UNIVERSITÄT ZU LÜBECK
INSTITUTE OF MEDICAL INFORMATICS

Seminar

Medical Image Computing and e-Health

WS 2020/2021

Automatic Detection and Segmentation of Brain Tumor Using Random Forest Approach

Leonard Brenk

Matr.-Nr.: 697947, Computer Science

Supervisor

Marja Fleitmann

LÃ¼beck, November 26, 2020

Contents

1 Motivation

The detection and treatment of tumors is one of today's greatest challenges for mankind. Finding mutating cells and preventing them from spreading is an extraordinarily difficult task. Usually tumors occur nested in non-affected tissues which makes their discovery and treatment heavily challenging. The current state of technology operating in that field has complications filtering the negative cells out and segmenting the pure tumor completely. Due to the variety of anatomical structures and the way they can differ from person to person, even for a trained professional it takes a lot of time to detect and fully segment a tumor in MRI scans - even more if three-dimensional scans are used. Therefore not only are the needed financial and personal requirements greatly inefficient but also the outcome can be flawed and incomplete based on the know-how of the experts. Machine Learning ensembles provide a promising approach to face that issue on a global, revolutionary scale.

Seeing that finding the tumor in an early stage has the highest impact on and can facilitate and enhance the treatment the detection is the key to success. The paper written by TODO AUTHOR in 2016 captures an experiment regarding this topic. It carries out a Random Forest based machine learning technique using multispectral volumetric MRI volumes. Training an algorithm to reliably find and segment tumor cells on MRI scans can due to its practically limitless resources in time and memory possibly perform on a higher level than multiple experts. Therefore the data undergoes several steps of pre-processing in order to be suitable for the application in Binary Decision Trees. Furthermore, after the employment of Random Forests the data is post-processed and then analyzed. In order to illustrate the accuracy of a decision the authors introduced a Dice Score, thereby Binary Decision Trees can be compared and graded. The machine learning method used in this paper belongs to the supervised learning techniques which means that the outcome of data the machine is trained with is already known and can be used to adjust and optimize the inner parameters.

The paper presents initial outcomes and recommendations regarding a complex brain tumor detection and segmentation system and its future implementation in a clinical context.

2 Introduction BDT & RDF

In this section the essential basics regarding Binary Decision Trees and Random Forest and their implementation are being explained an discussed.

2.1 Binary Decision Trees(BDT)

2.1.1 General

A Binary Decision Tree (BDT) is trained and employed in order to make a decision based on a data vector. It consists of multiple levels of two-way decisions until it reaches a leaf node at the end classifying the vector into a certain class. A BDT can be used to either deploy data into classes or predict values in the future using regression. However, this paper will concentrate on the ability to assign a label to a vector.

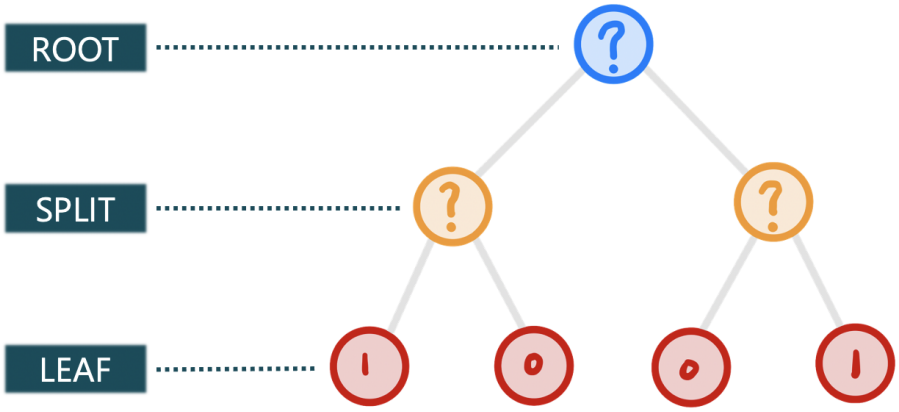


Fig. 1: The Structure of a BDT. Usually there are multiple levels of split nodes.

The root node is the starting point of a BDT. In order for a BDT to reach a decision it goes through several inner decision nodes which divide the data into two subgroups and forwards them to the next node. At each split node the data is divided again as long as the decisions within the data subset are distinguishable. If the decisions are unilateral, a leaf node is created. Such leaf nodes are then attributed to a class, also called a label. Thereby an inserted data vector can be classified into classes.

2.1.2 Building a BDT

A BDT can, amongst other options, be build from a table which contains multiple attributes and a column for the individual decision of that data row. When building a BDT a cycle is implemented recursively. The data shall be divided into two subgroups at a time. Then each subgroup is again divided into two subgroups. In the end there are as many subgroups as it takes to classify every single data vector correctly. The cycle starts with a given data set. For the first split an attribute needs to be picked which the data shall be split with. Having decided upon that attribute, a threshold is set to decided

wether a vector is forwarded to the left or right child node - depending on whether its value is higher or smaller than the threshold. Using the threshold the complete data set can then be divided into two subgroups. Now each subgroup is considered individually and the cycle begins again. The process ends when the decisions of a subgroup are all identical. Then the leaf node is created labeled with that decision.

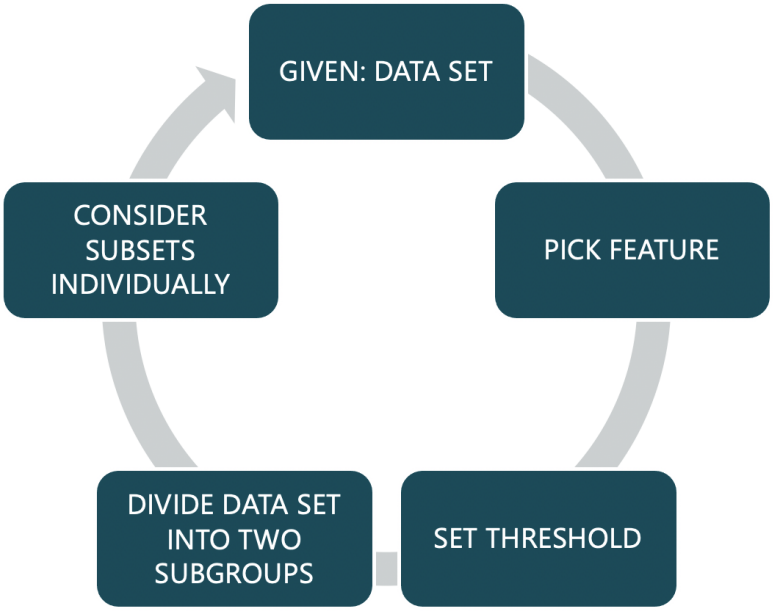


Fig. 2: The process of buidling a BDT.

The decision for a specific feature while building a BDT can influence the outcome greatly since the generated subgroups will be completely different. That is why there is need for a way to assess the most efficient way of splitting the data. For that the information entropy and information gain are used. The entropy describes the level of information of a variable (2.1.2). The information gain compares two entropies and calculates the difference(3). The idea is to test every attribute as a possible option for a split node and calculate which attribute would yield the highest gain in information after the split.

$$H(X) = -\sum_{i=1}^{n} P(x_i) \log_2 P(x_i)$$

Fig. 3: Entropy

$$\text{Information Gain } = \text{Entropy(Parent)} - \text{Average Entropy(Children)}$$

Fig. 4: Information Gain

2.1.3 Example BDT

Given a table of sample data:

Temperature	Rain	Windy	Humidity	Go outside ?
High	Yes	True	High	No
Low	No	False	Low	Yes
Low	No	True	Low	No
High	No	False	High	Yes
High	Yes	False	Low	No
Low	Yes	True	Low	No
High	No	False	Low	Yes
Low	Yes	True	High	No
Low	No	True	High	Yes
Low	No	False	Low	Yes
High	Yes	True	High	No
High	Yes	False	Low	No
Low	No	True	High	Yes
High	Yes	False	High	No

In order to decide which attribute to pick for the root node, the information gain must be calculated for all of them (Temperature, Rain, Windy and Humidity). For example Splitting with Temperature would look like this:

First we need to calculate the entropy of the complete table.

$$H(6,8) = -\left[\frac{6}{14}\log_2\left(\frac{6}{14}\right) + \frac{8}{14}\log_2\left(\frac{8}{14}\right)\right] = 0,9852$$

If we were to split at Temperature the table would be divided into two groups, one with high temperature and one with low temperature. Now the entropy for both of these subgroups needs to be calculated individually. The information gain for Temperature is the difference resulting from the complete entropy minus the average entropy of the new generated subgroups. In this case the information gain is $0,01615$.

Split at Temperature:

$$\text{Temperature} = \text{high: } H(2,5) = -\left[\frac{2}{7}\log_2\left(\frac{2}{7}\right) + \frac{5}{7}\log_2\left(\frac{5}{7}\right)\right] = 0,8631$$

$$\text{Temperature} = \text{low: } H(4,3) = -\left[\frac{4}{7}\log_2\left(\frac{4}{7}\right) + \frac{3}{7}\log_2\left(\frac{3}{7}\right)\right] = 0,9852$$

$$\text{Gain: } H(6,8) - H\left[\frac{7}{14}H(2,5) + \frac{7}{14}H(4,3)\right] = \boxed{0,01615}$$

Since the information gain is the value that needs to be maximized, the information gain is calculated for the remaining attributes as well. For the root node the attribute is selected that yields the highest information gain. In this case: Rain.

Split at Temperature:

$$\text{Temperature = high: } H(2,5) = -\left[\frac{2}{7}\log_2\left(\frac{2}{7}\right) + \frac{5}{7}\log_2\left(\frac{5}{7}\right)\right] = 0,8631$$

$$\text{Temperature = low: } H(4,3) = -\left[\frac{4}{7}\log_2\left(\frac{4}{7}\right) + \frac{3}{7}\log_2\left(\frac{3}{7}\right)\right] = 0,9852$$

$$\text{Gain: } H(6,8) - H\left[\frac{7}{14}H(2,5) + \frac{7}{14}H(4,3)\right] = 0,01615$$

Split at Windy:

$$\text{Windy = True: } H(2,5) = -\left[\frac{2}{7}\log_2\left(\frac{2}{7}\right) + \frac{5}{7}\log_2\left(\frac{5}{7}\right)\right] = 0,8631$$

$$\text{Windy = False: } H(4,3) = -\left[\frac{4}{7}\log_2\left(\frac{4}{7}\right) + \frac{3}{7}\log_2\left(\frac{3}{7}\right)\right] = 1,3781$$

$$\text{Gain: } H(6,8) - H\left[\frac{7}{14}H(2,5) + \frac{7}{14}H(4,3)\right] = 0,1354$$

Split at Rain:

$$\text{Rain = yes: } H(0,7) = -\left[\frac{0}{7}\log_2\left(\frac{0}{7}\right) + \frac{7}{7}\log_2\left(\frac{7}{7}\right)\right] = 0$$

$$\text{Rain = no: } H(6,1) = -\left[\frac{6}{7}\log_2\left(\frac{6}{7}\right) + \frac{1}{7}\log_2\left(\frac{1}{7}\right)\right] = 0,5916$$

$$\text{Gain: } H(6,8) - H\left[\frac{7}{14}H(0,7) + \frac{7}{14}H(5,2)\right] = 0,6894$$

Split at Humidity:

$$\text{Humidity = High: } H(3,4) = -\left[\frac{3}{7}\log_2\left(\frac{3}{7}\right) + \frac{4}{7}\log_2\left(\frac{4}{7}\right)\right] = 0,9852$$

$$\text{Humidity = Low: } H(3,4) = -\left[\frac{3}{7}\log_2\left(\frac{3}{7}\right) + \frac{4}{7}\log_2\left(\frac{4}{7}\right)\right] = 0,9852$$

$$\text{Gain: } H(6,8) - H\left[\frac{7}{14}H(3,4) + \frac{7}{14}H(3,4)\right] = 0$$

Having split the data with rain, two subgroups are created 2.1.3. Selecting the attribute for the next split node for a subgroup proceeds exactly like with the root node, only with a smaller set of data. The final tree will then look like this:

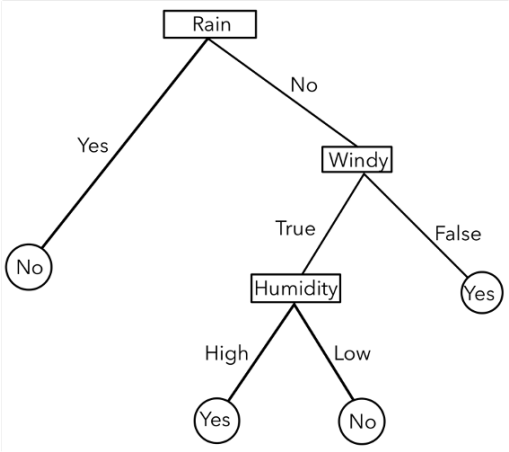


Fig. 5: The final BDT.

Temperature	Rain	Windy	Humidity	**Go outside ?**
High	Yes	True	High	**No**
High	Yes	False	Low	**No**
Low	Yes	True	Low	**No**
Low	Yes	True	High	**No**
High	Yes	True	High	**No**
High	Yes	False	Low	**No**
High	Yes	False	High	**No**

Temperature	Rain	Windy	Humidity	**Go outside ?**
Low	No	False	Low	**Yes**
Low	No	True	Low	**No**
High	No	False	High	**Yes**
High	No	False	Low	**Yes**
Low	No	True	High	**Yes**
Low	No	False	Low	**Yes**
Low	No	True	High	**Yes**

Fig. 6: The two generated subgroups when splitting with Rain at the root node.

2.1.4 Testing a BDT

After building a BDT it has to be ensured that it is working correctly and classifies properly. Therefore not all data is used for building the tree but a part is saved for testing purposes. Now that the BDT is build we can pick samples of the test data and insert it into the BDT. As this is a supervised learning ensemble the result is already known and can thereby be compared with the yielded outcome of the tree. If it is correct, the tree is working fine. The accuracy characterizing Dice Score is based on testing cases and describes their outcome in a numerical, understandable way.

2.1.5 Pro and Cons of BDT

A BDT can be used to classify and label data vectors which can be helpful for a lot of problems. As is it not limited to categorical values for classification but can also work with numerical values, a BDT is also able to predict future values through regression curves. Furthermore the technique and structure of a BDT and its decision finding process is intuitive and visualizable. Additionally the data used only requires little data pre-processing TODO WHY.

The problem with BDTs is that small changes in data can heavily affect the outcome of the decision. If the tree internalizes every detail of the training data it can overfit. That means that the BDT doesn't reflect the general input data but has adapted to the training data. For instance: If an algorithm was given scans of tumors that are compared to all other existing tumor scans rather dark than the BDT will set this as the general intensity of a tumor scan. Thereby brighter parts of tumor scans that the BDT has not seen yet will not be classified as tumor. In order to face that issue Random Forests have been introduced.

2.2 Random Forest(RDF)

Random Forest (RDF) overcome the obstacle of overfitting through the use of multiple BDTs. A tested vector is inserted into multiple trees which all yield a decision for that vector. One way to determine the overall final decision is to decide by a majority vote. On individual tree might be overfit at some points in the forest however this does not influence the final decision too much.

2.2.1 Bagging

While building a random forest one must build several trees. When for each tree only a subset of the training data is being used, it is called Bagging. Thereby randomness is introduced in the procedure and improves the reliability of the decision.

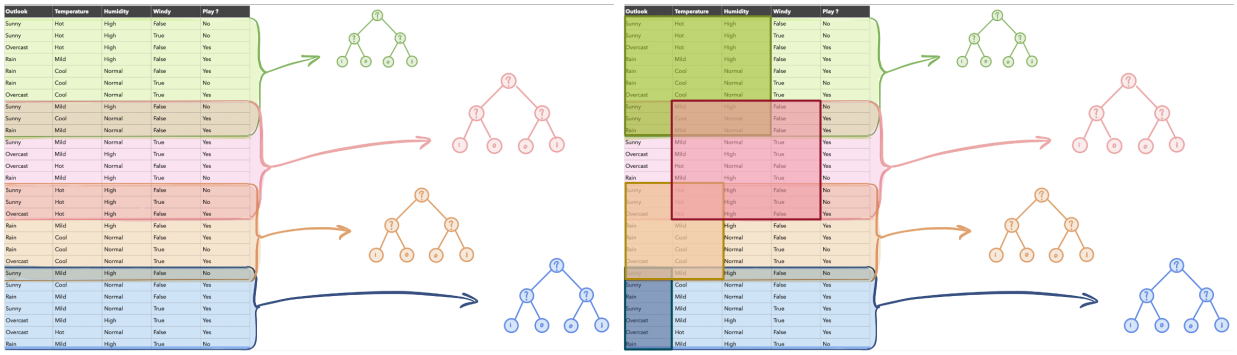


Fig. 7: Bagging(left): Each tree is trained by individual data set, Baggin +XY (right) Each tree is trained by an individual set of data and features

3 Application

3.1 Goal of the Paper

The paper aims to develop a reliable procedure based on machine learning ensembles to detect tumors on MRI scans in an early stage and segment them. For the execution of the experiment 12 records from the BRATS Data Set were used. Each record belongs to one patient and contains four different MRI scans (T1, T2, T1C, FLAIR) which display the same brain in different ways. Each record consists of approximately 1.5 Mio feature vectors. For every volume there is a truth image containing experts annotations for either an active tumor or edema cells. The algorithm is supposed to seperated between tumor cells, edema cells and negative cells.

3.2 Data & Pre-Processing

In order to address a three-dimensional pixel, also called voxel, a feature vector is generated. It looks like this:

$$T[x, y, z] = T1[x, y, z], T2[x, y, z], T1C[x, y, z], FLAIR[x, y, z]$$

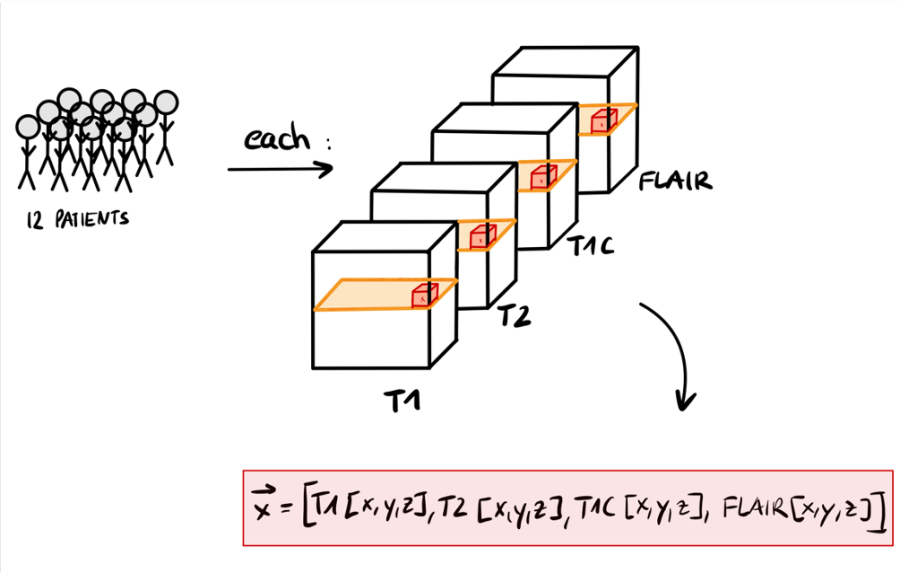


Fig. 8: The Feature vector is generated to address a specific cell in all four cells

Before the data is used in training and testing BDTs and RDFs it needs to be pre-processed. The first step is the histogram normalization. A histogram displays the amount of intensity values of a scan. In this case the middle 50% of the data shall be located between the intensity values 600 and 800. Also the minimum value of 200 and the maximum value of 1200 are set as boundaries. The second step involves the location of a considered voxel. Since the feature vector does not include any information regarding the voxel's location two more features have to be computed for each scan. The new feature vector will thereby consist of 12 elements. For the computation of the first added feature all direct neighbor voxels in a $3 \times 3 \times 3$-sized Cube are considered. The average intensity value of them is the first new feature value of the vector. The second feature is computed based on the average intensity value of the eight closest voxels of that slice TODO WHAT SLICE and two closest ones from the neighboring slices. That is what the new feature vector now looks like:

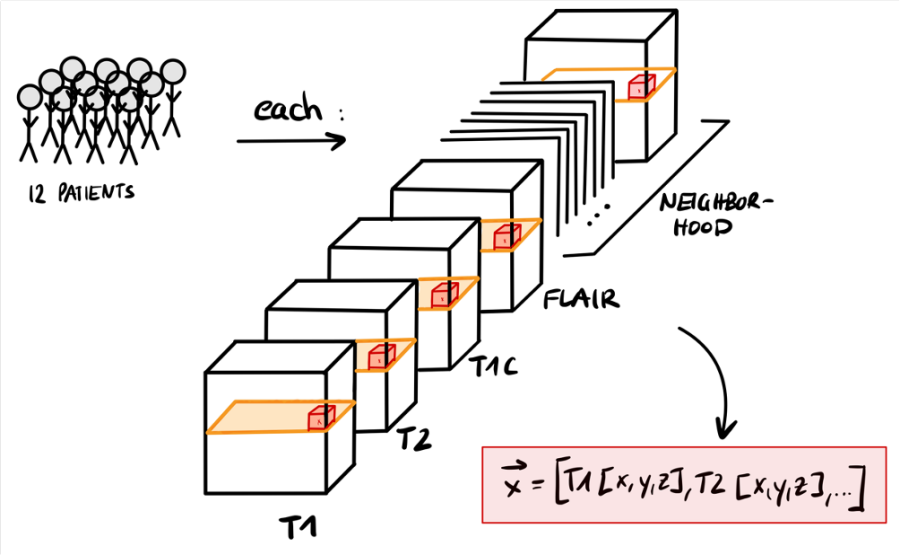


Fig. 9: The updated Feature vector contains twelve elements

The last step of pre-processing excludes the missing voxels from the average calculations for the neighborhood. Null values would distort the result and must thereby be ignored for the average intensity value.

3.3 Training and Testing

The data is now ready to be used for training BDTs. As explained in section 2.1 the tree is built, while the attributes like Rain or Temperature are now T1, T2, etc. As 1.5 Mio vectors are too many to use on training the tree samples have to be selected. When doing so and selecting e.g. 300 samples, then it means that 300 feature vectors representing tumor cells, 300 feature vectors representing edema and 300 feature vectors representing negative cells are used.

3.4 Post-Processing

Even though the BDT and RDF classifies most of the voxels reliably there are usually some negative cells which are labeled as tumor or edema due to the great variety of normal tissues they can belong to. In order to correct falsely labeled positives for each tumor and edema voxel a 250-voxel neighborhood is defined. The average of these are then compared with a pre-defined threshold which value is debatable. If the intensity of the voxel in question is less than the average intensities, it is relabeled into a negative voxel, if its higher it remains the same.

4 Results

5 Conclusion

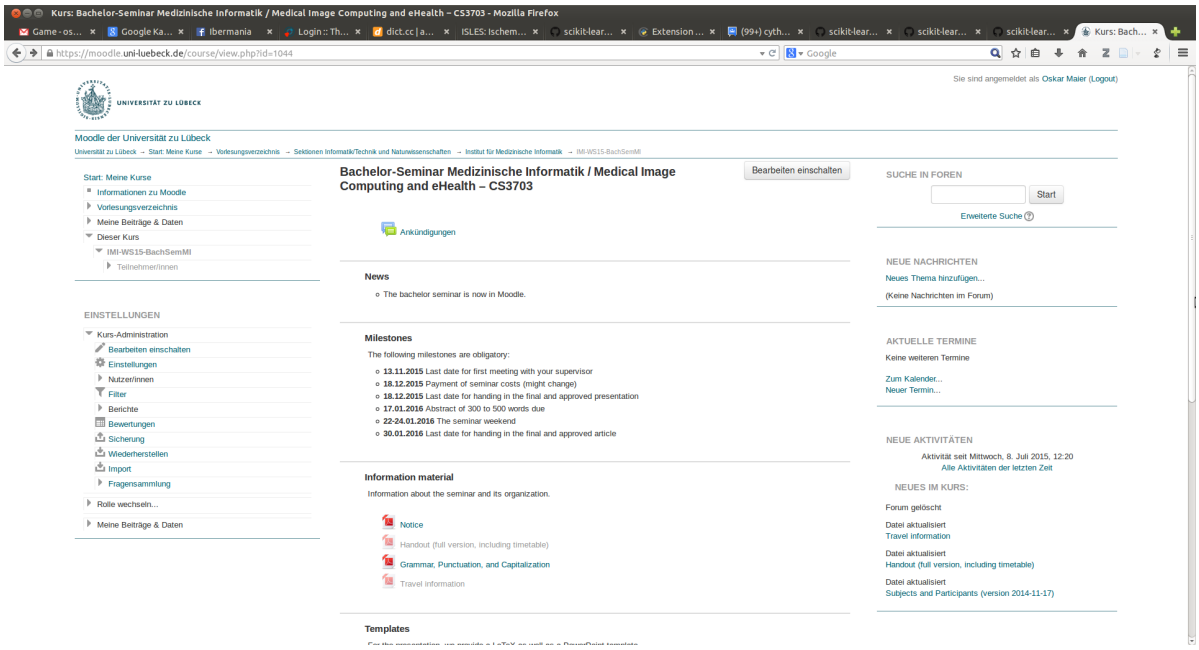
6 Sources & Appendix

Table 1: Topics of the presentations **two years ago**.

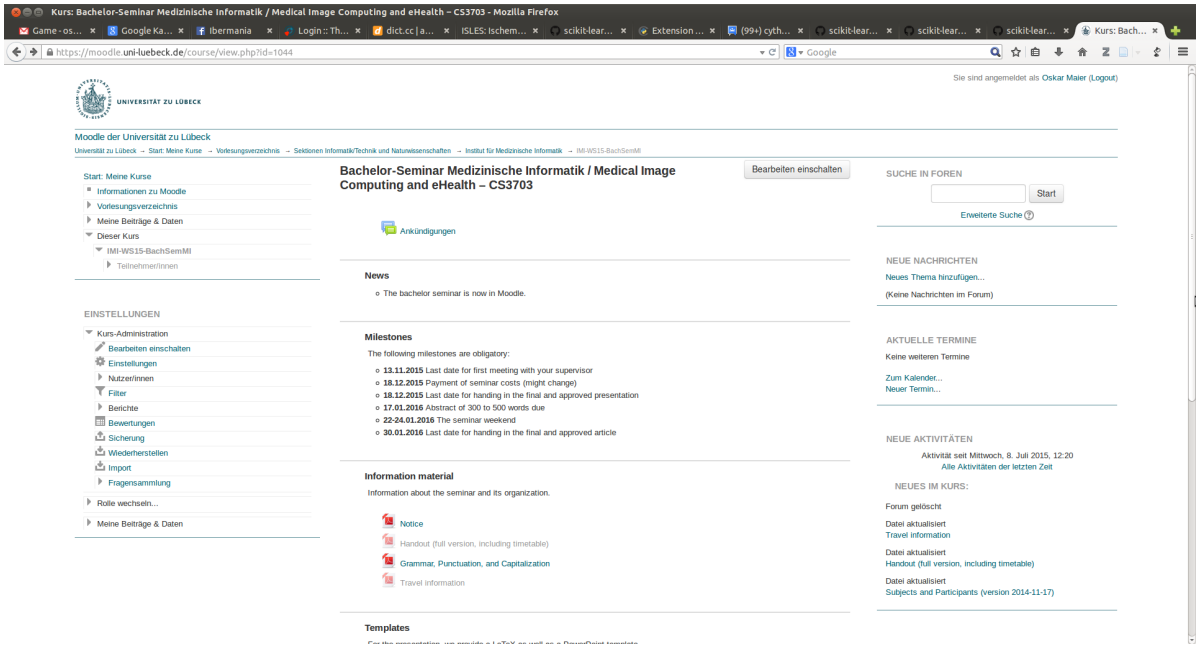
Speaker	**Topic [Literature]**	**Supervisor**
J. Niemeijer	Hough transforms	M. Wilms
D. Labitzke	Optimal Surface Segmentation in Volumetric Images – A Graph-Theoretic Approach (cf. [Li et al. 2006])	M. Wilms
A. Bostelmann	Graph Cuts for image segmentation	O. Maier
D. Conrad	Texture descriptors and their application to medical images	O. Maier
E. Franke	Image Segmentation Using Deformable Models: Parametric Deformable Models	J. Krï¿½ger
N. Broecker	Image Segmentation Using Deformable Models:Geometric Deformable Models	J. Krï¿½ger
L. Pankert	Visualization in Medicine: Volume Rendering with raycasting	J. Ehrhardt
T. Langer	Visualization in Medicine: Surface Rendering using the Marching Cubes Algorithm	J. Ehrhardt
M. Caspe	Volumetric Ultrasound Stitching	D. Fortmeier
H. Tï¿½nnies	Surface-based Palpation Haptics	D. Fortmeier
P. Kling	A Content Model for the ICD-11 Revision	J. Ingenerf
S. Heusel	MeSHy: Mining unanticipated PubMed information using frequencies of occurrences and concurrences of MeSH terms	J. Ingenerf
K. Soika	What is bioinformatics? An introduction and overview	B. Andersen
M. Licht	How (not) to protect genomic data privacy in a distributed network: using trail re-identification to evaluate and design anonymity protection systems	J. Ingenerf
J. Fleckner	Adverse events in medicine: Easy to count, complicated to understand, and complex to prevent	A.-K. Kock
A. Wiegmann	An automated technique for identifying associations between medications, laboratory results and problems	A.-K. Kock
J.-H. Mathes	Organization of Heterogeneous Scientific Data Using the EAV/CR Representation	B. Andersen
F. Simon	Structured Reporting: Patient Care Enhancement or Productivity Nightmare?	A.-K. Kock

References

Li, K., Wu, X., Chen, D.Z., Sonka, M., 2006. Optimal surface segmentation in volumetric images-a graph-theoretic approach. IEEE Transactions on Pattern Analysis and Machine Intelligence 28, 119–134.



(a)



(b)

Fig. 10: Two times ((a) and (b)) the Moodle page for this seminar.