In Proceedings of International Symposium on Digital Libraries, pp. 242-249, August 1995.

A Semantic Movie Indexing Scheme*

Rynson W. H. Lau Hong V. Leong Antonio Si
Department of Computing
The Hong Kong Polytechnic University
Kowloon, Hong Kong
Phone number: (852) 2766 - 7241
Fax number: (852) 2774 - 0842
Email: {cswhlau, cshleong, csasi}@comp.polyu.edu.hk

Abstract

Current content-based segmentation techniques for digitized movies are based on low level syntactic characteristics of individual movie frame such as color, texture, shape, and motion transition. While these techniques might work well for detecting physical commonalities among movie frames, they fall short in correlating with the semantic content of the frames. Performing transformations such as Discrete Cosine Transform only exploits its spatial content, but not the semantics. A *semantic mismatch* exists between what users perceive and what are actually used to characterize the movies. In this paper, we introduce a novel indexing scheme that supplements each movie frame with high level semantic description, and illustrate the idea and concept via intuitive examples. Users can thus initiate queries against the movie using high level textual descriptions rather than low level syntactic features, providing a faster response and a more natural interface.

Keywords: Multimedia Databases, Digital Libraries, Indexing

1 Introduction

With the advent in computing technologies in processing digitized audio and video data, integrating multiple "media" of data within an application becomes a reality. Incorporating so-called "multi-media" data in an application involves a time-consuming process of identifying and retrieving the appropriate images, audio or video segments from the multi-media database. Effective multi-media retrieval systems must be developed that will identify related multi-media data units and retrieve the appropriate ones for a user according to his/her requirements, in a way similar to database management system works for literal data today.

*This research is funded by The Hong Kong Polytechnic University Central Grant numbers 340/115 and 353/029.

Unlike literal data, data items of different media have their own characteristics. It is unlikely that a general purpose multi-media retrieval system could be developed for all different kinds of media; rather, a separate system should be developed for individual kind of multi-media data, tailored to its characteristic, and combined in a loosely-coupled manner, in much the same way as a federated database [6]. In this paper, we dedicate ourselves to primarily the type of video data. In this context, to support queries on a video stream, a video retrieval system in general will parse the video stream and execute a sequence of tasks [10]:

1. *Segmentation*: A video stream is usually partitioned into several video segments, each of which consists of a sequence of video frames. Ideally, all video frames within a segment are content-related, e.g., the frames showing the movement of a person from the left to the right side of the screen.
2. *Indexing*: Each video segment is tagged with a semantic description describing the content of the video segment.
3. *Query Processing*: Queries originating from users are evaluated against the index of the video stream. Relevant video segments or even video frames are retrieved and presented to users. In particular, two types of queries can be differentiated in this context:
   (a) A user should be able to identify movies of certain content. A sample query of this type would be:

   Return all movies about Star War.

   (b) A user should also be able to locate individual movie segment or frame within a movie that are of particular content. A sample query of this type would look like:

   Return all frames or segments about Darth Vader.

It is important to note that the effectiveness of query processing is confined to the effectiveness of the indexing function. Without a properly built index, queries supported by the system will be very much limited.

Existing segmentation techniques for digitized video are based on low level syntactic characteristics of individual video frame such as color, texture, shape, and motion transition [10]. While these techniques work quite well for detecting physical commonalities among video frames, they fall short in correlating with the semantic content of the frames. Exploiting the spatial information by transforming a video image from the time domain to the frequency domain does not offer additional value, since the spectrum has little relation to the content, or the semantics, of a video frame. Consequently, the indexing functions in contemporary proposed video retrieval systems are either not supported very well or are simply performed manually. Query processing, likewise, is supported poorly.

The main difficulty in supporting the indexing function is due to the distinction between what users actually perceive and what are actually used to characterize and represent the video data. As mentioned, video data is characterized and represented according to its low level syntactic characteristics while users perceive the video using much higher level concepts. Such a distinction in perception is recognized as *semantic mismatch*. In this paper, we describe the design of a video retrieval system for a special kind of digitized video data, digitized movies. This system is called **S***emantic* **M***ovie* **I***ndexing* **S***ystem* (*SMIS*). While the SMIS supports all the three functional capabilities described above, namely segmentation, indexing, and query processing, this paper will focus on the indexing and query processing functionalities since these functions are poorly supported in systems proposed in the literature.

The remainder of this paper is organized as follows. Section 2 is dedicated to a brief survey on existing techniques for video indexing. In Section 3, a high level architecture of SMIS is described. In Section 4, we describe our movie indexing scheme in detail and illustrate our indexing scheme with a working example. Finally, we offer brief concluding remarks in Section 5.

2 Related Work

Early video indexing systems are usually text-based [1, 5]. Keyword annotations are attached to individual video streams. Document retrieval techniques are employed to retrieve relevant video streams based on textual query supplied by a user. The keywords are usually selected and entered by human manually. Not only does this approach require substantial overhead in identifying the appropriate keywords for individual video stream, but it also lacks the flexibility to support ad hoc user queries. For instance, retrieving certain frames or segments within a video stream is impossible.

To overcome the problems of text-based video indexing systems, several so-called "content-based" segmentation systems have been proposed [4, 7]. The main idea of such systems is to partition a video stream into a sequence of video segments, each containing a set of video frames. The segmentation techniques are mainly based on the low level syntactic features of individual frames. Such features include color, texture, shape, and motion transition. Queries specified in terms of such features can therefore, be initiated at the granularity of video segment. While the features utilized in these techniques might be sufficient in detecting physical commonalities among video frames, they are not appropriate for query specification and processing due to their semantic mismatch with how the movie is perceived by users. For instance, when users look for a segment in the movie Star War, they will think of the target segment in terms of phrases like "about Darth Vader" rather than the pixel color or the shape of Darth Vader.

To supplement the pitfalls of video segmentation systems, a hybrid system which combines text-based indexing system and feature-based segmentation system has been proposed [10]. Techniques based on low level syntactic features are used to partition a video stream into a sequence of video segments; each video segment is further supplemented with a high level text-based description. Users can therefore issue queries using high level description against individual segments. This approach still suffers from the high overhead in identifying the appropriate keywords manually.

In SMIS, we propose an indexing system for digitized movies that again, employs the keyword-based approach to meaningfully describe the content of each movie segment. However, unlike previous keyword-based systems, we propose an automated method to assign keywords for each individual segment. Our method employs a hybrid approach which extracts keywords for each movie segment based on the subtitle of each frame within the segment; we also utilize voice recognition techniques to extract keywords from the audio signal of the movie for those movies with no subtitle. To obtain the subtitle of each frame, one could apply image recognition techniques to the frame or by decoding the textual data encoded on the movie. The extracted keywords therefore, characterize the semantics of the segment accu-

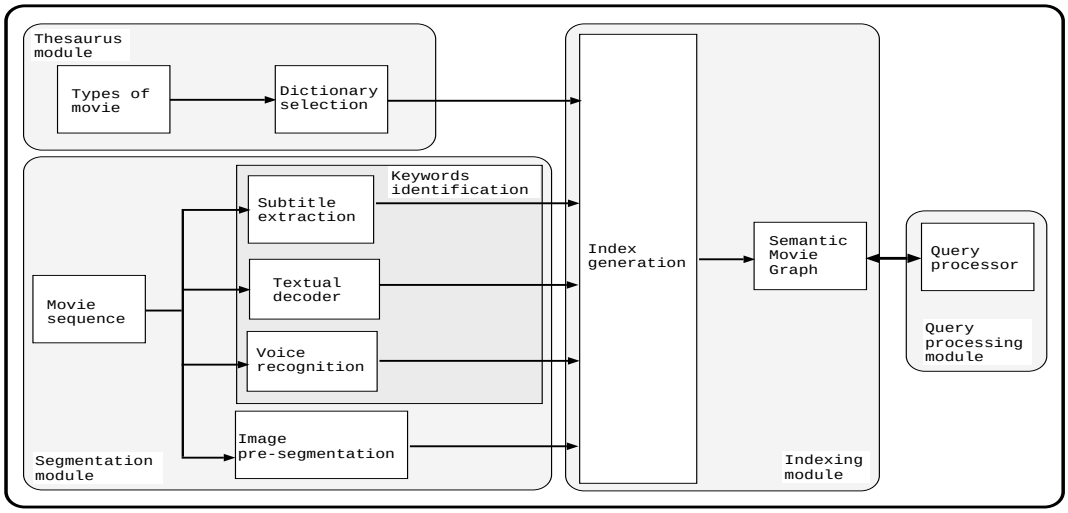

Figure 1: A semantic movie indexing architecture

rately. Users can then initiate queries based on the semantics of the movie.

3 The Architecture of SMIS

Figure 1 illustrates a top level architecture of SMIS, which consists of four interconnected modules: a *thesaurus module*, a *segmentation module*, an *indexing module*, and a *query processing module*. The functionalities and relationships of the modules are described as follows:

1. Thesaurus module: We assume that each movie is classified into one of several movie types such as love story, adventure, science fiction, and so forth. We further assume that each type of movie is associated with a dictionary containing a set of words commonly found in that particular type of movies. The thesaurus module identifies the type of the movie to be indexed and selects the most appropriate dictionary to be used in other modules. We are in the process of automating the establishment of the dictionaries for each movie type as well as classifying each movie into the appropriate type, but this module is not the focus of this paper and thus, will not be described in further detail here.
2. Segmentation module: For each movie stream, it is pre-segmented according to the low level syntactic characteristics of individual frames. The results from this process will be a set of *syntactic segments*. Currently, we employ existing segmentation techniques such as those proposed in [10, 11]. However, we are also investigating into techniques to improve the reliability and accuracy of segmentation and the result will be reported in future papers.

Once a movie stream is segmented, a keywords identification process is performed on the sequence of movie frames for each segment. This process identifies several keywords that best describe the content of the segment. We employ a hybrid approach which extracts the keywords from the subtitle of each frame of the segment as well as voice recognition techniques to extract the keywords from the audio signal of the movie [3]. The subtitle of each frame could be obtained either by applying image recognition techniques on the frame for those movies which have the subtitles hard printed on the frame. For those movies which have the subtitle encoded as textual data, we can simply extract the subtitle by a simple decoding task.

It is important to note that not every word is eligible to act as a keyword for a segment, e.g., the word "love" would be a keyword in an action movie, but not so in a love movie. A filtering method such as that employed in document retrieval [8] is employed to identify the keywords that best describe the semantic content of the movie segment. Intuitively, a two pass parsing is performed on each movie segment. In the first pass, a global analysis is performed on the whole movie stream. All the words that appeared in the movie stream are recognized[1]. A word distribution histogram is constructed and a set of keywords that best described the movie [8] are identified. In the second pass, a local analysis is performed on each segment. If a word

[1] This is achieved either through applying image recognition techniques on the subtitle of the movie or through applying voice recognition techniques on the audio signal of the movie, or even simply by retrieving the textual subtitle data encoded on the movie, if available.

recognized within the segment belongs to the set of keywords obtained in the first pass, the words is treated as a keyword for that particular segment.

3. Indexing module: Keywords for each syntactic segment are compared with the help of the selected dictionary, based on the movie type. For all segments whose keywords are similar, a new parent segment which subsumes the original segments is created. This generalizes to a segment graph called **S***emantic* **S***egment* **G***raph* (*SSG*). We call SSG a graph, and not a tree, since a segment within SSG might have multiple parents if its set of keywords is similar to that of several other segments.

   For each parent segment, a set of keywords that best describe the parent segment will be selected from the child segments. Each keyword set thus represents an index to a segment of the SSG. A multi-level index is therefore established, resembling the structure of the SSG. This index is called **S***emantic* **M***ovie* **G***raph* (*SMG*). Notice that the main difference between SSG and SMG is that SSG is a physical organization of movie segments while SMG is a logical indexing structure to SSG.
4. Query processing module: Once the indices to the segments are generated, users can initiate textual query against the movie in a way similar to document retrieval systems [9].

In summary, a movie will be classified by the thesaurus module, which also dictates the appropriate dictionary to be used by the indexing module. Concurrently, the movie frames are fed into the segmentation module which extracts the keywords from the subtitles and the digitized sound tracks, and/or performs a pre-segmentation based on video processing techniques. After the segments are constructed, the indices are built based on the best dictionary selected by the indexing module, which produces the semantic movie graph to be queried against through the the query processor interface. To illustrate, we will briefly describe how the two graphs in the indexing module are generated in next section along with a working scenario using the movie Star War as an example.

4 The Movie Indexing Scheme

We illustrate how SSG and SMG are established in this section. Each movie, M, is partitioned into a sequence of syntactic segments by the image pre-segmentation process. We denote a movie segment within the SSG by $S_{l,i}$ where l indicates the level where the segment locates within the SSG while i indicates the position of the segment within the set of segments at level l of the SSG. Similarly, we denote the keywords of $S_{l,i}$ by $K_{l,i}$ where l and i have their parallel meaning within the SMG. Such a set of keywords is referred to as the keyword set. Therefore, a movie, M, can be denoted by $M = \{S_{1,1},\ S_{1,2},\ ...\ S_{1,n}\}$ where n is the total number of segments belonging to M. This establishes the lowest level of the SSG. Each segment $S_{1,i}$ will be associated with a set of keywords, $K_{1,i}$, by keyword identification, i.e., $K_{1,i} = \{A_{1,i,1},\ A_{1,i,2},\ ...\ A_{1,i,m_{1,i}}\}$ where $m_{1,i}$ is the total number of keywords associated with segment $S_{1,i}$. This constitutes the lowest level of SMG. Note that the number of keywords in each keyword set might not be equal, i.e., $m_{1,i}$ might not be equal to $m_{1,j}$ if $i \neq j$.

For every pair of segments, $S_{1,i}$ and $S_{1,j}$, a fitness value, $f(K_{1,i}, K_{1,j})$, is calculated by Jaccard's score[2] [2]:

$$f(K_{1,i}, K_{1,j}) = \frac{|K_{1,i} \bigcap K_{1,j}|}{|K_{1,i} \bigcup K_{1,j}|} \qquad (1)$$

All segments whose fitness values are greater than a threshold, ε, are grouped into a collection. A new parent segment is created for each collection, modeling a supersegment-subsegment relationship. Note that a segment can have multiple supersegments if it belongs to multiple collections. A second level of the SSG is therefore created. For each newly created segment, $S_{2,k}$, a genetic algorithm [2] is employed to identify a set of best descriptive keywords, $K_{2,k}$, for $S_{2,k}$ from the keywords associated with $S_{2,k}$'s subsegments. This establishes the second level of the SMG. The fitness value between any two segments in the second level can thus be computed. Another higher level of segments and their corresponding keywords can be created recursively until the fitness values between any two segments within a particular level all fall below ε or only one segment remains in that level.

A sample SSG and the corresponding SMG index are depicted in Figure 2. As an example, in Figure 2, segment $S_{2,1}$ is a supersegment composed of $\{S_{1,1},\ S_{1,2},\ S_{1,3}\}$ and segment $S_{2,2}$ is a supersegment composed of $\{S_{1,3},\ S_{1,4},\ S_{1,5}\}$. Keyword set $K_{2,1}$ is generated by identifying certain keywords from $\{K_{1,1},\ K_{1,2},\ K_{1,3}\}$ that best describe segment $S_{2,1}$.

Keywords for the first level of SMG are generated through the keyword identification function in the segmentation module and act as a basis for generating keywords for higher levels. For each segment $S_{l,k}$ at a level $l > 1$ within SMG, we modify the genetic algorithm described in [2] to gen-

[2]Note that when computing the fitness value, the dictionary selected from the thesaurus module will be consulted, identifying synonym keywords.

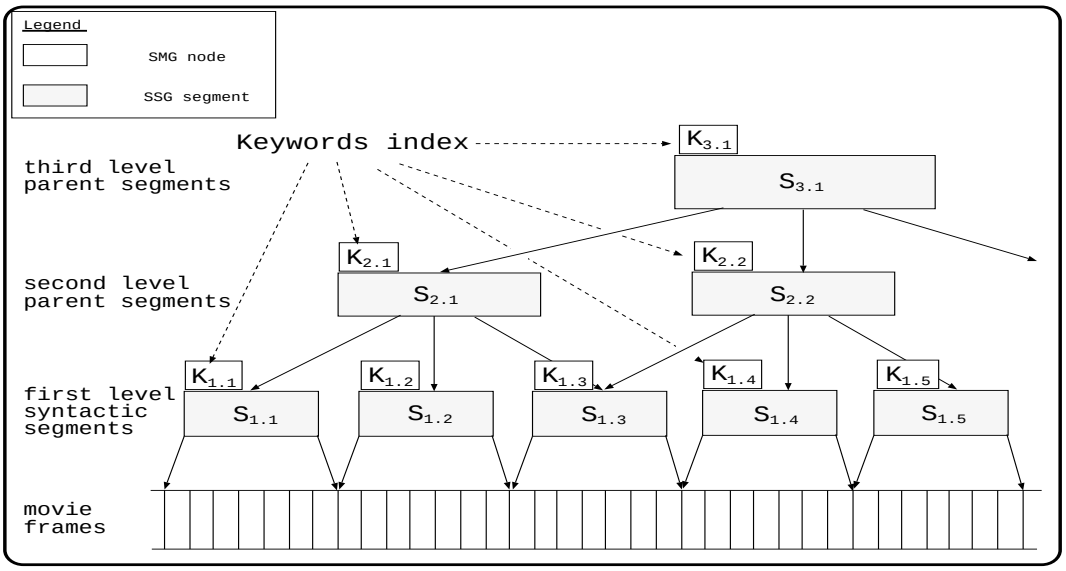

Figure 2: Semantic segment graph *versus* semantic movie graph

erate keywords for $S_{l.k}$. Assuming that $S_{l.k}$ has m subsegments, $\{S_{(l-1).1},\ S_{(l-1).2},\ \ldots\ S_{(l-1).m}\}$, the algorithm works as follows:

1. *Reference set generation*: Since every genetic algorithm requires a fixed set of reference keywords, we generate the reference set $R_{S_{l.k}}$ of $S_{l.k}$ by the union of all the keywords of its subsegments, i.e., $R_{S_{l.k}} = \bigcup_{i=1}^{m}\{K_{(l-1).i}\} = \{A_1,\ A_2,\ \ldots\ A_{|R_{S_{l.k}}|}\}$. Each subsegment $S_{(l-1).i}$ will be associated with a binary vector, $X_{S_{(l-1).i}}$, of length $|R_{S_{l.k}}|$ with respect to $R_{S_{l.k}}$, i.e., $X_{S_{(l-1).i}} = \{b_{(l-1).i.1},\ b_{(l-1).i.2},\ \ldots\ b_{(l-1).i.|R_{S_{l.k}}|}\}$. Each entry, $b_{(l-1).i.w}$, of the vector indicates whether $S_{(l-1).i}$ contains the keyword A_w or not.
2. *Reproduction*: A new set of m subsegments will be selected from the original set of subsegments of $S_{l.k}$, contributing to the generation of keywords for $S_{l.k}$. Each subsegment, $S_{(l-1).i}$, will have a probability p_i of being selected which is proportional to its fitness value with respect to the reference set, i.e., $f(K_{(l-1).i}, R_{S_{l.k}})$; therefore, a subsegment might be selected a multiple number of times if its probability value is high while a subsegment with low probability value might not be selected. This is in accordance with the principle of genetic inheritance. The result of this step will be a new set of subsegments $\{S'_{(l-1).1},\ S'_{(l-1).2},\ \ldots\ S'_{(l-1).m}\}$, and their associated keywords, possibly with duplicates.
3. *Crossover*: For each subsegment $S'_{(l-1).i}$, a random number r between 0 and 1 is generated. If $r < p_c$, $S'_{(l-1).i}$ will be selected for performing the crossover operation. Here, p_c indicates the *probability of crossover*. The binary vector of the two subsegments selected for crossover will be mated randomly as in [2]. The result from this step will be another new set of subsegments $\{S''_{(l-1).1},\ S''_{(l-1).2},\ \ldots S''_{(l-1).m''}\}$, and the corresponding keyword sets.
4. *Mutation*: For each subsegment $S''_{(l-1).i}$, a random number r between 0 and 1 is generated for every entry of its binary vector. If $r < p_m$, the entry is flip-flopped, simulating a mutation process. Here, p_m indicates the *probability of mutation*. The results from this step will be yet another new set of subsegments $\{S'''_{(l-1).1},\ S'''_{(l-1).2},\ \ldots S'''_{(l-1).m'''}\}$.
5. *Iteration*: The average fitness value between the keywords of each new subsegment and the keywords in $R_{S_{l.k}}$ is computed. Steps 2 to 4 are repeated until the average fitness values across several subsequent iterations remain relatively constant. The union of the keywords for the most currently generated subsegments will be assigned to segment $S_{l.k}$. Note that we do not actually change the subsegments of $S_{l.k}$; the creation of the new set of subsegments are merely for intermediate computation.

4.1 A Working Scenario

Due to space limitation, we will report detail experimental result in a future paper and only illustrate our indexing scheme through a simple example here. Consider the three movie segments as shown in Figure 3. For each segment, we have

Segment $S_{1.1}$:


He was betrayed and murdered, by a very young Jedi named Darth Vader;
a boy I was training. One of my brightest disciples; one of my greatest failures.

Segment $S_{1.2}$:



Enough of this! Vader, release him!

Segment $S_{1.3}$:



This is a fight you cannot win, Darth. Your power has matured since I taught you, but I too have grown much since our parting. If my blade finds its mark, you will cease to exist. But if you cut me down, I will only become more powerful.

Figure 3: Sample segments and subtitles

depicted several key frames within the segment along with their corresponding subtitles which contribute keywords to the segment.

When the specific subtitle of the segments depicted in Figure 3 is parsed, the following keywords are extracted for the corresponding segment according to the keyword identification process described in Section 3:

Segment	Keywords
$S_{1.1}$	$K_{1.1}$ = {Jedi, Darth, Vader}
$S_{1.2}$	$K_{1.2}$ = {Vader}
$S_{1.3}$	$K_{1.3}$ = {Darth, power, fight}

Based on the keywords of the segments, the Jaccard's score for fitness between any two segments will be computed as follows:

Segments	Jaccard's score
$S_{1.1}$, $S_{1.2}$	$f(K_{1.1}, K_{1.2}) = \frac{1}{3}$
$S_{1.1}$, $S_{1.3}$	$f(K_{1.1}, K_{1.3}) = \frac{1}{5}$
$S_{1.2}$, $S_{1.3}$	$f(K_{1.2}, K_{1.3}) = 0$

If ε is set to 0.2, $S_{1,1}$ and $S_{1,2}$ will be included in a new segment, since $f(K_{1.1}, K_{1.2}) = \frac{1}{3} \geq \varepsilon$. Similarly, $S_{1,1}$ and $S_{1,3}$ will be included. All the three segments will therefore be grouped into the same collection and a new segment $S_{2.1}$ will be created as the parent of segments $S_{1.1}$, $S_{1.2}$, and $S_{1.3}$. To establish the keywords for segment $S_{2.1}$, the following genetic algorithmic steps as described previously will be executed:

1. Reference set generation: As described previously, the reference set is generated as the union of $K_{1.1}$, $K_{1.2}$, and $K_{1.3}$, i.e., $R_{S_{2.1}}$ = {Jedi, Darth, Vader, power, fight}. The binary vector for each of the three segments with respect to the reference set will be:

Segment	Binary vector
$S_{1.1}$	$X_{S_{1.1}}$ = {1, 1, 1, 0, 0}
$S_{1.2}$	$X_{S_{1.2}}$ = {0, 0, 1, 0, 0}
$S_{1.3}$	$X_{S_{1.3}}$ = {0, 1, 0, 1, 1}

2. Reproduction: The Jaccard's score between each segment and the reference set is calculated as:

Segment	Jaccard's score
$S_{1.1}$	$f(K_{1.1}, R_{S_{2.1}}) = \frac{3}{5}$
$S_{1.2}$	$f(K_{1.2}, R_{S_{2.1}}) = \frac{1}{5}$
$S_{1.3}$	$f(K_{1.3}, R_{S_{2.1}}) = \frac{3}{5}$

   Assuming that $p_i = 0.5$, a possible set of segments chosen for reproduction would be $S'_{1.1}$ = $S_{1.1}$, $S'_{1.2}$ = $S_{1.3}$, and $S'_{1.3}$ = $S_{1.1}$. Note that segment $S_{1.1}$ have been chosen twice due to its high Jaccard's score[3].

3. Crossover: Assuming that $p_c = 0.5$ again, the segments chosen for crossover would be $S'_{1.1}$ and $S'_{1.2}$. A possible binary vector for the new crossovered segment, $S''_{1.1}$ would be $X_{S''_{1.1}}$ = {1, 1, 0, 1, 1} if the crossover point is at position 3.

4. Mutation: Assuming that $p_m = 0.6$, one possible mutated binary vector for the new mutated segment, $S'''_{1.1}$ would be $X_{S'''_{1.1}}$ = {1, 1, 1, 0, 0}.

5. Iteration: In this simple example, it would not be necessary to re-iterate the steps as there is only one segment left after crossover. The keywords chosen for segment $S_{2.1}$ would therefore be {Jedi, Darth, Vader}.

Once this SMG index has been established, segments $S_{1.1}$, $S_{1.2}$, and $S_{1.3}$ can be retrieved readily by the query processor for any user query containing the keywords {Darth, Vader}, or any query with the keywords {Jedi, Vader}. The search starts from the root, which is segment $S_{2.1}$ and branching downwards to the corresponding subsegments.

5 Conclusion

In this paper, we have described an indexing mechanism for digitized movies. We believe that different tailored techniques need to be developed for each type of video data, based upon the specific knowledge captured within the data. For video movies, the semantics captured within subtitles provide valuable additional knowledge to index movie segments meaningfully.

One might have noticed that the effectiveness of the indexing scheme presented in this paper largely depends on how well the original movie stream is segmented. We have found that current segmentation techniques fail to properly detect segment break in certain cases. For instance, if a camera is repeatedly flip-flopping between two persons in conversation, applying current segmentation techniques on such a movie stream will result in a large number of segments which might be classified more suitably to be subsumed by one single segment. We are currently working on new mechanisms to improve existing segmentation techniques.

Finally, we are also investigating into the effects of different threshold, ε, and different probability values, p_c and p_m, on the effectiveness of the scheme. Experiments to measure the effectiveness of the mechanism will then be conducted. In this context, traditional recall and precision metrics used in document retrieval can be fruitfully employed [9].

[3] Another possible set of segments might be $S_{1.1}$, $S_{1.3}$, and $S_{1.3}$, again due to the high Jaccard's score of $S_{1.3}$.

References

[1] A. Bertina, F. Rabitti, and S. Gibbs. Query Processing in a Multimedia Document System. *ACM Transaction on Office Information Systems*, 6(1):1–41, 1988.

[2] H. Chen. A Machine Learning Approach to Document Retrieval: An Overview and an Experiment. In 27^{th} *Hawaii International Conference on System Sciences*, pages 631–640, 1994.

[3] M. Christel, T. Kanade, M. Mauldin, R. Reddy, M. Sirbu, S. Stevens, and H. Wactlar. Informedia Digital Video Library. *Communications of the ACM*, 38(4):57–58, April 1995.

[4] T. Chua, S. Lim, and H. Pung. Content-based Retrieval of Segmented Images. In *Proceedings of ACM International Conference on Multimedia*, pages 211–218, 1994.

[5] T. Chua, H. Pung, G. Lu, and H. Jong. A Concept-Based Image Retrieval System. In *Proceedings of* 27^{th} *Hawaii International Conference on System Sciences*, pages 590–598, 1994.

[6] D. Fang, J. Hammer, D. McLeod, and A. Si. Remote-Exchange: An Approach to Controlled Sharing among Autonomous, Heterogenous Database Systems. In *Proceedings of the IEEE Spring Compcon, San Francisco.* IEEE, February 1991.

[7] A. Hampapur, R. Jain, and T. Weymouth. Digital Video Segmentation. In *Proceedings of ACM International Conference on Multimedia*, pages 357–364, 1994.

[8] Y. Maarek, D. Berry, and G. Kaiser. An information retrieval approach for automatically constructing software libraries. *IEEE Transactions on Software Engineering*, 1991.

[9] S. Robertson, C. Rijsbergen, and M. Porter. *Probabilistic models of indexing and searching*, pages 35–56. Butterworths, 1981.

[10] S. Smoliar and H. Zhang. Content-Based Video Indexing and Retrieval. *IEEE Multimedia*, 1(2):62–72, 1994.

[11] H. Zhang, C. Low, and S. Smoliar. Video Parsing and Browsing Using Compressed Data. *Multimedia Tools and Applications*, 1(1):89–111, 1995.