Insert here your thesis' task.

Czech Technical University in Prague
Faculty of Information Technology
Department of Software Engineering

Master's thesis

System for structure discovery in multimedia metadata in the project Open-Narra

Bc. Michal Kopp

Supervisor: doc. Ing. RNDr. Martin Holeňa, CSc.

26th June 2015

Acknowledgements

I would like to thank doc. Ing. RNDr. Martin Holeňa, CSc., supervisor of this thesis, for his help and patient guidance. I would also like to thank my family for their great support. Finally, I would like to thank Petr Pulc for his help with testing data.

Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular that the Czech Technical University in Prague has the right to conclude a license agreement on the utilization of this thesis as school work under the provisions of Article 60(1) of the Act.

In Prague on 26th June 2015 ....

Czech Technical University in Prague
Faculty of Information Technology
© 2015 Michal Kopp. All rights reserved.
This thesis is school work as defined by Copyright Act of the Czech Republic. It has been submitted at Czech Technical University in Prague, Faculty of Information Technology. The thesis is protected by the Copyright Act and its usage without author's permission is prohibited (with exceptions defined by the Copyright Act).

Citation of this thesis

Kopp, Michal. *System for structure discovery in multimedia metadata in the project Open-Narra.* Master's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2015.

Abstrakt

S rostoucí popularitou multimédií, zejména audiovizuálních dat, roste také potřeba porozumět jejich vnitřní struktuře. Tato práce se zabývá navrhnutím a prozkoumáním metody, s jejíž pomocí lze hledat strukturu v multimediálních datech. Metoda využívá integraci samoorganizujících map (SOM) a hierarchického shlukování za účelem nalezení vhodné struktury shluků v multimediálních datech. Výstup z každé samoorganizující mapy je vyhodnocen pomocí několika úrovní hierarchického shlukování s různým počtem shluků, namapovaných do SOM. V těchto namapovaných úrovních se nalezne ta, jejíž průměrná vnitroshluková vzdálenost je nejmenší. Tato úroveň poté určuje nejlepší shlukování pro danou samoorganizující mapu. Uvedený postup byl použit, v experimentech se čtyřmi rozdílně velkými mapami, na téměř 16000 segmentů ze zaznamenaných přednášek. V práci je uvedeno několik příkladů interpretace získaných výsledků.

Klíčová slova audiovizuální data, hledání struktury, shlukování, SOM, hierarchické shlukování

Abstract

With the quickly rising popularity of multimedia, especially of the audiovisual data, the need to understand the inner structure of such data is increasing. In this thesis, a method for structure discovery in multimedia data is proposed. It consists in integrating a self-organizing map (SOM) and hierarchical clustering to find a suitable cluster structure in the multimedia data. The output of every SOM is evaluated by various levels of hierarchical clustering with different number of clusters mapped to the SOM. Within those mapped levels, the one with the lowest average within-cluster distance is searched, which then determines the most appropriate clustering for the map. In experiments, the proposed approach was applied, with SOMs of four different sizes, to nearly 16 000 segments of audiovisual recordings of lectures. Several examples of the interpretation of obtained results are provided.

Keywords audio-visual data, structure discovery, clustering, SOM, hierarchical clustering

Contents

List of Figures

List of Tables

Introduction

In the past few decades, the amount of audiovisual data has grown rapidly. Every day, modern technology is accessible to more and more people. With these modern devices, such as smartphones or cameras, people are able to create large amounts of audiovisual data. However, expanding modern technology also means broader internet connection coverage and while the internet connection being available in new areas, it is also gaining more speed in the old ones. With a high speed internet connection available on many modern devices, these devices are not capable of only creating audiovisual content, but they are also able to share or publish it. And of course, what smart device it would be if it can not consume what it can produce. According to the Cisco's June 2014 Visual Networking Index (VNI) [1] , global data traffic reached 51 168 petabytes per month at the end of 2013. More than one third (17 506 petabytes) was video content and Cisco predicts that it will be almost one half of all Internet traffic by the end of 2018.

Growing popularity of audiovisual data is the main reason why there are often created many new sites containing audiovisual data. From the consumer point of view the the main purpose of such a site is to find the video the user is looking for. While the users typically have a very clear idea of the content they wants to get, the producer, publisher or anyone who uploaded the video can not know at the time of its uploading what the user in the future will be searching for.

When uploading a video on the YouTube or another similar site, the user is usually able to edit the title of the video and add some short text description. On some sites, users can choose a category which video corresponds to, or they can just simply add some tags, describing the video in a small set of simple words. This helps a lot when searching for the video, but when the tags of a video do not match searched phrase, the video will be ranked very low in result list or it will not be found at all.

The search engine of such a big site as the YouTube has to be very fast and relatively simple because of the amount of queries it has to process. According to [2], YouTube is the world second largest search engine with more than 3 billion searches a month.

Project Narra

An alternative approach would be to use the data which is a part of the audio-video itself (let us say we can extract it in a form of metadata) and try to find some structure in it, analyse it and enable it for knowledge extraction? That is the idea behind the Narra project at the Film and TV School of Academy of Performing Arts in Prague (FAMU). As presented on the Narra project website [3], "Narra is an open source software for visualizing, annotating and linking audio-visual media (video, photos, sounds) and text." From a high level point of view, it means that when the project is finished and all parts of it are working together, user can search for desired audio or video described by some common attributes such as audio or video title, length, date of publish etc. However, what is more important is an ability to link the data together and create structured representation which can be visualized as a tree of linked audio and video files. So, the whole media work can have multiple path which could tell multiple stories instead of linear story line.

Another great use of Narra would be an automatic annotation of the media files. All authors can annotate their work manually with any authors description they want. However, in many real world scenarios, there is a lot of audio-visual data created in the shooting process and only a small part of this data is used in the final media work. While the final media work will most probably be manually annotated by the author, other audio-visual data created during filming will not be annotated this way in the most cases. However, leaving this data not annotated would be a great loss, because there is a possibility that this data contains a lot of useful information and although it has not been used for the final author work, it could serve great to someone else.

At that point, a software capable of an automatic annotation comes very handy. Narra uses some tools for creating technical metadata about video, such as a size of the frame, duration, camcorder type, location and many more. Some of this metadata can be used for knowledge extraction and structure discovery in the audio and video data, however, there are also another important types of metadata that can be generated automatically and can help structure discovery a lot. The extracted texts from images or transformed speech from audio files to texts are good examples. Such metadata extraction methods are one of the main point of interest of Peter Pulc's master's thesis [4]. In his thesis, he used some metadata extraction methods on slides and videos recorded during various lectures. Petr is also involved in Narra project and methods for metadata extraction proposed in his thesis which will allow automatic addition of annotation (addition of metadata) are going to be integrated in the project. Further knowledge extraction and structure discovery, which is the main part of this thesis, is based on the same data generated by the methods that were proposed in the Peter's thesis.

As was originally intended by FAMU, I used an Artificial Neural Network for structure discovery in the data. The data was first clustered by Self-organizing map (SOM) Neural Network[5] and the results were compared to the hierarchical clustering.

Chapter 1

State-of-the-art

1.1 Hierarchical clustering

General purpose of clustering is grouping objects that are similar into same group. There are 2 main kinds of clustering – the number of clusters is a priori known or it is unknown, in which case the clusters are formed hierarchically. Hierarchical clustering can also capture all levels of similarities, with respect to how the similarity is defined.

The result of hierarchical clustering is a multilevel hierarchy of clusters, where two clusters at one level are joined as a cluster at the next level. At the bottom level every single cluster corresponds to a single observation.

Decision, which clusters will be joined on the higher level, is based on a similarity between these clusters, which is measured in respect of a chosen distance metric between observations / clusters at the previous level of the cluster hierarchy and a linkage criterion. The distance metric determines how similarity between two observation is calculated, while the linkage criterion determines how the distance metric is employed to calculate the similarity between two clusters. The Euclidean distance and the Ward's criterion were chosen as the metric and the linkage criterion in all experiments. The Euclidean distance was chosen because it the most common distance metric being used. The Ward's criterion (also called the minimum variance criterion or the inner squared distance) was chosen because it minimizes the total within-cluster variance after a potential merge of the two clusters. This variance is calculated as a weighted Euclidean distance between clusters' centres. The weight w is calculated as:

$$w_{r,s} = \sqrt{\frac{2n_r n_s}{(n_r + n_s)}} \tag{1.1}$$

where n_r and n_s are the number of elements in clusters r and s.

1.2 Self-organizing map

Self-organizing map (or Self-organizing feature map)[5] is a kind of an artificial neural network. The main idea behind SOM is that it can preserve topological relations of the input space, which is typically high-dimensional, in a low dimensional map (typically 2-D). That means the data which is similar in their original high-dimensional input space is also similar in the map. Due to this characteristics, the high-dimensional data whose relationships are often very hard to visualize in the original high-dimensional space can be relative easily visualized in the low-dimensional map. SOM does not only learn the distribution of the input data, but it also learns a topology.

Since in all experiments I used a 2-D map, which is also the most common type of SOM, the following description will be restricted just to that specific case.

SOM consists of one layer of neurons which are organized in some regular topology according to a topology function, which determines how the neurons in the map are organized. Most common are the grid, hexagonal and random topology. Also, a distance function, which measures some kind of distance between neurons needs to be selected. Common distance functions are Euclidean distance, Manhattan distance or the length of the shortest path between neurons.

Each neuron in the map is assigned a weight vector which is of the same dimension as the input vectors. The weight vectors are first initialized, most easily with samples from the input data or with small random values.

Training the SOM is an iterative process. In each iteration, one sample vector X is selected and it is presented to the map. A similarity between the selected feature vector and all weight vectors in the map is calculated, usually as Euclidean distance. The neuron that has the most similar weight vector is selected as a winning neuron c.

When the winning neuron is found, the weight vectors of the winning neuron and its neighbors are updated. They are moved closer to the presented input vector, with decreasing magnitude of changes according to the growing distance from the winning neuron. The update rule for a neuron i with weight vector $W_i(t)$ is:

$$W_i(t+1) = W_i(t) + \alpha(t)\theta(c,i,t)(X(t) - W_i(t)) \tag{1.2}$$

where $\alpha(t)$ is learning rate, which is from the interval $(0,1)$ and it is decreasing with increasing iterations, and $\theta(c,i,t)$ is the neighborhood function which depends on the distance between the winning neuron c and the neuron i, and may depend also on the iteration t.

Another possibilty for SOM training is the batch algorithm[6]. In each epoch, this algorithm presents the whole training data set to the network at once. The winning neurons for all the input data are selected and each weight vector is modified according to the position of all the input vectors for which it is a winner or for which it is in the neighborhood of a winner.

In all experiments with SOM, I used the MATLAB's Neural network toolbox implementation of SOM, which uses the batch algorithm by default.

1.3 Related work

Studies for utilizing SOM as a tool for clustering multimedia data were proposed in [7, 8].

1.3.1 PocketSOM

In the PocketSOM[7], the SOM is utilized for a creation of the map of songs. The songs are characterized by their acoustic attributes using Rhythm Patterns[9]. These attributes are used for the creation of a high dimensional vector of features which represents the original song. The PocketSOM utilization of the song's acoustic attributes leads to the extraction of many of these features. In [7], the extraction of the acoustic attributes generated more than thousand features. This is quite similar to approach to the multimedia clustering proposed in this thesis. However, I included even more features describing the content of multimedia files because I was able to use more than one kind of features due to the multimodality of the audiovisual data.

There are also some other differences, in particular between the main purpose of the PocketSOM and clustering method proposed in this thesis. The main purpose of the PocketSOM is to provide the user with an ability to build his or her own playlists. The user can choose the songs by virtually drawing a path in the map. The songs which are mapped to the nodes (neurons) on the path, are selected to be included in the playlist and the user has to decide which of them will be selected for the final playlist creation. It is quite obvious that the user is involved a lot in the generation of the playlists. However, structure discovery method proposed in this thesis is user-independent.

Till the SOM has been trained, PocketSOM and application of SOM proposed in this thesis proceed in a similar way. However, after the map has been trained, PocketSOM starts involving the user, whereas this application integrates the SOM with hierarchical clustering.

1.3.2 Music Map

Method proposed in this thesis has also some similarities with the Music Map[8], which also utilizes SOM for the music playlist generation. Both, the Music Map and this application utilize SOM to create a two dimensional map of similarities in the input data. However, the purpose of each of both applications is rather different. The Music Map is used for playlist generation from a collection of songs. The user's preferences for the songs that should be included in the playlist are transformed to an optimization criterion for the path planing. Then the algorithm that traverses the map and optimizes the criterion is executed, and the resulting optimal path determines the generated playlist. Although the user is not involved in the Music Map playlist generation as much as in the PocketSOM playlist generation, the main idea remains the same – to provide the user a possibility to create a playlist that corresponds to his or her demands. In the Music Map, the trained SOM is used as a background layer which is able to discover and preserve some structure in the data, although the main purpose is to traverse the map and generate the final product – the playlist. Differently to the Music Map and PocketSOM, this application does not primarily focus on any final product generated from the map. It rather tries to discover structure in the employed data.

There is also another difference between the Music Map and this method – the dimensionality of the feature vector. In the Music Map, every song is described by nine features. However, in this application are used use more than five thousands features. This also means that the training of the map takes much more time.

1.3.3 SOM as a clustering tool

Studies utilizing SOM as a clustering tool were also proposed in [10, 11]. In [10], a SOM is used as a one of the clustering tools for the analysis of embryonic stem cells gene expression data. The important difference compared to approach used in this thesis is that each neuron represents one cluster with a strict boundaries, so there are as many clusters as neurons in the SOM. Then it is easy to measure the within cluster distance and between cluster distance to provide an information about the overall clusters' quality.

The study[11] deals with cluster quality evaluation. Similarly as in [10], a SOM is used as a clustering tool where every neuron corresponds to a one cluster and it is treated this way when quality of a clustering solution is being measured. Thus, the overall number of clusters is determined by the size of the 2-D map.

Approach used in this thesis, on the other hand, relies on combining the information from SOM with information from hierarchical clustering.

1.3.4 SOM Summary

The first two utilizations of SOM, where it is used as a fundamental part in a higher concept, show how it can be used for the dimensions reduction while preserving the basic relations between the data. In both applications, the map servers as a data organizing layer which is thereafter traversed in a manual or some more sophisticated automatic way. The PlaySOM was constructed and tested on a high-multidimensional data, all generated through the automatic extraction algorithm, whereas in the case of Music Map, there was a relatively low-dimension metadata created mostly by annotating the input data from an external source of information.

Both proposed usages of SOM are conceptually quite close to usage of SOM in this thesis. Basically, the main difference is in the interpretation of map result since there is no usage of any path planning in this system and also it is user-independent. However, some proposed traversing techniques could be also incorporated in the future as it could help when finding similarities and/or differences in the data.

1.4 SOM cluster quality evaluating methods

An output from the SOM can be interpreted as a clustering solution so we could consider using some well know methods and criteria for cluster quality evaluation, such as Calinski-Harabasz[12], silhouette[13] or Davies-Bouldin[14]. The problem is that all of these methods expect that every single observation belongs to exactly one cluster. However, this assumption can not be easily applied to the SOM output if it is interpreted in the way proposed in this thesis.

In the SOM output, an every input observation is assigned to a one node (neuron), so when a single node is interpreted as a one cluster, the "every single observation – one cluster" condition is satisfied and the methods can be used. However, this interpretation of SOM output is not exactly what we want because there usually are more clusters distributed across many neurons and moreover those distributed clusters often have no strictly distinguishable borders, so they could overlap (which is something we may or may not want, according to our intention).

In previously mentioned studies [8] and [7] about SOM being utilized in playlist generation, there basically is no automatic cluster quality evaluation, because it is not the main purpose of the maps used in these applications. However, in studies [10] and [11] the SOM is primarily used as a clustering tool and its output is interpreted as a clustering solution so the quality of clusters is measured.

- First study[10], which is about comparison of clustering tools, utilizes SOM as a one of them. First, the map size was set to the desired number of clusters. Only numbers of clusters that have the square root of an integer were chosen, so only the squared maps were used. Then a training started. Once the training had been completed, cluster evaluation methods (such as Calinski-Harabasz, silhouette) were applied. A cluster in the SOM was defined as any single node in the map. The quality was measured in comparison to other clustering tools in respect to some well defined criteria.

- Second study[11] proposed SOM as another clustering tool. Clusters and number of clusters were chosen similarly as in [10]. The quality was also measured when interpreting every neuron as a cluster, comparing results to another clustering algorithm.

The main disadvantage in both usages proposed in [10] and [11] is that both of them assume that every single node in the map is exactly one cluster. Thus, common methods for a cluster quality evaluation can be applied since clusters are easily distinguishable and do not overlap. However, this is not the case. In the data supplied from Narra project (and also particularly in the data2.2 finally used in this thesis) we do not apriori know the number of clusters the data should be separated to. Thus we can not say that the clusters do not overlap nor that they corresponds to just single neurons.

The method for measuring quality of clustering solutions of the SOM proposed in this thesis is based on the different approach, described in details in section 3.1.

Chapter **2**

Employed data

Mutlimedia is usually seen as a combination of text, audio, still images, animation, video, or interactivity content forms. When speaking about structure discovery in the data, we have to know something about the content of a data itself. For example, if we want to compare two videos, we can not just say "both are videos, so the are the same." Of course, from the high-level point of view, if we do not need to know anything else than that the both files are the video files, we can say they are of the same type. But when we go further with our requirements and we want to measure similarity between them, we will need to involve more information.

As a starting point, we can choose some well known attributes of the video, such as duration, size of frame, codec etc. But this could still not be enough for our purposes. When we really want to compare one video to another, we would need an information about what is in their content. We would like to know if someone speaks in the video and what was said if so. We would also need a knowledge about the scenes, whether it was shot outside or inside, we would like to know what important things are in the videos.

To provide this sort of information about the videos, we first need to perform some knowledge discovery in the considered multimedia data. To analyse what was said in a video, we can use speech recognition which results in a text file containing recognized words. To analyse scenes, we can make histograms of colours and/or Speeded-Up Robust Features (SURF) analyses at some defined points in a time. We can extract many other kinds of information. All this information could be processed even further to make it more compact, precise etc.

All the processed information we store about a video is metadata. It is a data about a video, a data that describes a data. Metadata will be used for a structure discovery in a multimedia files as it contains the most information about the multimedia itself.

In the beginning of this thesis I mentioned that, for the topic of the thesis, there are two sources of data. The first source is the Narra project which is in development by FAMU, and the second is the data from Petr Pulc's master's thesis.

At the beginning, it was intended to use the data from the Narra project. Later it has turned out, that the data from Narra is not yet available so for the purposes of this thesis, I used the same data as Petr Pulc used in his master's thesis with some minor changes. This will be covered later in this chapter.

At the beginning of this chapter, I am going to briefly introduce some basic concepts of Narra and its data, even though I did not used any data from the project in this thesis. The reason is that in my opinion, solutions proposed later in this thesis can be used within Narra itself once the Narra data are available.

2.1 Data from the Narra project

Narra is a project in development by FAMU. Once completed, it should provide many great functions when working with audio-video content. Some of the main ideas behind the project, as stated on the official Narra website[1] are following:

- **Software as a tool for collaborative storytelling** – Creating media works which can have multiple paths, multiple versions. The software can be used to create, visualize and navigate a tree of linked stories.
- **Software as an editing tool** – Video editors can have hours and hours of audio-video material. They can annotate it a then organized into some structured search categories. Narra itself will make automated annotations and add them to the users' content. For example the software can perform a speech to text recognition and add the recognized words to the attached text file. It can also add an information about location, size of video, length etc.
- **Software as a tool for audiovisual scholarship** – Social scientists using video in their research need extensive annotation capabilities. Once there is enough amount of information about the audio-video file, Narra can apply external searches for additional associated information and display it along with the original file. Narra can be used to search trough an external research project's files and show founded information alongside the original audio-video.

When we look at the main functions of Narra listed above we see that in every point there are big needs for annotated data (we can also call these annotations as metadata). Thus, Narra needs to create and store them in some structured way. For the data storage, authors decided to use the MongoDB[2]

[1]Narra website: `http://www.narra.eu`

[2]MongoDB website: `https://www.mongodb.org/`

Attribute name	Sample value
audio_channels	"2"
audio_codec	"aac (mp4a / 0x6134706D)"
audio_proxy	"http://narra-storage/..."
audio_sample_rate	"48000"
author	"Jmeno Prijmeni"
bitrate	"3735"
colorspace	"yuv420p(tv, bt709)"
duration	"4.84"
frame_rate	"25.0"
height	"720"
library	"Richard Stallman"
location	"Sarajevo, Bosnia and Herzegovina"
name	"00081E-2"
resolution	"1280x720"
shot_type	"Medium Shot, Handheld Shot"
size	"2260148"
thumbnail_00000	"http://narra-storage/..."
timecode	"11:42:12:05"
transcription_0	"Freedom means having control of your own life."
type	"video"
url	"http://narra-ingest/..."
video_codec	"h264 (Main) (avc1 / 0x31637661)"
video_proxy_hq	"http://narra-storage/..."
video_proxy_lq	"http://narra-storage/..."
width	"1280"

Table 2.1: Narra – video fragment metadata

At the time I was given a copy of the Narra test database, there were about hundreds of short video fragments and every single one had about 25 metadata attached which described it. While the MongoDB is a document oriented database and there is no strict restriction on semantic of any document, it was pretty clear that every video fragment could have any type of metadata attached. Table 2.1 shows common metadata used for describing any video fragment.

If we look at the table, we can see that there are just a few attributes which could be of any use for structure discovery in data. The location, the transcription and maybe author. These are attributes that bring some information about content of video itself – the best seem to be attribute transcription. Other attributes are basically just a technical description of the audio-video fragment.

Of course we could search trough these attributes to find similar media content which is very useful function and it will be used in Narra plenty of times for sure. However, that is not a goal of this part of the project nor it is the goal of this thesis.

2.2 Data from CTU's A/V club

The data in which we would like to search structure has been created over the period of several years during the Weeks of Science and Technology, a two-week science festivals organized by the Czech Academy of Sciences. Altogether, the data includes more than 100 hours of multimedia content. For the purposes of further preprocessing and data consistency, only lectures and their related content in the Czech language have been chosen.

All lectures were recorded with Mediasite recorder, a platform maintained by Sonic Foundry, Inc. The idea behind the Mediasite platform is to enable streaming of lectures as easy as possible. The main element of the system is Mediasite server which stores the lectures and streams them to the internet. On the side of the lecturer, only audio and video sources are connected to the device.

The data consist of three main modalities – synchronized video with sound and slides from lecturer's presentation. The slides are captured directly from the lecturer's presentation through the VGA/DVI card of Mediasite, which is supplied with the same signal as the projector.

The signal input for the slides is continuously monitored and any change to it above given threshold is considered as a new slide. Mediasite stores the image alongside with the timestamps of slide transition used for the synchronization during playback.

Audio is recorded through Mediasite just as lecturer speaks. It is synchronized with the video and slides by the use of timestamps of slide transitions.

The Mediasite platform only provides a video signal in the resolution up to 240 rows and the bitrate of 350Kbps. However, for purpose of the lectures the high-definition video is not necessary, because we have the slides in a good resolution synchronized with the videos. Also, during the lecture, there are usually not many changes happening on the scene, since only the lecturer is supposed to move significantly.

2.2.1 Data composition

The recording of a lecture consists of the following synchronized parts - motion video, sound, sequence of images (slides).

2.2.1.1 Slides

Mediasite recorders are capable of capturing DVI/VGA input. Any greater change to this input (above some threshold) is considered as a new slide which is attached by the timestamps of the change.

Slides are then processed by an Optical Character Recognition tool. Various OCR tools were experimented with. First, it was `cuneiform` tool, which provided good results only for black-on-white text. This of course is no satisfactory so another tool called `tesseract-ocr` was used. This is now an open-source software released under the Apache License 2.0. Tesseract binarises the input image to simplify recognition process and although Petr tried to binarise it by himself, results did not improved.

OCR only works for slides with some text. However, there are some slides without text, for example slides with just an image. Thus, Petr was thinking about a use of some image recognition tool (database), but it showed up that it would be very difficult also because Google's Image Search, which is the best tool for such a thing, has no public API. Also it would be huge amount of work to make image recognition working. Applying visual feature extraction, which will be described in the next few sections, also does not work because part of the text (characters' outline have a great gradient) will be taken as a points of interest, which we do not want.

2.2.1.2 Audio

Sound signal can be a great source of information. In this thesis, I used sound processed through a speech recognition tool. Such an output, after some processing is applied, will usually be quite similar to an output of OCR to some extent. This is due to the fact that our input videos are recorded lectures. Thus, unless speaker explains broader context of topic or unless topic is changed unpredictably, text in the slides should correspond to what speaker says. This is be also different according to what lecture is about or on which event it is presented. Sometimes slides serves as a studying material, sometimes they are just a guideline for a lecturer.

Speech recognition is quite difficult but also quite often studied problem. Czech language is considered as one of more difficult mainly because of its inflection. There are some projects in the Czech Republic, mainly at universities, that study the speech recognition of Czech language problem. However, these projects are not open source so they were not used since Petr's framework was intended to be an open source. He used, at the time of creating the framework, publicly available Google's API[3] for speech recognition.[4]

[3]API url: `https://www.google.com/speech-api/v2/recognize`

[4]It seems a key for the API is no longer available to all public. It is still available to members of `chromium-dev` group though.

The limitation to this solution was that only 25 seconds or shorter audio files were supported by the API. I found[5] that as of now the maximal length of an audio file was shortened to just 15 seconds and also only 50 request per day can be made.

Nevertheless, audio files were successfully sliced to acceptable length and were processed through Google's speech recognition tools. As proposed in Petr's thesis, 11% of audio clips were not recognized a transcribed due to splitting, background noise (music) or just nature of Google's speech recognition service.

2.2.1.3 Video

Video is even more information-full than audio. There are many ways we could extract some information right from the video, for example face recognition, object recognition, camcorder movement etc. The higher the resolution/quality is the better extraction can be applied. However, this is not the case of our data. Mediasite recorded videos only in a resolution of 240 rows. Yet this is not a problem, because we had full quality slides recorded or handouts synchronized with the video. With respect to the resolution of the videos only visual feature and histogram extraction were applied. Both extractions were done on one frame from the middle of the part of video that corresponds to a one slide. Single frame from the video part should be sufficient since the lectures usually do not contain fast movements and are quite static.

SURF – Extraction of the local image features is a process of finding some significant points in the picture. Point are searched according to their neighbourhood gradient. Then, points are attached with values describing their neighbourhoods. Then the SURF is calculated from these points (for detailed description of SURF calculation, see [15]).

Histogram – SURFs are made only from grey scale images. It would be good if we have some information about colours in the video. For that reason we use histogram. Every image is described by a count of pixels with a certain colour value in all three colour channels.

2.2.2 Data preprocessing

The multimedia data described in the previous section is saved by the Mediasite system as audio, video and slides files in their respective formats. However, for the purposes of knowledge discovery in that data, including the discovery of its inner structure, we need all the data to be preprocessed to a suitable form.

[5]`https://github.com/gillesdemey/google-speech-v2`

Because there are three modalities in the data, I treated each one separately, so an appropriate feature extraction tools can be applied .

First, Optical Character Recognition (OCR) system for the slides was Used. The slides usually contain some text related to the topic or the subtopic of the lecture, so an OCR can give us a good insight into the content of the slides. The downside of this approach are slides, which contains, for example, just an image. For these slides OCR often returns nothing. However, we still have the other modalities we can rely on.

To this end, the OCR system Tesseract was used, which is an open-source software made publicly available by Google. The problem of the OCR output is that it sometimes contains a lot of unrecognized characters and misspelled words, so a basic text processing was applied. All punctuation is removed. To reduce the dimensionality of the data before further processing, only the words found in the spellchecking dictionary are considered. Also, a stemmer is applied.

Second, a speech recognition tool was used for the audio files. Because only lectures in the Czech language are considered, Google's API for speech recognition, which supports this language, was used. The result of the API call is a recognized speech converted to a text. To reduce misspelled words and the dimensionality of the data, the same methods as for OCR are applied, resulting in a text document for every slide, aggregated from the OCR and the speech recognition.

The text data returned from the speech and slide recognition have to be transformed into term-by-document matrices. For term weighting on cleaned text data, a tf-idf (term frequency – inverse document frequency) scheme has been used [16]:

$$W_{t,d} = (n_{t,d}/max_t(n_{t,d})) \cdot log_2(n_D/n_{D,t}) \tag{2.1}$$

where $n_{t,d}$ is the count of term t in document d, n_D denotes the number of documents in the employed corpus and $n_{D,t}$ is count of documents from the corpus that contain at least one occurrence of term t. Logarithmic weighting in idf part is used to not penalize relatively frequent terms as much.

As the resulting matrices had tens of thousands of columns there is a need to minimize the impact of the curse of dimensionality. Thus, the Latent Semantic Analysis [17] has been used. To this end, the k largest eigenvalues of the term-by-document matrix of weights (2.1), corresponding to the most significant concepts, and their associated eigenvectors are found first. Let U_k, Σ_k and V_k be the matrices resulting from the singular value decomposition of the term-by-document matrix, and k denotes the number of the largest eigenvalues.

To obtain dense matrices of significantly lower dimensions, a concept-by-document matrix [18] is then computed as:

$$C_k = \Sigma_k V_k^\top \tag{2.2}$$

Third, the video is used by two different extraction methods – Speeded Up Robust Features (SURF) and colour histogram. Both methods work with an image, so just one frame from the centre of the video's time span related to a slide is taken. SURF is used to find visual descriptors in the image. We have almost 16 000 slides, therefore the same number of images taken from the videos, and each of these images produced nearly a hundred SURF descriptors – summing up to over a million descriptors in total. To reduce the dimensionality, a dictionary of SURF descriptors concepts was created by k-means clustering of the original SURF descriptors with k empirically set to 400. The detailed description of the SURF usage with our data can be found in [4].

Because the matrix of these 400 descriptor concepts was still very sparse, I decided to cluster it even further, by hierarchical clustering. This resulted in 32 final descriptor concepts.

The histogram describes each image (taken from the video) by count of pixels with certain colour value in each colour channel. The RGB coding with 1 byte depth is used. Each colour count is relatively scaled to the size of the image.

All three kinds of features are combined into feature vectors, their resulting dimension is 5800 features.

Chapter 3
Analysis and design

3.1 Integrating hierarchical clustering with SOM

During each epoch of the batch algorithm that is used in the training process of the SOM, each vector of values of input features is assigned to the neuron that is the winner for that input feature vector. Thus, after the training process is completed, each neuron in the map could be interpreted as a cluster which is formed by a subset of the input feature vectors, those for which it is a winner.

In many scenarios, for example in studies [10] and [11], the output of SOM is interpreted in that way. However, this means that every neuron in the map is interpreted as a single cluster.

In this thesis, I use a different approach – an integration of SOM and hierarchical clustering. First, SOM is trained with all the input feature vectors. The resulting map serves as a reference for all clustering solutions obtained by cutting the cluster hierarchy. These solutions are mapped to that reference SOM and then evaluated.

After the reference SOM has been created, hierarchical clustering is performed. The cluster hierarchy is cut at many different levels and the clusters at the respective level are used as a starting point to calculate the distance between their respective vectors positioned in the map. When evaluating a solution, all feature vectors that form a cluster by hierarchical clustering are taken, the positions of their respective neurons in the map are found and the distance between each pair of them is calculated according to the distance function used in the map. From these distances between all pairs, the average distance is calculated, which is interpreted as a within-cluster distance.

From those within-cluster distances, an average distance of the whole solution is calculated. That average distance is used as a measure of quality of each clustering solution obtained by cutting the cluster hierarchy. I also interpret it as a measure of similarity between the map output and the hierarchical clustering output – the smaller the within-cluster distance is, the more similar

SOM size	# of clusters range	step	# of solutions
8×8	16 – 256	4	61
12×12	16 – 500	4	122
16×16	32 – 800	8	97
20×20	32 – 1000	8	122

Table 3.1: Settings used in the performed experiments

the outputs of SOM and hierarchical clustering are. I also consider that the more similar they are (the smaller the within-cluster distance is), the better the number of clusters produced from cutting the tree suits the map.

3.2 Evaluation Methodology

For experimental evaluation, There is a need to train several differently sized SOMs first. Data from all available lectures has been used for the training to make the SOMs as precise as possible. However, with 5800 features, the data dimensionality is quite high and so is the number of feature vectors – 15953. Due to the time complexity of SOM training and also with respect to the size of our data, only relatively small maps were used. Namely, only the squared maps with sizes of 8×8, 12×12, 16×16 and 20×20 neurons are considered. The number of training epochs was set empirically, taking into account the time taken by the training process, to 200 for the three smaller maps and 150 for the largest one.

Even though outputs from many levels of the hierarchical clustering are needed, the cluster hierarchy was constructed only once. It is important to realize that the hierarchy must be constructed from the same set of feature vectors as the SOMs to make the comparison possible. In all considered clustering solutions, the same cluster hierarchy was considered, cut at an appropriate level to obtain a solution with the desired number of clusters.

A different number of clustering solutions for different sized SOMs was used. I started comparison with hierarchical clustering output containing just a few clusters and ended with an output containing three or for times more clusters than the number of neurons present in the respective map. Different steps for increasing the number of clusters, depending on the map size, were also introduced.

In Table 3.1, the empirically chosen values for the experiments are shown. The number of clusters range is an interval which determines, together with the step size, how many clustering solutions produced by cutting the tree were evaluated.

3.3 Interpretation of SOMs results

The results of SOMs are also interesting to interpret. When we take a result of SOM and look at the original input data, we can easily find out if the SOM made a great clustering. In experiments, I have selected several SOMs' outputs and compared them with their respective input data. By those comparison we can easily see how similar the original input data mapped close together are.

In most of the examples, the result of such comparisons are quite expected. Some of them are a little harder to interpret, but when the nature of the SOM and our testing data are taken into account, it shows that SOMs behave according to expectations and are able to find similarities and/or dissimilarities in the data.

Chapter 4

Realisation

4.1 Evaluation Results

In this section I am going to use three types of figures to illustrate the measured results. Because these figure types are used repeatedly, their interpretation will be first shortly described.

The first type shows the map topology (which is hexagonal in all experiments) with a relative distribution of the input feature vectors. Each neuron is represented as a white hexagon with a blue patch. The bigger the blue patch at each neuron is, the more feature vectors has been mapped to that neuron. Also in smaller maps, the numbers representing the total number of feature vectors mapped to the neurons are shown in each hexagon.

The second type is a neighbor distances map. It shows the distances between weight vectors of the neurons. The small regular hexagons are the neurons and the lines connecting them represents their direct neighbor relations. The colored patches between the neurons show how close each neuron's weight vector is to the weight vectors of its neighbors. Color range varies from yellow to black, where the darker color means the greater distance between the weight vectors.

The third type is a figure showing an average within-cluster distance depending on the number of clusters produced from the hierarchical clustering, which were mapped to the SOM as described in Section 3.1.

In both types of images which show topology of the map, the neurons are ordered in a hexagonal grid. When we want to speak about any particular neurons, we need them assigned with some serial numbers. For this purpose, in further text the same numbering system as in the Matlab will be used. The neuron in the lower left corner of the map is the first one, with the serial number equal to 1. The number of the neighbour neuron to its right is 2 and the numbers of other neurons in the same row linearly grows till the end of the row. Then, the next serial number is assigned to left-most neuron in the following upper row. The numbers grows in that way, from left to right, from bottom to top. The highest serial number, which is equal to the number of neurons in the map, has the neuron in the top-right corner.

4.1.1 8×8 map

I started the evaluation with the smallest, 8×8 map. A relative distribution of the input feature vectors onto the map is shown in Figure 4.1.

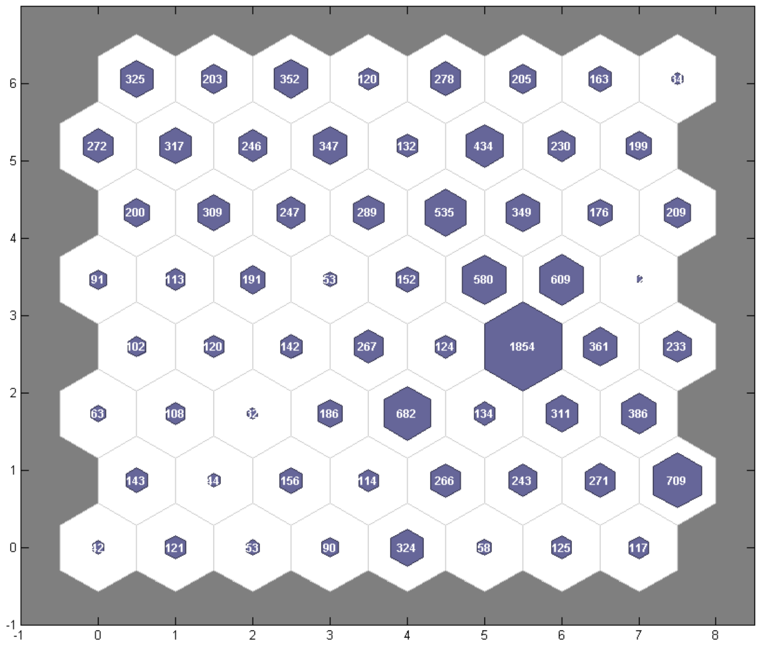

Figure 4.1: Relative distribution of the input feature vectors alongside with the total numbers of vectors mapped to each neuron in the 8×8 map.

From the distribution of feature vectors shown in Figure 4.1, we can see that among the 15 953 avilable feature vectors, 1854 vectors are mapped to the single neuron with the the highest number of assigned feature vectors. That indicates that there are many quite similar input vectors in the data. This is even more evident when we look at the neurons in its neighborhood, which are also quite a lot populated with the input vectors in contrast with the lower-left part of the map.

4.1.1.1 The most populated neuron

The neuron with the most mapped feature vectors, the one with the serial number 30, has more than twice as many feature vectors mapped as any other neuron in this map. When we look at the input data, we find that many different slides, examples are shown in Figure 4.2, are mapped to this neuron.

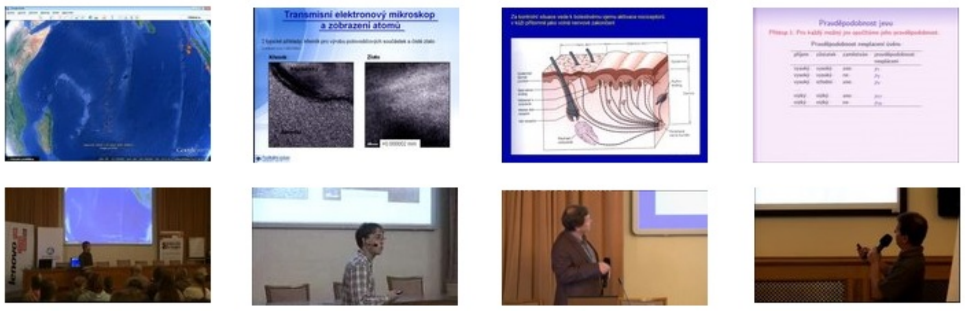

Figure 4.2: The examples of input data. First row are the slides, frames from the videos are in the second row. Unlike the others, last example is quite regular slide.

To understand this behaviour, we have to take a closer look at the map, its training process and also the input data itself. In each iteration of SOM training, the winning neuron for every feature vector is chosen. This means, that similar feature vectors have the same winning neuron and so they are mapped to it. However, in this case we have a map of size 8×8 neurons and nearly 16 000 feature vectors, so there should be approximately 250 feature vectors mapped to each neuron in average. It means that each neuron should be the winning neuron for approximately 250 feature vectors. However, there are some small similar subsets of input data, which are very different from others. These subsets of input data formed their own small distant clusters, which means that some other clusters have to be larger so that all the input data and its respective feature vectors are mapped.

The second reason for the dissimilarity between these slides is that the slides provide just one modality of our input data. Only the OCR is performed on the slides. Despite that, this modality can be crucial for computing the similarity, depending on the other modalities. Nearly a half of input data mapped to this neuron seems to be slides created from video playback or they are just images without text. This data also has in common that there is a little of recognized speech, thus the data are very similar in the speech-to-text modality and when there is a little to no text recognized by the OCR, the similarity is even bigger.

The most important difference in feature vectors is due to the SURFs and histograms produced from one frame taken from the video, but this difference can also be just a small one. In the case of lectures, the majority of lecture halls seem very similar – white walls and brown desks accompanied by projection screen.

However, there are also some inputs mapped to this neuron that seem to be regular in a common sense of a lecture – a slide with some text, video from lecture hall and average speech-recognized audio.

Quite interesting is that usually not the whole sequence of similar input data from one lecture is mapped to this neuron (mostly similar in the slides modality, because the OCR and speech recognition produced a little to nothing in the case of many input vectors of this most populated neuron). However, the input data from the sequences, which are not present in this neuron, are in the most cases mapped to its direct neighbours. It supports the idea of large clusters spread over more neighbouring neurons.

4.1.1.2 The least populated neuron

Let us investigate the neuron with the lowest number of mapped feature vectors.

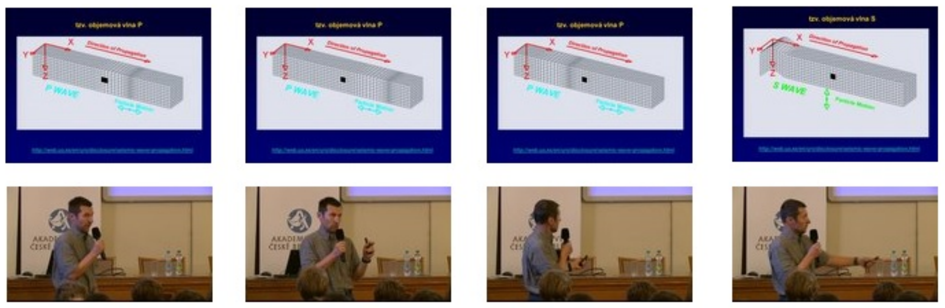

Figure 4.3: The examples of input data. First row are the slides, frames from the videos are in the second row.

It is the neuron with the serial number 40. Only 12 out of almost 16 000 feature vectors are mapped to this neuron, which is much below the average. Because of this relatively low number of inputs mapped to it, we can have a look at a great part of them. Figure 4.3 shows the examples of the slides and frames from the video which were mapped to this neuron. All of these 12 slides are very similar, with a little exception of the last one, which is also shown in Figure 4.3 as a fourth image. The OCR on these slides produced just a few words and because the slides are nearly the same, so is the OCR output.

A little more interesting situation is in the speech recognition output. It seems that the output is the same for two or three slides in the row and then it changes. In all these groups of the same recognized speech, the number of recognized words is about an average for each lecture, so it seem to be normal lecture.

From the slides, OCR and speech recognition outputs it seems that there are some duplicities in the input data. With the exception of the last input, the slides are nearly the same (there are little differences in the images, but not in the text, although the OCR sometimes recognized one or two more words). Also the output from the speech-to-text tool is exactly the same for groups of two or three inputs. However, when we look at the frame taken from the middle of the video time span, we can see that the image from the camcorder is always a little different. The lecturer in each frame is in a little different position. This means that the time span of the videos related to their respective slides are in these cases always different, and so is the frame from the middle of the time span. However, we can also suppose that these timespans overlap, because of the same speech-to-text outputs.

These "duplicities" are quite strange, because it is not very clear how they are produced. Some duplicities may occur during a recording by the Mediasite system. It scans the signal for the projector and when there is a change to the signal above some threshold, it produces a new slide with a related time span in the video signal. So, when there is, for example, a bigger noise in the projector's signal, it produces a new slide even though the slide did not changed. But this is not the case of this neuron, because all the inputs have a little different slides and when the duplicities are created the way described above, their respective timespans are different, so they are not related to the same audio. Moreover, the duplicities created by the Mediasite system had been removed before the experiments with the SOMs.

Another interesting observation about the data mapped to this neuron is, that it is not the whole sequence of the very similar slides. First, similar slides with a little different images are mapped to the direct neighbours of this neuron, which is quite similar to the case of the most populated neuron. Second, the inputs that have the similar slides, but their frames from the videos do not show the lecture hall, instead they show the projected slides, are mapped to some more distant neurons. So, in this example, the SOM was able to find the difference in the frame extracted from the middle of the video's time span and mapped those inputs to further location in the map.

4.1.1.3 The most distant neighbors

In Figure 4.4, which shows the distances between neighbors, we can see that between the neurons with the highest number of vectors, the distances are the shortest. Conversely, neurons in the lower-left corner are the most distant from each other. When we look at the input data, we find that the segments of video-recordings and associated slides projected during that segment indeed substantially differ from the remaining segments of all recorded lectures. First, the camcorder during the time span of respective segment was set to shoot only the projected slide, instead of the whole lecture hall. Second, the projected slides in most cases were examples of web pages. Moreover, in 35 out of

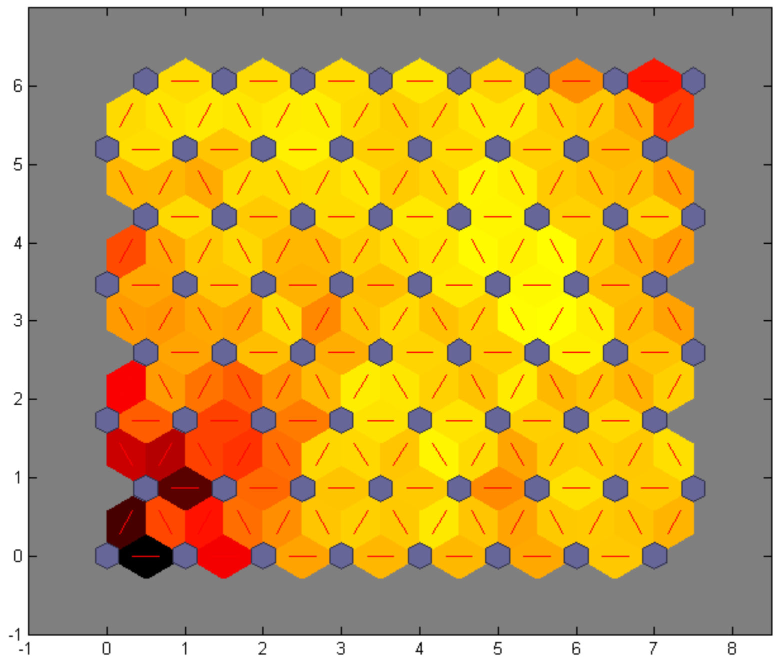

Figure 4.4: The neighbor distances in the 8×8 map.

42 slides mapped to the neuron in the bottom-left corner, the examples of web pages were taken from the Wikipedia, so they contain a lot of text and hyperlinks. The image taken from the video, containing just a slide instead of the whole lecture hall, caused the visual features and histogram to be very different. A lot of text on a Wikipedia page instead of only few lines of text on a typical slide caused the main difference in OCR.

4.1.1.4 The closest neighbors

The closest neighbors are neurons with the serial numbers 30, 38 and 39, as is indicated in Figure 4.4 by the lightest colour of the patches between them. Alongside the neuron number 30, which is by far the most populated neuron in the map, those other two neurons also belongs to the ones with the most feature vectors mapped to them. As is particularly described is Section 4.1.1.1, the input data which belongs to the most populated neuron are usually not the whole sequences of the most similar slides in the row. Some data from those sequences are missing there, so they are mapped to an another neuron. And so, when we have a look where those missing inputs were mapped, we find that it usually is one of these other two neurons

This is the main reason why these three neurons have the lowest neighbor distance in the whole map. The sequences of the very similar slides are

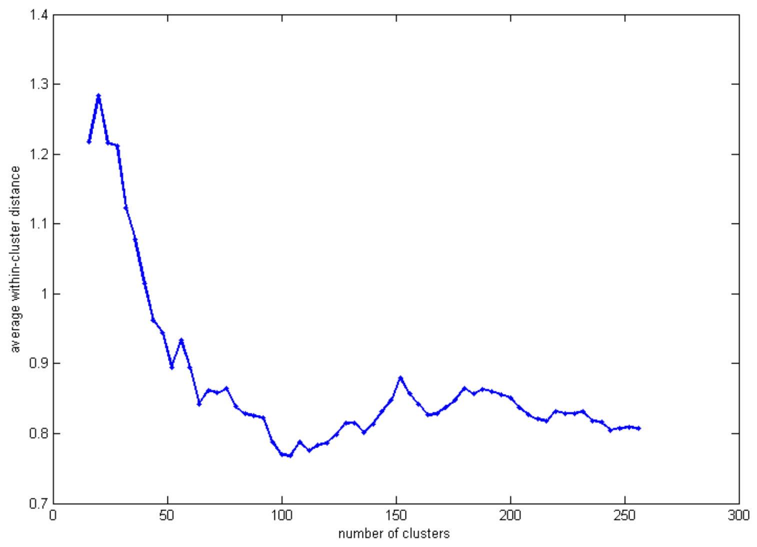

Figure 4.5: A within-cluster distance, measured as the length of the shortest path between neurons, dependent on the number of clusters produced from the hierarchical clustering, mapped to the 8×8 map.

distributed between those neurons. Sometimes, the sequences are also split between other neurons in that location of the map. We can see in Figure 4.4 that neurons located to the top of the neurons 38 and 39 also have very low neighbor distances.

4.1.1.5 SOM and hierarchical clustering fusion

Figure 4.5 shows that in the map of size 8×8 neurons, an average within-cluster distance drops down to its local minimal value when the cluster hierarchy is cut at an appropriate level to form about a hundred clusters. It means that the hierarchical clustering should produce more clusters than the number of neurons in the map, which is only 64.

4.1.2 12×12 and 16×16 maps

Let me now pass to the 12×12 and 16×16 maps. In Figures 4.6 and 4.10 we can see similar, one highly populated neuron, like in the 8×8 map. In bigger maps, the total number of vectors mapped to that neuron is lower, but in comparison to other neurons, the relative count is the same or even higher.

In Figures 4.7 and 4.11, we can see that there are some neurons, mostly on the left side of the map 4.7 and in the top-center, lower-left and top-right corners in 4.11, that have a large distance to their respective neighbors. The situation is quite similar to that of the smaller map in Figure 4.4. The

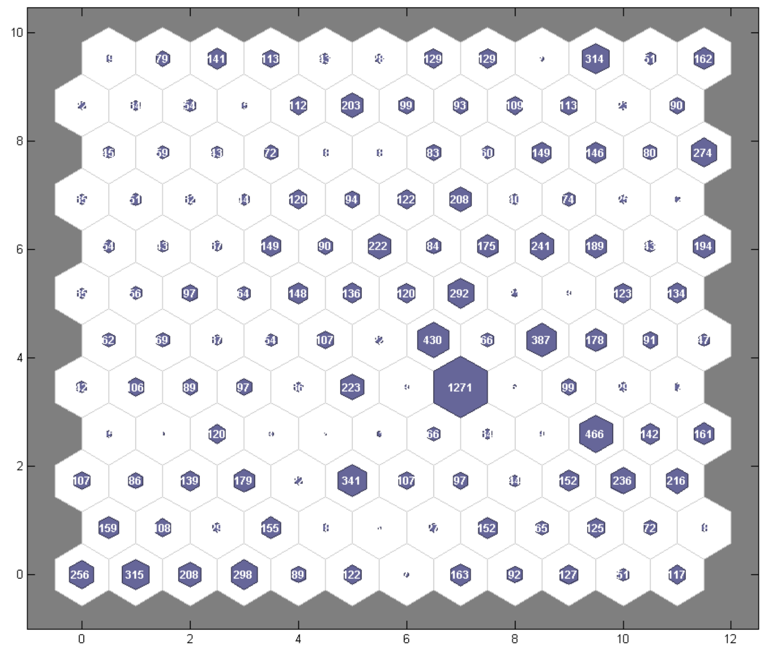
Figure 4.6: Relative distribution of the input feature vectors alongside with the total numbers of vectors mapped to each neuron in the 12 × 12 map.

darkest areas just moved around to another physical location in each of those two maps, but the relationships seems to be similar and those neurons tend to stay together.

It is also interesting, that in those larger maps a few neurons are more distant from their respective neighbors than others in their neighborhoods, which indicates that there is some input data which is quite different from the rest in the most features. This can be seen in the map as a darker "star" in a lighter neighborhood.

Let us have a look at the input data mapped to these interesting areas.

4.1.2.1 The most populated neurons

In the 12×12 map, a neuron with the most inputs mapped to it is the neuron with the serial number 56, as shown in figure 4.6. When we compare it to the most populated neuron in the smallest map, it is obvious that this neuron in the larger map has lower number of inputs mapped to it. Also, from Figure 4.7 it is clear, that the average distance of its neighbors is higher than in the case of the smallest map.

When we look at the input data mapped to this neuron, we find that the situation is quite similar to the one in the smaller 8×8 map. The majority of input data mapped to this neuron are inputs, where the slides contain

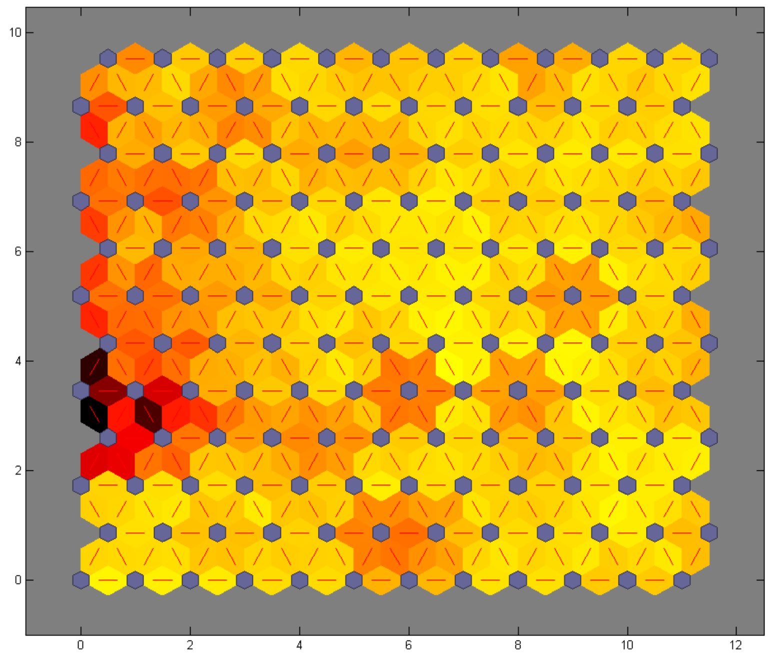

Figure 4.7: The neighbor distances in the 12 × 12 map.

pictures or the whole slide is an image. It is also usually a sequence of similar inputs and also some inputs in the sequence are missing. However, there is a difference between the input data mapped to this neuron and the input data mapped to the one in the smallest map. Unlike the one in the smallest map, this neuron does not contains as many inputs with the "regular" slides. There are still some cases where the input data mapped to this neuron are the slides without big images, but it is much lower number then in the smallest map. This is the main difference in the number of data mapped to the most populated neurons in both maps.

The second most populated neuron (with the serial number 67) in this map also happens to be the closest neighbor of the most populated neuron, as shown in Figure 4.7 by the lightest colour of the patch between them. The input data mapped to that neuron are mostly the missing inputs from sequences mapped to the neuron 56. Table4.1 shows examples of some sequences mapped to these two neurons.

Figure 4.8 shows an example of sequence which is also listed in Table4.1. We can see that the slides shown in the example are quite similar in colours and overall layout. Although some of them do not contain an image (a graph), their were mapped close together. This is mainly because we have the other modalities available than just the slides.

In Figures 4.9 and 4.12, which show the within-cluster distance dependence on the number of clusters formed by cutting the cluster hierarchy, we can not

Sequence	Neuron 56	Neuron 67	Other neurons
7068 – 7078	7068, 7069, 7072, 7073, 7075	7070, 7074, 7076, 7078	7071, 7077
213 – 228	213 – 221, 224, 226, 227	222, 223, 225	228
11169 – 11185	11169, 11171 – 11174, 11176 – 11180, 11185	11175, 11181 – 11184	11170

Table 4.1: Examples of distributed sequences. Numbers are the serial numbers of input data.

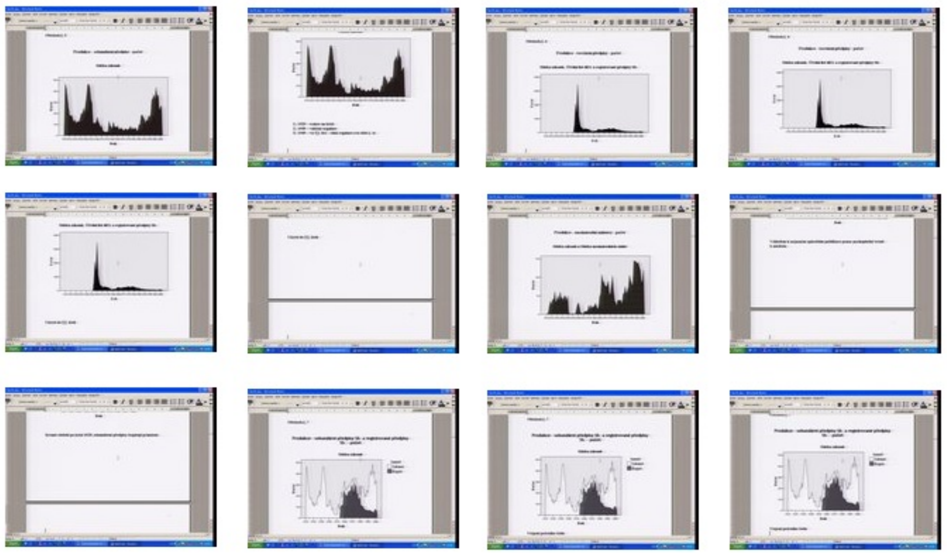

Figure 4.8: An example of sequence of slides, starting with slide number 11169 and ends with 11180. Slide 11169 is in the top-left corner and the series linearly grows in row to the right. Slide 11180 is in the bottom-right corner.

find a clear local minimum. The value of distance drops down quickly with the increasing number of clusters, forming an elbow in the figure, which is at just about a few less clusters than is the number of neurons in the map.

Even though the most suitable number of clusters in the map could not be define precisely this way when there is no distinct local minimum, locating an elbow on the curve gives us a great possible candidate when we are searching for a solution with a good average within-cluster distance and the least number of clusters.

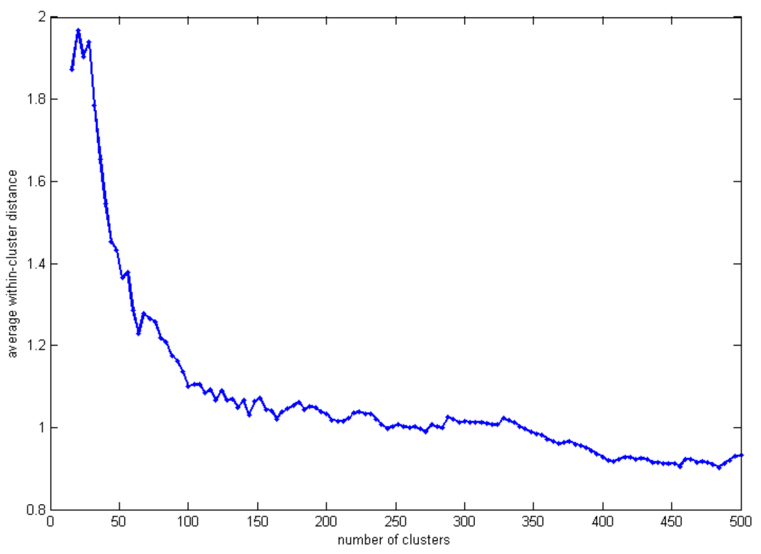

Figure 4.9: A within-cluster distance, measured as the length of the shortest path between neurons, dependent on the number of clusters produced from the hierarchical clustering, mapped to the 12×12 map.

4.1.3 20×20 map

Finally, let me pass to the largest, 20×20 map. Figure 4.13 shows that there is still one neuron with significantly more input vectors mapped to it than to any other neuron. A great part of the map is low populated with feature vectors, even though there are some neurons with small neighborhoods which are populated a little more.

In Figure 4.14, we can see that the small area of neurons distant from each other is still present, situated at the top of the map, almost in the right corner. Alongside with Figures 4.1, 4.6 and 4.10, we can also see that as the map grows, there are more distant neurons and the whole map becomes darker. This is due to the higher number of neurons which allows more distinct input feature vectors to form their own clusters.

The 20×20 map also has other interesting characteristics. In Figure 4.15, there is once again a local minimum, though in this case it can be found within the interval of 400 and 500 clusters. It seems that clustering solutions produced by hierarchical clustering and clusters formed in the map are very similar in that interval, because there is exactly 400 neurons in this map.

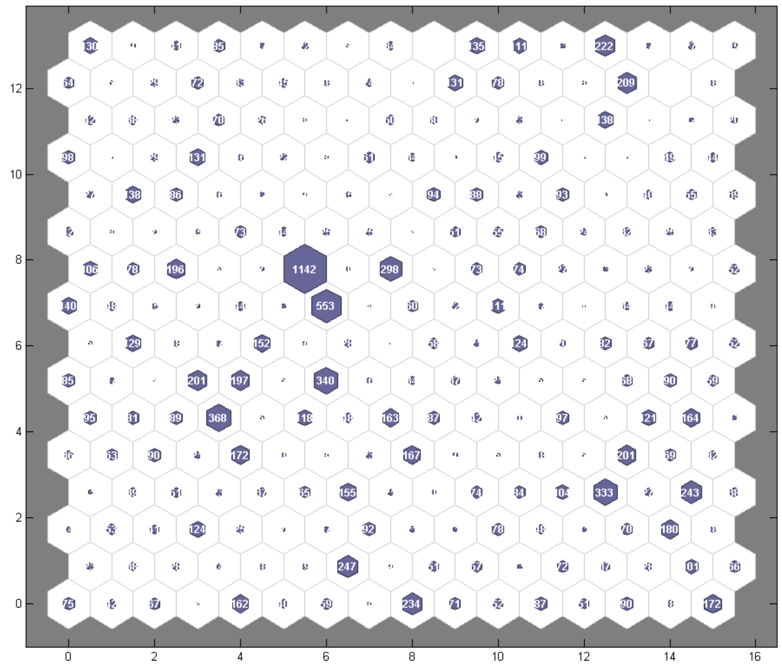

Figure 4.10: Relative distribution of the input feature vectors alongside with the total numbers of vectors mapped to each neuron in the 16 × 16 map.

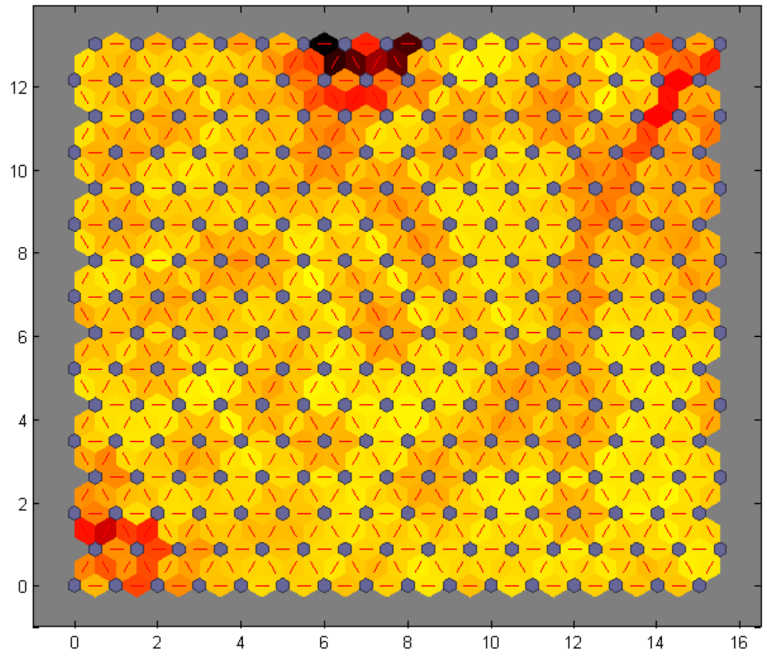

Figure 4.11: The neighbor distances in the 16 × 16 map.

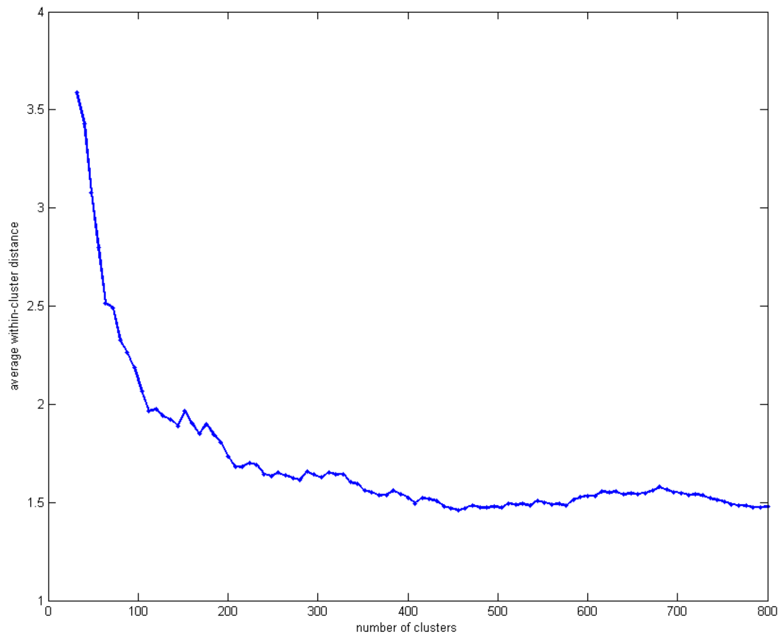

Figure 4.12: A within-cluster distance, measured as the length of the shortest path between neurons, dependent on the number of clusters produced from the hierarchical clustering, mapped to the 16 × 16 map.

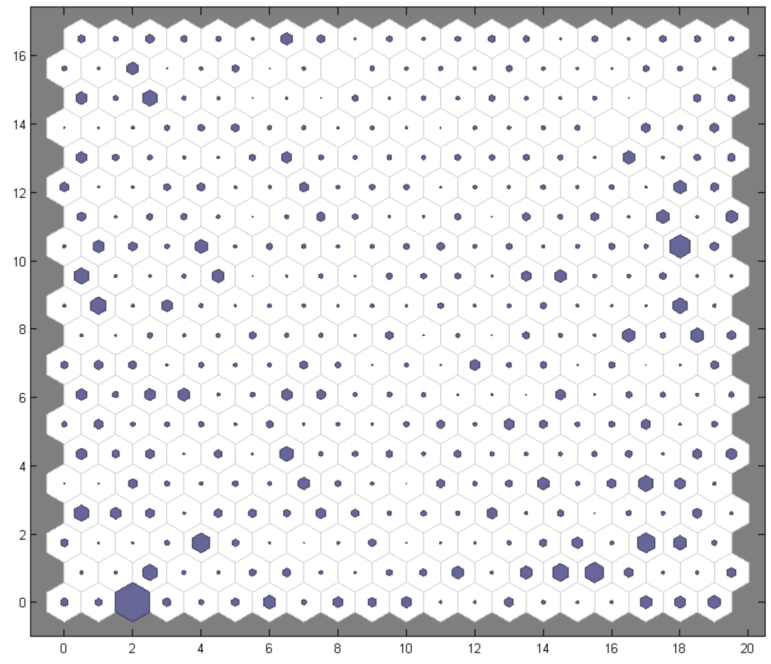

Figure 4.13: Relative distribution of the input feature vectors alongside with the total numbers of vectors mapped to each neuron in the 20 × 20 map.

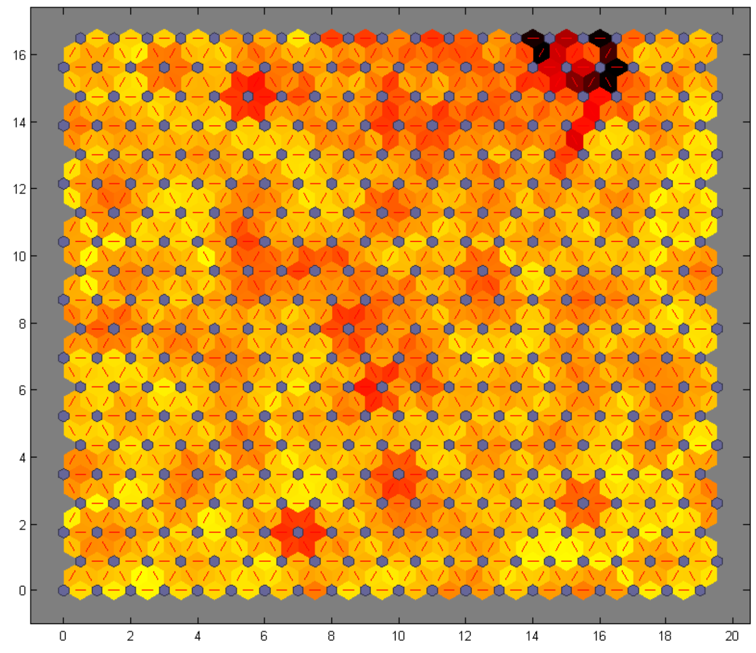

Figure 4.14: The neighbor distances in the 20 × 20 map.

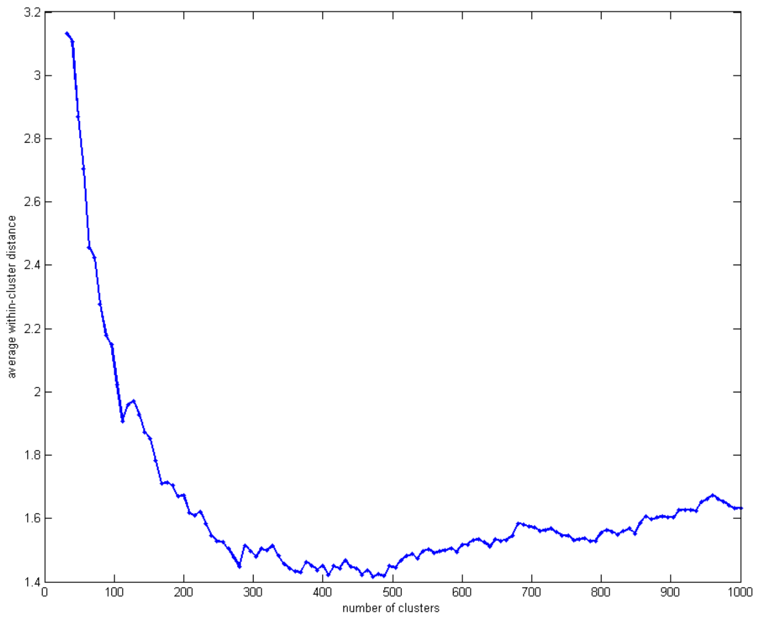

Figure 4.15: A within-cluster distance, measured as the length of the shortest path between neurons, dependent on the number of clusters produced from the hierarchical clustering, mapped to the 20 $\times$ 20 map.

Conclusion

In this thesis, a method for structure discovery in multimedia data was proposed. This method integrates two common unsupervised approaches for data clustering. The input data are clustered by differently sized self-organizing maps and outputs are evaluated by various levels of hierarchical clustering with different number of clusters, mapped to the SOM.

The SOM has been used mainly because of its nature ability to preserve topological relations of the high-dimensional input space in a low dimensional map, so those relations can be easily visualised. The ability to visualise the output turns out to be very useful when it comes to interpreting the data. It consists in four modalities with nearly six thousands attributes, so it is a too much complex to see the relations in its original dimension.

Hierarchical clustering has been used as a second unsupervised clustering tool. Its output has been used for the evaluation of the results of SOM. The main purpose of this evaluation is to find the best number of clusters appropriate for the map.

The results of the four self-organizing maps have been examined and interpreted in the context of the available audio-visual data. These results indicate that SOMs can help to recognize structures in multimedia data. The examples of results reveal that groups of similar data spread over more neurons or that a small group of very specific data can be easily identified. Even though that all experiments showed that the most suitable number of clusters should be equal to the number of neurons in the map, or even greater, it is still a great indicator what number of clusters we should choose when we want to use hierarchical clustering itself or when we need a less number of clusters in the map than the number of neurons.

Even though the system for the structure discovery in multimedia data, originally intended for the Film and TV School of the Academy of Performing Arts in Prague, have not been implemented under the Narra project, because the project has not been filled with data for such a system at the time of writing this thesis, the methods used in this thesis could be later also used in the project, once the data is available.

The methods themselves can also be further developed, for example to provide more automatic evaluation. One possible solution can be map traversing, which has been successfully used in the Music Map[8].

Bibliography

[1] Cisco Visual Networking Index: Forecast and Methodology, 2013 — 2018. 2014. Available from: `http://www.anatel.org.mx/docs/interes/Cisco_VNI_Forecast_and_Methodology.pdf`

[2] YouTube - The 2nd Largest Search Engine. 2015. Available from: `http://www.mushroomnetworks.com/infographics/youtube---the-2nd-largest-search-engine-infographic`

[3] Narra. 2015. Available from: `http://narra.eu/`

[4] Pulc, P. *Research into approaches to the classification of audiovisual recordings of lectures and conferences.* Master's thesis, Czech Technical University in Prague, Faculty of Information Technology, 2014.

[5] Kohonen, T. Self-organized formation of topologically correct feature maps. *Biological cybernetics*, volume 43, no. 1, 1982: pp. 59–69.

[6] Vesanto, J.; Himberg, J.; Alhoniemi, E.; et al. Self-organizing map in Matlab: the SOM Toolbox. In *Proceedings of the Matlab DSP conference*, volume 99, 1999, pp. 16–17.

[7] Neumayer, R.; Dittenbach, M.; Rauber, A. PlaySOM and PocketSOM-Player: Alternative Interfaces to Large Music Collections. In *Proceedings of the 6th International Conference on Music Information Retrieval (ISMIR'05)*, London, UK: Queen Mary, University of London, September 11-15 2005, ISBN 0-9551179-0-9, pp. 618–623.

[8] Hartono, P.; Yoshitake, R. Automatic playlist generation from self-organizing music map. *Journal of Signal Processing*, volume 17, no. 1, 2013: pp. 11–19.

[9] Rauber, A.; Pampalk, E.; Merkl, D. Using Psycho-Acoustic Models and Self-Organizing Maps to Create a Hierarchical Structuring of Music by

Musical Styles. In *Proceedings of the 3rd International Conference on Music Information Retrieval*, Paris, France, October 13-17 2002, pp. 71–80.

[10] Chen, G.; Jaradat, S. A.; Banerjee, N.; et al. Evaluation and comparison of clustering algorithms in analyzing ES cell gene expression data. *Statistica Sinica*, volume 12, no. 1, 2002: pp. 241–262.

[11] Lamirel, J.-C.; Cuxac, P.; Mall, R.; et al. A new efficient and unbiased approach for clustering quality evaluation. In *New Frontiers in Applied Data Mining*, Springer, 2012, pp. 209–220.

[12] Caliński, T.; Harabasz, J. A dendrite method for cluster analysis. *Communications in Statistics-theory and Methods*, volume 3, no. 1, 1974: pp. 1–27.

[13] Rousseeuw, P. J. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, volume 20, 1987: pp. 53–65.

[14] Davies, D. L.; Bouldin, D. W. A cluster separation measure. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, , no. 2, 1979: pp. 224–227.

[15] Bay, H.; Tuytelaars, T.; Van Gool, L. Surf: Speeded up robust features. In *Computer vision–ECCV 2006*, Springer, 2006, pp. 404–417.

[16] Ramos, J. Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning*, 2003.

[17] Deerwester, S. C.; Dumais, S. T.; Landauer, T. K.; et al. Indexing by latent semantic analysis. *JAsIs*, volume 41, no. 6, 1990: pp. 391–407.

[18] Skopal, T.; Moravec, P. Modified LSI model for efficient search by metric access methods. In *Advances in Information Retrieval*, Springer, 2005, pp. 245–259.

Appendix A
Acronyms

API Application Programming Interface

DVI Digital Visual Interface

FAMU Film and TV School of Academy of Performing Arts in Prague

OCR Optical Character Recognition

RGB Red-Green-Blue

SOM Self-Organising Map

SURF Speeded-Up Robust Features

VGA Video Graphics Array

Appendix B

Contents of enclosed CD

```
README.txt ...................... the file with CD contents description
src ..................................the directory of source codes
  thesis .............the directory of LaTeX source codes of the thesis
  scripts ......................the directory of MATLAB scripts
text ...................................... the thesis text directory
  thesis.pdf ......................the thesis text in PDF format
```