WITH SEMANTICS AND HIDDEN MARKOV MODELS TO AN ADAPTIVE LOG FILE PARSER

Nadine Kuhnert and Andreas Maier

Pattern Recognition, Friedrich-Alexander University, Erlangen-Nuremberg, Germany

ABSTRACT

We aim to model an adaptive log file parser. As the content of log files often evolves over time, we established a dynamic statistical model which learns and adapts processing and parsing rules. First, we limit the amount of unstructured text by clustering based on semantics of log file lines. Next, we only take the most relevant cluster into account and focus only on those frequent patterns which lead to the desired output table similar to Vaarandi [10]. Furthermore, we transform the found frequent patterns and the output stating the parsed table into a Hidden Markov Model (HMM). We use this HMM as a specific, however, flexible representation of a pattern for log file parsing to maintain high quality output. After training our model on one system type and applying it to a different system with slightly different log file patterns, we achieve an accuracy over 99.99%.

KEYWORDS

Hidden Markov Models, Parameter Extraction, Parsing, Text Mining, Information Retrieval

1. INTRODUCTION

Today, almost any computer system documents its performed actions, events, warnings, and errors in the form of log files. A lot of information is generated during operation and written mostly into text, xml or xes files. For some systems several hundreds of lines per second are generated per second which add up to a huge amount of data ready to be interpreted. By parsing the log files, valuable information is extracted which can then be further processed into knowledge. Furthermore, systems are built to receive software updates. With those, also log file contents and their patterns might change. For example, some Key Performance Indicators (KPIs) could have been introduced with a certain software version and not logged by older versions, yet. Kuhnert et al. [1] covered this issue of the information of body region only logged by Magnetic Resonance Imaging systems having the latest software version. They trained clustering methods in order to learn the examined body region from the scan parameters. Thus, they applied their learnt clustering algorithm to logged scan parameters from earlier software versions and could complete the examined body region information in the respective result tables. Furthermore, another problem of changing logged events is that rigidly implemented parsing rules will fail and lead to incomplete extracted data. Practice has shown that in some cases patterns are flexible enough, in other cases patterns are manually adjusted in time before patterns fail due to changed log file content. However, those two described scenarios do not always apply which leads to failing patterns and missing output data. As this is the very first step of turning raw data into actionable insights, failing patterns are a major problem for all subsequent data analysis steps. Figure 1 depicts exactly this scenario, where two systems produce log files during operation. The systems differ in their software version. This leads to varied key event tags which are crucial for the parser. In Figure 1, we marked the difference in the log files in grey. Applying the same, rigid parser can result in complete tables for system A on the one hand, however incomplete resulting tables for system B on the other.

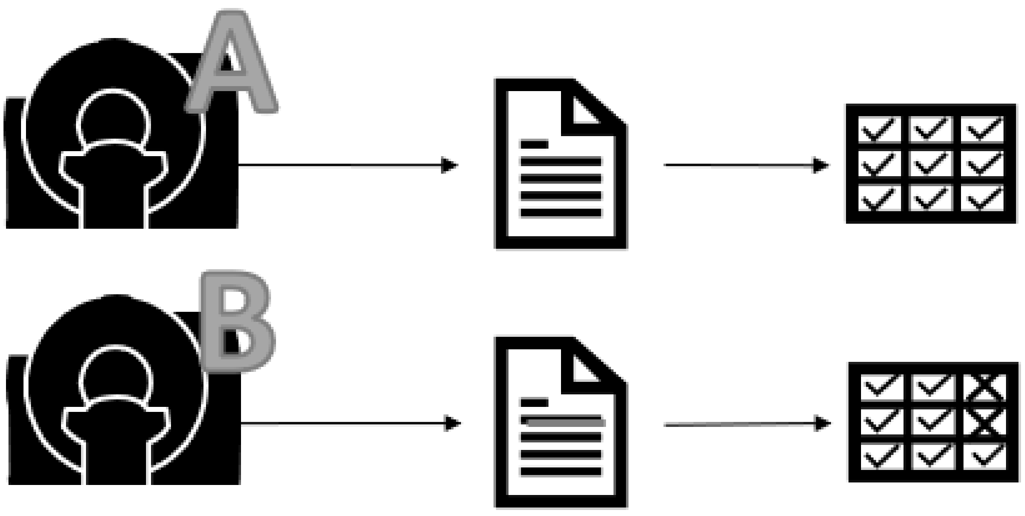

Figure 1. Problem Statement: Emerging log file entries processed by rigid parsers will lead to missing entries in output data.

Our target is a model which parses text and is flexible to adapt automatically to gradual changes in the text. In this example, we use text from log files produced by Computed Tomography (CT) systems during operation. Next to medical imaging data, CTs constantly write events into text files which are subject to our research. As described in Maier et al. [2], during acquisition, a CT system applies X-rays from different angles and records multiple projection images. Thus, 3-D reconstruction enables cross-sectional views of the examined objects. The examined body is exposed to X-rays which can harm healthy tissue. Patient’s health and regulatory limitations require to measure the radiation dose being applied to the patient. The exposed radiation dose is one Key Performance Indicator (KPI) which is recorded in an event log file and is denoted as Computed Tomography Dose Index (CTDI).

2. STATE OF THE ART

The high importance of turning plain log files into usable knowledge correlates with the large number of literature handling log file analysis. Already in 1993, Hansen and Atkins [3] applied algorithms for system monitoring and notification.

The most common methods of extracting information from log files base on detecting known fault types using regular expressions [4][5]. This requires knowledge about the exact pattern upfront, which is not always given. Thus, several data mining approaches have been applied in order to discover trends and correlations without prior knowledge [6][7][8][9]. For example, Vaarandi [10] used clustering algorithms to find frequent patterns as well as identify anomalous log file lines.

Since log file entries are discrete, sequential data, applying Markov models is natural choice. Already in 1966, the statistical concept of Hidden Markov Models (HMM) was presented by Baum and Petrie [11], whereas Rabiner [12] took that concept further into practice. A HMM is a statistical signal model with unobservable (hidden) states whose likelihood only depends on the preceding state (Markov property). Emissions, also called outputs, are observable states connected to the hidden states by emission probabilities. Thus, a HMM is fully described by a set of hidden states, emissions, starting probabilities, transition probabilities and emission probabilities. By setting up a HMM, three fundamental problems can be addressed. First, the evaluation problem can be tackled using the forward-backward algorithm. Thus, the probability of a particular output sequence given the model can be determined. Secondly, given the model and a given output sequence, we want to find the most likely sequence of hidden states. This is solved by the Viterbi algorithm. Lastly, the so-called learning problem addresses finding the most likely set of state transition and emission probabilities given a set of emissions, which is solved by the Baum-Welch algorithm and used in fitting new data to a previously learnt HMM.

Furthermore, Yamanishi and Maruyama [9] addressed the issue of evolving sequences of events in the field of network failure monitoring and proposed to combine HMM mixture with adaptive learning of parameters to achieve dynamic modeling and adaptive tracking. More general term of data transformation for an increase of information content is data wrangling which is discussed by Endel et al. [13] in "how to make data useful again".

Classification is a Machine Learning (ML) technique which addresses the task of assigning data to classes. For example, Duda et al. [14] introduce classification by describing the task of categorizing images of fish to salmon or sea bass. Another application is to categorize lines of text according to similar semantics which means that text with similar meaning is likely to be classified equally. Dave et al. [15] classified product reviews by combining sentiment analysis per review with training a classifier to separate the relevant reviews from all available data.
Huge amounts of unstructured text are often tackled with various Information Retrieval Techniques which are discussed widely in literature [16]. In Sentiment Analysis, often Latent Semantic Analysis (LSA) [17] is used to extract contextual usage and meaning of words. LSA is also known as Latent Semantic Index (LSI) and is a ML approach to train a model on unstructured collection of text. It learns latent topics by first turning text into matrix form using bag of word (BoW) model. Subsequently, this matrix is decomposed by applying Singular Value Decomposition (SVD) as illustrated in
Figure. Thus, the $m\,x\,m$ term document matrix is factorized into an $m\,x\,n$ singular matrix holding term assignment per topic, an $n\,x\,n$ diagonal matrix containing topic importance, and lastly an $n\,x\,m$ singular matrix with topic distribution across documents.

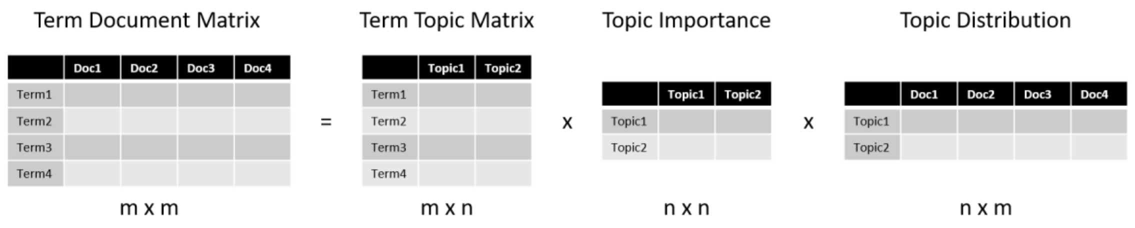

Figure 2. Singular Value Decomposition of a Term Document Matrix used in Latent Semantic Analysis.

The effectiveness of LDA models highly depends on the chosen number of topics. By considering each topic as a cluster one can find the optimum number of topics. Thus, the optimum number of topics is the one which corresponds to the most effective clustering. A measure for the effectiveness of a cluster is the Silhouette coefficient as introduced by Kaufman and Rousseeuw [18]. The silhouette coefficient is calculated as

$$Silhouette = \frac{b-a}{\max(a,b)}$$

where a stands for the mean intra-cluster distance and b holds the mean nearest-cluster distance for each sample. Thus, the Silhouette coefficient can take values between -1 and 1, where -1 denotes the most misclassifications and +1 the ideal case. Huang [19] discussed different common distance measures like Euclidean distance representing the distance between two points, Cosine similarity holding the cosine of the angle between vectors, Jaccard coefficient describing similarity as the intersection divided by the union of objects, and others.

Another way to determine the optimum number of topics is by measuring the coherence in topics [20] and choosing the number of topics that is most coherent. The coherence measure is calculated using the average and median of pair wise word similarity scores of words in a topic.

3. MATERIALS AND METHODS

Our goal is to build an adaptive log file parser. Thus, we want to extract specific, reoccurring information out of the raw log file despite of emerging, according entries and rules because of software version updates. Here, we present our approach using the example of amount of dose, called “ctdi”, which is applied during a Computed Tomography (CT) scan. We reach our goal by implementing a processing pipeline as illustrated in
Figure.

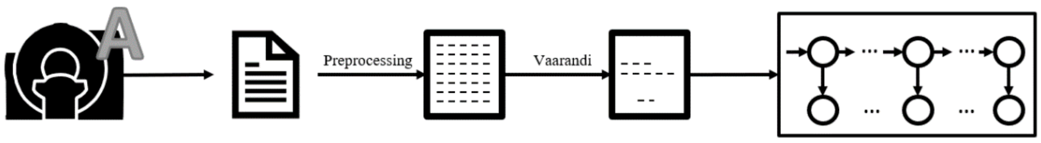

Figure 3. Processing pipeline of our adaptive parser using HMM.

3.1. Preprocessing

First, we apply several text preprocessing steps such as tokenization, stemming and lower capitalization. Furthermore, we remove all English stopwords and punctuation and receive a cleared set of tokens.

Semantics carry precious information which we take into consideration by training a LSA model. We applied LSA in order to limit the big data set to the topics of interest. Thus, we first turned the tokens into TF-IDF representation. Afterwards, we determined the optimal number of clusters by calculating the average silhouette score on a small data set using cosine and Euclidean distance as similarity measures. For that, we first determine the number of topics by calculating the silhouette coefficient for several numbers of topics. The highest found silhouette coefficient determines the optimum number of topics. This enables us to set up the LSA and cluster the log file lines into topics.

Based on that preprocessed and prefiltered data set we implemented and applied a slightly modified version of Vaarandi's "data clustering algorithm for mining patterns from event logs” [10] in order to find common structures. Vaarandi’s original approach consists of mainly three steps. First, the frequency of words is calculated and only the most frequent words are taken into account. Next, based on the found most frequent words cluster candidates are constructed. They contain one or more frequent words and are found in the same line of text. Afterwards, for each cluster candidate the according number of occurrences is calculated and reported as support value. In the third and last step the final clusters are selected from the cluster candidates by filtering the support values greater or equal to a given threshold.

In our approach, because of software adaptions we often face irregular order of tokens. As different orders of events do not have an impact on the desired parsing results, we do not consider the word's position and order unlike suggested by Vaarandi. As we focus on the dose value in this paper, we can assume to find only one kind of KPI to be parsed per line and parsing pattern. Thus, we use the number of KPI values as a threshold rather than setting or optimizing a specific threshold as Vaarandi proposed. The result is a vector of clusters representing frequently occurring types of lines. We further reduce the resulting clusters to the number of expected distinct KPIs (e.g. “ctdi”) and choose the most likely clusters.

3.2. Hidden Markov Model

The preprocessed data is subsequently utilized to set up and train a HMM. The found cluster is interpreted as a parsing pattern and determines the states of our HMM. The starting probability p_{si} reflects the frequency of the tokens, accordingly. A first-order Markov model depicts a sequence of states, where one state's likelihood can be predicted using its conditional probability given one preceding state. Therefore, next to the probability of the HMM's entry point, information about the subsequent state's likelihood $pt_{i,j}$ of occurrence is stored in a transition probability matrix by utilizing the concept of bigrams. A schematic presentation of the model is displayed in Figure 4. The initial starting probability is indicated by arrows drawn from *Start* to the respective $state_i$. Circles in the middle lane represent the different states. Arrows between the states illustrate likelihood of one state being followed by the other. Furthermore, dotted lines connect states to emissions and specify the emission probability, respectively.

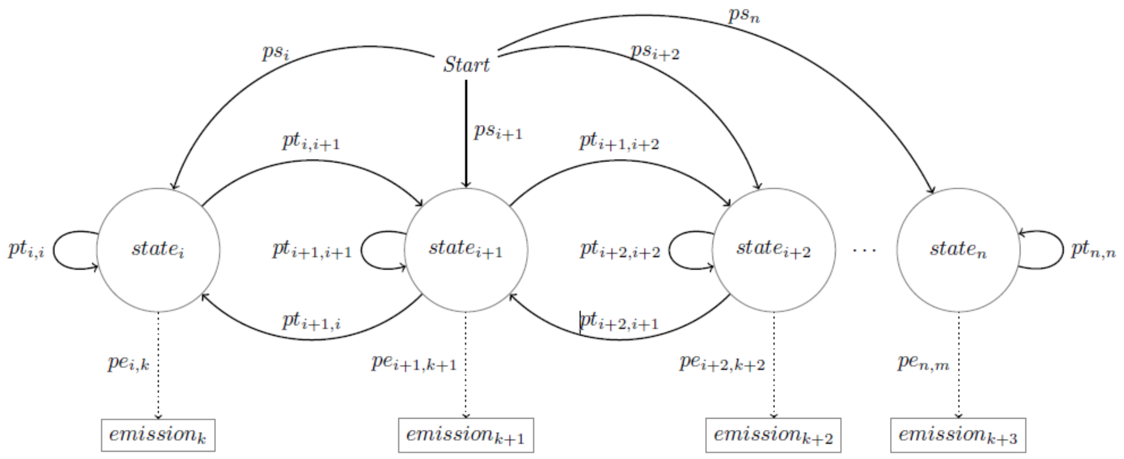

Figure 4. Parsing one element represented by a HMM.

Thus, our HMM is built and fully described by a vector of states, emissions, starting, transition and emission probability matrix. The states are hidden and represent entries in the original log file. Emissions can be observed and found as dose values in the output tables of the parsing process. $pe_{i,k}$ describes the probability that a certain emission k follows a given state i. This defines our HMM exhaustively.

Once we built the HMM representing a flexible version of a parsing pattern for dose values on one system type (Data *A*), we can apply the abstract pattern to data of a different system type (Data *B*) in order to extract the relevant dose information. Furthermore, we find the most probable states for a new observation sequence using Viterbi [12]. This is illustrated in Figure 5, as well as using Baum-Welch [12] to fit our model to new data. This implies that we update the transition and emission probabilities which connect the states with each other and states with emissions, accordingly. The HMM representing a flexible version of the parsing pattern is adapted to new Data B in two ways, using Viterbi as well as Baum-Welch, in order to get the correctly parsed output table.

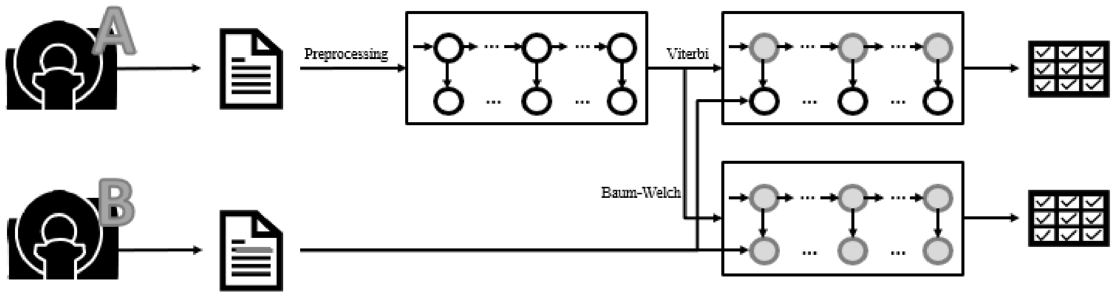

Figure 5. Pipeline of training the HMM on Data A and applying the implicit pattern on Data B.

4. RESULTS

In this chapter, we present the results of our flexible log file parser as well as the intermediate results along the introduced pipeline. We trained our new, flexible parsing model on one data set from system A. For testing of the desired adaptability, we tested our model on a different data set originating from another system B as demonstrated in Figure 6. The input data constitutes log file entries and accordingly parsed values of applied dose values from one CT system over the period of December 2018, further referenced by data set *A*. In order to test our approach on rigid patterns where the algorithm does not have to adapt, but only parse in the same way as it has been trained on, we split *A* into training and testing using a stratified split. In the following, we will refer to 70% of data set *A* by A_{train} and use the term A_{test} for the remaining 30% of *A*. Thus, we can test the trained algorithm on the very same system. Furthermore, we will call data from a different CT system of type B from December 2018 for testing purposes of our algorithm's adaptability as data set B_{test}, accordingly. The split is visualized in Figure 6.

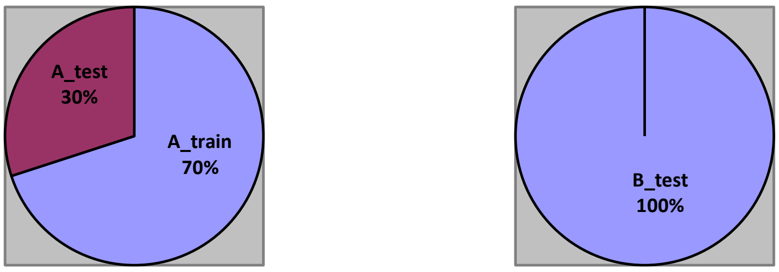

Figure 6. Two data sets from different CT system types are used, whereas the data of system A is split into training and testing for rigid parsing patterns. Further testing for adaptability of the trained model is performed on data of system B.

4.1. Preprocessing Results

Analogously to our processing pipeline, we present intermediate results along the pipeline and evaluate all steps in order to judge their success and discuss their importance. Figure 7 shows an exemplary, anonymized example event text of a raw log file which carries information about a specific scan. Among that information, also the amount of applied dose can be found. This event text is tied to a time stamp, event type and event ID.

```
&Load scan protocol&,@Patient
LOID@=#2.0.123456#,@Scan@=#1#,@ScanUID@=#1.3.12.2.1107.5.1.4.83004.1234567890
#,@Scan protocol name@=#rot00#,@Organ characteristics@=#MlOrgCharAbdomen#,@Body
size original@=#MlAdult#,@Scan entry name@=#rot00#,@Kind@=#MlRot#,@Entry
Mode@=#standard#,@AutoRange@=#Cont#,@kV@=#120#,@mAs@=#250#,@CARE
Dose@=#Off#,@AEC@=#Off#,@CTDI@=#16.660#,@DLP@=#59.975#,@Slice@=#0.6#,@
Scan start@=#MlRangeStartAuto#,@Slice Width Collimated@=#60#,@No Of Acquisition
Slices@=#60#,@CBC@=#Off#,@Scan trigger@=#MlScanTriggerAuto#,@No of
scans@=#1#,@Examination
time@=#0.500000#,@ScanTime@=#1.000#,@RotTime@=#0.500#,@RotKind@=#Normal#,
@CurrentPeak@=#250#,@DoseModulationType@=#MlNoModulation#,@Focus@=#MlSmal
lFocus#,@Anodespeed
A@=#120#,@StartDelay@=#2.000#,@NoOfClustersPerRange@=#1#,@RevolAngle@=#360
#,@Contrast@=#false#,@Begin Pos@=#517.000#,@Readings
A@=#2304#,@Scandirection@=#cr-
ca#,@MasterXray@=#On#,@Service@=#On#,@CycleTime@=#0.00#,@ZigZagReconVolum
e@=#0.00#,@ZigZagScanTime@=#0.00#,@EndPos@=#517#,@SpecialMeas@=#None#
```

Figure 7. Example of an anonymized, raw event text.

As described in Section 2, this event text is further processed into stemmed tokens. We present in Figure 8 the remaining word stem where punctuation as well as stopwords have been removed. In order to avoid misleading mismatches of values that are rounded differently, we reduce the digits after the decimal point to two.

```
'load', 'scan', 'protocol', 'paty', 'loid', '2.0.123456, 'scan', '1.00', 'scanuid',
'1.3.12.2.1107.5.1.4.83004.1234567890, 'scan', 'protocol', 'nam', 'rot00', 'org', 'charact',
'mlorgcharabdom', 'body', 'siz', 'origin', 'mladult', 'scan', 'entry', 'nam', 'rot00', 'kind', 'mlrot',
'entry', 'mod', 'standard', 'autorang', 'cont', 'kv', '120.00', 'mas', '250.00', 'car', 'dos', 'off', 'aec',
'off', 'ctdi', '16.66', 'dlp', '59.98', 'slic', '0.60', 'scan', 'start', 'mlrangestartauto', 'slic', 'wid', 'collim',
'60.00', 'no', 'of', 'acquisit', 'slic', '60.00', 'cbc', 'off', 'scan', 'trig', 'mlscantriggerauto', 'no', 'scan',
'1.00', 'examin', 'tim', '0.50', 'scantim', '1.00', 'rottim', '0.50', 'rotkind', 'norm', 'currentpeak',
'250.00', 'dosemodulationtyp', 'mlnomodulation', 'foc', 'mlsmallfocus', 'anodespee', 'a', '120.00',
'startdelay', '2.00', 'noofclustersperrang', '1.00', 'revolangl', '360.00', 'contrast', 'fals', 'begin',
'pos', '517.00', 'read', 'a', '2304.00', 'scandirect', 'cr-ca', 'masterxray', 'on', 'serv', 'on', 'cycletim',
'0.00', 'zigzagreconvolum', '0.00', 'zigzagscantim', '0.00', 'endpo', '517.00', 'specialmea', 'non'
```

Figure 8. The example event text represented as preprocessed tokens.

All further calculations are based on those preprocessed tokens. Thus, we set up a TF-IDF using these tokens to calculate distance matrices and, moreover, average silhouette scores to find the optimal number of clusters. Figure 9 illustrates for Euclidean and Cosine distance measure the resulting silhouette scores for two to 50 clusters in steps of two. As the silhouette score using Cosine distance reports the highest value for a number of 32 clusters, we performed a PCA and visualized the spread, accordingly. Figure 10 holds the plot of TF-IDF reduced to 32 clusters which are highlighted by different colors. Consequently, we select the cluster among the found 32 clusters which contains the highest number of emissions. Thus, we focus only on the so-called “ctdi-cluster” containing most dose values and similar entries.

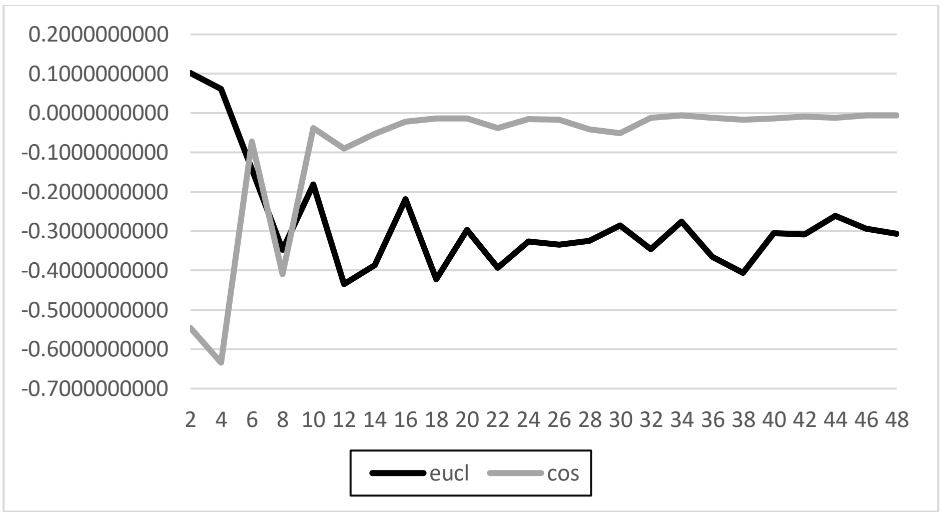

Figure 9. Silhouette score for Euclidean and Cosine distance measure for 2 to 50 clusters.

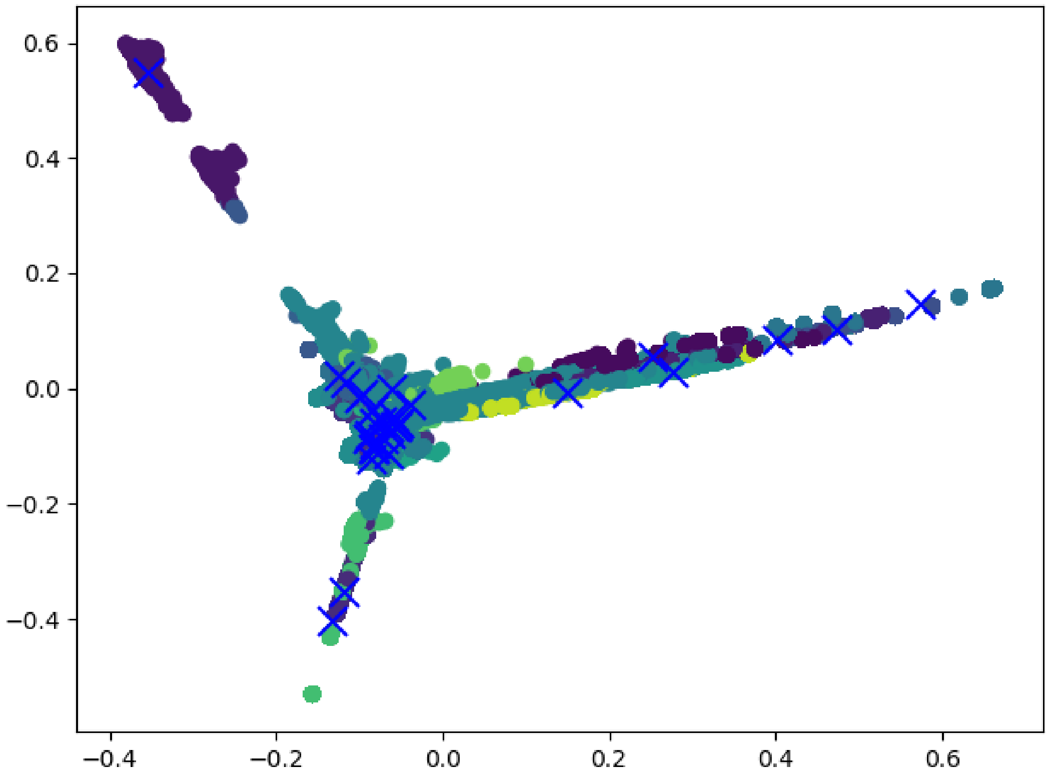

Figure 10. PCA for TF-IDF mapped to 32 clusters.

Moreover, by applying an adapted version of Vaarandi [10] to the “ctdi-cluster”, we detect the appropriate text that contains the desired dose information, automatically. Furthermore, we find again clusters in the reduced but still large amount of log file content. We learnt that a subset of the tokens is the representative, common structure applicable to all lines. In Figure 11 we highlight the found cluster items among the list of the semantically prefiltered and preprocessed tokens. Moreover, the elements of this cluster are used as states of the HMM and states the basis for all following steps.

'load', 'scan', 'protocol', 'paty', 'loid', '2.0.123456, 'scan', '1.00', 'scanuid', '1.3.12.2.1107.5.1.4.83004.1234567890, 'scan', 'protocol', 'nam', 'rot00', 'org', 'charact', 'mlorgcharabdom', 'body', 'siz', 'origin', 'mladult', 'scan', 'entry', 'nam', 'rot00', 'kind', 'mlrot', 'entry', 'mod', 'standard', 'autorang', 'cont', 'kv', '120.00', 'mas', '250.00', 'car', 'dos', 'off', 'aec', 'off', 'ctdi', '16.66', 'dlp', '59.98', 'slic', '0.60', 'scan', 'start', 'mlrangestartauto', 'slic', 'wid', 'collim', '60.00', 'no', 'of', 'acquisit', 'slic', '60.00', 'cbc', 'off', 'scan', 'trig', 'mlscantriggerauto', 'no', 'scan', '1.00', 'examin', 'tim', '0.50', 'scantim', '1.00', 'rottim', '0.50', 'rotkind', 'norm', 'currentpeak', '250.00', 'dosemodulationtyp', 'mlnomodulation', 'foc', 'mlsmallfocus', 'anodespee', 'a', '120.00', 'startdelay', '2.00', 'noofclustersperrang', '1.00', 'revolangl', '360.00', 'contrast', 'fals', 'begin', 'pos', '517.00', 'read', 'a', '2304.00', 'scandirect', 'cr-ca', 'masterxray', 'on', 'serv', 'on', 'cycletim', '0.00', 'zigzagreconvolum', '0.00', 'zigzagscantim', '0.00', 'endpo', '517.00', 'specialmea', 'non'

Figure 11. The identified cluster elements are highlighted which are found among the preprocessed tokens.

4.2. HMM Set Up Results

We built the HMM based on the preprocessed tokens, found clusters, and emissions. In order to assemble a flexible but precise parser, we found the token which is followed most often by emissions, automatically. In our case, we found the token “ctdi” to emit most values, correctly. We integrated here a flexible version as well, as also the trigger for an emission can be subject to changes. For example, “ctdivol” is an equivalent term for a dose value and could be the trigger for an emission in log files produced by other software versions.

In order to turn our model into an adaptive parser and successfully retrieve complete information from new data sets, we set up the parsing pattern depending on the trained HMM and the most emitting state. If we find all elements of the pattern as tokens in the event text, the emitted value after “ctdi” is considered as a found emission. The emission found in the example of Figure 11 is “16.66”.

4.3. Results for Training on Data Set A_train and Testing on Data Set A_test

As we determined an adaptive version of a parsing pattern, we can now apply the pattern in order to parse dose information out of the event text. First, we test if the built parser is representative and works without necessary adaption correctly. Thus, we trained our model on A_{train} and then tested it directly without further fitting on A_{test}. We achieved an accuracy of 99.99% and sensitivity of 98.24%. The respective confusion matrix is given in Table 1.

Table 1. Confusion matrix after training on A_{train} and testing on A_{test}.

		True Parsing	
		Positive	Negative
HMM Parsing	Positive	3293	0
	Negative	59	856393

4.4. Results for Training on Data Set A and Testing on Data Set B

After testing our model without the necessity for adaption, we now test the trained model on a different data set which holds slight variations in the respective event text. Thus, we can test our model on adaptability and flexibility. Thus, we trained our model on Data A and applied it to Data B. Figure 12 shows an excerpt of the differences in the event texts produced by both systems, respectively. The main difference is highlighted in yellow, where Data Set A contains “ScanUID” which was changed to “StudyLOID” in the event text produced by system B.

&Load scan protocol&,@Patient LOID@=#2.0.107559#,@Scan@=#1#,@ScanUID@=#1.3.12.2.1107.5.1.4.83004.1234567890#,@Scan protocol name@=#rot00#,@Organ characteristics@=#MlOrgCharAbdomen#,@Body size original@=#MlAdult#,@Scan entry name@=#rot00#,@Kind@=#MlRot#,@Entry Mode@=#standard#,@AutoRange@=#Cont#,@kV@=#120#,@mAs@=#250#,@CARE Dose@=#Off#,@AEC@=#Off#,@CTDI@=#16.660#,@DLP@=#59.975#,@Slice@=#0.6#,@Scan start@=#MlRangeStartAuto#,@Slice Width Collimated@=#60#,@No Of Acquisition Slices@=#60#,@CBC@=#Off#	&Load scan protocol&,@Patient LOID@=#4.0.123727110#,@Scan@=#1#,@StudyLOID@=#1.3.12.2.1107.5.1.4.73307.0987654321#,@Scan protocol name@=#1_HeadSequence#,@Organ characteristics@=#MlOrgCharHead#,@Body size original@=#MlAdult#,@Scan entry name@=#Topogram#,@Kind@=#MlTopo#,@Entry Mode@=#standard#,@AutoRange@=#None#,@kV@=#80#,@mA@=#20#,@CARE Dose@=#Off#,@AEC@=#Off#,@CTDI@=#0.023#,@DLP@=#0.597#,@Slice@=#0.6#,@Scan start@=#MlRangeStartConsole#,@Slice Width Collimated@=#60#,@No Of Acquisition Slices@=#6#,@CBC@=#Off

Figure 12. Example of an anonymized, raw event text from System A compared to System B, respectively. The main difference in tokens is highlighted in yellow.

Without adapting the model but applying the model on data generated from a different system, we observe an accuracy of 99.84% and sensitivity of 78.35%. The respective confusion matrix is given in Table 2.

Table 2. Confusion matrix after training on A and testing on B without adaption.

		True Parsing	
		Positive	Negative
HMM Parsing	Positive	2183	314
	Negative	603	571163

After fitting the model to the new Data B using Baum-Welch and again applying the adapted parsing rules we receive an accuracy of 99.78% and hit rate of 100.0%. The confusion matrix can be found in Table 3, accordingly.

Table 3. Confusion matrix after training on A, fitting model to B and testing on B.

		True Parsing	
		Positive	Negative
HMM Parsing	Positive	773	1255
	Negative	0	577558

Furthermore, we adapted the model by applying Viterbi to the learnt HMM with the emissions of Data B. This gave us results with 99.28% accuracy and hit rate of 56.44%. We present the respective confusion matrix in Table 4, accordingly.

Table 4. Confusion matrix after training on A, fitting only states to B and testing on B.

		True Parsing	
		Positive	Negative
HMM Parsing	Positive	2701	2028
	Negative	2085	578290

5. DISCUSSION

We built a flexible parser which adapts to gradual changes in log files. We tested our model on different set ups with two different data sources in order to evaluate and further improve our approach. In the following, we discuss the results in more detail and propose how to interpret those.

5.1. Discussion on Intermediate Results along the Pipeline

We presented the intermediate results of our first processing step using the example of one single line of log file content. We applied tokenization, stemming and removed stopwords and presented the resulting tokens in Figure 8. Thus, we could reduce complexity and set the base for further processing. Subsequently, we took the meaning of words into account and applied LSA. The optimum number of clusters being 32 mirrors the huge variety of information contained in our data set. Thus, we sliced the input into semantically similar clusters and reduced the input to the clusters containing the events of interest. Finding the most decisive tokens out of the entire data set works well as shown in Figure 11. The identified cluster elements primarily describe reoccurring elements and important items of lines which distinguish desired lines from others correctly.

5.2. Discussion on Results for Training on Data Set A_train and Testing on Data Set A_test

Furthermore, we used the found clusters in combination with the most emitting state as parsing pattern and tested our model on its basic functionality of parsing without adaption. In order to judge the quality and significance of our model, we trained our model on Data A_{train} and double checked the fundamental structure by directly testing on A_{test}. This implies that our model does not have to adapt to any changes in the input, yet. We could parse almost all desired values and achieved an accuracy of 99.99% and hit rate of 98.24%. As the confusion matrix presented in Table 1 shows in zero false positives, we did not parse wrong values. However, we missed to detect a few values. We explain the few false negatives with some very rare occurrences of line types which only occurred in A_{test} and, thus, have not been trained on.

5.3. Discussion on Results for Training on Data Set A and Testing on Data Set B

Our major goal was to build an adaptive parser which can parse gradually changing inputs without missing relevant information. In order to evaluate our model on flexibility, we trained our model on data set A and tested on B. Without fitting our model to data B, we already achieve an

accuracy of 99.84%. Having a closer look at the confusion matrix presented in Table 2, we see that we miss some true values while others are found to be false positive. As we now trained on the full data set A, we also trained on the rare cases of lines which have been missed before.
After we fitted our model to the emissions of data set B, we accomplished a hit rate of one hundred percent. We achieve that as applying Baum-Welch and fitting our model to B implies reducing the restrictiveness of the cluster and its length to only two distinct clusters. This leads to no false negative but 1255 false positives which means that we find wrong values to carry dose information.

Finally, Viterbi algorithm was applied to the model. This amplified our clusters to such an extent that we parsed almost as many false negatives and false positives as we found true positives. The model got sensitive and more descriptive towards wrong event text lines and patterns and led to a hit rate of 56.44%.

6. Conclusions and Future Work

We built a flexible parsing model which is capable to adapt to gradual changes in input structures. Our model is adaptive to slight changes in input log file and, thus, parses new input with very high accuracy. By constant learning and fitting our model using Baum-Welch, we continuously adapt our parsing rules to the changes in the input data. However, sensitivity should be improved, and further analysis is needed to judge the generality of our model. We found that applying Baum-Welch leads to a very well applicable parsing pattern, whereas Viterbi delivers too restrictive rules.

The good results of high accuracy mean that the combination of HMMs with text preprocessing and latent semantic analysis support the construction of new, flexible, learning models for information retrieval. By applying HMMs in combination with text processing and semantics, we contribute to research around text data mining and parsing. Thus, we enriched understanding and importance of this field of research’s opportunities and added a new perspective on flexible parsing. Furthermore, as our research combines semantic analysis, parsing, language processing and pattern recognition, it also contributes to the field of information retrieval. Due to our knowledge, we are the first who apply semantic analysis and HMMs to machine written text in order to build a flexible, automatically adapting parser. In practice, this is a starting point to automate information retrieval in log files for any system. Therefore, data analysts can base their algorithms on stable, high quality, preprocessed data. Thus, companies installing this model holistically, can reduce their maintenance costs for parsers drastically while maintaining high quality business insights throughout their systems lifecycles.

In next steps and future work, the model should be enhanced to parse several tokens at a time and not be limited to one aspect of information. In our example, an ideal system would parse all values belonging to all tokens from “Patient LOID” to “SpecialMeas”. Furthermore, current manual parsers do not only parse values but also time frames, durations and distances. Thus, in order to fully replace a manually assembled parser, the model should be enhanced to detect reoccurring patterns of line pairs. Their relationship should be detected automatically, as well as the determining information. In further research, the algorithm should be extended to even more complex patterns and parse KPIs that are found as a combination from more than two lines in the event text.

REFERENCES

[1] Kuhnert, N., Lindenmayr, O., & Maier, A. (2017, May). Classification of body regions based on MRI log files. In *International Conference on Computer Recognition Systems* (pp. 102-109). Springer, Cham.

[2] Maier, A., Steidl, S., Christlein, V., & Hornegger, J. (Eds.). (2018). *Medical Imaging Systems: An Introductory Guide* (Vol. 11111). Springer.

[3] Hansen, S. E., & Atkins, E. T. (1993, November). Automated System Monitoring and Notification with Swatch. In *LISA* (Vol. 93, pp. 145-152).

[4] Prewett, J. E. (2003, June). Analyzing cluster log files using logsurfer. In *Proceedings of the 4th Annual Conference on Linux Clusters*. Citeseer.

[5] Sahoo, R. K., Oliner, A. J., Rish, I., Gupta, M., Moreira, J. E., Ma, S., ... & Sivasubramaniam, A. (2003, August). Critical event prediction for proactive management in large-scale computer clusters. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 426-435). ACM.

[6] Han, J., Pei, J., & Yin, Y. (2000, May). Mining frequent patterns without candidate generation. In *ACM sigmod record* (Vol. 29, No. 2, pp. 1-12). ACM.

[7] Ma, S., & Hellerstein, J. L. (2001, April). Mining partially periodic event patterns with unknown periods. In *Proceedings 17th International Conference on Data Engineering* (pp. 205-214). IEEE.

[8] Stearley, J. (2004, September). Towards informatic analysis of syslogs. In *2004 IEEE International Conference on Cluster Computing (IEEE Cat. No. 04EX935)* (pp. 309-318). IEEE.

[9] Yamanishi, K., & Maruyama, Y. (2005, August). Dynamic syslog mining for network failure monitoring. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining* (pp. 499-508). ACM.

[10] Vaarandi, R. (2004, November). A breadth-first algorithm for mining frequent patterns from event logs. In *International Conference on Intelligence in Communication Systems* (pp. 293-308). Springer, Berlin, Heidelberg.

[11] Baum, L. E., & Petrie, T. (1966). Statistical inference for probabilistic functions of finite state Markov chains. *The annals of mathematical statistics*, *37*(6), 1554-1563.

[12] Rabiner, L. R. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, *77*(2), 257-286.

[13] Endel, F., & Piringer, H. (2015). Data Wrangling: Making data useful again. *IFAC-PapersOnLine*, *48*(1), 111-112.

[14] Duda, R. O., Hart, P. E., & Stork, D. G. (2012). *Pattern classification*. John Wiley & Sons.

[15] Dave, K., Lawrence, S., & Pennock, D. M. (2003, May). Mining the peanut gallery: Opinion extraction and semantic classification of product reviews. In *Proceedings of the 12th international conference on World Wide Web* (pp. 519-528). ACM.

[16] Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., & Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American society for information science*, *41*(6), 391-407.

[17] Schütze, H., Manning, C. D., & Raghavan, P. (2008, June). Introduction to information retrieval. In *Proceedings of the international communication of association for computing machinery conference* (p. 260).

[18] Kaufman, L., & Rousseeuw, P. J. (2009). *Finding groups in data: an introduction to cluster analysis* (Vol. 344). John Wiley & Sons.

[19] Huang, A. (2008, April). Similarity measures for text document clustering. In *Proceedings of the sixth new zealand computer science research student conference (NZCSRSC2008), Christchurch, New Zealand* (Vol. 4, pp. 9-56).

[20] Röder, M., Both, A., & Hinneburg, A. (2015, February). Exploring the space of topic coherence measures. In *Proceedings of the eighth ACM international conference on Web search and data mining* (pp. 399-408). ACM.

AUTHORS

Nadine Kuhnert studied medical engineering to receive her Bachelor's degree in 2013 and finished her Master's in Information and Communication technology in 2015 at the University of Erlangen-Nuremberg. From 2015 to 2018, she joined Siemens Healthineers and worked on usage data analysis and managed a consultative approach based on market and usage data in order to guide customers to the best fitting options and upgrades for their medical imaging systems. Since 2019, Nadine Kuhnert is product manager of a customer service portal of Siemens Healthineers.

Since 2016, she is working on her PhD on the topic of data wrangling applied to medical imaging system log files at the pattern recognition lab in Erlangen, Germany.

Prof. Dr. Andreas Maier studied Computer Science, graduated in 2005, and received his PhD in 2009 at the University of Erlangen-Nuremberg. From 2009 to 2010, he started working on flat-panel C-arm CT as post-doctoral fellow at the Radiological Sciences Laboratory in the Department of Radiology at the Stanford University. From 2011 to 2012 he joined Siemens Healthcare as innovation project manager and was responsible for reconstruction topics in the Angiography and X-ray business unit.

In 2012, he returned the University of Erlangen-Nuremberg as head of the Medical Reconstruction Group at the Pattern Recognition lab. In 2015 he became professor and head of the Pattern Recognition Lab. Since 2016, he is member of the steering committee of the European Time Machine Consortium. In 2018, he was awarded an ERC Synergy Grant "4D nanoscope". Current research interests focus on medical imaging, image and audio processing, digital humanities, and interpretable machine learning and the use of known operators.