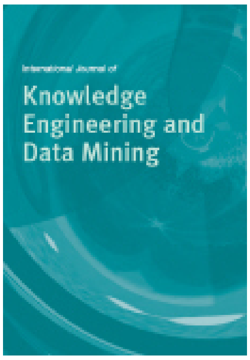


International Journal of Knowledge Engineering and Data Mining

ISSN online: 1755-2095 - ISSN print: 1755-2087
https://www.inderscience.com/ijkedm

Method to integrate speaker identification, speech recognition, and information retrieval algorithms for speaker-based information retrieval

Muhammad Muneeb

DOI: 10.1504/IJKEDM.2022.10049052

Article History:

Received:	15 December 2021
Last revised:	12 July 2022
Accepted:	04 February 2022
Published online:	10 October 2022

Method to integrate speaker identification, speech recognition, and information retrieval algorithms for speaker-based information retrieval

Muhammad Muneeb

Department of Electrical Engineering and Computer Science,
Khalifa University of Science, Technology and Research (KUSTAR),
Abu Dhabi, UAE
Email: muneebsiddique007@gmail.com

Abstract: This article proposes speakers' voice-based information (audio and video) retrieval systems, which combines speaker identification, speech recognition, and information retrieval algorithms. Information retrieval systems encompass system structure and a way to query the system for information retrieval. This article illustrates both, including how it is deployed on top of existing systems. The input to the system is a speaker voice sample and a text query. Based on the speaker's voice, the size of the corpus is reduced, and based on the text query, documents are retrieved and ranked. For the speaker identification, we used the LPC coefficient, for voice recognition, we used a Python speech recognition library, and for ranking, we used cosine similarity and TF-IDF. Other algorithms can replace any intermediate modules depending on the system, like crime investigation, news analysis, and lecture retrieval. We demonstrated the proposed method on simulated data generated from online websites.

Keywords: audio retrieval; information retrieval; speaker identification; TF-IDF; voice recognition.

Reference to this paper should be made as follows: Muneeb, M. (2022) 'Method to integrate speaker identification, speech recognition, and information retrieval algorithms for speaker-based information retrieval', *Int. J. Knowledge Engineering and Data Mining*, Vol. 7, Nos. 3/4, pp.234–251.

Biographical notes: Muhammad Muneeb obtained his MSc in Computer Science from the Khalifa University, Abu Dhabi, UAE. He is currently working as a research associate in the same institute under the supervision of Dr. Samuel. He like to work on inter-discipline problems and have an interests in algorithms, automation, genetics, medical image analysis, and optimisation.

1 Introduction

In computer science, researchers work on algorithm mutation (Muneeb and Raza, 2021), energy-efficient algorithms (Raza et al., 2021), and frameworks (Muneeb et al.,

2022) to obtain an efficient workflow to perform a task. The same is the case for information-retrieval systems. The rapid increase in video data invokes the necessity for efficient indexing and retrieval systems (Rasheed et al., 2020; Smith and Chen, 2005; Zhang and Smoliar, 1994). In this article, we proposed two systems for audio-based information retrieval systems. The first system focuses on annotation-based approaches (Patil and Nemade, 2016; Wang and Zhang, 2009), which employ textual information obtained from converting audio to text. The second system integrates the speaker voice (Intechopen, 2021; Hill, 2007) and the first system for speaker voice-dependent content retrieval. As we know, there is much information generated every day on the internet. There are many websites like YouTube and Netflix on which a lot of video information is available (Budzinski et al., 2021). Currently, the videos on the websites are retrieved based on the video's title and using some ranking algorithm like TF-IDF (Bafna et al., 2016; Kim and Gil, 2019; Sammut and Webb, 2010). Some advancements are made in retrieving audio from the video is extracted and converted to text using the audio transcription. For a given query, the audio's text can be used for retrieval rather than matching the video's title. Information retrieval system based on this approach is not that difficult to implement. Like the inverted index approach for document retrieval, we can repeat this process for the audio text. In this article, we combined speaker recognition with the information retrieval system. When a person types some query like 'a cat', then there is a possibility that many people speak it. So with speaker voice recognition (Ramos, 2003; Leu and Lin, 2017), we can retrieve the videos in which a specific person speaks that particular phrase. We build the speaker-based system on top of the existing system so that current technologies can adopt it.

The following paragraph provides an overview of the proposed system.

Suppose we have the audio files of two people in our system, speaker A and speaker B. Speaker A and B both say one sentence, 'I am a human'. When you type 'I am a human, B', all videos that have 'human' in audio and spoken by person 'B' must be returned by the system. When you type 'I am a human, A', all videos that have 'human' in audio and spoken by a person 'A' must be returned by the system.

2 Related work

This section highlights the advancement made in the audio-based information retrieval system and different approaches used by the researchers. This article (Hou and Zhou, 2013) proposed a method that uses both audio signal (for rough retrieval) and visual features (for refining retrieval) extracted from the shot and keyframe level. This article (Hiriyannaiah et al., 2020) used text associated with the video for content-based video retrieval (CBVR), which is similar to the approach proposed in our article, but we also included the speaker-based retrieval. This article (Patel, 2012) reviews the features that are extracted from video data for indexing and retrieval along with similarity measurement methods for content-based retrieval. This article (Radha, 2016) combines the text information from the visual slide frame and audio signal to retrieve the video from the lecture video database. This article (Mohamadzadeh and Farsi, 2016) proposed a methodology that combines the local and global colour, texture, and motion features of the video for information retrieval. This article (Chechik et al., 2008) proposed a technique that is similar to our proposed information retrieval system. They proposed a system to find sound recordings (audio documents) based on their acoustic features.

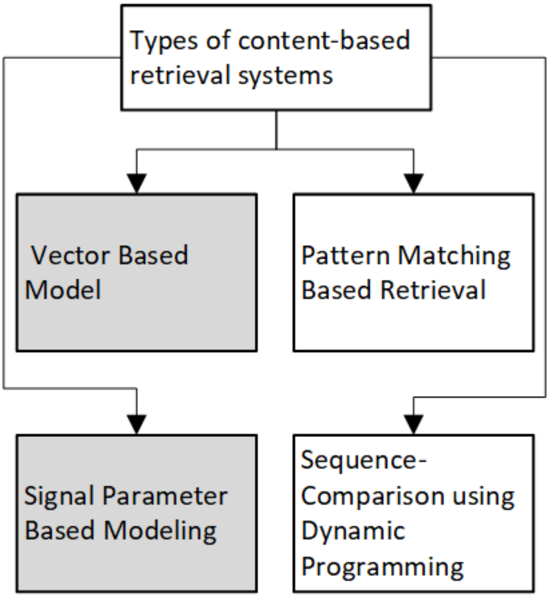
Their content-based approach differs from retrieval approaches that index media files using metadata such as file names and user tags. They used a machine learning approach for retrieving sounds and incorporated the following features

1 free-form text queries rather than sound sample-based queries

2 searches by audio content rather than via textual metadata.

Figure 1 summarises the four types of content-based retrieval systems. Our proposed system combines a vector-based model (for the similarity between the query and the documents) and a signal parameter-based model (for the speaker voice identification).

Figure 1 This diagram shows the four types of content-based retrieval systems


Types of content-based retrieval systems
Vector Based Model
Pattern Matching Based Retrieval
Signal Parameter Based Modeling
Sequence-Comparison using Dynamic Programming

Notes: 1 Signal parameter-based modelling is characterised by signal or acoustical parameters applicable to audio objects. To model an object, it uses both frame-level as well as global parameters.

2 In a vector-based model, both the query and object are characterised as vectors in terms of n-dimensional space. A measure of similarity between the query and each object is computed, and results are ranked accordingly.

3 In pattern matching-based retrieval, a sequence of characters represents both the queries and the documents, and their similarity is computed based on how similar the two sequences are.

4 Sequence-comparison using dynamic programming uses the concept of edit distance; edit distance is the cost of changing the source sequence (source string) into the target sequence (target string).

The proposed system comprises four main sub-steps, which are speakers identification (Furui, 2009; Kabir et al., 2021), speech recognition (Jain and Rastogi, 2019), information retrieval algorithms, and the data structure and algorithm to store and query the system. There are various ways for speakers identification like machine learning (Jahangir et al., 2021), mel-frequency cepstral coefficients (MFCC) (Leu and Lin, 2017), and responses from a model of the auditory periphery (Islam et al., 2016). We opted for LPC coefficients which is an efficient way of speaker identification (Chauhan et al., 2019). For the second sub-step, speech recognition, there are many methods like deep learning (Nassif et al., 2019), and Gaussian mixture model (GMM) acoustic models (Povey et al., 2010), which is also used in google speech recognition API. For the third sub-step, information retrieval, there are many methods like Word2Vec (Altszyler et al., 2016; Ma and Zhang, 2015) and TD-IDF (Kim and Gil, 2019) which we used. Lastly, the way to store speakers' information and query the system is done using a modified version of TF-IDF.

The other methods can replace methods used in sub-steps without affecting the other steps. For instance, speaker recognition can be replaced with deep learning, speech recognition can be replaced with MFCC, and information retrieval can be done using Word2Vec, ultimately leading to multiple systems. In this article (Mitrović et al., 2010), researchers highlighted the best methods for each sub-step.

3 Application

This section elaborates on the possible application of the proposed system.

3.1 Crime investigation

This system can assist in crime investigation. In any legal investigation, many people are involved, and police take statements from each person. For the judiciary, it is not easy to listen to each person's statement. When thinking about a specific angel in the investigation, a person can query the audio files based on a particular speaker's voice, and the query-specific audio file will be returned.

3.2 News analysis

In most interviews, like hard talk, there is an exchange of meaningful conversation that can shape international politics. Two people talk to each other: one is the host, and the other is the guest. People can search for specific words spoken by a specific person rather than listening to many videos.

3.3 Lecture retrieval

Due to the COVID situation, most schools, colleges, and universities switched to online classes. They use learning management software to deliver lectures. There are about 30–40 lectures on average for each course, with a duration of 45–75 minutes. If some students want to search for the specific word in the lecture, it would be impossible due to the many videos. The proposed system can act as a module in the existing learning management system by assisting students in searching for a specific word and a particular professor's voice in all the videos.

4 Methodology

The following text explains the two proposed systems.

4.1 Dataset for system 1

For system 1 we generated ten audio files (.mp3 extension) for five speakers using the website (From Text to Speech – Free Online TTS Service, http://www.fromtexttospeech.com/). Each audio file was converted to the following specs using this website (Convert Audio to WAV, https://audio.online-convert.com/convert-to-wav).

- convert .mp3 files to .wav
- audio channels: stereo
- bit resolution: 16 bit
- sampling rate: 16,000 Hz.

Audio 1 for each speaker is generated using the same text extract from an online form related to machine learning. Similarly, audio 2 to 10 for each speaker contains the same text.

4.2 System 1

This first system is speaker-independent, and in this system, we showed how the video files are retrieved based on the text in the audio of the video. Label each video file in sequence like video 1, video 2, and video N. Label each audio file correspondingly as shown below.

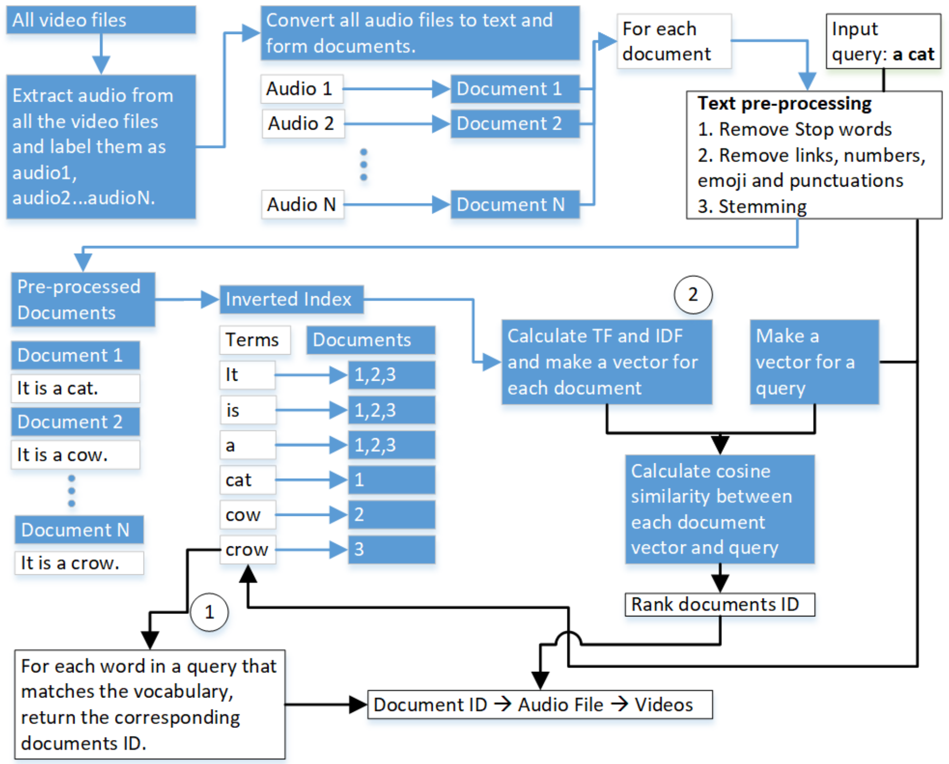
- video 1 –> audio 1
- video 2 –> audio 2
- ...
- video N –> audio N.

Convert each audio file to textual form. Many libraries are available, but we used Python speech recognition for text transcription and speech recognition (Torfi, 2018). Google speech recognition or Python library can also be used. Convert each audio file in the form of a text document and label them correspondingly, as shown below.

- video 1 –> audio 1 –> document 1
- video 2 –> audio 2 –> document 2
- ...
- video N –> audio N –> document N.

Figure 2 shows the flow of system 1.

Figure 2 System 1: extract audio from all the videos and name each audio file correspondingly (see online version for colours)


All video files
Extract audio from all the video files and label them as audio1, audio2...audioN.
Convert all audio files to text and form documents.
Audio 1
Audio 2
Audio N
Document 1
Document 2
Document N
For each document
Input query: a cat
Text pre-processing
1. Remove Stop words
2. Remove links, numbers, emoji and punctuations
3. Stemming
Pre-processed Documents
Document 1
It is a cat.
Document 2
It is a cow.
Document N
It is a crow.
Inverted Index
Terms
Documents
It
is
a
cat
cow
crow
1,2,3
1,2,3
1,2,3
1
2
3
2
Calculate TF and IDF and make a vector for each document
Make a vector for a query
Calculate cosine similarity between each document vector and query
Rank documents ID
1
For each word in a query that matches the vocabulary, return the corresponding documents ID.
Document ID → Audio File → Videos

Notes: Convert each audio to text and form a document for each audio file. For each document, perform the text processing and make an inverted index. After that, there are two ways to retrieve the documents. The first is to match each word in the query with the terms and return the documents. The second is to form vectors for each document and query. Find the similarity between a query vector and each document and rank results based on the similarity index.

The next step is to pre-process each document. Following are the default text pre-processing (Hickman et al., 2020; Hasanah et al., 2018) steps which are performed on the text of each document in any information retrieval system and also used in other applications like sentiment analysis and text classification. We use nltk (Loper and Bird, 2002) Python library for text pre-processing and librosa (McFee et al., 2015) for audio processing.

- Lower case – Convert each word to lower case.
- Stop words – Remove all stop words.
- Punctuation – Remove punctuation.
- Apostrophe – Remove apostrophe.

- Single characters – Remove single character.
- Stemming – The process of reducing inflected words to their root word. For example, ‘flooding’ is stemmed as ‘flood’.
- Lemmatisation – The process of grouping the inflected forms of a word so that it can be analysed as a single item. For example, ‘better’ is lemmatised as ‘good’.
- Converting numbers – Remove the numbers or convert them to words. For example, nine can be changed to nine.

For document retrieval, we selected the inverted index technique. The inverted index data structure is a key component of a standard search engine indexing algorithm. It improves query performance by locating documents that include the term ‘X’. Creating a forward index, which holds lists of words per document, necessitates sequential iteration across each document and to each word. Technically, the time, memory, and processor resources required to conduct such a query are not always available. An inverted index data structure is created rather than listing words per document in the forward index, which lists documents per word. The query now is resolved by jumping to the word ID in the inverted index, which is generated after the inverted index is created (Mahapatra and Biswas, 2011).

We can query the inverted index or the second option is to use TF-IDF, which is the product of two statistics, term frequency, and inverse document frequency. There are various ways of determining the exact values of both statistics. These numerical statistics are intended to reflect how important a word is to a document in a collection or corpus. They are used as a weighting factor in information retrieval searches, text mining, and user modelling. The TF-IDF value increases proportionally to the number of times a word appears in the document. It is offset by the number of documents in the corpus containing the word, which helps to adjust that some words appear more frequently in general. The simplest way to compute TF-IDF is to convert everything to a vector and compute the cosine similarity. The length of the vector is the same as the total_vocab variable, which has an index for all the unique tokens, so the final data structure is a set of vectors having dimensions (total number of documents, total_vocab). Cosine similarity measures the similarity between two vectors of an inner product space. It is measured by the cosine of the angle between two vectors and determines whether two vectors are pointing in roughly the same direction. Consider the following terms to understand the calculation of the TF-IDF.

- $TF\text{-}IDF$ = term frequency $*$ inverse document frequency.
- d – document (set of words)
- N – count of corpus
- $corpus$ – the total document set
- t – term (word)
- v – document vector
- q – query vector.

$$TF(t,d) = count\ of\ t\ in\ d/number\ of\ words\ in\ d \tag{1}$$

$$DF(t) = corresponds\ to\ the\ number\ of\ documents\ containing\ the\ term\ t\ in\ the\ corpus \tag{2}$$

$$IDF(t) = N/DF \tag{3}$$

$$IDF(t) = \log(N/(DF+1)) \tag{4}$$

$$TF - IDF(t,d) = TF(t,d) * \log(N/(DF+1)) \tag{5}$$

$$\cos\theta = v.q/(|v| * |q|) \tag{6}$$

Although there are various formulas to calculate the TF in particular, the logarithmic factor and the augmented factor.

Equations (1), (2), (3), (4) and (5) shows the TF-IDF. Equation (6) shows the process of calculating the cosine similarity between one document vector v and the query vector q. Each document in the corpus is ranked based on the cosine similarity. After ranking the documents based on the query, we can list the corresponding audio and video file as shown below:

- document 1 ranked by cosine similarity –> audio 1 –> video 1
- document 2 ranked by cosine similarity –> audio 2 –> video 2
- ...
- document N ranked by cosine similarity –> audio N –> video N.

The following sub-section explains the input and output for system 1.

4.2.1 Input and output for system 1

The first step is to make a database for system 1:

$$V_1, V_2, ..., V_n \tag{7}$$

$$A_1, A_2, ..., A_n \tag{8}$$

$$D_1, D_2, ..., D_n \tag{9}$$

$$C_1, C_2, ..., C_n \tag{10}$$

In equations (7), (8), (9) and (10), V represents video (convert video to audio), A represents audio (convert audio to text), D represents text (convert text to vectors), and C represents vectors respectively. n means n^{th} video, audio, text document and a vector.

Input query to system 1 is any text query, which will be converted to vector and processed in the following way:

$$D_q \to C_q, \text{ where } q \text{ represents query} \tag{11}$$

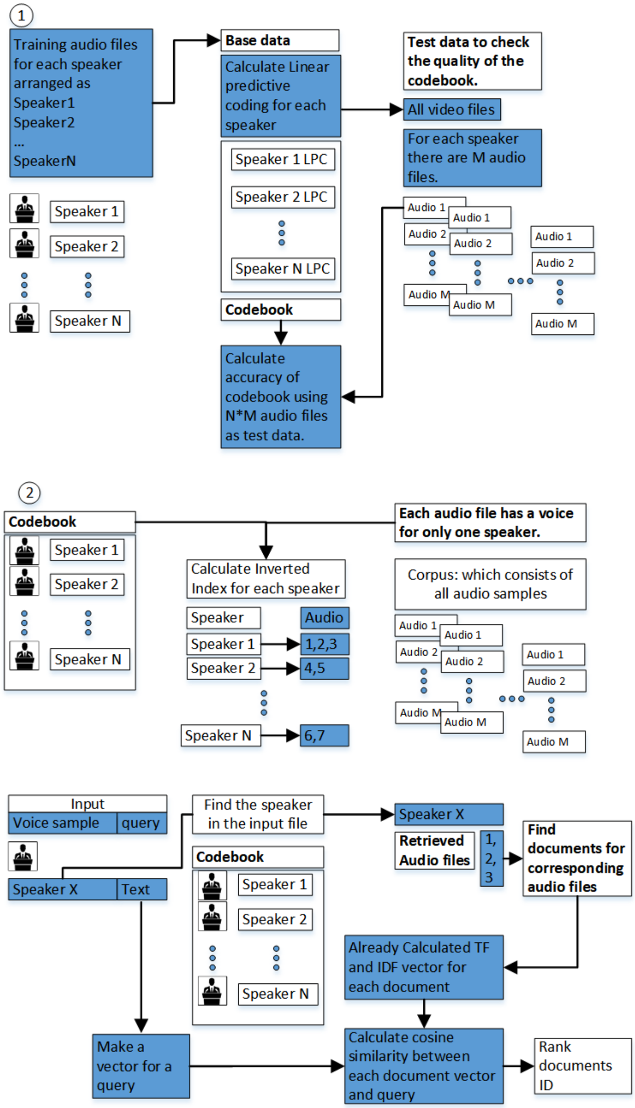
In equation (11), D_q is the text query, which is converted to a vector C_q. Match C_q with the vector of each document $C_1, C_2, ..., C_N$ and return results in the following form.

$$C_1 \to D_1 \to A_1 \to V_1 \tag{12}$$

$$C_2 \to D_2 \to A_2 \to V_2 \tag{13}$$

$$C_N \to D_N \to A_N \to V_N \tag{14}$$

Figure 3 This diagram shows the overview of the proposed system 2 (see online version for colours)


1
Training audio files for each speaker arranged as Speaker1 Speaker2 ... SpeakerN
Speaker 1
Speaker 2
Speaker N
Base data
Calculate Linear predictive coding for each speaker
Speaker 1 LPC
Speaker 2 LPC
Speaker N LPC
Codebook
Calculate accuracy of codebook using N*M audio files as test data.
Test data to check the quality of the codebook.
All video files
For each speaker there are M audio files.
Audio 1
Audio 2
Audio M
2
Codebook
Speaker 1
Speaker 2
Speaker N
Calculate Inverted Index for each speaker
Speaker
Audio
Speaker 1
1,2,3
Speaker 2
4,5
Speaker N
6,7
Each audio file has a voice for only one speaker.
Corpus: which consists of all audio samples
Input
Voice sample
query
Speaker X
Text
Find the speaker in the input file
Codebook
Speaker 1
Speaker 2
Speaker N
Speaker X
Retrieved Audio files
1, 2, 3
Find documents for corresponding audio files
Already Calculated TF and IDF vector for each document
Make a vector for a query
Calculate cosine similarity between each document vector and query
Rank documents ID

Notes: The first step is to train the features for each speaker. We assumed that in each audio file, there is one speaker. There is a single audio file used to calculate the LPC coefficients for each speaker. Calculate the LPC coefficients for each person and store them. The next step is to evaluate the performance of the LPC coefficient. Test the performance on the test set, consisting of five audio files for each speaker. Calculate the classification accuracy. In the second step, we have the codebook for each speaker, and we can index the corpus of audio files at this stage. Convert audio files to documents using the voice recognition library and make a TD-IDF vector for each audio file. The data structure looks like this speaker X –> audio Y –> document Z –> TD-IDF vector.

Equations (12), (13) and (14) show the processing when input is provided to the system, and at the end, videos are returned based on the similarity rank.

4.3 System 2

The dataset comprises of three datasets for five speakers, generated using this website (From Text to Speech – Free Online TTS Service, http://www.fromtexttospeech.com/). The first set is the training set, consisting of 1 audio file for each speaker. The corpus or base set consists of five audio files for each speaker, also treated as a test set. For querying the base set, we used a single audio file for each speaker. Figure 3 the overview of the working of system 2.

System 2 combines speaker identification, speech recognition, and information retrieval. Let's start with the speaker identification part. There are many ways for speaker identification, like MFCC coefficients, but we used LPC coefficients for speaker identification. LPC predicts future values based on prior samples, whereas MFCC extracts characteristics while taking into account the nature of the speech (Koolagudi et al., 2016). For speaker identification, both MFCC and LPC can be used. This sub-step is a black box, and the input and output to both MFCC and LPC are the same so that any algorithm can be used.

By obtaining filter coefficients equivalent to the vocal tract and reducing the mean square error between the input speech and estimated speech, linear predictive coding (LPC) identifies the speaker, cancels the noise signals generated by the mouth during the conversation, and keeps those signals that are voice. A linear weighted aggregate of preceding samples forecasts speech samples at a particular period. Following that, each frame of the windowed signal is auto-correlated, and the highest auto-correlation value determines the linear prediction analysis order (Anjum et al., 2020; Wang and Xu, 2014; Ratanpara and Patel, 2015). Calculate the LPC coefficients and form a codebook as shown below.

- speaker 1 –> LPC coefficients for speaker 1
- speaker 2 –> LPC coefficients for speaker 2
- ...
- speaker 5 –> LPC coefficients for speaker 5.

In the corpus, we have many audio files, and we assumed that each audio file contained only one person's voice. For testing the efficiency of LPC coefficients, classify each audio based on the speaker's voice, and calculate accuracy. We tested the system on 25 audio files, five files for each speaker. The accuracy was about 100%, but in realistic settings, if it deviates, recalculate LPC coefficients to increase the performance on the test set. After classifying each audio sample, store information in the following data structure.

- speaker 1 –> video 1, 3 –> audio 1, 3 –> document 1, 3
- speaker 2 –> video 2, 4 –> audio 2, 4 –> document 2, 4
- speaker 3 –> video 5 –> audio 5 –> document 5.

Represent the data structure in the following format for better understanding.

- speaker 1 –> video 1 –> audio 1 –> document 1
- speaker 1 –> video 3 –> audio 3 –> document 3
- speaker 2 –> video 2 –> audio 2 –> document 2
- speaker 2 –> video 4 –> audio 4 –> document 4
- speaker 3 –> video 5 –> audio 5 –> document 5.

After this point, the processing of the documents is the same as that of the previous system. The documents are converted to TF-IDF vectors, as shown below.

- speaker 1 –> video 1 –> audio 1 –> document 1 –> TF-IDF vector 1
- speaker 1 –> video 3 –> audio 3 –> document 3 –> TF-IDF vector 2
- speaker 2 –> video 2 –> audio 2 –> document 2 –> TF-IDF vector 3
- speaker 2 –> video 4 –> audio 4 –> document 4 –> TF-IDF vector 4
- speaker 3 –> video 5 –> audio 5 –> document 5 –> TF-IDF vector 5.

We have the indexed corpus at this stage, and we can query the system. For system 2 you can input the speaker voice and the word.

1 get video

2 extract the audio signal

3 find LPC for the new audio signal

4 find the speaker from the corpse, which resembles the LPC coefficient in the new audio file

5 find the audio files and corresponding documents for that speaker

6 query those documents and return the results as shown in Figure 4.

The following sub-section explains the input and output for system 2.

4.3.1 Input and output for system 2

The first step is to make a database for system 2:

$$S_1, S_2, ..., S_m \tag{15}$$

$$V_1, V_2, ..., V_n \tag{16}$$

$$A_1, A_2, ..., A_n \tag{17}$$

$$D_1, D_2, ..., D_n \tag{18}$$

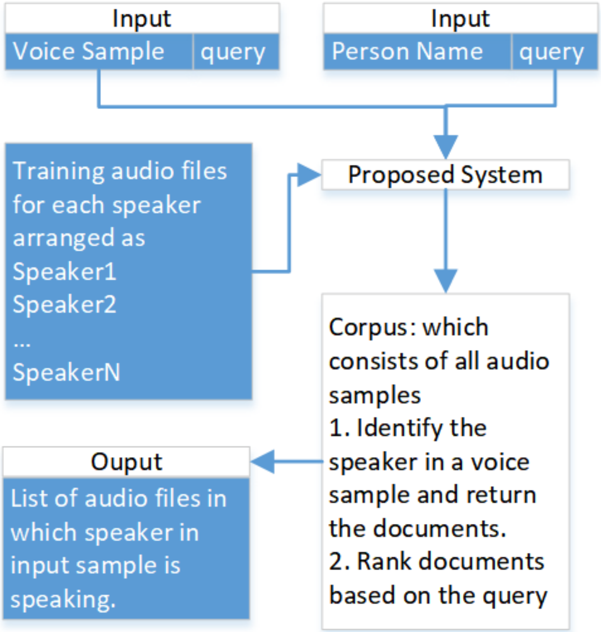
$$C_1, C_2, ..., C_n \tag{19}$$

In equations (15), (16), (17), (18) and (19), S represents a speaker, V represents video (convert video to audio), A represents audio (convert audio to text), T represents text (convert text to vectors), and C represents vectors respectively. n is the number of videos, m represents the number of speaker, and $m <= n$ because each video contains voice for only one speaker.

$$S_1V_1, S_2V_2, ..., S_{train_m}V_{train_m} \ (calculate \ LPC \ coefficents) \tag{20}$$

Equation (20) shows training files: for speaker X, audio file X extracted from video X is used to calculate the LPC coefficient, where X is a number to differentiate among speakers and videos.

Figure 4 Querying system (see online version for colours)


Input

Voice Sample | query

Input

Person Name | query

Proposed System

Training audio files for each speaker arranged as
Speaker1
Speaker2
...
SpeakerN

Corpus: which consists of all audio samples
1. Identify the speaker in a voice sample and return the documents.
2. Rank documents based on the query

Ouput

List of audio files in which speaker in input sample is speaking.

Input query to system 2 is any text query D_q (which is converted to vector) and a audio sample A_q, for which LPC coefficient are calculated.

Step 1 From A_q find S_q, and then extract all documents for speaker S_q.

Step 2 From D_q find C_q, which is a query vector.

Step 3 Match C_q with the vector of each document $(C_1, C_2, ..., C_N)$ of speaker X.

$$S_q \to C_1 \to D_1 \to A_1 \to V_1 \tag{21}$$

$$S_q \to C_2 \to D_2 \to A_2 \to V_2 \tag{22}$$

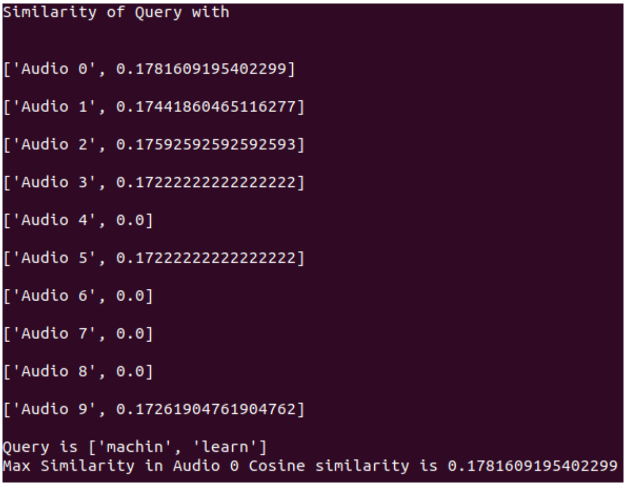
$$S_q \to C_N \to D_N \to A_N \to V_N \tag{23}$$

Equations (21), (22) and (23) show the processing when input is provided to the system, and at the end videos are returned based on the similarity rank.

5 Results

This section illustrates the results of the proposed systems on the generated dataset, and the following are the reasons to evaluate the system on generated data rather than using existing text-based documents.

Figure 5 Result of querying system 1 (see online version for colours)



```
Similarity of Query with

['Audio 0', 0.1781609195402299]

['Audio 1', 0.17441860465116277]

['Audio 2', 0.17592592592592593]

['Audio 3', 0.17222222222222222]

['Audio 4', 0.0]

['Audio 5', 0.17222222222222222]

['Audio 6', 0.0]

['Audio 7', 0.0]

['Audio 8', 0.0]

['Audio 9', 0.17261904761904762]

Query is ['machin', 'learn']
Max Similarity in Audio 0 Cosine similarity is 0.1781609195402299
```

Figure 6 LPC coefficients for speaker 1 (see online version for colours)

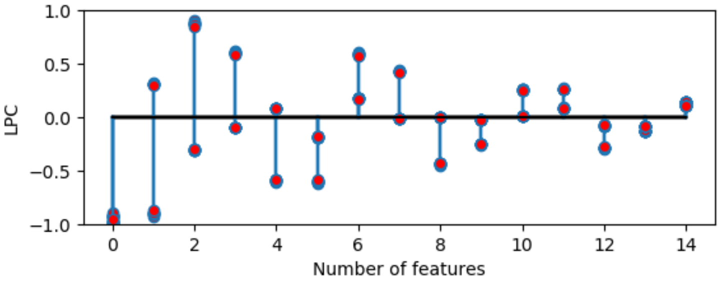
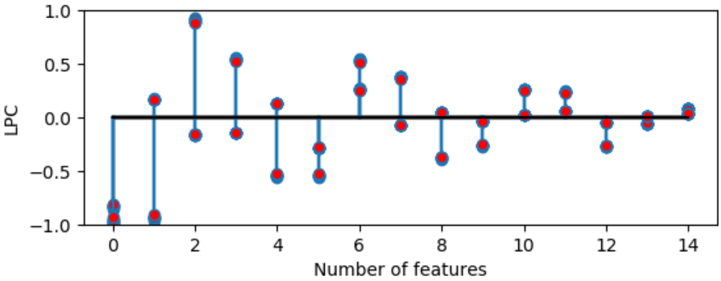


Figure 7 LPC coefficients for speaker 2 (see online version for colours)



There are many documents and query databases to benchmark the system, but there is no associated speaker information with those documents. Even if we use those documents, we would not be able to evaluate the first two steps: speaker identification and speech recognition. We tried to do the same thing with BBC News talk shows like Hard Talk, but again there is a limitation in the speaker identification. Audio files should be converted to a specific format for speaker identification and speech recognition, but commercial tools are required for that conversation. Another issue that arises when testing the real system is the amount of noise in the audio signal. The last issue is the overlapping sound signal when both speakers are talking, and it is not easy to separate speaker voices. So, after struggling with these issues, we decided to test the prototype version of the system on the generated dataset, which is easy to handle.

Figure 8 LPC coefficients for speaker 3 (see online version for colours)

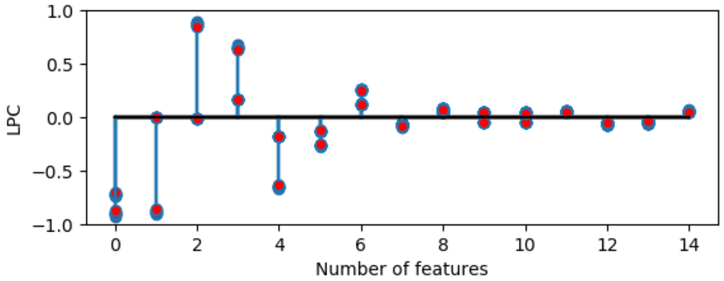


Figure 9 LPC coefficients for speaker 4 (see online version for colours)

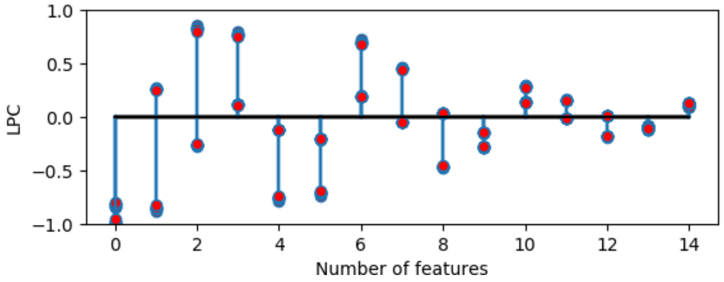
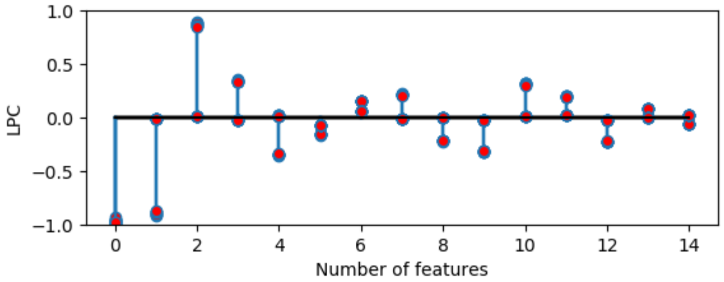


Figure 10 LPC coefficients for speaker 5 (see online version for colours)

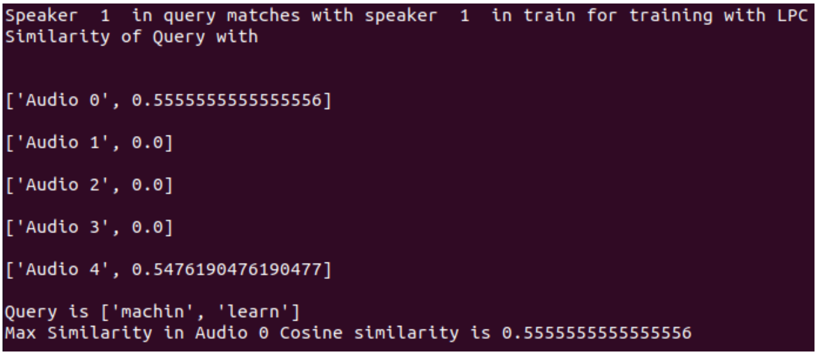


What follows explains the system 1 and system 2 performance on the generated dataset.

5.1 System 1

For system 1, the corpus consists of ten audio files, two from each speaker. This system is speaker-independent, so convert each audio file into text and also calculate TF-IDF vectors. The input query is ‘machine learning’, which after pre-processing or query normalisation becomes [‘machin’, ‘learn’]. Convert a query into a TF-IDF vector and measure the cosine similarity between the documents and query. The results show that audio 0 has the best similarity with the query. Figure 5 shows the result when system 1 is tested for a particular query.

Figure 11 Result of querying system 2 (see online version for colours)



```
Speaker  1  in query matches with speaker  1  in train for training with LPC
Similarity of Query with


['Audio 0', 0.5555555555555556]

['Audio 1', 0.0]

['Audio 2', 0.0]

['Audio 3', 0.0]

['Audio 4', 0.5476190476190477]

Query is ['machin', 'learn']
Max Similarity in Audio 0 Cosine similarity is 0.5555555555555556
```

5.2 System 2

Figures 6, 7, 8, 9 and 10 show the LPC coefficient for five speakers. The corpus consists of 25 audio files, five from each speaker. The input voice sample is for speaker 1, correctly identified by the system, and keywords are ‘machine learning’, which after pre-processing or query normalisation becomes [‘machin’, ‘learn’]. Figure 11 shows the result of the above query.

6 Limitations and concerns

This section highlights the concerns and limitations of the processed system.

There are many documents and query databases to benchmark the system, but there is no associated speaker information with those documents. Even if we use those documents, we would not be able to evaluate the first two steps: speaker identification and speech recognition. We tried to do the same thing with BBC News talk shows like Hard Talk, but again there is a limitation in the speaker identification. Audio files should be converted to a specific format for speaker identification and speech recognition, but commercial tools are required for that conversation. Another issue that arises when testing the real system is the amount of noise in the audio signal. The last issue is the overlapping sound signal when both speakers are talking, and it is not easy to separate speaker voices. So, after struggling with these issues, we decided to test the prototype version of the system on the generated dataset, which is easy to handle.

FFmpeg Python library can be used for video to audio, but there is one issue. The sub-step, speech recognition, is based on the google speech recognition library, and it requires an audio signal with a specific frequency for further processing. So, if you use FFmpeg, you have to convert the audio file to a particular format for further processing, but if the sub-step contains a method that can do speech recognition for any audio file, then FFmpeg is recommended.

7 Conclusions

We illustrated in the article how to integrate a speaker identification with typical information retrieval system technologies with minor modifications in the data structure. For speaker identification, rather than using LPC coefficients, MMFC coefficients can also be used. A library like google speech recognition can be used for voice recognition. We can use WordNet rather than TF-IDF for document retrieval, or a semantic-based information retrieval algorithm can be deployed.

References

Altszyler, E., Sigman, M. and Slezak, D.F. (2016) 'Comparative study of LSA vs. Word2Vec embeddings in small corpora: a case study in dreams database', *ArXiv*, abs/1610.01520.

Anjum, M.F., Haug, J., Alberico, S.L., Dasgupta, S., Mudumbai, R., Kennedy, M.A. and Narayanan, N.S. (2020) 'Linear predictive approaches separate field potentials in animal model of Parkinson's disease', *Frontiers in Neuroscience*, April, Vol. 14, p.394.

Bafna, P., Pramod, D. and Vaidya, A. (2016) 'Document clustering: TF-IDF approach', in *2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT)*, pp.61–66.

Budzinski, O., Gaenssle, S. and Lindstädt-Dreusicke, N. (2021) 'The battle of YouTube, TV and Netflix: an empirical analysis of competition in audiovisual media markets', *SN Business & Economics*, August, Vol. 1, No. 9, p.116.

Chauhan, N., Isshiki, T. and Li, D. (2019) 'Speaker recognition using LPC, MFCC, ZCR features with ANN and SVM classifier for large input database', in *2019 IEEE 4th International Conference on Computer and Communication Systems (ICCCS)*, pp.130–133.

Chechik, G., Ie, E., Rehn, M., Bengio, S. and Lyon, D. (2008) 'Large-scale content-based audio retrieval from text queries', in *Proceeding of the 1st ACM International Conference on Multimedia Information Retrieval – MIR'08*, ACM Press.

Convert Audio to WAV [online] https://audio.online-convert.com/convert-to-wav (accessed 12 September 2021).

From Text to Speech – Free Online TTS Service [online] http://www.fromtexttospeech.com/ (accessed 11 August 2021).

Furui, S. (2009) '40 years of progress in automatic speaker recognition', in *Advances in Biometrics*, pp.1050–1059, Springer, Berlin, Heidelberg.

Hasanah, U., Astuti, T., Wahyudi, R., Rifai, Z. and Pambudi, R.A. (2018) 'An experimental study of text preprocessing techniques for automatic short answer grading in Indonesian', in *2018 3rd International Conference on Information Technology, Information System and Electrical Engineering (ICITISEE)*, pp.230–234.

Hickman, L., Thapa, S., Tay, L., Cao, M. and Srinivasan, P. (2020) 'Text preprocessing for text mining in organizational research: review and recommendations', *Organizational Research Methods*, November, Vol. 25, No. 1, pp.114–146.

Hill, D.R. (2007) 'Speaker classification concepts: past, present and future', in *Lecture Notes in Computer Science*, pp.21–46, Springer, Berlin, Heidelberg.

Hiriyannaiah, S., Singh, K., Ashwin, H., Siddesh, G.M. and Srinivasa, K.G. (2020) 'Deep learning and its applications for content-based video retrieval', in *Hybrid Computational Intelligence for Pattern Analysis and Understanding*, pp.49–68, Elsevier.

Hou, S. and Zhou, S. (2013) 'Audio-visual-based query by example video retrieval', *Mathematical Problems in Engineering*, pp.1–8.

Intechopen (2021) *Voice Identification using Classification Algorithms*, Intechopen [online] https://www.intechopen.com/chapters/68705 (accessed 21 September 2021).

Islam, M.A., Jassim, W.A., Cheok, N.S. and Zilany, M.S.A. (2016) 'A robust speaker identification system using the responses from a model of the auditory periphery', *PLoS One*, July, Vol. 11, No. 7, p.e0158520.

Jahangir, R., Teh, Y.W., Nweke, H.F., Mujtaba, G., Al-Garadi, M.A. and Ali, I. (2021) 'Speaker identification through artificial intelligence techniques: a comprehensive review and research challenges', *Expert Systems with Applications*, June, Vol. 171, p.114591.

Jain, N. and Rastogi, S. (2019) 'Speech recognition systems – a comprehensive study of concepts and mechanism', *Acta Informatica Malaysia*, January, Vol. 3, No. 1, pp.1–3.

Kabir, M.M., Mridha, M.F., Shin, J., Jahan, I. and Ohi, A.Q. (2021) 'A survey of speaker recognition: fundamental theories, recognition methods and opportunities', *IEEE Access*, Vol. 9, pp.79236–79263.

Kim, S-W. and Gil, J-M. (2019) 'Research paper classification systems based on TF-IDF and LDA schemes', *Human-Centric Computing and Information Sciences*, August, Vol. 9, No. 1.

Koolagudi, S.G., Vishwanath, B.K., Akshatha, M. and Murthy, Y.V.S. (2016) 'Performance analysis of LPC and MFCC features in voice conversion using artificial neural networks', in *Proceedings of the International Conference on Data Engineering and Communication Technology*, Springer, Singapore, August, pp.275–280.

Leu, F-Y. and Lin, G-L. (2017) 'An MFCC-based speaker identification system', in *2017 IEEE 31st International Conference on Advanced Information Networking and Applications (AINA)*, pp.1055–1062.

Loper, E. and Bird, S. (2002) 'NLTK', in *Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics*, Association for Computational Linguistics.

Ma, L. and Zhang, Y. (2015) 'Using Word2Vec to process big text data', in *2015 IEEE International Conference on Big Data (Big Data)*, IEEE, October.

Mahapatra, A. and Biswas, S. (2011) 'Inverted indexes: types and techniques', *International Journal of Computer Science Issues*, July, Vol. 8, No. 4, pp.384–392, Mahebourg.

McFee, B., Raffel, C., Liang, D., Ellis, D., McVicar, M., Battenberg, E. and Nieto, O. (2015) 'librosa: audio and music signal analysis in Python', in *Proceedings of the 14th Python in Science Conference*, SciPy.

Mitrović, D., Zeppelzauer, M. and Breiteneder, C. (2010) 'Features for content-based audio retrieval', in *Advances in Computers*, Vol. 78, pp.71–150, Elsevier.

Mohamadzadeh, S. and Farsi, H. (2016) 'Content based video retrieval based on HDWT and sparse representation', *Image Analysis & Stereology*, April, Vol. 35, No. 2, p.67.

Muneeb, M. and Raza, Z. (2021) 'Tree-based blockchain architecture for supply chain', *International Journal of Blockchains and Cryptocurrencies*, Vol. 2, No. 2, p.143.

Muneeb, M., Raza, Z., Ul Haq, I. and Shafiq, O. (2022) 'SmartCon: a blockchain-based framework for smart contracts and transaction management', *IEEE Access*, Vol. 10, pp.10719–10730.

Nassif, A.B., Shahin, I., Attili, I., Azzeh, M. and Shaalan, K. (2019) 'Speech recognition using deep neural networks: a systematic review', *IEEE Access*, Vol. 7, pp.19143–19165.

Patel, B.V. (2012) 'Content based video retrieval systems', *International Journal of UbiComp*, April, Vol. 3, No. 2, pp.13–30.

Patil, N.M. and Nemade, M.U. (2016) 'Content-based audio classification and retrieval: a novel approach', in *2016 International Conference on Global Trends in Signal Processing, Information Computing and Communication (ICGTSPICC)*, IEEE, December, pp.599–606.

Povey, D., Burget, L., Agarwal, M., Akyazi, P., Feng, K., Ghoshal, A., Glembek, O., Goel, N.K., Karafiát, M., Rastrow, A., Rose, R.C., Schwarz, P. and Thomas, S. (2010) 'Subspace Gaussian mixture models for speech recognition', in *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp.4330–4333.

Radha, N. (2016) 'Video retrieval using speech and text in video', in *2016 International Conference on Inventive Computation Technologies (ICICT)*, Vol. 2, pp.1–6.

Ramos, J.E. (2003) *Using TF-IDF to Determine Word Relevance in Document Queries*.

Rasheed, J., Jamil, A., Yahyaoui, A. and Madey, A.S.A. (2020) 'Automatic video indexing and retrieval system for turkish videos', in *2020 28th Signal Processing and Communications Applications Conference (SIU)*, pp.1–4.

Ratanpara, T. and Patel, N. (2015) 'Singer identification using MFCC and LPC coefficients from Indian video songs', in *Emerging ICT for Bridging the Future – Proceedings of the 49th Annual Convention of the Computer Society of India (CSI)*, Vol. 1, pp.275–282, Springer International Publishing.

Raza, Z., Ul Haq, I., Muneeb, M. and Shafiq, O. (2021) 'Energy efficient multiprocessing solo mining algorithms for public blockchain systems', *Scientific Programming*, October, pp.1–13.

Sammut, C. and Webb, G.I. (2010) *Encyclopedia of Machine Learning*, Springer, USA.

Smith, M.A. and Chen, T. (2005) 'Image and video indexing and retrieval', *Multimedia Systems and Techniques*, p.993, Elsevier.

Torfi, A. (2018) 'SpeechPy – a library for speech processing and recognition', *Journal of Open Source Software*, July, Vol. 3, No. 27, p.749.

Wang, F. and Xu, W. (2014) 'A comparison of algorithms for the calculation of LPC coefficients', in *2014 International Conference on Information Science, Electronics and Electrical Engineering*, Vol. 1, pp.300–302.

Wang, X-J. and Zhang, L. (2009) 'Annotation-based image retrieval', in *Encyclopedia of Database Systems*, pp.85–88, Springer, USA.

Zhang, H.J. and Smoliar, S.W. (1994) 'Developing power tools for video indexing and retrieval', in Niblack, C.W. and Jain, R.C. (Eds.): *Storage and Retrieval for Image and Video Databases II*, SPIE, April.