J. Yaghob (Ed.): ITAT 2015 pp. 150–158
Charles University in Prague, Prague, 2015

Search for Structure in Audiovisual Recordings of Lectures and Conferences

Michal Kopp[1], Petr Pulc[1], and Martin Holeňa[2]

[1] Faculty of Information Technology, Czech Technical University in Prague
Thákurova 9, 160 00 Prague
koppmic2@fit.cvut.cz, petrpulc@gmail.com
[2] Institute of Computer Science, Czech Academy of Sciences
Pod Vodárenskou věží 2, 182 07 Prague
martin@cs.cas.cz

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 case study, we propose a method for structure discovery in recorded lectures. The method consists in integrating a self-organizing map (SOM) and hierarchical clustering to find a suitable cluster structure of the lectures. The output of every SOM is evaluated by various levels of hierarchical clustering with different number of clusters mapped to the SOM. Within these mapped levels we search for the one with the lowest average within-cluster distance, which we consider the most appropriate number of clusters for the map. In experiments, we applied the proposed approach, with SOMs of four different sizes, to nearly 16 000 slides extracted from the recorded lectures.

1 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. Growing popularity of audiovisual data is the main reason why many new sites containing such data are created. From the consumer's point of view, the main purpose of such a site is to find a video concerning the topic they are interested in.

Usually, any video file that is being uploaded to some video sharing site is annotated with some keywords by the uploader. Beside that, it also contains some metadata such as resolution, frame rate or length of the video. Consequently, any search in the site is restricted to the keywords provided by the uploader and/or some technical data about the video. It does not rely on any relationship between the contents of the videos themselves. This would need to first discover some structure in the data available at the site.

We investigated the use of a self-organizing map (SOM) [1], which is a kind of artificial neural network, for clustering the videos with respect to their content. First, some features are extracted from the videos and then the map is built as a two-dimensional grid of neurons, so it can be easily visualized. According to the training principle of SOMs, the videos which are similar in their features are placed close to each other in the map. Consequently, the closer the videos represented by their respective vectors in the map are, the more similar they should be.

The most simple clustering of the original data would be based on individual neurons, i.e., all feature vectors mapped to the same neuron would form one cluster. However, there is more information involved in the SOM than could be represented by such a simple clustering. To have a clue what SOM-based clustering would be most suitable, we complement SOM with hierarchical clustering.

2 Related Work

Studies for utilizing SOM as a tool for clustering multimedia data were proposed in [3, 4]. In the PocketSOM [3], the SOM is utilized for a creation of the map of songs. The songs are characterized by their acoustic attributes using Rhythm Patterns [5]. 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 [3], the extraction of the acoustic attributes generated more than thousand features. This is quite similar to our approach to the multimedia clustering proposed in this paper. However, we included even more features describing the content of multimedia files because we were able to use more than one kind of features due to the multimodality of our audiovisual data.

There are also some other differences, in particular between the main purpose of the PocketSOM and our clustering method. 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, our structure discovery method is user-independent.

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

Our method has also some similarities with the Music Map [4], which also utilizes SOM for the music playlist generation. Both, the Music Map and our 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, our 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 our method – the dimensionality of the feature vector. In the Music Map, every song is described by nine features. However, in our application we use more than five thousands features. This also means that the training of the map takes much more time.

Studies utilizing SOM as a clustering tool were also proposed in [6, 7]. In [6], 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 our approach 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[7] deals with cluster quality evaluation. Similarly as in [6], 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.

Our approach, on the other hand, relies on combining the information from SOM with information from hierarchical clustering.

3 Case Study Audiovisual Data

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.

4 Proposed Structuring Approach

4.1 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. Because we have three modalities in the data, we treated each one separately, so we can apply appropriate feature extraction tools.

First, we used Optical Character Recognition (OCR) system for the slides. 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, we used the OCR system Tesseract, 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 we consider only lectures in the Czech language, 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 [9]:

$$W_{t,d} = (n_{t,d}/max_t(n_{t,d})) \cdot log_2(n_D/n_{D,t}) \quad (1)$$

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 and we would like to minimize the impact of the curse of dimensionality, the Latent Semantic Analysis [8] has been used. To this end, the k largest eigenvalues of the term-by-document matrix of weights (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 [10] is then computed as:

$$C_k = \Sigma_k V_k^\top \quad (2)$$

Third, the video is used by two different extraction methods – Speeded Up Robust Features (SURF) and color histogram. Both methods work with an image, so just one frame from the center 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. Because the matrix of these 400 descriptor concepts was still very sparse, we 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 color value in each color channel. We use the RGB coding with 1 byte depth. Each color 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.

4.2 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 our 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' centers. The weight w is calculated as:

$$w_{r,s} = \sqrt{\frac{2n_r n_s}{(n_r + n_s)}} \quad (3)$$

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

4.3 SOM Clustering

Self-organizing map (or Self-organizing feature map)[1] 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 our experiments we 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:

$$W_i(t+1) = W_i(t) + \alpha(t)\theta(c,i,t)(X(t) - W_i(t)) \qquad (4)$$

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[2]. 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 our experiments with SOM, we used the MATLAB's Neural network toolbox implementation of SOM, which uses the batch algorithm by default.

4.4 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 [6] and [7], 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 case study, we use a different approach – an integration of SOM and hierarchical clustering. First, we train SOM 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. Algorithm 1 and the following paragraphs describe how the average withing cluster distance of a clustering solution is calculated.

Algorithm 1 An average within-cluster distance

```
map ← trained SOM
S ← clustering solution obtained by cutting cluster hierarchy
for all c ∈ S do
    WCD[c] ← WITHINCLUSTERDISTANCE(map,c)
end for
SolutionAvgWCD ← avg(WCD) //over all clusters c

function WITHINCLUSTERDISTANCE(map,c)
    for each pair p of input vectors u, v ∈ c do
        N_u ← FINDNEURON(u,map)
        N_v ← FINDNEURON(v,map)
        Dist[p] ← dist(N_u, N_v)
    end for
    return avg(Dist) between all pairs
end function

function FINDNEURON(v, map)
    return Neuron N in the map map to which the input
    vector v is mapped.
end function
```

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, we calculate the average distance, 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. We also interpret it as a measure of similarity between the map out-

put and the hierarchical clustering output – the smaller the within-cluster distance is, the more similar the outputs of SOM and hierarchical clustering are. We 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.

5 Experimental Evaluation

5.1 Evaluation Methodology

For our experimental evaluation, we 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, we considered only the squared maps with sizes of 8×8, 12×12, 16×16 and 20×20 neurons. 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 we need outputs from many levels of the hierarchical clustering, 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. We 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. We also introduced different steps for increasing the number of clusters, depending on the map size.

In Table 1, the empirically chosen values for of our 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.

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 1: Settings used in the performed experiments

5.2 Evaluation Results

In this section we are 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 our 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 4.4.

We started the evaluation with the smallest, 8×8 map. A relative distribution of the input feature vectors onto the map is shown in Figure 1.

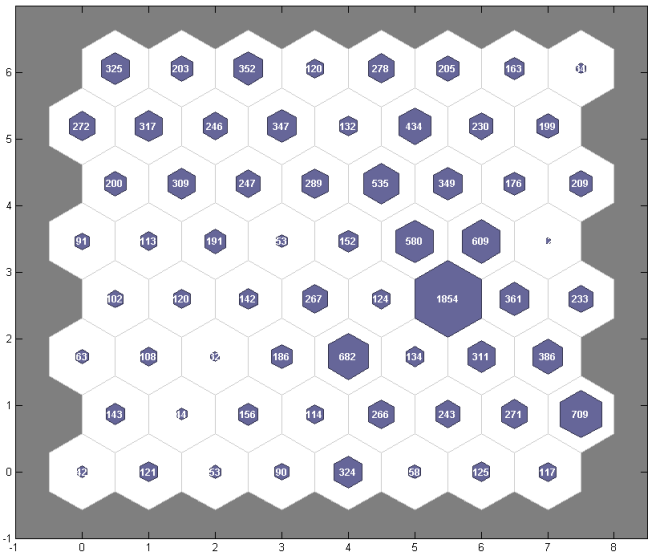

Figure 1: Relative distribution of the input feature vectors alongside with the total numbers of vectors mapped to each neuron in the 8×8 map.

From the distribution of feature vectors shown in Figure 1 we can see that from the total number of 15 953 feature vectors, there are 1854 feature vectors mapped to a single neuron. 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.

The neuron with the most mapped feature vectors has them mapped more than twice as many as any other neuron in this map. When we look at the input data, we find that many different slides are mapped to this neuron.

Another reason for the large number of feature vectors mapped to this neuron is that these feature vectors provide just one modality of the available input data. Namely, 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. So, even though these slides look different, the OCR produces little to no text, which can result in great similarity between them, depending on other modalities.

These feature vectors also have in common that there is little of recognized speech, thus they are very similar in the speech-to-text modality. The only really 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 typical examples of a lecture – a slide with some text, video from lecture hall and an 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. However, the input data from the sequences, which are not present in this neuron, are in the most cases mapped to its direct neighbours. This behaviour supports the idea of large clusters spread over more neighbouring neurons.

In Figure 2, which shows the distances between neighbors, we can see that the distances are the shortest between the neurons with the highest number of vectors. 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 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.

Figure 3 shows that in the map of size 8×8 neurons,

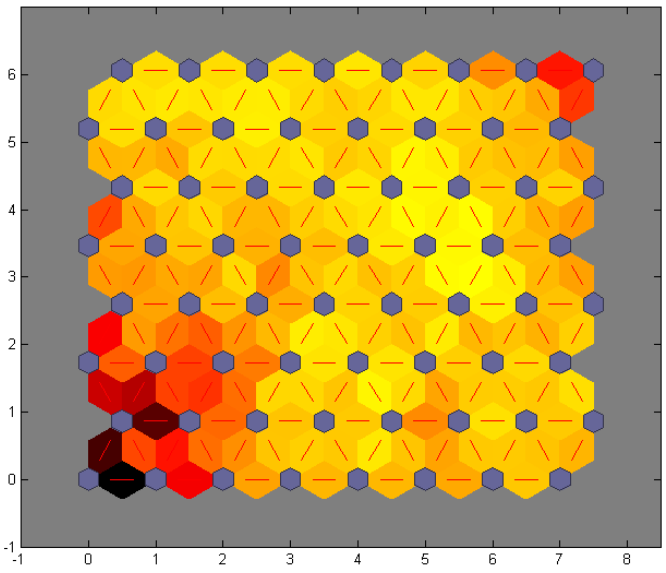

Figure 2: The neighbor distances in the 8×8 map.

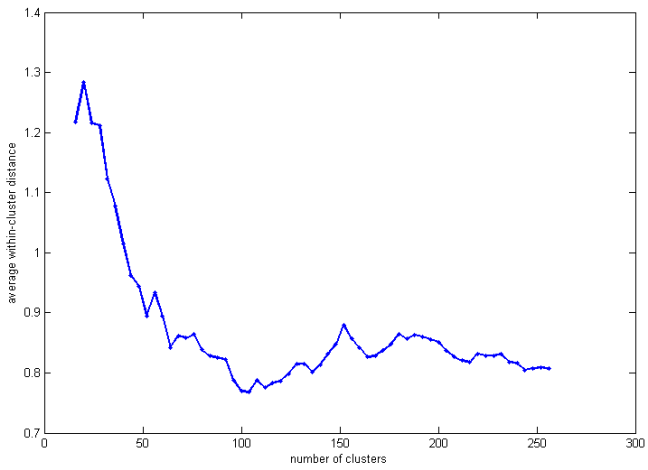

Figure 3: A within-cluster distance, measured as the length of the shortest path between neurons, dependent on the number of clusters produced from the hierarchical clustering, mapped to the 8×8 map.

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.

Let us now pass to the 12×12 and 16×16 maps. In Figures 4 and 7 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.

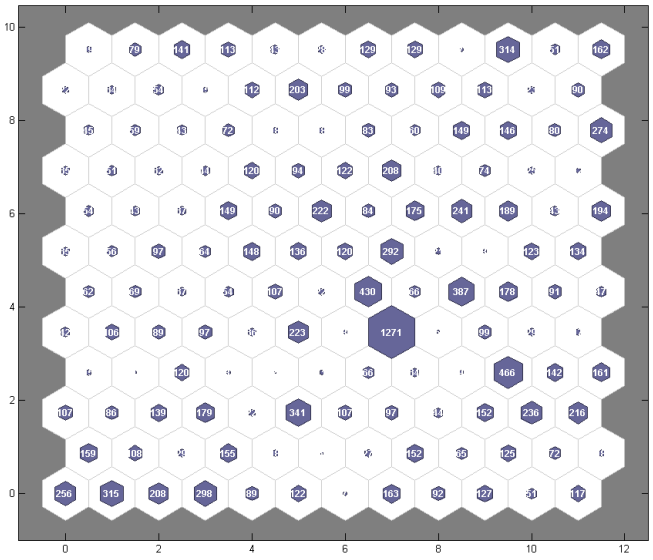

Figure 4: Relative distribution of the input feature vectors alongside with the total numbers of vectors mapped to each neuron in the 12×12 map.

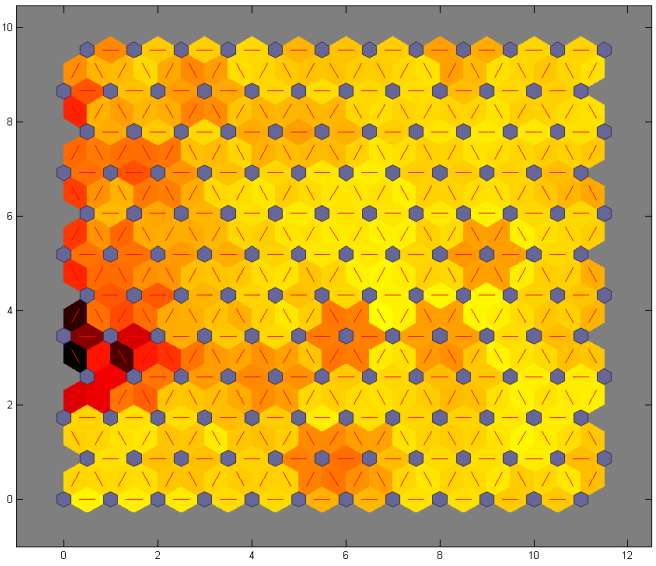

Figure 5: The neighbor distances in the 12×12 map.

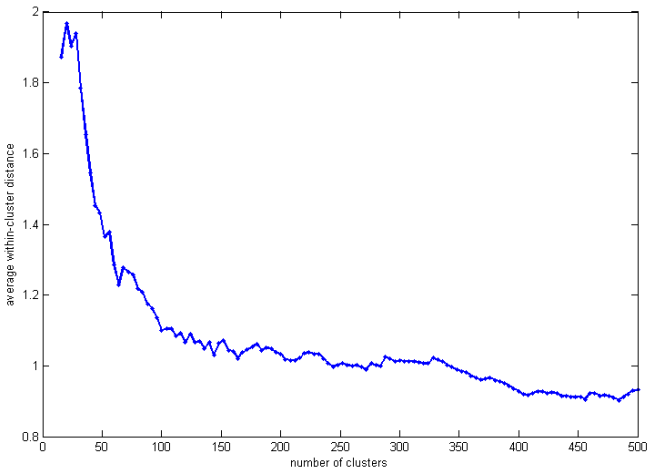

Figure 6: A within-cluster distance, measured as the length of the shortest path between neurons, dependent on the number of clusters produced from the hierarchical clustering, mapped to the 12×12 map.

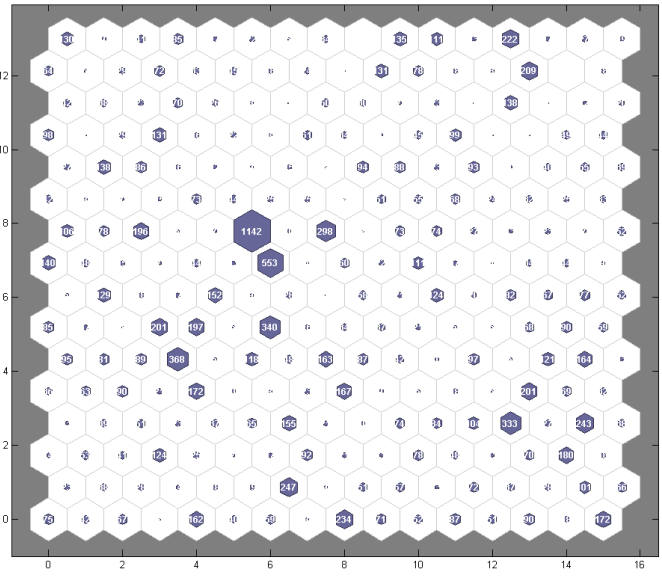

Figure 7: Relative distribution of the input feature vectors alongside with the total numbers of vectors mapped to each neuron in the 16×16 map.

In Figures 5 and 8, we can see that there are some neurons, mostly on the left side of the map 5 and in the top-center, lower-left and top-right corners in 8, that have a large distance to their respective neighbors. The situation is quite similar to that of the smaller map in Figure 2. The 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.

In Figures 6 and 9, which show the within-cluster distance dependence on the number of clusters formed by cutting the cluster hierarchy, we can not 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

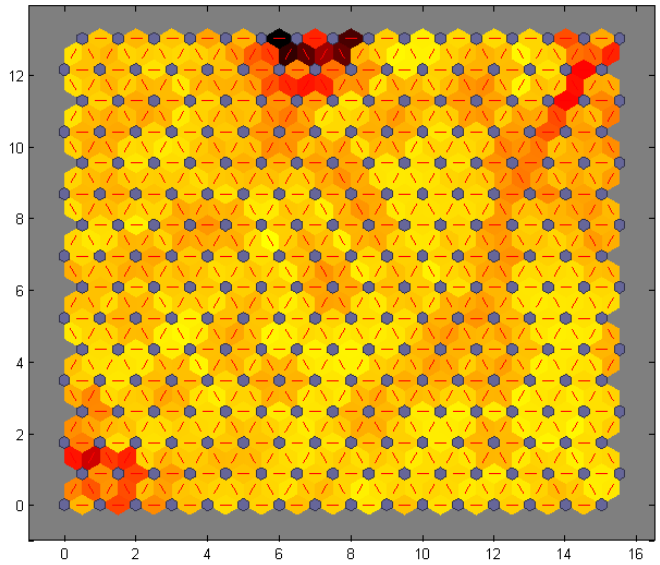

Figure 8: The neighbor distances in the 16×16 map.

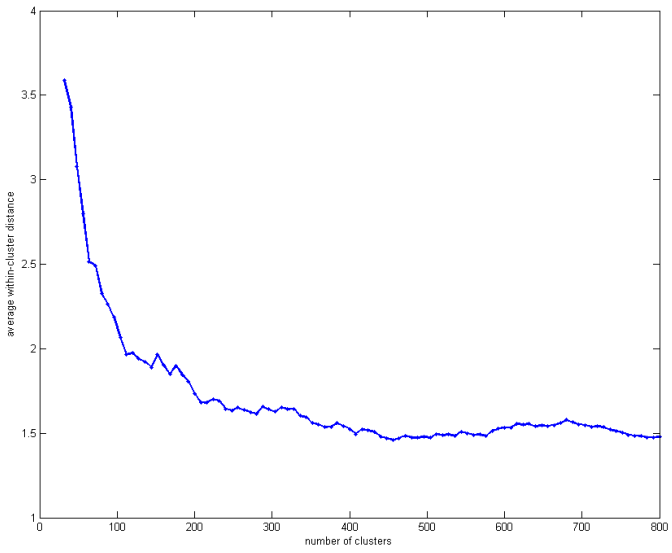

Figure 9: A within-cluster distance, measured as the length of the shortest path between neurons, dependent on the number of clusters produced from the hierarchical clustering, mapped to the 16×16 map.

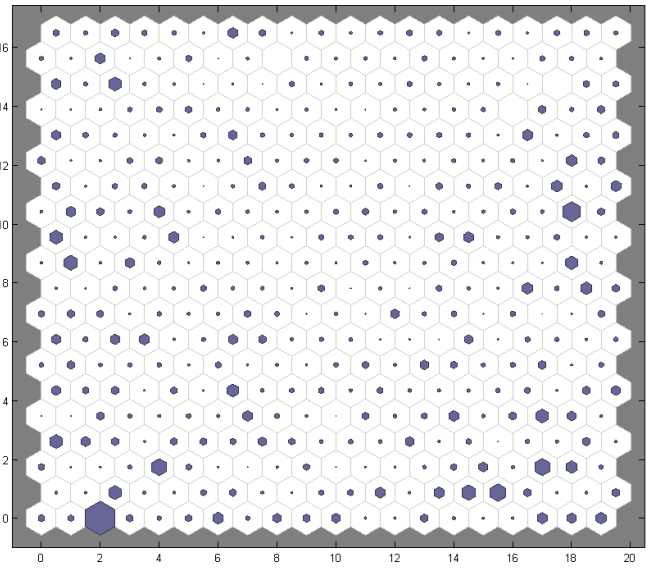

Figure 10: Relative distribution of the input feature vectors alongside with the total numbers of vectors mapped to each neuron in the 20×20 map.

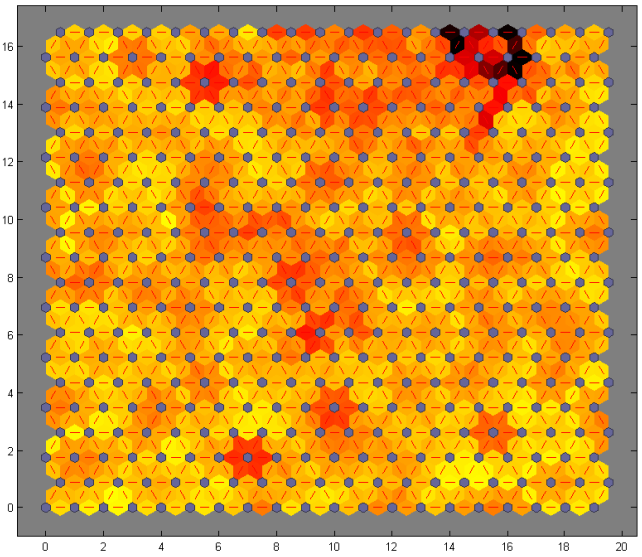

Figure 11: The neighbor distances in the 20×20 map.

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.

Finally, let us pass to the largest, 20×20 map. Figure 10 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 11, 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 1, 4 and 7, 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 12, 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

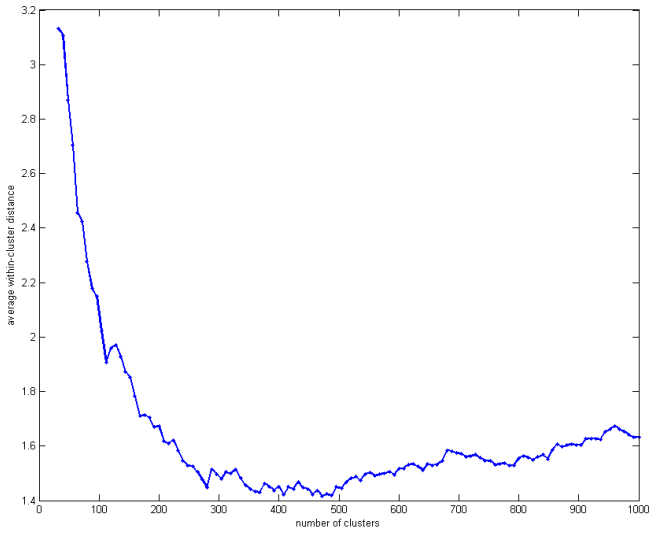

Figure 12: A within-cluster distance, measured as the length of the shortest path between neurons, dependent on the number of clusters produced from the hierarchical clustering, mapped to the 20×20 map.

the map are very similar in that interval, because there is exactly 400 neurons in this map.

6 Conclusion

The objective of this paper was to provide a proof of concept for the possibility to improve structure discovery in complex multimedia data through the integration of two unsupervised approaches – hierarchical clustering and self-organizing maps. For that proof of concept, we have used a case study of more then 100 hours of audiovisual recordings of lectures from several years of a Czech science festival called Week of Science and Technology. The approach we propose relies on the one hand on the flexibility and universality of hierarchical clustering, on the other hand on the suitability of SOM for multimedia data, convincingly documented by Pitoyo Hartono. Our approach has been much inspired by his paper [4]. However, the need to integrate SOM with hierarchical clustering is a consequence of the fact that our data are much more complex, both from the point of view of involved modalities, and from the point of view of the number of attributes.

We have tested the proposed approach with 4 differently sized self-organizing maps. Although space limitations allowed us to present only two examples of the interpretaion of the obtained results in the context of the original audio-visual data, even these examples indicate that SOMs indeed help to recognize structure in that data. In the first provided example, the SOM organized the most common data into a large cluster spread over few neighbouring neurons and in the second example, the SOM discovered a specific group among the nearly 16000 considered segments of video-recordings and their associated slides, which substantially differs from the others.

In the future, we want to further elaborate our approach, taking into account the results obtained in the case study presented here. We also intend to apply it to further complex multimedia data sets, in particular to data nowadays being collected within the project Open Narra at the Film and TV School of the Academy of Performing Arts in Prague.

Acknowledgement

The research reported in this paper has been supported by the Czech Science Foundation (GAČR) grant 13-17187S. Access to computing and storage facilities owned by parties and projects contributing to the National Grid Infrastructure MetaCentrum, provided under the programme "Projects of Large Infrastructure for Research, Development, and Innovations" (LM2010005), is greatly appreciated.

References

[1] Kohonen, T.: Self-organized formation of topologically correct feature maps. Biological Cybernetics **43** (1982), 59–69

[2] Kohonen, T.: Self-organizing maps. Springer Series in Information Sciences, 1995, 2nd ed. 1997

[3] 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, 618–623, 2005

[4] Hartono, P., Yoshitake, R.: Automatic playlist generation from self-organizing music map. Journal of Signal Processing **17 (1)** (2013), 11–19

[5] 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 Symposium on Music Information Retrieval, 2002, 71–80,

[6] Chen, G., Jaradat, S. A., Banerjee, N., et al.: Evaluation and comparison of clustering algorithms in analyzing ES cell gene expression data. Statistica Sinica **12.1** (2002), 241–262

[7] Lamirel, J. C., Cuxac, P., Mall, R., et al.: A new efficient and unbiased approach for clustering quality evaluation.

[8] Deerwester, S. C., Dumais, S. T., Landauer, T. K., et al.: Indexing by latent semantic analysis. JASIS **41 (6)** (1990), 391–407

[9] Ramos, J.: Using tf-idf to determine word relevance in document queries. In: Proceedings of the First Instructional Conference on Machine Learning, 2003

[10] Skopal, T., Moravec, P.: Modified LSI model for efficient search by metric access methods. In: Advances in Information Retrieval, 2005, 245–259