, pages 49–55, April 1936. URL `http://ir.isical.ac.in/dspace/handle/1/1268`.

[13] R. Mason. From distance education to online education. *The Internet and Higher Education*, 3(1-2):63 – 74, 2000. ISSN 1096-7516. doi: 10.1016/S1096-7516(00)00033-6. URL `http://www.sciencedirect.com/science/article/B6W4X-430XMJH-5/2/f6a918ff13ff5b143cfae6ba4c9091ce`.

[14] M. McCandless, E. Hatcher, and O. Gospodnetic. *Lucene in Action, Second Edition: Covers Apache Lucene 3.0*. Manning Publications Co., Greenwich, CT, USA, 2010. ISBN 1933988177, 9781933988177.

[15] D. McNeill. What gestures reveal about thought. In *Hand and Mind*, chapter 9. The University of Chicago Press, 1992.

[16] K. A. Mohamed. *Concepts and solutions for efficient handling of the digital ink*. PhD thesis, Albert-Ludwigs-Universität Freiburg, März 2009. URL `http://www.freidok.uni-freiburg.de/volltexte/7361/`.

[17] D. Rubine. Specifying gestures by example. *SIGGRAPH Computer Graphics*, 25(4):329–337, 1991. ISSN 0097-8930. doi: 10.1145/127719.122753. URL `http://doi.acm.org/10.1145/127719.122753`.

[18] S. Swigart. Easily write custom gesture recognizers for your tablet pc applications. Technical Report 1, November 2005. URL `http://msdn.microsoft.com/en-us/library/aa480673.aspx`. Online.

[19] C. D. Worth. Xstroke: Full-screen gesture recognition for x. In *Proceedings of the USENIX '03*, pages 187–196, 2003.

[20] P. Ziewer. Navigational indices and full text search by automated analyses of screen recorded data. In J. Nall and R. Robson, editors, *Proceedings of the World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education*, E-Learn, pages 3055–3062, Washington, DC, USA, 2004. AACE. URL `http://www.editlib.org/p/11112`.