

Institute for Visualization and Interactive Systems

University of Stuttgart
Universitätsstraße 38
D-70569 Stuttgart

Master Thesis

Real-time Analytics and Monitoring of ML-Applications using Visual Analytics

Shaista Sultana

Course of Study:	M.Sc. INFOTECH
Examiner:	Prof. Dr. Thomas Ertl
Supervisor:	Dr. Dennis Thom, Dr. Sebastian Klenk
Commenced:	February 1, 2018
Completed:	August 1, 2018

Abstract

In the quest for scientific developments and advancements in the society, machine learning applications are becoming part of almost every process in the industries. The world is heading towards the utilization of experience gained by machines. What if the experience is gained from faulty dataset or the predictions of the machine learning algorithm are wrong due to some other reason? This would corrupt the entire system and lead to an enormous loss of time and money. So, it is important to take a note of the performance of the machine learning application before deploying it. In the current scenario, understanding of machine learning model is a continued field of research. The visual analytics approach which involves interactive visualization and explorative analysis of dataset can be exploited in the model development process, as it integrates human-knowledge with the power of machines. In this thesis, a contribution is made in this area for monitoring the machine learning application in real-time and analysis of the same using Visual Analytics to address the problem of concept drift. An approach is designed and a software implementation is done for its demonstration. Interactive visualizations have been provided for the actual dataset and the predictions obtained from the machine learning model. A simulation for continuous arrival of data streams has been developed. The idea is to recognize the right point of time at which a new model needs to be trained. Tools have been integrated for further interactive analysis of this dataset. As the data from a News Agency has been used, analysis of textual data and its visualization have formed a significant part of the visual explorative analysis.

Contents

1. Introduction	8
1.1 Problem statement	8
1.2 Goal	9
1.3 Outline.....	9
2. Fundamentals	10
2.1 Basic theoretical background.....	10
2.1.1 Information Visualization.....	10
2.1.2 Visual Analytics	11
2.1.3 Machine learning	14
2.1.4 Visualizations using the concept of stacked graphs.....	16
2.1.5 Topic Modelling.....	20
2.1.6 Circle Packed Graph	20
2.2 Related works	22
2.2.1 Machine Learning and Visual Analytics.....	22
2.2.2 Concept drift	25
3. Approach.....	27
3.1 Specific requirements	27
3.2 Design of approach	28
3.2.1 Database	28
3.2.2 Backend logic	29
3.2.3 Classifier	34
3.2.4 Visualization Controller.....	35
3.2.5 Frontend (Visualization).....	35
4. Implementation	36
4.1 Algorithm	36
4.2 Tools and Libraries Used	49
5. Case study	50
5.1 Classifier trained on 15,000 records data.....	50
5.1.1 Setup	50

5.1.2	General behavior of the classifier and visual analysis	50
5.1.3	Deviations observed in the behavior of classifier and visual analytics	54
5.1.4	Discussion on this case study	58
5.2	Classifier trained on 150 records of data	59
5.2.1	Setup	59
5.2.2	Monitoring and analysis of general behavior of the classifier	59
5.2.3	Discussion on this case study	64
6.	Conclusion	65
6.1	Summary	65
6.2	Outlook	66
7.	Bibliography	67

List of figures

Figure 2-1: Process of Visual Analytics [KKE+10]	13
Figure 2-2: Optimal separating hyperplane for SVM	16
Figure 2-3: Stacked graph functions [BW08]	18
Figure 2-4: Conventional stacked graph with $g_0 = 0$ [BW08]	18
Figure 2-5: ThemeRiver Example [BW08]	19
Figure 2-6: Streamgraph example [BW08]	19
Figure 2-7: Analogy between Treemaps and circle-packed graphs.[Rib12]	21
Figure 2-8: Pipeline of Machine Learning [LWL+17]	23
Figure 2-9: Interactive Model Analysis [LWL+17]	24
Figure 2-10: An illustration of supervised learning.....	25
Figure 3-1: Basic backend architecture of software	30
Figure 4-1: Integration of languages for the development of software for the thesis.....	37
Figure 4-2: GUI for the User.....	39
Figure 4-3: Streamgraphs for actual and predicted values of “ressort”	41
Figure 4-4: Selection of a portion of the Streamgraphs using the slider.	42
Figure 4-5: Update of fields below Static Streamgraph header	42
Figure 4-6: Zoom-in view for the portion selected by the slider	43
Figure 4-7: Update of textboxes after mouse clicks at selected position of the ribbons.	44
Figure 4-8: Parameters for applying LDA algorithm.	45
Figure 4-9: Circle packed graph	46
Figure 4-10: Zoomed-in view of the central inner circle inside Figure 4-9.	47
Figure 4-11: Text-box below the circle-packed graph	48
Figure 5-1: A screenshot of the continuous Streamgraphs showing their general behavior (case study 1).	51
Figure 5-2: Static observation of the Streamgraphs shown in Figure 5-1	52
Figure 5-3 : Circle-packed graph for “textForUi” corresponding to Figure 5-1.	53
Figure 5-4: Selected portion of continuous Streamgraphs	54
Figure 5-5: Actual data corresponding to 3a.m. for the blue ribbon.....	55
Figure 5-6: Predicted data corresponding to 3a.m. for the blue ribbon.	55
Figure 5-7: Circle-packed graph	56
Figure 5-8: A zoomed-in view of the biggest inner circle of Figure 5-7.	57
Figure 5-9: “textsForUi” which contain the words from Figure 5-8.	58
Figure 5-10: A screenshot of the Streamgraphs showing their general behavior	60
Figure 5-11: Circle-packed graph for “textForUi” corresponding to Figure 5-10.	61
Figure 5-12: A zoomed-in view of the biggest circle inside Figure 5-11.	62
Figure 5-13: Text-area for “textForUi” containing words of the circle shown in Figure 5-12.	63
Figure 5-14: A zoomed-in view of the 2 nd largest circle inside Figure 5-11.	64

1. Introduction

This chapter describes the existing problem and its criticality which we have tried to solve with this thesis. It focusses on the motivation which led to the idea of this work. It then throws light on the objective which we intend to achieve. Then, the structure of this document is described in the Outline section.

1.1 Problem statement

As the world is heading towards an era of digitization, machine learning applications are gaining tremendous significance. Machine learning applications are, therefore, being used in almost every sphere of life, ranging from day-to-day to safety-critical fields. For instance, they are widely being used in insurance sector to handle damage treatment, healthcare systems to monitor health and perform diagnosis, financial sector for market analysis and even in security applications. Failure in machine learning models can cause drastic and irreparable damages, resulting in loss of economy and even life. Failure of a security system may grant unauthorized access leading to severe data loss. A flaw in medical diagnosis can cost human lives. Incorrect market predictions could have a huge negative impact on the economy relying on these results. Thus, it is extremely important to verify the correctness of machine learning models being used.

As data and trends change over time, machine learning models start degrading, eventually becoming obsolete. Retraining the model again and again with every single change of data and each new incoming data, would not solve the problem, until and unless we know if the model is improving or not. So, identifying the right point of time to adapt a new predictive mechanism is a prime concern. Real-time monitoring and analysis of machine learning algorithm have a huge potential to solve this problem of maintaining up-to-date machine learning applications.

Several statistical methods have been proposed to monitor the quality of machine learning model and attempts have been made in the area of visual analytics to understand the model and build the model using interactive visualizations. Thomas and Cook [TC05] have provided deep insight in the area of visual analytics, which involves Information Visualization as well as the ability of analytical reasoning.

This work is an attempt to monitor the machine learning application in real-time to find deviations in its performance so that decisions can be made on its continuity of the use of the trained model. This combines the visual analytics approach of understanding and development of the model with the statistical method of monitoring and analyzing the model in real-time. A software has been developed and explanations are provided for each visualization to demonstrate the approach discussed above. Moreover, two case studies have been conducted to understand the viability of the software.

1.2 Goal

The goal of this thesis is to monitor real-time performance of machine learning application and analyze the same using visual analytics. So, visual explorative tools are given to the user to understand the normal behavior of the machine learning model and deviations in the same.

To achieve this aim, visual analytics approach [KMS+08] is used, where *automated data analysis* is done along with data collection, preprocessing, and interactive visualizations. In this work, static news dataset is simulated to provide real-time streaming. Then, actual labels are compared to predicted labels received from the classifier. Interactive visualization tools are developed to accomplish the comparison objective. Inspection of data is automated with the help of topic modelling and visualization of its result, so that users can probe into the details only upon necessity.

1.3 Outline

The rest of this master thesis document is structured in the following manner. First of all, an insight is provided on the basic theoretical background required to gain an understanding of this thesis. This includes the conceptual knowledge of information visualization, visual analytics, machine learning terms, topic modelling, and different forms of visualization used in this work. Then, relevant contributions made by different researchers in this area of work are included and a discussion is made of how this work has utilized the existing knowledge. Chapter 3 focusses on the approach in which the specific requirements to solve the problem are mentioned and the approach to achieve the objective of this thesis is described. Chapter 4 explains the implementation of the software developed to demonstrate the approach discussed in the previous chapter. The tools and libraries used in the software are also enumerated in this chapter. In Chapter 5, two case studies are presented to analyze the working of the developed software. Evaluations are done and the results are presented with visual explanations. A discussion is made on the obtained results. Eventually, the Conclusion summarizes the proposed technique concisely and discusses how this approach can be a contribution to solve the problem of analyzing the machine learning model. Furthermore, future possibilities for improvement and expansion of this work are proposed.

2. Fundamentals

In this chapter, an explanation of the basic concepts used in this work is provided and an analysis of the existing work related to this thesis is done. An understanding of the theoretical background is essential to dive into this work. In the section of related works, contributions in the field of machine learning, visual analytics, and concept drift are provided. Then, a discussion is made about how this work is an adaptation or extension of the existing work.

2.1 Basic theoretical background

In the theoretical background, visual analytics and related terms are described. Also, the concept of machine learning, its types and applications are included as the thesis is focused on real-time analytics of machine learning applications using visual analytics. This section throws light on the basic concepts of the important terms used in this thesis.

2.1.1 Information Visualization

As the world is heading towards technological advancement, the amount of data is growing at a rapid pace, due to which humans are heading towards the development of different mechanisms to store this enormous volume of data. Humans have been able to succeed in achieving the demand for storage of this incredible amount of data. But, this data is of hardly any usage until and unless we can extract useful information from the same. Information Visualization comes into play to provide a reasonable and effective solution to this problem of visualizing the massive and diverse data.

Ben Shneiderman [Shn96] had come up with the *Visual Information-Seeking Mantra*:

“Overview first, zoom and filter, then details on demand.”

He had mentioned the importance of visual approaches over textual ones for different types of data and cited the contributions made by early researchers in that area. For example, one way of dealing with *1-dimensional* data, which refers to the linear data type which includes a *string of characters*, was using the bifocal display technique [SAM82] for the presentation of this type of data. This technique maintains the full view of the entire dataset and enables the detailed view of sections of the same, maintaining the continuity across the border. For 2-dimensional data visualizations like *geographic maps, floorplans, or newspaper layouts*, users usually seek information such as the distance between regions, paths, size of locations, etc. Experts in the area of information visualization have used *spatial displays of document collections* for the further improvement of such visualizations. For *3-dimensional* objects of the physical world which possess the characteristics of mass and volume, it is crucial to have *high-resolution images* of the human body to excel in the area of medical sciences. Another important application of Information Visualization is the visualization of *temporal* data as timelines play a significant role in *project management*, display of historical data, *medical records*, where there is a possibility of overlapping entries at a particular instance of time. An early contribution in this area was done by Mackinlay et al., 1991 [MRG+91], where they had proposed the technique of Perspective Wall which offers an interactive 3-D wall, with the front segment is for viewing details and two side-regions represent the context. For *multi-dimensional* data, it is a common practice to represent n -attributes of a data-item as points in n -

dimensional space. Another data structure, *tree*, is widely used for portraying the relationships between the parent node and child nodes along with their attributes and features. Researchers have come up with various visualization techniques for trees, which are commonly used in daily life. For example, *treemaps* are very effective in viewing the directory structure of our personal computers. The network data type comes into picture to overcome the shortcomings of trees when there can be a one-to-many association between different items. Some of its advantages include the facilitation of traversal in the graphs along the shortest path. Early attempt in visualization of network type data structure included visualization of the content structure of the World Wide Web.

Furthermore, Schneiderman [Shn96] had mentioned the explanation of the process of information visualization which comprises of the following tasks:

- Overview: Here, the user gains a holistic view of the entire data. The user gets to look at the complete dataset as well as a detailed view by the side of it.
- Zoom: User has the control over the factor by which he wants to zoom in the interesting parts of the data.
- Filter: This gives the user more control over restricting the parts of the data that are visible to him so that he can concentrate more on the exciting parts. Employing sliders, scrollbars, and hiding fields can be major contributors to achieve this aspect.
- Details on demand: This enables the user to focus on the particulars when needed. This may involve a pop-up on a button-click for the selected item.
- Relate: This is used to view *relationships among items*. For example, in movie reviews, a user can select a particular actor's name and see the names of all the movies in which he performed.
- History: This gives the user the flexibility to keep track of events which have already passed by so that he can have a view of them, whenever desired.
- Extract: Extraction permits the user to select parts of information and save them so that they can be mailed, printed, dragged to another window, etc.

This thesis strives to include the above-mentioned tasks in the approach to solve the problem of real-time monitoring of machine learning application. In the software developed for demonstration of the approach, users are first given an overview of the results obtained by using the classifier model, then they have been provided with the options of zoom and filter. Also, users have the ability to query for detailed information and view connections among items. Moreover, users can also inquire for past data and view visualizations related to that.

2.1.2 Visual Analytics

Visual analytics can be considered to be an extension of the field of information visualization. Thomas and Cook have provided a deep insight into Visual Analytics in their pioneering work: "Illuminating the Path: The Research and Development Agenda for Visual Analytics" [TC05]. According to them,

"Visual analytics is the science of analytical reasoning facilitated by interactive visual interfaces."

They were one of the early contributors in the area of Visual Analytics. They have described visual analytics as a multifaceted area, which covers the aspects ranging from the visual representation and analytical reasoning of information to human interaction with this information, along with the methods to communicate the appropriate context of the result to the designated users. Not only this, but visual

analytics also enables the transformation of the extensive data into a form that can be used for analysis. This facilitates humans to get a deeper understanding of the data, which helps them come to better conclusions. This has led visual analytics to contribute significantly in the areas of physics, nuclear science, engineering and software analytics as well as day-to-day applications like environmental monitoring, financial applications, biology, medicine, and health.

Daniel Keim and Mansmann [KMS+08] mention the peculiar strength of visual analytics in the fact that it allows decision makers to focus their intellect on the process of analysis, as it integrates the power of machines and humans. Methods from knowledge discovery in databases, statistics and mathematics contribute significantly in the analytical process, whereas, the ability of humans to comprehend, collaborate, interact, present, judge and disseminate, add to the sheer competence of visual analytics, which turns out to be a promising area of research.

Visual analytics is a transformation from data to information. It involves *data preprocessing*, *visualization functions* and *user interactions*. There does not exist a clear line of separation between information visualization and visual analytics as visual analytics combines the strength of information visualization and *automated analysis*. Therefore, the mantra of visual analytics is [KMS+08]:

*“Analyse First -
Show the Important -
Zoom, Filter and Analyse Further -
Details on Demand”*

Keim et al. [KKE+10] presented their definition of Visual Analytics as:

“Visual analytics combines automated analysis techniques with interactive visualizations for an effective understanding, reasoning and decision making on the basis of very large and complex datasets”.

Figure 2-1 shows the Visual Analytics Process. Keim et al. [KKE+10] had vividly described the process of Visual Analytics using this figure, where the phases of the Visual Analytics Process are mentioned in the ovals and the arrows indicate how one stage proceeds to the next. In most of the cases, integration of heterogeneous data sources is needed before analytical procedures can be applied. So, the transformation of data (preprocessing) is required as shown in the figure. Once the transformation process has been done, the analyst has the choice to either apply visual analytics or automated analysis, as portrayed by the arrows emerging from “data” in the figure. The lower arrow illustrates the use of data mining for discovering patterns in the dataset and creation of models. After the creation of the model, the analyst can visualize the same for his own assessment, where the human knowledge can be utilized. If the analyst is not satisfied by the results, he/she can tweak the parameters or even change the existing algorithm used for data mining, unless satisfactory results are obtained. Thus, the cycle between model development and visualization is continued until the achievement of satisfactory results. If the pathway shown by the upper arrow is used first, extraction of information from data is done by mapping data structures to visual structures, where the data exploration process can be enhanced by human interactions with the visualization, which involves zooming, filtering, etc. This is similar to the process of Information Visualization, discussed in the previous section. Information obtained from visual data exploration can serve as an aid to the process of model development. The output of this process of visual analytics is the knowledge about the data, obtained as a result of interaction between humans, data,

models and visualization. This knowledge can serve as a feedback to select new inputs or help in the improvement of the data.

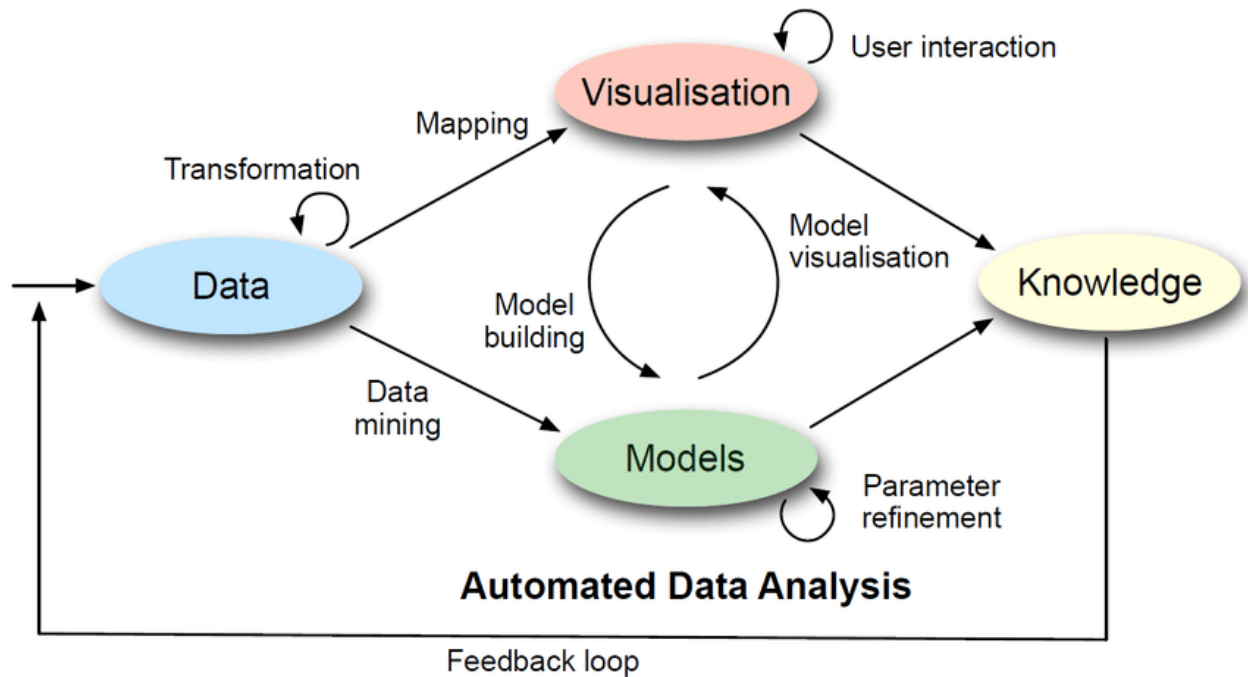


Figure 2-1: Process of Visual Analytics [KKE+10]

As the figure shows, data can be mapped to its visualization or models can be directly created from the data. Whether visualization is done in the first step or models are developed, in both the cases, an interplay between visualization and models takes place. The final result is the knowledge gained by the analyst.

As real-time analysis of machine learning algorithm is to be done using visual analytics, the described procedure has been adapted to meet the objective of this work. As the textual data for the realization of the approach used in the thesis is composed of several fields, which are not relevant in the context of this work, they have been filtered out and only useful data has been extracted from the database. Next, the textual data is preprocessed, which involves cleaning the data and converting the textual data into algebraic form, which can be used for training the classifier (discussed in Chapter 3). Now, an automated analysis of this data is done by building a classifier model from a portion of this data and choosing a machine learning algorithm. As the data provided is already labelled, the actual labels are compared to the labels generated from the classifier model, with the help of interactive visualizations. The classifier model is trained iteratively until satisfactory results are achieved for the user. This is done by selecting a different machine learning algorithm for training the classifier, and even changing the input data for training the classifier model. In this thesis, the goal is to find if the machine learning algorithm is performing well or not by comparing the actual labels with the predictions from the classifier model. So, analytical tools for analysis of textual data related to the labels, have also been incorporated in the visualizations, so that humans can understand whether the deviations in the behavior of classifier are due to sudden changes in the data or due to unsatisfactory classifier model.

2.1.3 Machine learning

Machine learning is a branch of Artificial Intelligence, which gives computers the capability to learn with the help of data that we feed to them, giving them the ability of problem-solving. This is unlike the standard software programs, where a set of executable instructions is given to the computer to perform a particular task. In order to accomplish the task of problem-solving, the machine learning algorithm is “trained” with large volumes of data, so that it can learn and improve itself. Machine learning has been in existence for a very long period of time. The term ‘Machine learning’ was coined by Arthur Samuel in 1959. According to Arthur Samuel:

"Machine learning is a field of study that gives computers the ability to learn without being explicitly programmed."

In this thesis, the basic concepts of machine learning were learnt from the book by Shalev-Schwartz and Ben-David, 2014 [SB14]. Nowadays, machine learning is used in every sphere of life, ranging from day-to-day applications like speech and voice-recognition to space research. For example, machine learning is widely used in the area of image-recognition to identify images and classify them. It is now possible to upload pictures on the search-engine websites and obtain data associated with them. Image-tagging in social networking websites also utilizes this principle. Nowadays, we can feed our smartphone with voice instructions and search the web, play music, control the connected devices, etc. Machine learning has made a significant contribution in this area of speech recognition. Machine learning is also widely used in the area of medical diagnosis to determine the associated risks of diseases and their severity. This is done by monitoring the health conditions of the patients and keeping a track record of the same.

Machine learning tasks can be broadly categorized into supervised, unsupervised and semi-supervised learning. In supervised learning, the model is trained with labelled data, i.e., the data to be fed to the model includes the inputs as well as their associated outputs, so that the model can make predictions about the unknown categories or values linked to the test dataset. In other words, in supervised learning, one tries to estimate a function that maps input to output, based on the observations made earlier. Supervised learning can further be categorized into classification and regression. A regression problem arises when one needs to predict the actual real value associated with the test dataset, not just a class label, for example, prediction of stock prices of companies. For example, regression can be used for prediction of stock price of a company, with the help of past trends in data. Given a dataset of the stock price of a company versus date, regression can be used for prediction of the value of the stock price for the upcoming month or so. Thus, regression is about continuous values, i.e., mapping a point to a real value using the trend observed for a collection of points whose output values are already given. Whereas, a classification problem occurs when data is to be categorized into a given number of classes. Classification can be binary as well as multi-class. Binary classification involves assigning the output variable to one of the two class labels. For example, detecting whether a mail is spam or not spam is a typical problem of binary classification. Whereas, predicting the category associated with a particular mail can be considered to be a multi-class classification problem, i.e., finding out whether the mail is about sports schedule, studies schedule, travel-itinerary, or job-related is a multi-class classification problem.

In contrast to supervised learning, the data fed to an unsupervised learning model does not include the outputs associated with the input data, i.e., the model is trained with unlabeled data. The goal of unsupervised learning is that the system figures out its own model. Different types of unsupervised learning include clustering and anomaly detection. In a typical problem of clustering, similar items are grouped together. Grouping together similar types of news items from a given set of unlabeled newspaper data can be considered to be an excellent example of clustering. Anomaly detection involves detection of anomalous data from a given dataset. For example, a car engine manufacturer has a set of data about the functional car engines, which includes features like power output, combustion efficiency, etc. Now, when a newly manufactured engine is to be tested, the manufacturer can use anomaly detection technique to test the functionality by checking if its associated data is consistent with the previous values or not.

In case of semi-supervised learning, input data comprises both labelled as well as unlabeled data. Semi-supervised learning combines the advantages of both supervised and unsupervised learning. As labelling of data by the humans is an extraordinarily time-taking and expensive task, and this results in the availability of less amount of labelled data. So, the ability to combine labelled as well as unlabelled data (available in abundance), for training the model can prove to be of significant advantage.

In this thesis, a set of labelled textual data is used to train the classifier model. The dataset is composed of news reports and they are assigned their respective labels of “genre”, “cities”, “version”, “ressort”, etc. The field “ressort” corresponds to the category to which the news item belongs like politics, sports, economy, etc. So, to illustrate the approach of this thesis, the classifier model was trained so as to predict the “ressort” label of the test-set. This decision was made because “ressort” is related to “textForUi”, i.e., the words present in the “textForUi” have an influence on the field “ressort”. There are several classification techniques that could be used to deal with textual dataset. An initial attempt was made to use the Naïve Bayes classifier, but the trained model was found to have very low accuracy. Then, Support Vector Machine was used for text classification and the evaluation results obtained were satisfactory. So, the Support Vector Machine (SVM) model was trained to address the multiclass classification problem.

If we have a set of labelled training data with two classes, the SVM algorithm builds a hyperplane to separate the two classes to categorize the new data samples into one of the two classes. Originally, SVM was developed by Cortes and Vapnik, 1995 [CV95] to solve binary classification problems. Cortes et al. used the idea of non-linear mapping of input vectors to a *feature space of very high dimension*, and then constructing a *linear decision boundary*, with an ability to generalize the trend as much as possible. The data on which the prediction is to be made is then mapped into this same feature space, and the decision of the prediction is determined by the side of the decision boundary on which the data point lies. Now comes the problem of creating *optimal separating hyperplanes* between the two separable classes. This was solved by Cortes and Vapnic by maximizing the margin between the closest points of the two classes as shown in Figure 2-2, where the points lying near the boundary are the support vectors and the optimal separating hyperplane is in the middle of the margin.

As SVMs are primarily intended to solve binary classification problems, some techniques have been developed to solve the issues of multi-class classification. Some of these are *one-against-all* and *one-against-one*. Milgram et al. [MCS06] had presented a comparative analysis of the two techniques and described these strategies. In the *single machine approach*, multi-class SVM is created by unravelling a

single optimization problem, whereas, in the *divide and conquer* approach, the multi-class problem is decomposed into binary sub-problems, and then, an SVM is built for each of them. In the *one-against-all* strategy of decomposition, one SVM is created for each class by training the samples in such a way that they can distinguish themselves from all the remaining classes, and identify themselves as belonging to their own class. In the *one-against-one* strategy, one SVM is built for each pair of classes. So, if there are n number of classes, $n(n-1)/2$ SVMs are constructed to separate the samples of one class from the others. In order to perform the classification of an unlabeled sample, the technique of obtaining maximum votes is used, in which each SVM votes for a single class. The class with the maximum pairwise decision takes the lead and is assigned as the label of the sample.

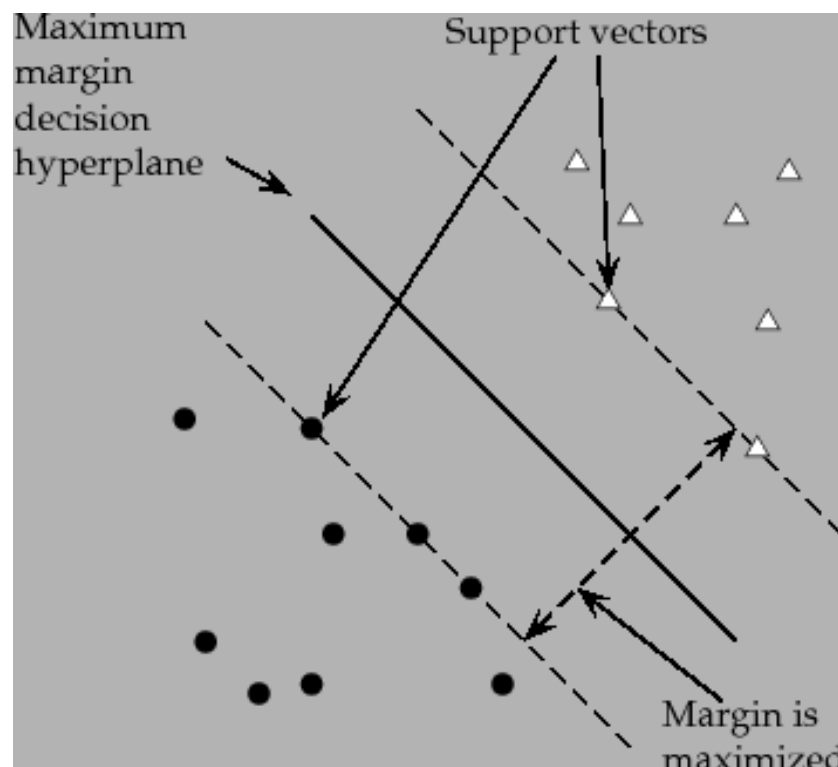


Figure 2-2: Optimal separating hyperplane for SVM¹

There could be many possibilities of setting up a hyperplane to separate the classes. But, an optimal separating hyperplane is the one that maximizes the distance between it and the nearest points on each side of the boundary.

2.1.4 Visualizations using the concept of stacked graphs

In 2000, Havre, Hetzler, and Nowell [HHN00] presented the idea of ThemeRiver, which they described as: “A prototype system that visualizes thematic variations over time within a large collection of documents. The “river” flows from left to right through time, changing width to depict changes in thematic strength of temporally associated documents. Colored “currents” flowing within the river narrow or widen to indicate

¹ <https://nlp.stanford.edu/IR-book/html/htmledition/support-vector-machines-the-linearly-separable-case-1.html>

decreases or increases in the strength of an individual topic or a group of topics in the associated documents."

The author mentioned a vital objective of Information Visualization as the presentation of information to facilitate pattern detection directly and swiftly. So, visualizing the trends over a period becomes an essential aspect in this regard. The patterns in ThemeRiver support the observation of *trends, relationships, anomalies, and structure in the data, unexpected occurrence or nonoccurrence of themes or topics*. The primary purpose of the design of ThemeRiver is the visualization of changes in themes over a period of time. For example, people may be interested in observing the changes in the theme of theatre plays with reference to changes in the social events and incidents. So, ThemeRiver provides the users with a visualization that enables the users to come up with more explorative questions and curiosities.

Furthermore, in addition to the use of variable width to represent changes in strength, ThemeRiver's transition from one time period to another takes place in the form of *smooth and continuous curves*, which gives it an aesthetic aspect. This novel approach attracted a large audience at that time. *Currents* in the ThemeRiver adjust themselves according to the occurrence of their respective themes. If a theme does not exist for a time-duration and then appears again, the corresponding current vanishes for that period and then appears again.

A user study conducted by the authors to compare ThemeRiver and Histogram received an overwhelming response in support of this design as it was highly understandable for them. Users were able to identify the *macro trends* over time very easily. However, they also suggested some improvements, for example, the visibility of numeric values associated with the currents, like that in a histogram. They also suggested the addition of features like the ability to see the *number of documents* during a time-interval and text of the documents, related to a current. So, the authors came up with the incorporation of such features in their visualization to *explore the 1990 Associated Press (AP) newswire data from the TREC5 distribution disks, a set of over 100,000 documents*. This research shows the importance of continuous curves for the visualization, which enhances the understandability of the users as well as the aesthetic appeal of the visualization. Addition of interactive features to such visualizations can be a major contribution to the field of visual analytics.

In 2008, Byron and Wattenberg [BW08] made a significant contribution towards the analysis of *layered graphs*. They described the algorithms behind the different graphs and gave a *mathematical analysis* to show the relations between them. They had mentioned how the ThemeRiver proved to be the initial contributor to strengthen the ability of traditional stacked graphs by *creating a smooth interpolation from discrete data*, and by incorporating the *layers*, which were *stacked* symmetrically on both sides of the x-axis, in contrast to the histograms where the stacking begins above the x-axis [BW08]. The four most important factors taken into consideration for stacked graphs were *the shape of the overall silhouette*, which is responsible for the *overall slope* and the *curvature* of different *layers*, *ordering of the layers*, *in accordance with the aesthetic aspects, labels, and the choice of colors*, to perceive the different layers.

As described in the paper, *the shape of the baseline and the order of the layers* affect the geometry of the stacked graph. If a set of n *real-valued non-negative functions*, $f_1 \dots f_n$ represent the time series, the crest of the layer associated with the i th time series, can be calculated by the function [BW08]:

$$g_i = g_0 + \sum_{j=1}^i f_j \quad (\text{Equation 2.1})$$

Figure 2-3 is an illustration of the above formula for $n = 2$.

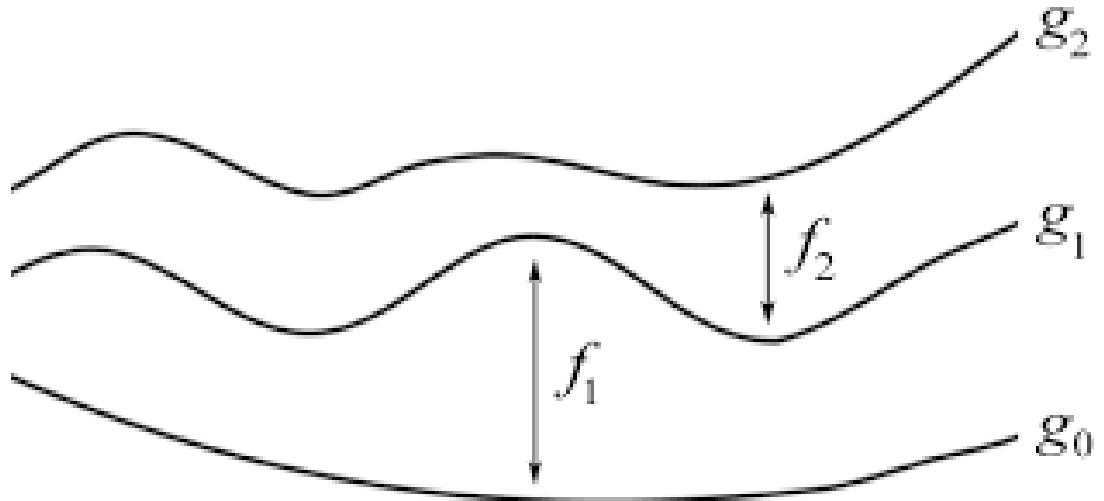


Figure 2-3: Stacked graph functions [BW08]

These graphs illustrate (Equation 2.1 for calculation of the top of the layer, i.e., g_i for $n = 2$

The choice of g_0 determines the type of layered graph. In the conventional stacked graph, $g_0 = 0$. An example of such type of graph can be seen in Figure 2-4.

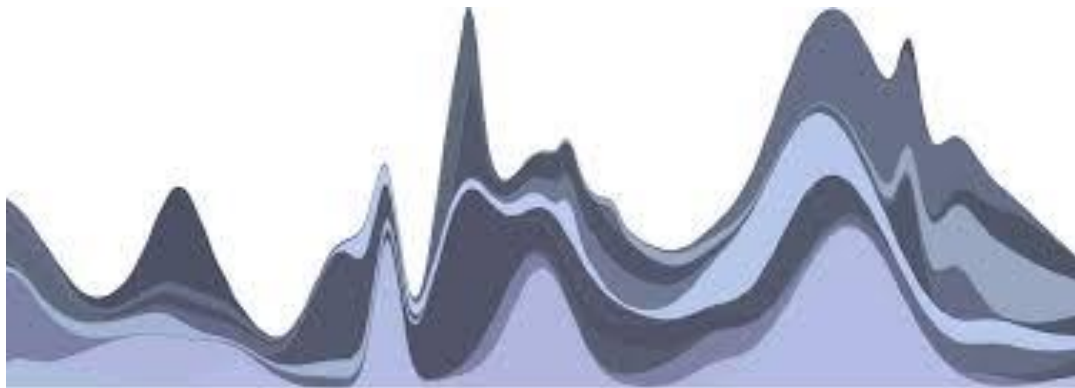


Figure 2-4: Conventional stacked graph with $g_0 = 0$ [BW08].
 g_0 refers to the baseline from which the graph is started.

In the ThemeRiver, g_0 is chosen in such a way that [BW08]:

$$g_0 + g_n = 0 \quad \text{Equation 2.2}$$

After mathematical calculations, it is found that the *silhouette is very near to the x-axis* at every point and the *slopes of the top and bottom of the silhouette* are very low. This results in the graph being very less “spiky.” Another advantage of this choice of baseline is that it *minimizes wiggle* in the graph. These lead to an improvement of the *legibility* of the graph. Figure 2-5 is a typical representation of a ThemeRiver for the same dataset that was used for the traditional stacked graph.

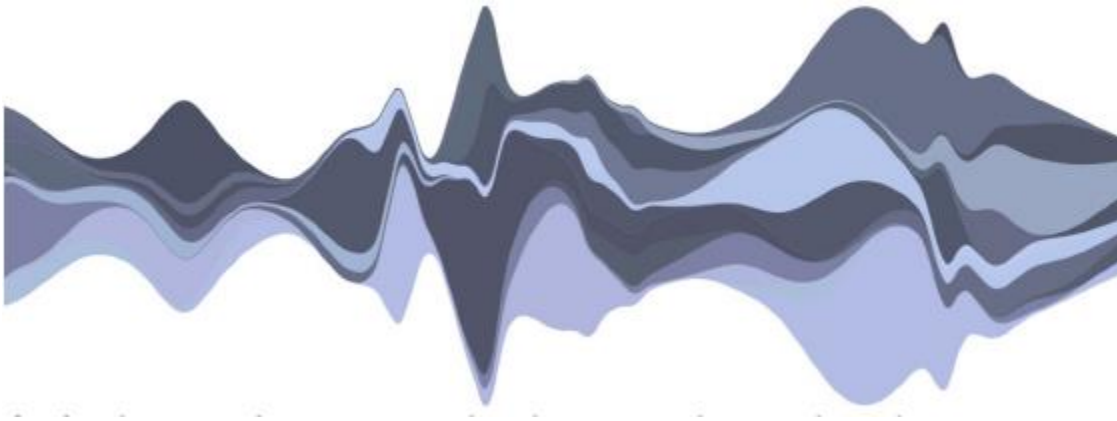


Figure 2-5: ThemeRiver Example [BW08]

Furthermore, in Streamgraphs, the optimization of the weighted-wiggle (*mean of squares of slopes between the midpoints of each layer, weighted by thickness of the layer*) gives a value of baseline. This enhances the aesthetics of the graph to a great extent, thereby, improving the understandability. Figure 2-6 uses this algorithm to display the Streamgraph for the same dataset that has been used in the earlier graphs.

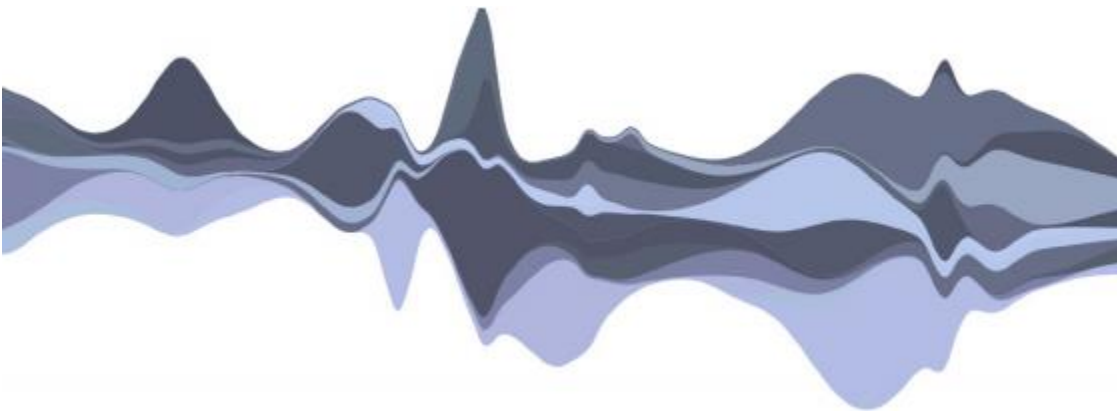


Figure 2-6: Streamgraph example [BW08]

After the observations and analysis of the paper by Byron and Wattenberg [BW08], it was decided to use Streamgraphs for the visualization of the results of the machine learning algorithm due to its high level of understandability for the user its and aesthetic aspect.

2.1.5 Topic Modelling

Topic modeling is the process of disintegrating an *unstructured* collection of documents into clusters representing similar topics [Ble12]. This is very helpful in sorting the *corpus* (even extremely huge in terms of size) in line with the identified *themes*. Therefore, topic modelling has helped in the organization and statistical analysis of documents, which can never be done manually by humans.

A topic is defined as:

“Distribution over a fixed vocabulary”

A common method of topic modelling which has been used in the thesis is Latent Dirichlet allocation (LDA). Blei, Ng and Jordan [BNJ03] described LDA as:

“A generative probabilistic model for collections of discrete data such as text corpora. LDA is a three-level hierarchical Bayesian model, in which each item of a collection is modeled as a finite mixture over an underlying set of topics.”

The assumption of LDA is that the topics exist before the generation of documents and for each document, the words are generated by a step-by-step sequence in which a distribution over topics is selected at random in the first step and then, for every word of the document, a topic from the 1st step is chosen randomly and then, a word from the associated distribution is selected randomly [Ble12].

To facilitate the analysis of the visualizations presented in the thesis, the user can view the news reports from which the labels have been generated. But, the news reports consist of long pieces of textual data, and reading the entire text would be a time-taking and daunting task for the human. So, topic modelling using LDA has been done to cluster similar words together, and the results are visualized in the form of circle-packed graph, so that the user can quickly understand what the document is about and then investigate further.

2.1.6 Circle Packed Graph

In this thesis, the result of topic modelling is visualized in the form of circle-packed graph, which are similar to Treemaps in terms of structure, and use circles contained inside circles for the hierarchical representation. The attributes of the circles, like area and colors can be adjusted to represent the weight assigned to each circle [Rib12]. An analogy between Treemaps and circle packed graphs is shown in Figure 2-7. This type of graph was chosen to display the results of topic modelling because of its aesthetic aspect and its ability to portray related words beautifully.

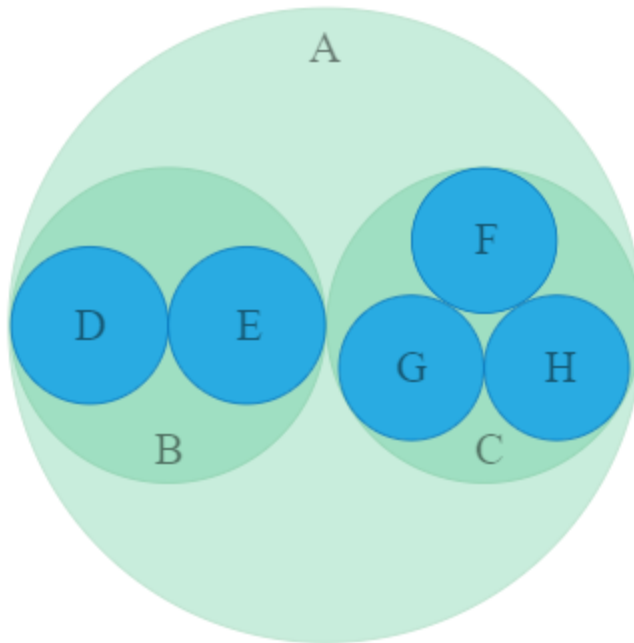
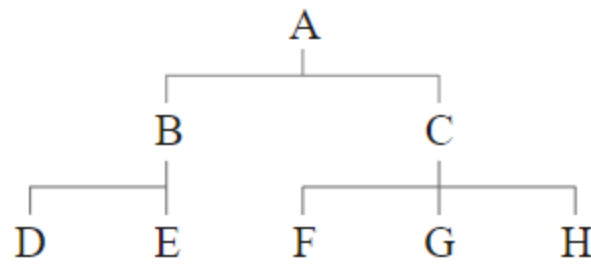


Figure 2-7: Analogy between Treemaps and circle-packed graphs.[Rib12]

As shown in the figure, A contains B and C. B contains D and E, and C is composed of F, G, and H. Both these graphs are similar in terms of showing the hierarchy and they may be used interchangeably. But, the circle packed graph seems to be aesthetically pleasing to the user.

2.2 Related works

In this section major contributions by past researchers in the area of machine learning and visual analytics are discussed. Then, a comparison is made on how this resembles the existing work and how it tries to overcome the shortcomings of the previous approaches. Further, concept drift is introduced to show the significance of real-time analysis of machine learning algorithm.

2.2.1 Machine Learning and Visual Analytics

Researchers are proposing methodologies to integrate machine learning with visual analytics. They have developed tools and platforms to improve the performance of classifiers by exploiting Visual Analytics for the same.

Seifert et al. [SSG10] integrated the approach of automatic analysis and information visualization to present an *Information Landscape* to the users enabling them to understand the topical structure of the data. As data keeps updating in real life scenario, the intention was to dynamically improve the training set fed to the classifier, and then use visualization to evaluate and refine the machine learning model. Choo et al. [CLK10] presented the *ivisClassifier*, a system that employed visual analytics for classification. The user can first use the visual interface to get an insight of the data, which is mapped into the reduced dimensional space and then label the new data. Settles [Set11] presented an interface for semi-supervised active learning to improve the accuracy of the training system. In this system, the learning algorithm can present new datasets to the human annotator for labelling and use these labelled examples for improving the performance of the classifier.

Marcus et al. [MBB+11] had come up with a platform, called *TwitInfo* for exploration of tweets in real-time. TwitInfo utilized Information Visualization which enabled users to gather insights about trending events and sudden ones with the interface provided to them. Also, it had the capability of sentiment analysis, and incorporated a new algorithm for peak-detection and labelling.

Heimerl et al. [HKB+12] presented three prototypes to improve text document retrieval by training binary classifiers interactively. Two of these approaches incorporated interactive visualizations, which rendered the documents, when required as well as the state of the classifier so that the analyst could gain insights about the data. Users could then utilize their knowledge, along with the knowledge gained from the visualizations to create training sets and their own labels, thereby incorporating the visual analytics loop in the process of classifier model development. The approach used in this thesis utilizes a similar idea for training the classifier, i.e., a classifier is built using a part of the dataset, results are visualized and explorative analysis is done on the data to examine and justify the classifier performance.

This work is also inspired by Bosch et al. [BTH+13] who had developed a system called *Scatterblogs2* for real-time monitoring and analysis of microblog messages using visual analytics. In the initial step, analysts can train the classifier model by visual interactions using the past data available to them, and then, the tools of the first step can act a support for monitoring, analysis and identification of relevant messages. Thus, analysts can use this approach to create task-tailored message filters. Users can also gain an overview of the dataset using Contentlens and LDA Topicview. The Prolix system (to predict box office revenues and ratings of movies) [KBT+13] was another contribution to improve classifier performance and understand the behavior of machine learning model using visual analytics. Analysts could refine machine

learning algorithms by gaining insights from visual explorations of texts obtained from social media content.

Liu et al. [LWL+17] had published an analysis on a visual analytics approach for a better understanding of the machine learning models. Despite the application of machine learning in myriads of areas, machine learning models are treated as a black box by the users. Understanding the ‘how’ and ‘why’ of a machine learning model is a task of significant importance to improve their performance otherwise, development of an efficient model can be a daunting job of continuous hit-and-trial. *Interactive model analysis* can be a great support to solve this challenge. The ultimate objective is to achieve better machine learning models (*reliable and accurate*) with the aid of *interactive visualizations*. A typical pipeline of machine learning is shown in Figure 2-8. In the initial step, relevant features are extracted from the input data. Then, the model is *trained, tested and refined*, on the basis of the outcomes of evaluation and skills of data scientists. This turns out to be a problem which needs a lot of time and does not guarantee the reliability of the model as well.

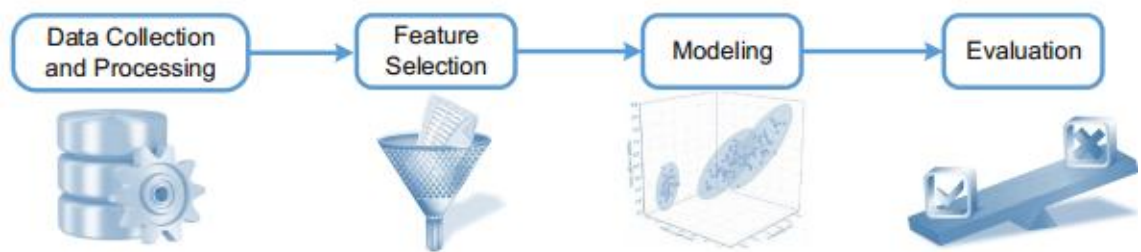


Figure 2-8: Pipeline of Machine Learning [LWL+17]

The initial step involves the collection of data and preprocessing the same, then, relevant features are selected from the input. Thereafter, the model is trained using a machine learning algorithm. The model is evaluated using performance metrics.

Visual analytics tends to improve the machine model iteratively by offering an explanation of the model with the help of *interactive visualizations*. Liu et al. explain that:

“In interactive model analysis, machine learning models are seamlessly integrated with state-of-the-art interactive visualization techniques capable of translating models into understandable and useful explanations for an expert.”

Figure 2-9 explains this approach, which focuses on *understanding, diagnosis and refinement* of the model. Understanding the behavior of machine learning models and figuring out the reason for their deviations with each other is an important step for model development. Diagnosis is identifying the problem due to which the machine learning model fails to achieve the desired performance, so that data scientists can exercise better options while training, for example, choosing a different classifier. Refinement involves suggesting experts to achieve advancement in the performance of the model. Users can utilize their knowledge by considering factors like features, type of classifier, and the training dataset, which can have a significant impact on the model. Paiva et al. [PSP+15] have mentioned about rebuilding the model in case of sudden changes in the distribution of classes. This methodology can prove to be of

great usage to facilitate convergence of the training process in adapting to the evolution of the dataset for achieving robust classification.

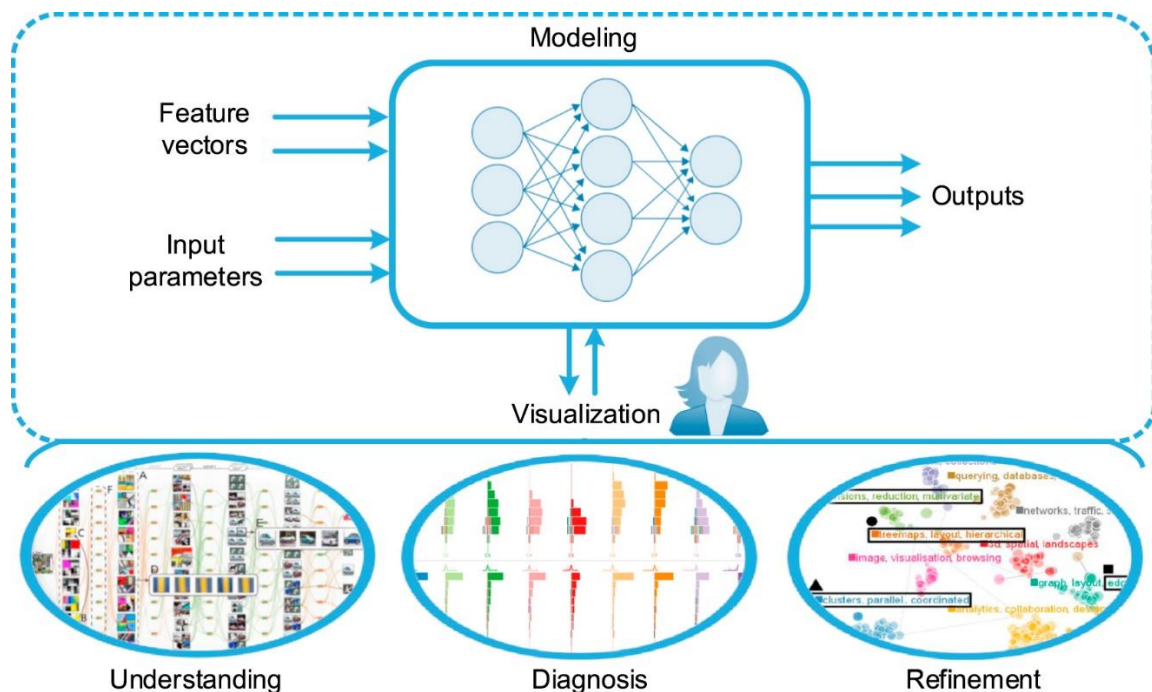


Figure 2-9: Interactive Model Analysis [LWL+17]

The figure shows the importance of visualization in understanding, diagnosis and refinement of the machine learning model.

As a contribution to this area, Charles D. Stolper, Adam Perer, and David Gotz [SPG14] have explained the idea of progressive visual analytics:

“This method is based on the idea that analytic algorithms can be designed to produce semantically meaningful partial results during execution. These progressive results can then be integrated into interactive visualizations that allow users to immediately explore partial results, examine new results as soon as they are computed, and perform new rounds of exploratory analytics without waiting for previous analyses to complete.”

Hence, a significant portion of the time that was earlier spent waiting for the final result can now be devoted to the analysis of partial results. This gives the analyst more control over the choice of classifier, feature and the datasets. It even offers the flexibility to rebuild the model in case the early analyses show that significant valuable results haven't been achieved. Thus, accuracy and robustness of machine learning models can be significantly improved by the tight coupling of visual analytics to the model.

2.2.2 Concept drift

“Concept drift primarily refers to an online supervised learning scenario when the relation between the input data and the target variable changes over time.” [GZB+14]

Zliobaite et al. [ZPG16] have aptly discussed the overview of *concept drift applications*. They have mentioned the importance of the analysis of data in real-time manner, before *patterns and relations* in the data becomes obsolete. This is a critical issue in building supervised machine learning model, which assumes the existence of a static source of data in contrast to the reality where data may arrive from different source distributions, leading to changes in pattern as illustrated in Figure 2-10 [ZPG16]. In their paper, they have identified the areas where concept drift is a critical issue and discussed the significance of designing machine learning models which would be capable of diagnosing themselves, and adjusting to the changing environment.

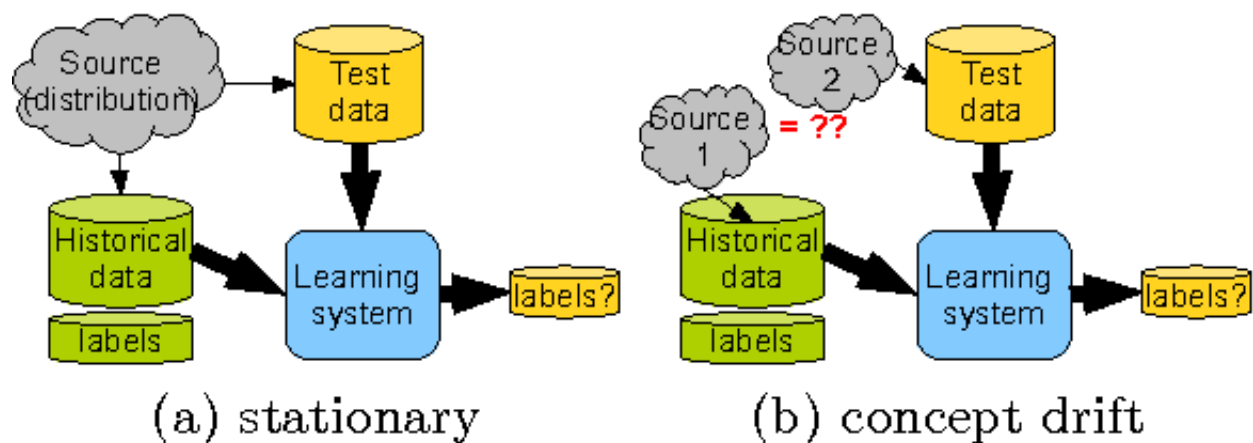


Figure 2-10: An illustration of supervised learning with static source of data (a) and taking concept drift under consideration (b). It can be seen that in reality, test dataset may be very different from the training dataset due to changes in the sources of data in real-time. [ZPG16]

Zliobaite et al. [ZPG16] have broadly categorized the tasks where concept drift can occur into three dimensions: “a type of the learning task, environment from which data comes, and online operational settings.” It is important to consider the type of data coming in, i.e, it should be noted whether the data is *sequential*, *relational*, or *complex* and how it is arriving (in clusters or continuous streams). Also, different tasks need different treatments in line with their nature. For example, a classification task needs to be dealt differently than a clustering task. During the design phase of an *adaptive learning system*, the reason for change of dataset should be observed minutely. This which will help in the prediction of future trends. For example, in this thesis, the cause of sudden deviations in the news dataset have been noted to be abnormal incidents. In some cases, drift in the economic situation of a country may also be a source of changes in news reports. Systems will have to be designed differently on the basis of the availability of

ground truth. For instance, the availability of actual labels in this work have enabled the comparison of predictions from the reality.

Gama et al. [GMC+04] had proposed the Drift Detection Method (DDM) which monitors the number of errors produced by the model where the number of errors is represented as a binomial random variable. The machine learning model trained here is based on the assumption that the error rate will decrease with the increase in the size of the training dataset, if the distribution of samples remains constant. Thus, an increase in the error rate during the training process indicates that the distribution of samples in the training set is changing and once the probability of misclassification reaches a certain level, concept drift is assumed to be true, training process of a new model is started. This method of drift detection is useful when changes are abrupt, but, has problems while detecting changes occurring at slow rates. Baena-García et al. [BCF+06] suggested the Early Drift Detection Method for detection of extremely gradual changes. This method considers the distance between the classification errors, and detects a concept drift when this parameter reaches a certain level.

Tsymbol [TPC+06] had suggested a *dynamic integration approach for ensembles* to handle concept drift by trying to overcome the shortcomings of the existing methods of that time pointed out by Brodley and Lane [BL96]. Tsymbol [TPC+06] states that:

“In ensemble learning, a set of models built over different time periods is maintained and the best model is selected or the predictions of models are combined.”

Brodley[BL96] had described the importance of utilization of *diversity of an ensemble* in the process of integration of different classifier models for the improvement of accuracy of predictions. Tsymbol [TPC+06] built the classifier models after short time-periods, selected the classifier with the best performance for that time-window, integrated this classifier with the base classifier, and then repeated the procedure before coming up with the final model of the classifier. This approach was thus capable of handling the *local concept drift* because importance was given to the *“local accuracy in the neighbourhood of the current test instance, instead of the global classification accuracy.”* [TPC+06]

Thus, it is observed that these methods of retraining the classifier again and again requires utilization of resources and time, and, hence, turns out to be an expensive approach. So, finding out the right point of time for retraining of classifier is of crucial significance. This work utilizes the concepts learnt from these research works to solve the problem of detecting concept drift and integrates the idea of Visual Analytics for understanding the machine learning model in real-time. An attempt is made to get an idea of the classifier performance for real-time data, and analyze the same with the help of interactive visualizations provided to the user

3. Approach

The goal of this thesis is to monitor and analyze the results of machine learning algorithm with the aid of visual analytics. The objective of this work is to support humans in verifying the correctness of machine learning algorithm with the help of visualizations and a graphical user interface provided to them, through which they can analyze the textual data on which the algorithm is applied for prediction. This chapter will first focus on the requirements and then present the approach to achieve the objective.

3.1 Specific requirements

This section provides an overview of the input fields and output obtained from the system. The requirements of the software include:

- **Visualization of actual labels:** The user should be able to get the visualization of the number of instances of a particular label and the count of labels at a particular instance of time in the form of a real-time graph. This requirement is to enable the user to be aware of the ground truth so that he/she can compare the actual values to the output from the classifier model.
- **Visualization of predicted labels:** The user should be able to get the visualization of the number of instances of a particular predicted label and the count of predicted labels at a specific instance of time in the form of a real-time graph. The presence of counts of predicted labels will enable the user to find deviations in the behavior of the model.
- **Control of refresh-rate of application:** The user should be able to change the refresh time of the application, i.e., the time after which the front-end queries the backend for data. If the user wants to observe the continuous graphs carefully, he/she can decrease the refresh-rate by increasing time after which the query is made from the backend. The user can increase the refresh rate if he/she just wants to observe the trends quickly.
- **Time-span of the graph:** The user should be able to select the time-duration of visibility of the Streamgraphs. The user should be flexible to decide about the number of hours whose data he/she wants to see in the front-end at a particular instance of time. With this functionality, the user can see the trend over a long period of time in one go, if he/she wants to. So, the decision of the number of hours to observe the trend should be left to the user.
- **Zoom-in on demand:** The user should be able to select a particular interval's Streamgraph, to display it statically, so that he/she can inspect the data for the chosen specified duration. This is one of the basic ideas of Visual Analytics that lets users zoom-in the important parts of the visualization.

- **Display of labels:** The user should be able to see the label corresponding to each layer of the graph. This should be applicable for the actual as well as predicted labels. The visualization would not be meaningful if the user is unable to see what each layer corresponds to. So, it is essential that the user gets the information about the layers in the Streamgraphs.
- **Display of data:** User should be able to see the textual data corresponding to each layer, i.e., the user should have the option to view the data from which the label has been generated. This would let the user analyze the reason behind the generation of a particular label. As this functionality should be present in the predicted labels as well, the user can use his/her knowledge to understand the behavior of the classifier model and decide if something is extremely wrong or not.
- **Visualization of words:** User should be able to perform the topic modelling of the textual data. Also, the user should be able to visualize the result of topic modelling in the form of a circle-packed graph, so that he/she can investigate the deviations in the results obtained from the classifier. The presence of most important words in the graph will help the user get a quick idea of the underlying data before investigating the complete text.

3.2 Design of approach

To achieve the objective of the thesis, a software has been developed to meet the requirements which have been discussed in the previous section of this work. The software comprises of the database, backend logic, the classifier, the visualization controller, and the frontend. The data is stored in the database and the backend logic is responsible for interaction with the database, preprocessing of textual data, and converting the data into a form that can be sent to the frontend for visualization. The visualization controller acts as a bridge between the backend and the frontend as the rest endpoints are implemented here. The frontend has the logic for visualization of data received by it.

3.2.1 Database

The basic idea of the thesis is to have the real-time data in the form of continuous streams to be sent to the front-end for visualization and analysis. To accomplish this, a real-time environment has been simulated from the static dataset which has been provided from the Institute for Visualization and Interactive Systems (VIS) of the University of Stuttgart².

The data was made available in the form of a mongodump folder, which is a combination of collection.bson file and collection.metadata.json file. This data is imported to MongoDB using Studio3T³. The MongoDB collection has 100,000 documents, obtained from the German news agency Deutsche Presse-Agentur (DPA)⁴. Each document is made up of 40 fields. A typical example of a document is shown in Table 3.1.

² <https://www.vis.uni-stuttgart.de/>

³ <https://studio3t.com/>

⁴ <https://www.dpa.com/de/>

Field	Content
_id	593a7cd46a853f6024f8ebde
date	2017-03-08T17:11:52+01:00
ressort	Vm
textForUi	Magdeburg (dpa/sa) - Für die zweieinhalb Monate alten Vierlinge bei\nden Weißen Löwen hat der Magdeburger Zoo jetzt Namen gefunden. Online...
genre	Meldung
credit	Dpa
city	Berlin
version	5
...	...

Table 3.1: Example of a document of the MongoDB data

For the thesis, “date,” “textForUi” and “ressort” are taken into consideration, where:

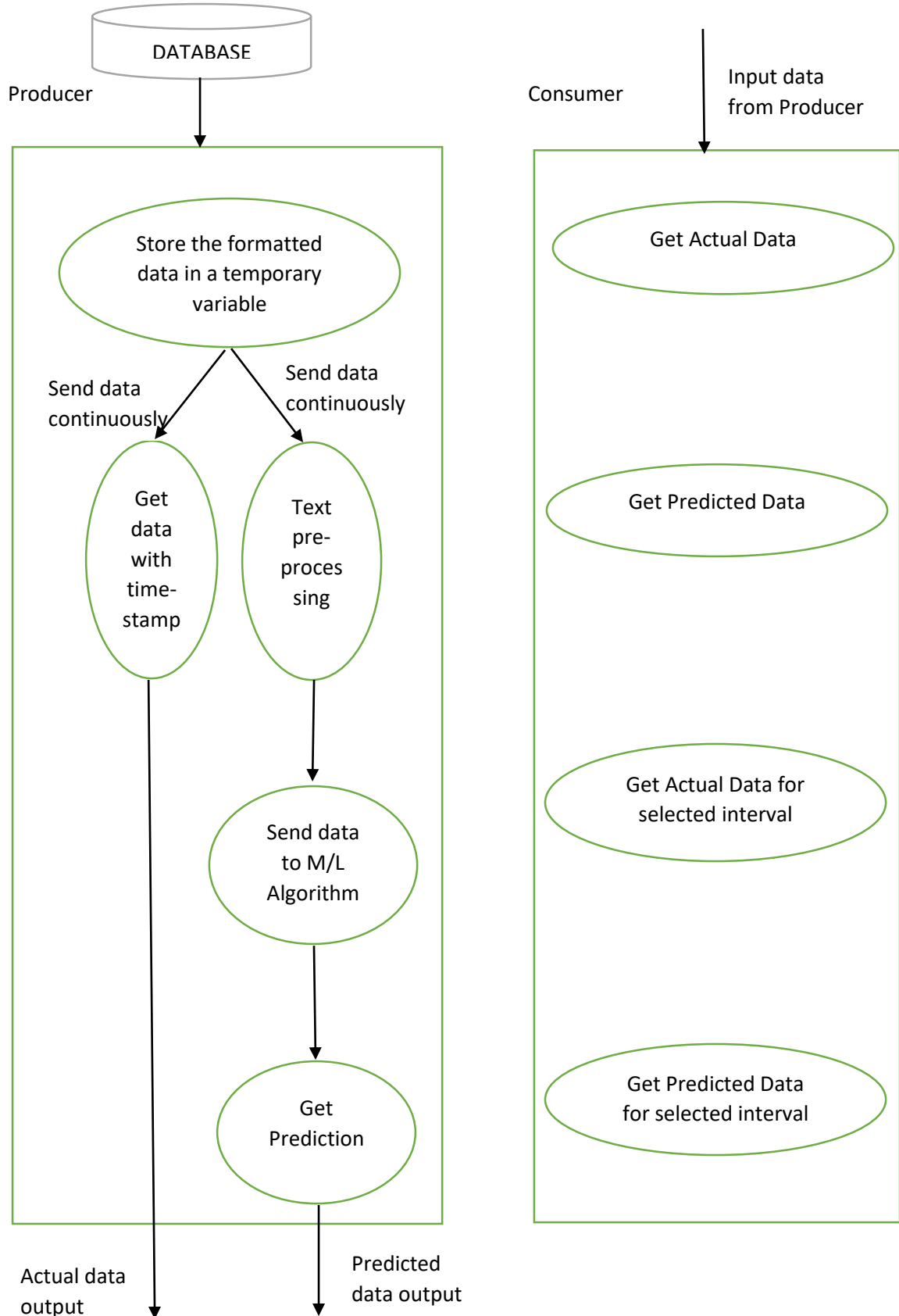
“date” is the timestamp associated with the text field, “textForUi” is the text field from which the “ressort” is to be predicted by the classifier, so that the comparison between the actual values of the “ressort” and predicted values of the “ressort” could be performed after visualization to monitor the classifier behavior, “ressort” is the category of the text field, i.e., the type of news the text is related to. For example, the “ressort” value of “wi” refers to “wirtschaft”, i.e., economy, “pl” refers to political news, etc.

3.2.2 Backend logic

The underlying architecture of the backend logic is given in Figure 3-1. It is composed of Producer, Consumer, and LDAHandler which are described in this section. The Producer does the task of gathering data from the database and sending it to classifier model to obtain the results. The Consumer is responsible for querying the data from the Producer on request from the frontend. The LDAHandler performs topic modelling by applying the Latent Dirichlet allocation algorithm to the selected parts of data which is to be analyzed.

Approach

Figure 3-1: Basic backend architecture of software



Producer: The Producer Thread is responsible for the tasks including the establishment of connection with the database, retrieval of data from the database, simulation of a real-time environment to show that the data is going out from the Producer continuously, cleaning of text before sending to the classifier, and sending the data to the classifier to obtain the predictions. On request from the Consumer, the Producer sends real-time response to the Consumer for the chosen time-interval.

After receiving the data (values of fields “textForUi”, “date” and “ressort” of all documents) from the database in a temporary data-structure, the Producer thread keeps on updating another data structure with the values in a real-time fashion. The data received from the database is stored in a data structure, TreeMap temporarily. Now, another TreeMap extracts the data from the previous TreeMap and stores it, which is sent to the Consumer, on receipt of a request from it. After this storage of values in the TreeMap, the entries from the temporary TreeMap are cleared. This is to show that the current TreeMap is updated with time, as is the case of a real-time scenario. As the dataset belonging to a period of one minute is not very dense, so the system has been speeded up in such a way that each minute corresponds to an hour. When the consumer requests for the data, the data is sent to the consumer which further, sends it to the frontend. Algorithm 1 represents the pseudocode for the Producer Thread.

Algorithm 1. PseudoCode of Producer Thread

```

1. Data: documentsFromMongoDB docs
2. BEGIN
3.     producerMapTemp    // TreeMap for temporary storage of all the data from the DB
4.     producerMapActual  // TreeMap which holds continuously updated data of DB
5.     producerMapPredicted // TreeMap which holds continuously updated data of classifier
6.     FOR Document d : docs
7.         add d.date to producerMapTemp.key
8.         add d.date, d.ressort and d.textForUi to producerMapTemp.value
9.     ENDFOR
10.    load Machine learning model in R
11.    appStartTime ← time.now()
12.    dataStartTime ← producerMapTemp.firstKey()
13.    WHILE (true) DO
14.        currentTime ← time.now()
15.        refreshTime ← (currentTime - appStartTime) * speedFactor + dataStartTime
16.        newEntries ← producerMapTemp.getEntriesWithKeyLessThanRefreshTime
17.        FOR Entry e : newEntries
18.            add e.date to producerMapActual.key
19.            add e.date, e.ressort and e.textForUi to producerMapActual.value
20.            preprocess e.textForUi
21.            send cleanedTextForUi to classifier in R
22.            predictedRessort ← output from classifier
23.            add e.date to producerMapPredicted.key
24.            add e.date, predictedRessort, e.textForUi to
                producerMapPredicted.value

```

Approach

```
25.          ENDFOR
26.          clear newEntries
27.          Thread.sleep(2 seconds) // sleep the thread to simulate real-time environment
28.    ENDWHILE
29. END
```

Text preprocessing before input to the classifier is essential to prepare the textual data before training the classifier model. Haddi, Liu, and Shi, 2013 [HLS13] had discussed the importance of preprocessing as an essential aspect before text classification. According to Haddi et al., preprocessing involves cleaning and preparation of the text for feeding to the classifier, to remove the noise and uninformative parts, which do not contribute to the *orientation* of the textual data. The presence of noise causes an increase in the dimension of the textual input, degrading the performance of the classifier. Providing the classifier with pre-processed data helps in the improvement of its accuracy and performance, including its speed. Vadehra, Grossman and Cormack, 2013 [VGC17] have used cleansing steps in their work before training the classifier, from which some have been used in this thesis, as mentioned below:

1. Remove all URLs from the texts
2. Convert the entire text to lowercase
3. Replace all symbols with space
4. Replace multiple spaces with single space
5. Replace tabs and newlines with single space
6. Replace ü with ue, ä with ae, ö with oe, ß with ss (as the data consists of German text)
7. Remove the stop words

An example of a raw text and its cleaned version is given below:

Raw text string: "Magdeburg (dpa/sa) - Bei\neiner Online-Abstimmung stimmte die Mehrheit der mehr als 48 000\nNutzer dafür, das einzige Weibchen im Quartett Nala zu taufen, wie\nder Tierpark am Mittwoch mitteilte. Zoogeburten gelten als Zuchterfolge.\n\n# Notizblock\n\n## Aktualisierung\nFotohinweis ergänzt\n\n## Internet\n- [Zoo zum Namensvoting](<http://dpaq.de/or5GZ>)"

Cleaned text string: "magdeburg online abstimmung stimmte mehrheit nutzer einzige weibchen quartett nala taufen tierpark mittwoch mitteilte zoogeburten gelten zuchterfolge notizblock aktualisierung fotohinweis ergaenzt internet zoo namensvoting"

As can be seen from the above example, the dimension of the text has been reduced to a great extent after cleaning. Now, the Producer Thread sends the cleaned "textForUi" to the classifier, and the classifier returns the predicted "ressort" to the Producer.

Consumer: The consumer has the task of querying the Producer for actual data obtained from the database and the predicted data (which is the output of the classifier) on request from the frontend. Thus, the Consumer has the implementation of the following functions:

1. `getTextAndActualRessort(duration)`
This method queries the producer for the “textForUi”, the actual “ressort” which has been obtained from the DB, the “date” and constructs a “JSONArray” out of it, which is ready to be sent to the frontend. The parameter “duration” comes from the frontend and refers to the time duration for which the query is to be made. This query is made continuously from the frontend for the visualization of the continuous real-time graph.
Please refer to Algorithm 2.
2. `getTextAndPredictedRessort (duration)`
This method queries the producer for the “textForUi”, the predicted “ressort” which has been obtained from the classifier, the “date” and constructs a “JSONArray” out of it, which is ready to be sent to the frontend. The parameter “duration” comes from the frontend and refers to the time duration for which the query is to be made. This query is made continuously from the frontend for the visualization of the continuous real-time graph. The pseudocode of this method is similar to that of the previous method, only with the variation that the predicted “ressort” is obtained instead of the actual one from DB.
3. `getTextAndActualRessortFromStartTime(duration, startTime)`
This method queries the producer for the “textForUi”, the actual “ressort”, the “date” and constructs a “JSONArray” out of it. The query is made on the basis of a selected time slot for which the real-time graph is to be analysed.
Please refer to Algorithm 3.
4. `getTextAndPredictedRessortFromStartTime(duration, startTime)`
This method queries the producer for the “textForUi”, the predicted “ressort” the “date” and constructs a “JSONArray” out of it. The query is made on the basis of a selected time slot for which the real-time graph is to be analysed. The pseudocode of this method is similar to that of the previous method, only with the variation that the predicted “resort” is obtained instead of the actual one from DB.

Algorithm 2. Pseudocode of `getTextAndActualRessort (duration)`

1. Data: JSONArray `ressortAndTextDB`
2. BEGIN
3. SortedMapRecentData `recentData` ← `getDataFromProducer(duration)`
4. FOR MapEntry `e` : `recentData` DO
5. add `e.value` to `ressortAndTextDB`
6. ENDFOR
7. END

Algorithm 3. Pseudocode of `getTextAndActualRessortFromStartTime(duration, startTime)`

```
1. Data: JSONArray ressourtAndTextForSelectedInterval
2. BEGIN
3.     SortedMapRecentData recentData ← getDataFromProducer(selected interval)
4.     FOR MapEntry e : recentData DO
5.         add e.value to ressourtAndTextForSelectedInterval
6.     ENDFOR
7. END
```

LDAHandler: The task of LDAHandler is to use the latent dirichlet allocation (LDA) technique to discover topics to which the text corresponds and to identify words which are related to each other. This is done only when a request is sent from the front-end to perform the explorative analysis on the data associated with a particular prediction. LDA Handler receives the “textForUi” for a specific duration of time and then infers the topic distribution and the most significant words related to the topic. This data is then sent to the front-end for visualization in the form of word-clouds. In this work, LDA has been done using Mallet [CK02].

3.2.3 Classifier

For this thesis, Support Vector Machine (SVM) has been used as the algorithm for the multi-class text classification. An initial attempt was made to train the Naïve Bayes classifier, but it was found that the performance in terms of accuracy was not good and using the SVM classifier model to make predictions fetched much more accurate results. In this work, SVM training and classification is done with the help of the `e1071` package of R [CRA17] which uses the one-against-one technique for multi-class classification, because of its speed of performance.

As the SVM Classifier is to be trained on textual data, feature selection is an important step before training. This thesis took ideas from the paper by Haddi et al., [HLS13], where they came up with experimental reports on the accuracy improvement with appropriate feature selection. “Words” are taken into consideration for features in problems of text mining. A specific weight is attached to the text (which is treated as the feature) to determine its importance in the document. There are myriad ways of assigning weights to the words. Most commonly used ones are *Feature Frequency (FF)*, *Term Frequency Inverse Document Frequency (TF-IDF)*, and *Feature Presence (FP)*.

Alsmearat, Al-Ayyoub, and Al-Shalabi, 2014 [AAA14] had discussed the Bag of Words model to address a text mining problem in the Arabic language. In this model, the text is represented as a set of terms (words) and the feature vectors are calculated by the frequency of occurrence of these terms in the document. It is a simple model, where the order of the words and the grammar are not taken into account, only the frequency count of words is preserved. In this thesis, the Bag of Words model has been used for the representation of text data as vectors for machine learning algorithm. The basis of this model is the Document Term Matrix (DTM). In Document Term Matrix, the words of the corpus form the columns of the matrix, the documents form the rows, and the corresponding frequency count of the terms fills up the matrix. For example, if we have two documents as mentioned below:

D1 = “Das Auto hat einen Unfall”

D2 = “Das Auto fährt”

The DTM of the above-mentioned documents can be represented as given in Table 3.2

	Das	Auto	hat	Einen	Umfall	fährt
D1	1	1	1	1	1	0
D2	1	1	0	0	0	1

Table 3.2 DTM Representation

After the words have been represented as an algebraic model, the dataset is ready to be fed to the classifier for training. Following the standard method of building the machine learning model, the dataset is divided into a training set and a test set. The training set is fed to the classifier, which uses it to build the machine learning model, by learning the weight of words from the DTM. Now, every element of the test set is used to evaluate the machine learning model, which has been built on the basis of the training set. Then, accuracy and error are measured on the basis of the performance of the classifier model on the test set.

With this method, an average accuracy of around 84% has been achieved for the model trained on a subset of 10000 records of the news data, which can be considered to be a reasonable value for the multi-class classification problem to be dealt in the thesis.

3.2.4 Visualization Controller

In this thesis, the RESTful web services have been developed using Spring Boot [SOF18]. So the REST endpoints have been created in the Visualization Controller, which acts as a bridge between the frontend and the backend logic. All the HTTP requests sent from the frontend are handled by the Visualization Controller.

3.2.5 Frontend (Visualization)

The approach used in the thesis is to present the user with an interactive interface so that he/she can discern any abnormalities in the performance of classifier model and then perform statistical analysis of the same. An idea of the same could be the comparison between actual labels and labels from classifier via Streamgraphs so that the user can easily spot the difference between the two graphs. Thus, the frontend logic is implemented to perform visualizations and meet the specific requirements mentioned in the beginning of this chapter. The next step could be performing a statistical analysis of the corresponding text data from which the labels have been predicted. This objective can be achieved through visualization after topic modelling of the data

4. Implementation

The approach discussed in the previous chapter is realized with the help of a software which implements the backend logic, training of the classifier model, visualization controller, and the frontend logic. This chapter starts with the algorithm which includes the information about how different programming languages have been used for the development of the software. It also includes the description of the functions which have been implemented. Illustration of the visualizations are presented with the help of screenshots that have been taken from a running version of the Software. The next section of this chapter gives an overview of the tools and libraries used in the software.

4.1 Algorithm

In order to monitor the results of the classifier in real-time and perform analysis of the same, a software has been developed using MongoDB database, Java for the backend, R for machine learning and HTML and JavaScript for visualization. This section describes how different programming languages have been used in the development of the software and the implementation of functions. An integration of the different programming languages is shown in Figure 4-1. The visualization part of the thesis to meet the requirements is also given in detail in this section.

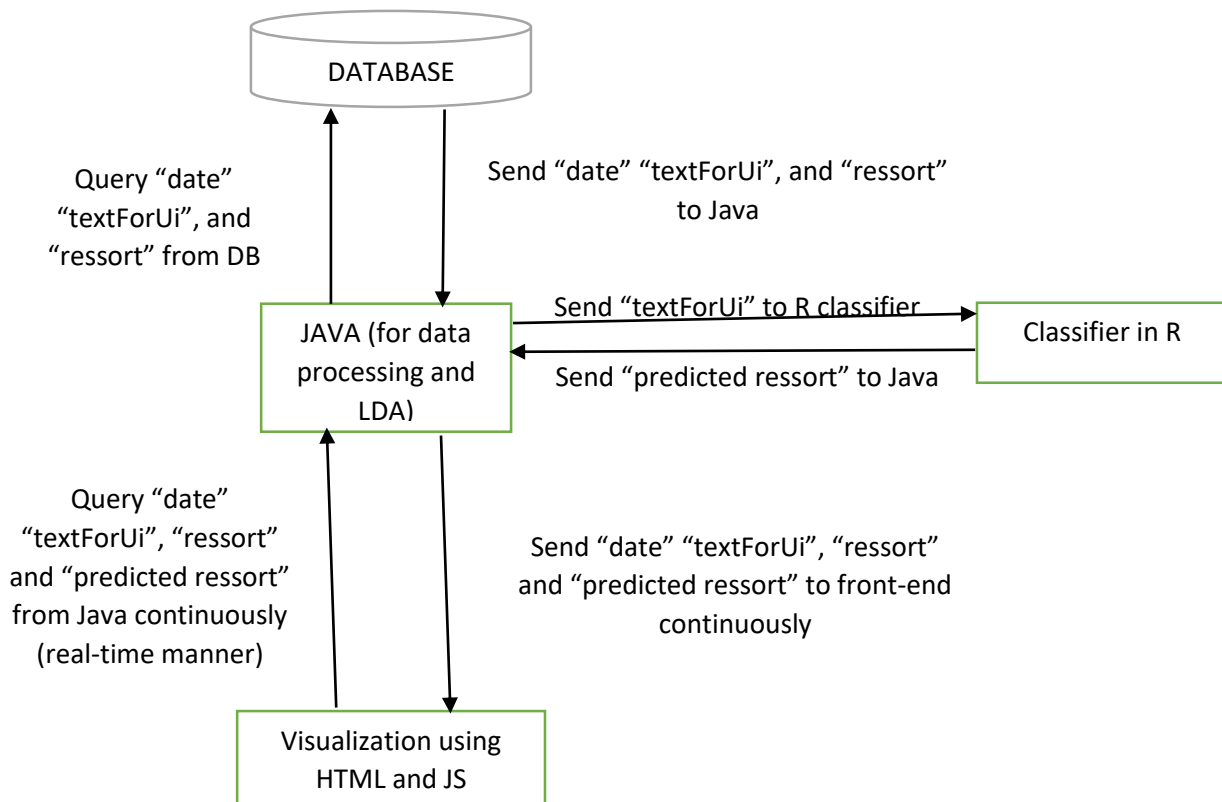


Figure 4-1: Integration of languages for the development of software for the thesis.

Java queries the MongoDB for the entire dataset and stores it. A mechanism is implemented in Java to simulate a real-time environment. The frontend queries Java continuously for the most recent data. Java responds to the frontend with the data, timestamp, actual label and the predicted label (which it receives from R). Now, the visualization is done using JS and HTML.

As shown in Figure 4-1, the backend logic interacts with the database. This part is implemented in JAVA, which has the Producer class which queries the database and simulates a real-time environment to ensure the availability of data to the frontend as continuous streams. The backend part consists of the Consumer class which continuously enquires the Producer class for data, when the frontend requests for the same. The classifier model has been trained using the programming language R. So, to use the classifier model to make predictions, a connection is established between JAVA and R. REST endpoints have been developed using Spring [SOF18] to enable a method of communication between backend and frontend. Finally, visualization is done using HTML and JavaScript. Details about the different classes and functions are given in this section.

The Producer Thread is implemented in Java and implements the `run()` method. The logic going on inside the Producer class has already been discussed in the previous Chapter. The Producer first establishes a connection with the MongoDB, extracts the entire useful dataset, i.e., “date”, “textForUi”, and “ressort” of 100,000 documents and stores this data as a *TreeMap* (data structure in Java), so that the keys are

Implementation

arranged in order of timestamps. After this, the connection to the database is closed as the entire data is present in the data structure.

As the backend logic is implemented in JAVA, an endeavor was made to use Weka 3.8.2 [Wai17] from JAVA code. But, building the classifier model using Weka was very difficult because of its large memory requirement and slow speed. So, the SVM classifier was trained using the programming language R. To call a function of R from Java, RServe [Urb03] package has been used in this thesis. A server instance of RServe is started, and the Java client communicates with this over TCP/IP.

Now, the Producer loads the machine learning model by calling a function of R by using RServe [Urb03]. The idea of a connection between R and Java has been taken from Codophile [Cod15]. The snippet to start an instance of RServe server in R is shown in Algorithm 4. To obtain the classifier output, the Producer sends the cleaned “textForUi” to the classifier implemented in R and then gets the prediction from the classifier.

Algorithm 4. Starting RServe in R [Cod15]

```
Library(Rserve)
Rserve()
```

Next, another TreeMap is updated with the data in a real-time manner. The key of this Map is the timestamp, and the value is a JSONObject of “date”, “textForUi”, and “ressort”. This TreeMap has the actual labels of “ressort” obtained from the database. An update of a similar TreeMap is also done, where the JSONObject has the “predicted ressort”, received from the SVM classifier.

Now comes the role of the Consumer class, which is also implemented in Java and is responsible for fetching the data from the Producer class on request from the front-end and converting it into a JSONArray, which is ready to be sent to the front-end. The conversion into a JSONArray enables an efficient way of data transfer from backend to frontend and the vice-versa. The main purpose of the Consumer class is to achieve a level of modularity in the code, so that the code may become reusable.

On receipt of a request from the front-end, the LDAHandler class performs the task of topic-modelling. To train the Latent Dirichlet Allocation (LDA) Model, the Java API of MALLET [CK02] has been used in this work. This implementation is an adaptation of the work in the blog-post of Welleck [Wel14]. The first step for topic modelling is to convert the SortedMap received by the LDAHandler class from the Consumer (on request from the frontend) into an InstanceList Object so that it can be used by the Mallet Model. Now, an LDA model is trained on this data. The method “getSampledDistribution()” of Mallet helps in the inference of topic distribution. With the LDA model in hand, it is now possible to have an insight on the topics of which the dataset is composed and the most frequent words of different topics.

As mentioned earlier, the Visualization Controller has the Rest endpoints, with URL Mappings which correspond to “/streamgraphActual”, “/streamgraphPredicted”, “/streamgraphActualFromStartTime”, “/streamgraphPredictedFromStartTime”, and “/performLDAAActual”. The URL mapping “/streamgraphActual” has the request parameter of the selected time duration for which the streamgraph of the actual data should be visible to the user. For example, if the user chooses to visualize the data of

the last 5 hours, then the data of the last 5 hours is sent to JavaScript for visualization. Thus, the data of this time period has to be sent continuously to the front-end. The return type of this URL is a JSONArray to JavaScript. "/streamgraphPredicted" is similar to "/streamgraphActual" with the only difference that it is responsible for handling predicted data. It has the request parameter of the selected time duration for which the streamgraph of the predicted data should be visible to the user. This returns a JSONArray to JavaScript for visualization. The URL "/streamgraphActualFromStartTime" has the request parameter of the "startTime" and the "duration" for which the Streamgraph of actual labels is to be visualized. With the help of this URL, a portion of the Streamgraph can be viewed statically. This URL returns a JSONArray to the front-end, so that it can be used for visualization. "/streamgraphPredictedFromStartTime" has the request parameter of the "startTime" and the "duration" for which the Streamgraph of predicted labels is to be visualized. "/performLDAActual" has "startTime" and "duration" as the request parameter as it performs the topic modelling of the "textForUi" for the duration for which the static Streamgraphs are displayed. The Json format returned by this URL is used to visualize the topic models in the front-end.

Finally, the user is presented with an HTML page comprising of input fields, text areas, buttons, and slider. Their functionalities are controlled by JavaScript. The visualizations have been implemented using the D3 library of JavaScript [BDH+11]. The user is presented a simple Graphical User Interface as shown in Figure 4-2.

Monitoring Results of Machine Learning Algorithm

Continuous Streamgraph	Static Streamgraph	Topic Modeling
Duration (hrs): <input type="text" value="5"/>	Duration (hrs): <input type="text" value="5"/>	Number of Topics: <input type="text" value="5"/>
Refresh time (sec.): <input type="text" value="1"/>	Start Date: <input type="text" value="2016-12-17T11:00"/>	Number of Words: <input type="text" value="10"/>
<input type="button" value="Get Actual Data"/>	<input type="button" value="Get Actual Data"/>	
<input type="button" value="Get Predicted Data"/>	<input type="button" value="Get Predicted Data"/>	

Figure 4-2: GUI for the User.

The user can enter the duration of which he/she needs the continuous Streamgraphs, select portions of continuous Streamgraphs by entering parameters in the input fields below the Static Streamgraph header, and enter number of topics and number of top words for each topic that would be displayed as circle-packed graph after topic modelling.

There is a possibility to obtain two Streamgraphs for comparative visual analysis, one for the continuous data coming from the database and the other for the continuous data which is output from the classifier. The user can then select parts of the continuous Streamgraphs and zoom in to obtain static Streamgraphs, to analyze the same. The data visualized in the form of Streamgraph is the number of "ressort" coming from the backend. Each colored ribbon of the Streamgraph corresponds to a single type of "ressort".

Implementation

In the GUI of Figure 4-2, the user can type the number of hours, for which the Streamgraph can be visualized in the “duration” field for Continuous Streamgraph. For example, when it is set to default as 5 hours, the user can see the Streamgraph of 5 hours which keeps on getting updated continuously. In the field of “refresh time”, the user can enter the time after which the Streamgraph is refreshed, or updated so that it appears in the form of a continuous graph. Two fields exist for taking parameters for Static Streamgraph. In the “duration” field, the number of hours is to be entered by the user for which he/she wants to analyze the Streamgraph. Moreover, this field can be updated automatically with the help of the slider in the continuous Streamgraph. In the “Start date” field, the user has the option to enter the starting time from which he/she wants to zoom in the Streamgraph. This field can also be updated automatically with the help of the slider in the continuous Streamgraph.

The frontend has the implementation of a number of features to facilitate human interaction. For example, once the button “Get Actual Data” is clicked, the event for the generation of Streamgraph is triggered, and the continuous streamgraph for the actual values of “ressort” is displayed. Similarly, on click of the button “Get Predicted Data”, the event for the generation of Streamgraph is triggered, and the continuous streamgraph for the predicted values of “ressort” is displayed. The coordinate axes of the Streamgraphs present another set of valuable information. The x-axis gives the respective time associated with the “ressort”. The y-axis gives the value of the count of a particular “ressort” at an instance of time and the total count of all the “ressort” at that moment. Each “ressort” is linked to a specific colour to maintain uniformity between the two graphs. This proves to be very useful when there is a zero-count of a particular “ressort” in one graph and a non-zero value in the other graph.

An example of visualization is shown in Figure 4-3. The Streamgraph at the top represents the actual values of “ressort”, whereas, the one at the bottom represents the predicted values of “ressort”. The two graphs are displayed together, to give a visual aid to the user for comparative purposes.

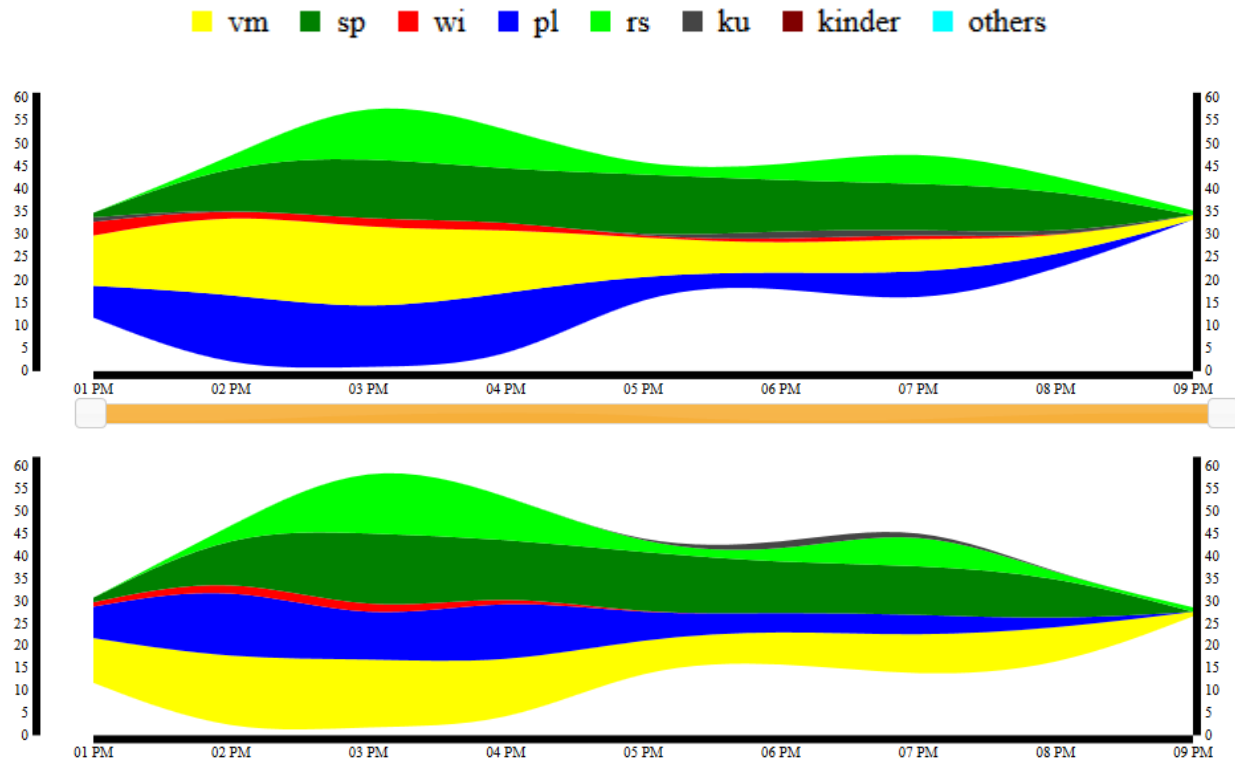


Figure 4-3: Streamgraphs for actual and predicted values of “ressort”.

The graph at the top represents the actual values and count of labels, whereas the one at the bottom illustrates the predicted values and count of labels.

There is a slider introduced in between the two Streamgraphs, so that the user can select portions of the Streamgraphs to zoom in the same to perform explorative and statistical analysis. For instance, when the user selects a portion of the Streamgraph using a slider as shown in Figure 4-4, then the “duration” and “start date” fields get updated automatically as shown in Figure 4-5. When the user clicks on the buttons of static Streamgraph, he/she is able to obtain the Streamgraphs for the selected time-span (Figure 4-6).

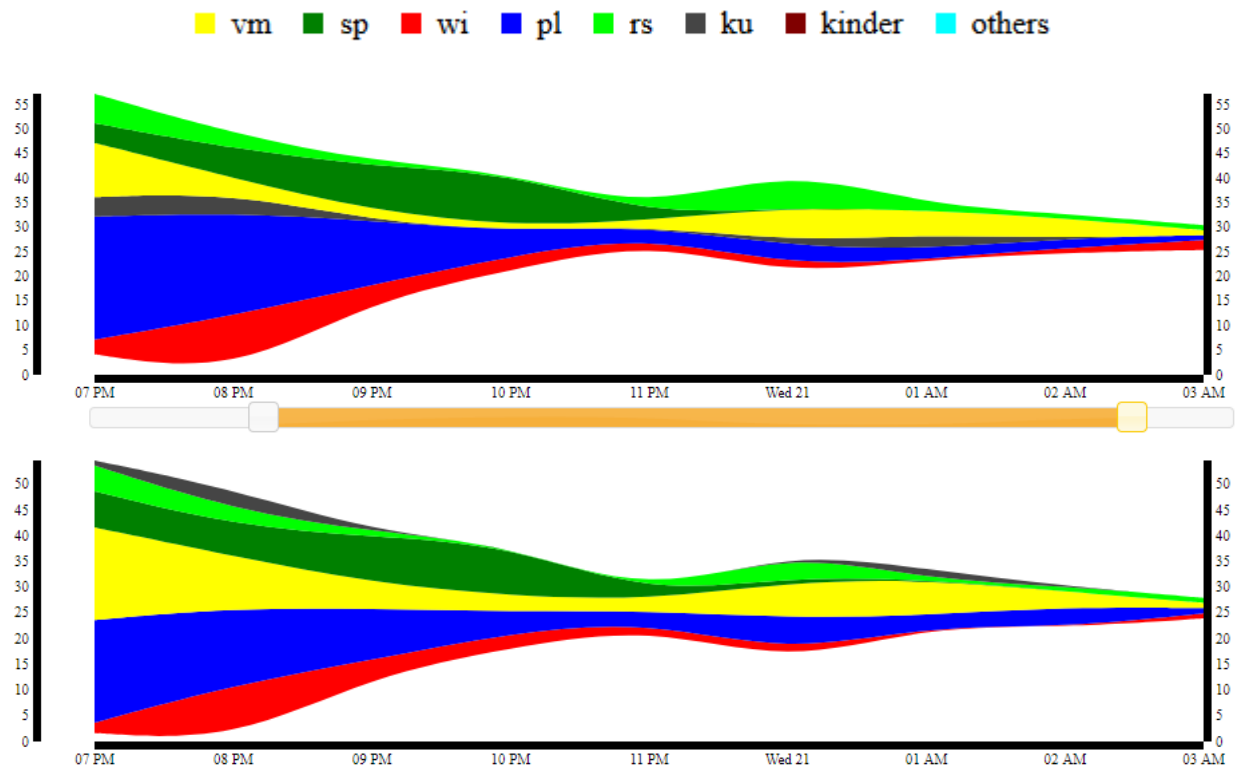


Figure 4-4: Selection of a portion of the Streamgraphs using the slider. This figure illustrates the use of a slider to select portions of the continuous Streamgraphs to obtain a zoomed in view, so that the user can investigate interesting parts of the graph.

Static Streamgraph

Duration (hrs):

Start Date:

Figure 4-5: Update of fields below Static Streamgraph header after selection of a portion of the Streamgraphs of Figure 4-4. The “start date” indicates the time of start of the slider and “duration” refers to the period till which the selection is made by the slider.

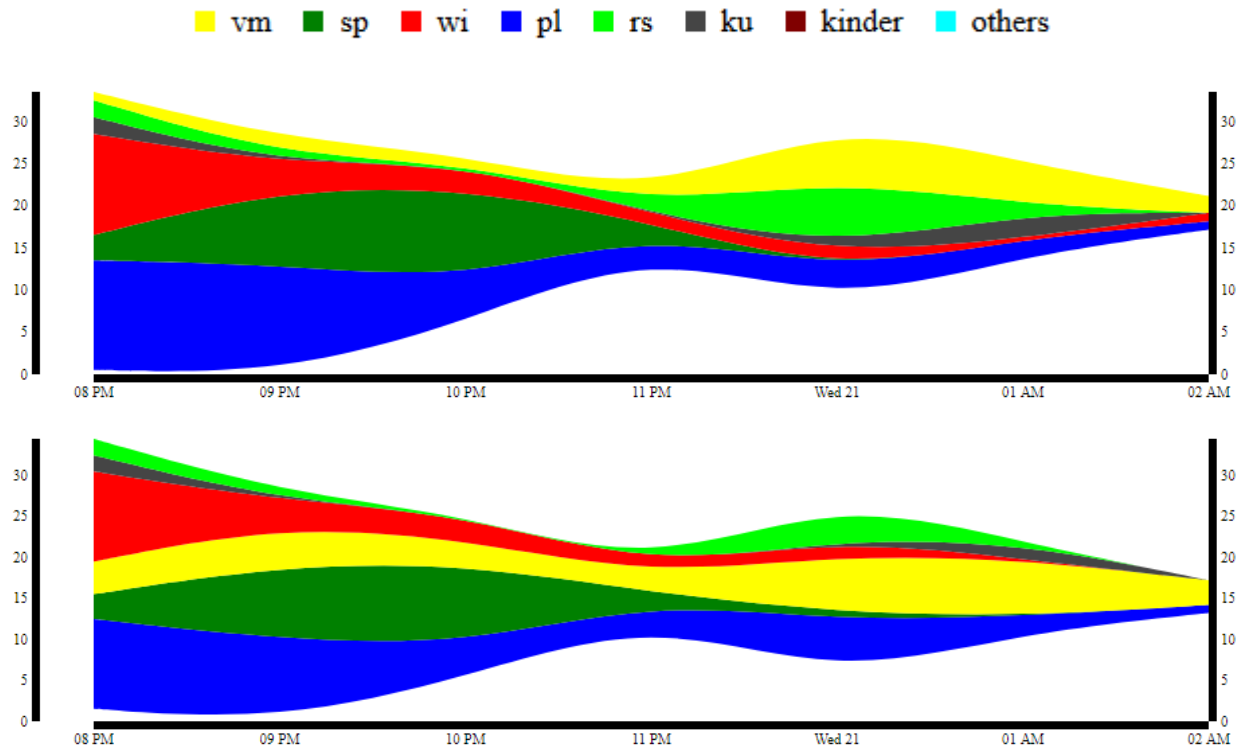


Figure 4-6: Zoom-in view for the portion selected by the slider in the continuous Streamgraph of Figure 4-4. The upper graph represents the actual values of “ressort” coming from the database and the lower one represents the “ressort” predicted by the classifier

There are textareas in the GUI which get updated on clicks in the Streamgraph. For instance, when a user clicks on a layer of a Streamgraph at a particular point of time, the corresponding value of the number of “ressort”, its label and the associated “textForUi” is displayed in the textarea near the graph. This functionality exists for both the Streamgraphs, the one representing the actual values and the other representing the predicted values. There is another textarea which represents all the text data associated with the chosen time-interval for the Streamgraphs.

For instance, in Figure 4-7, mouse clicks are done at 1 a.m. in the yellow ribbons. So, the textarea at the top is updated with the name and count-value of actual “ressort” associated with the yellow label, and the “textForUi” corresponding to 1a.m. The second textarea is updated with the name and value of the count of predicted “ressort” associated with the yellow label and the “textForUi” corresponding to 1a.m., whereas, the last textarea represents the “textForUi” for the entire time-slot selected by the slider.

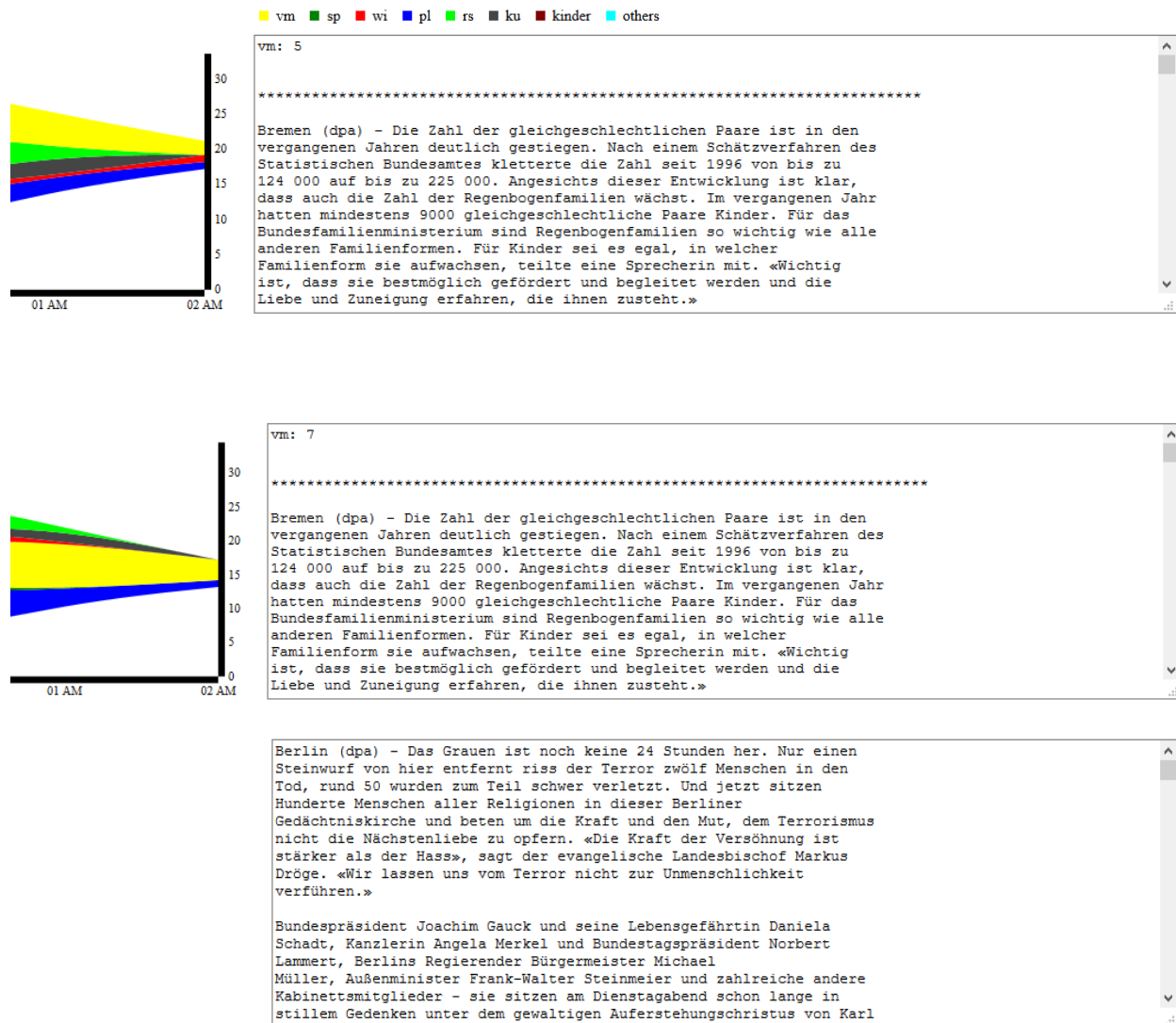


Figure 4-7: Update of textareas after mouse clicks at selected position of the ribbons.

The user has clicked at the yellow ribbons at 1a.m. in the two graphs and the corresponding labels, their count and the corresponding “textForUi” for that particular hour are shown in the textareas. The last textarea shows the “textForUi” for the entire selected time-duration.

Following the mouse-click, LDA for the text corresponding to the static Streamgraph is performed by the LDAHandler, and then, the corresponding result of topic modelling is sent to the front-end for visualization. To visualize the most important topics and their most important words, visualization is done in the form of “Zoomable Packed Circles”. The idea of this type of visualization is taken from the work of Welleck [Wel14]. D3.js [BDH+11] has its implementation of this type of *enclosure diagram*, which helps in the explanatory representation of the hierarchy. For instance, when mouse-click is done in the graph and topic modelling is done with parameters given from frontend as shown in Figure 4-8, the result obtained in the front-end is as shown in Figure 4-9, where the outer circle represents the entire set of topics, the inner circles represent the topics and the innermost circles are the words which form the topic. The size

of inner circles corresponds to their relative significance in comparison to other members. For example, the innermost circle with the word “berlin” has the greatest importance, signifying that the most frequent word in this topic is “berlin”. For the circles contained in the outermost circle, the size of the biggest circle shows that the topic composed of the words “berlin”, “anschlag”, “deutschland”, etc. is the most important.

Topic Modeling

Number of Topics:

Number of Words:

Figure 4-8: Parameters for applying LDA algorithm.

The number of topics into which the “textForUi” corresponding to static streamgraph is decomposed, can be specified here. Also, the number of top words for each topic can be given by the user.



Figure 4-9: Circle packed graph

obtained after topic modelling of “textForUi” corresponding to data in Figure 4-4 for 6 topics and 12 top words in each topic. The innermost circles represent the top words of a particular topic and the circles contained just inside the outermost circles are a representative of the topics of which the document is composed.

The circles with the smaller radii don’t have words inside them as they are very small circles. To inspect the words, the user has the option to zoom in the circles to get a bigger picture of each of the circles. An illustration of the zoomed-in view of the central inner circle of Figure 4-9 is shown in Figure 4-10.

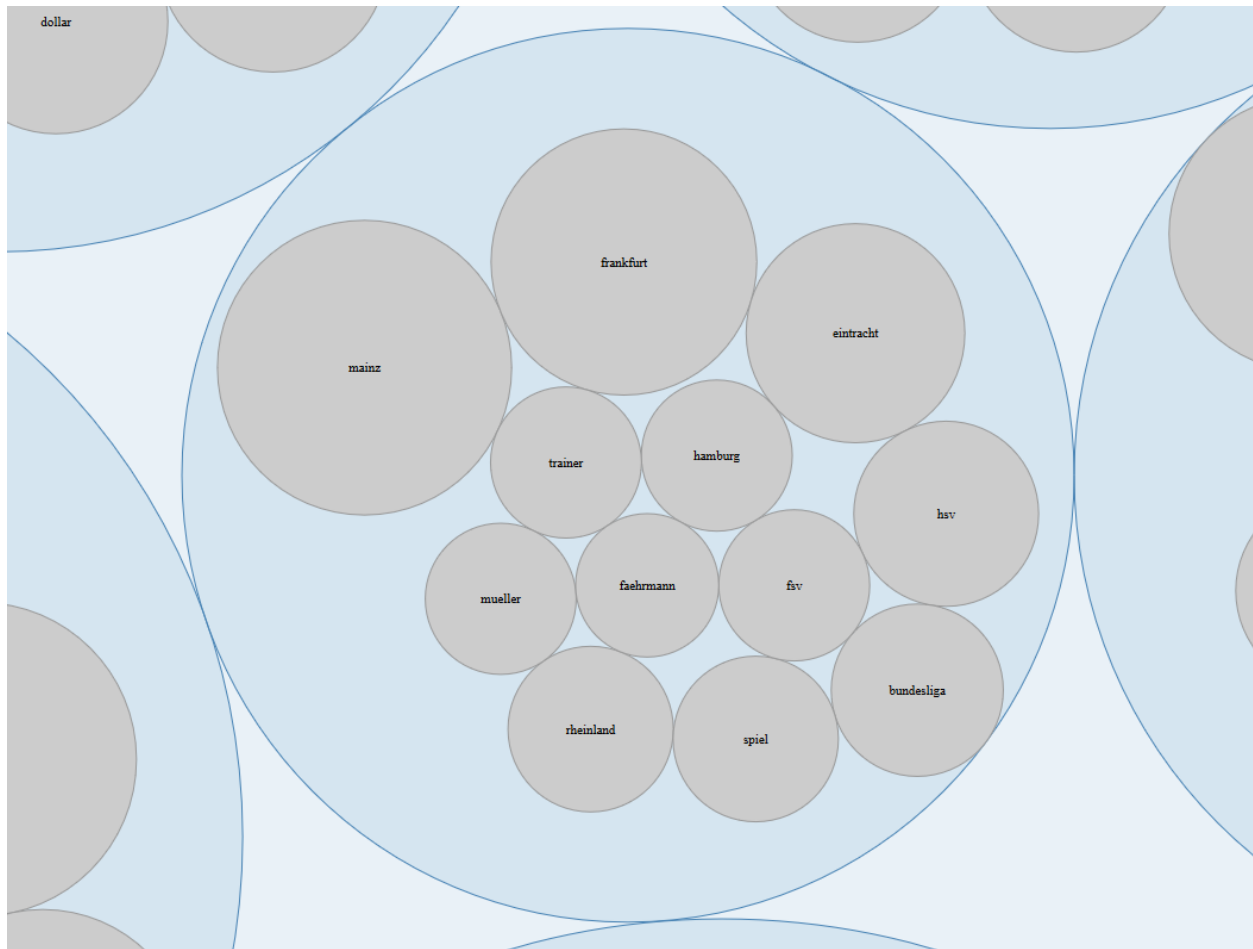
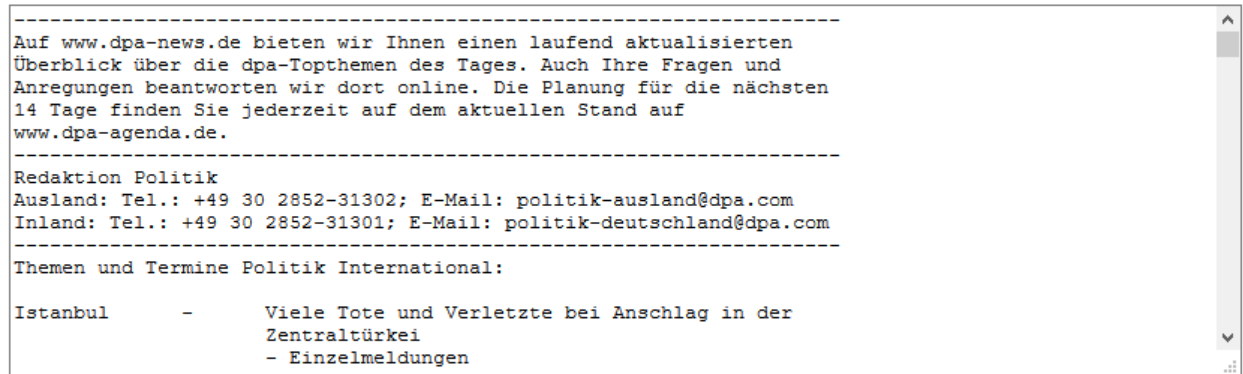


Figure 4-10: Zoomed-in view of the central inner circle inside Figure 4-9. Words which couldn't be seen in the previous figure due to lack of space can be seen clearly in this image.

Furthermore, when the circles are clicked to be enlarged, there is a textbox below the circle-packed graph which gets updated by the "textForUi" documents containing the words corresponding to the topic whose linked circle is clicked. So, along with the zoomed-in view of Figure 4-10, the text-box is updated with data as shown in Figure 4-11.



```
-----
Auf www.dpa-news.de bieten wir Ihnen einen laufend aktualisierten
Überblick über die dpa-Topthemen des Tages. Auch Ihre Fragen und
Anregungen beantworten wir dort online. Die Planung für die nächsten
14 Tage finden Sie jederzeit auf dem aktuellen Stand auf
www.dpa-agenda.de.
-----
Redaktion Politik
Ausland: Tel.: +49 30 2852-31302; E-Mail: politik-ausland@dpa.com
Inland: Tel.: +49 30 2852-31301; E-Mail: politik-deutschland@dpa.com
-----
Themen und Termine Politik International:

Istanbul      -      Viele Tote und Verletzte bei Anschlag in der
                    Zentraltürkei
                    - Einzelmeldungen
```

Figure 4-11: Text-box below the circle-packed graph

This gets updated with those “textForUi” documents which consist of the words contained in the circle which has been clicked to be enlarged. This example contains data corresponding to words inside the circle shown in Figure 4-10.

4.2 Tools and Libraries Used

This section briefly describes the tools and libraries used for the development of the Software. Tools have been used as they ease the implementation process by providing user-friendly interfaces. The use of Integrated Development Environments (IDEs) have supported greatly during the whole development phase for code-compilation, debugging and building the application.

The IDE Studio3T⁵ has been used in the thesis for import of MongoDB data from the mongodump folder, which was provided by the Institute for Visualization and Interactive Systems (VIS), University of Stuttgart. Also, the presentation (in Studio3T) of dataset in Tree view, table view and JSON view was continuously used to verify the correctness of results, by comparing the data obtained in the front-end and that in the database. Studio3T has an extremely user-friendly interface for the same.

As the backend of the software is developed in Java, the Eclipse IDE⁶ has been used for development, and the Eclipse plugin for Maven⁷ integration has been used. Eclipse was chosen because of it is free to use and has several simple shortcuts to facilitate JAVA development. Maven has been used for importing various dependencies used in the Eclipse project. The use of Maven eased the difficulty of downloading and including specific Java Archives in the build path of the project. This was extremely helpful because many external libraries have also been used in the thesis. For example, OpenNLP⁸, an open source Java library by Apache has been used in this work to perform tokenization of sentences so that the stopwords can be removed from them. Furthermore, Mallet [CK02], a *Java-based package*, has been used in this thesis for topic modelling. The implementation of LDA in the *Mallet topic modelling toolkit* has been used for the statistical analysis of the texts associated with the labels in the front-end. The RESTful web services have been developed using Spring Boot [SOF18], offering the feature of Embedded Tomcat as the server, due to which stand-alone applications can be built very easily.

To train the SVM classifier and to use the model to make predictions, R has been used as the programming language. RStudio⁹ has been used as the IDE for writing the R code to perform machine learning for prediction of labels. With RStudio, the installation of packages becomes very easy. For example, “tm package”¹⁰ of R has been used for the removal of German stopwords, conversion to lowercase, other text preprocessing tasks and creation of Document Term Matrix, before the prediction algorithm is applied. Furthermore, the “caret package”¹¹ has been used for partitioning the data into training and testing set. For training the SVM for multiclass classification, “e1071 package” [CRA17] of R has been used.

The visualizations in the front-end including Streamgraphs and circle-packing after topic modelling are adaptations of the examples of the D3 library [BDH+11] of JavaScript. D3 offers several examples of interactive visualizations on its website, which can be adapted to suit the requirements of the visualizations.

⁵ <https://studio3t.com/>

⁶ <https://www.eclipse.org/>

⁷ <https://maven.apache.org/>

⁸ <https://opennlp.apache.org/>

⁹ <https://www.rstudio.com/>

¹⁰ <https://cran.r-project.org/web/packages/tm/index.html>

¹¹ <https://cran.r-project.org/web/packages/caret/vignettes/caret.html>

5. Case study

In this chapter, an application of the approach discussed in this thesis is demonstrated using two case studies. The case studies were conducted on 25th July, 2018, using the news data from the German news agency Deutsche Presse-Agentur (DPA) ⁴ collected by the Institute for Visualization and Interactive Systems, University of Stuttgart (VISUS). Thus, the approach for real-time monitoring and analysis of machine-learning algorithm has been evaluated with the help of these case studies and the functionalities of the software have been tested as well. Two case studies have been conducted, one using a classifier model, trained on larger set of data, which yielded high accuracy results of prediction, and the other using a classifier model, trained on only a small set of data, with low accuracy results of prediction.

5.1 Classifier trained on 15,000 records data

In this case study, an SVM classifier model is trained on 15,000 records of DPA data, by the approach discussed in Section 3. This model is then used to predict the “ressort” of “textForUi” for all the 100,000 documents in the DPA data. It took about 40 minutes to train the multi-class SVM classifier on a PC with 8 GB RAM, Intel Core i7 Processor, and 64-bit Operating System. 10-fold cross-validation was done to evaluate the performance of the classifier, and the average classification accuracy was found to be over 84%. The machine learning application, so-implemented as in Section 4 is then monitored and analyzed using the software developed in this thesis, which gives a visualization aid to the user to justify the performance of the classifier.

5.1.1 Setup

First of all, the SVM classifier was trained on the data as mentioned and the model was then saved to be used by the application. Now, the database and RServe were started. The application was then started and the model trained on this data was loaded by the application for predictions. The Streamgraphs for the actual and predicted values of “ressort” were observed.

5.1.2 General behavior of the classifier and visual analysis

The application is started, and the visualization in the HTML page is observed carefully. In most of the cases, the visualization of the two Streamgraphs (one for actual values of “ressort” and the other for predicted values of “ressort”) are found to be similar. This similarity was seen directly by looking at the similar shapes and wiggles in the Streamgraphs. An example screenshot of similar Streamgraphs is shown in Figure 5-1. This screenshot has been taken for the data corresponding to 17th December 2016 from 9 p.m. to 18th December 2 a.m.

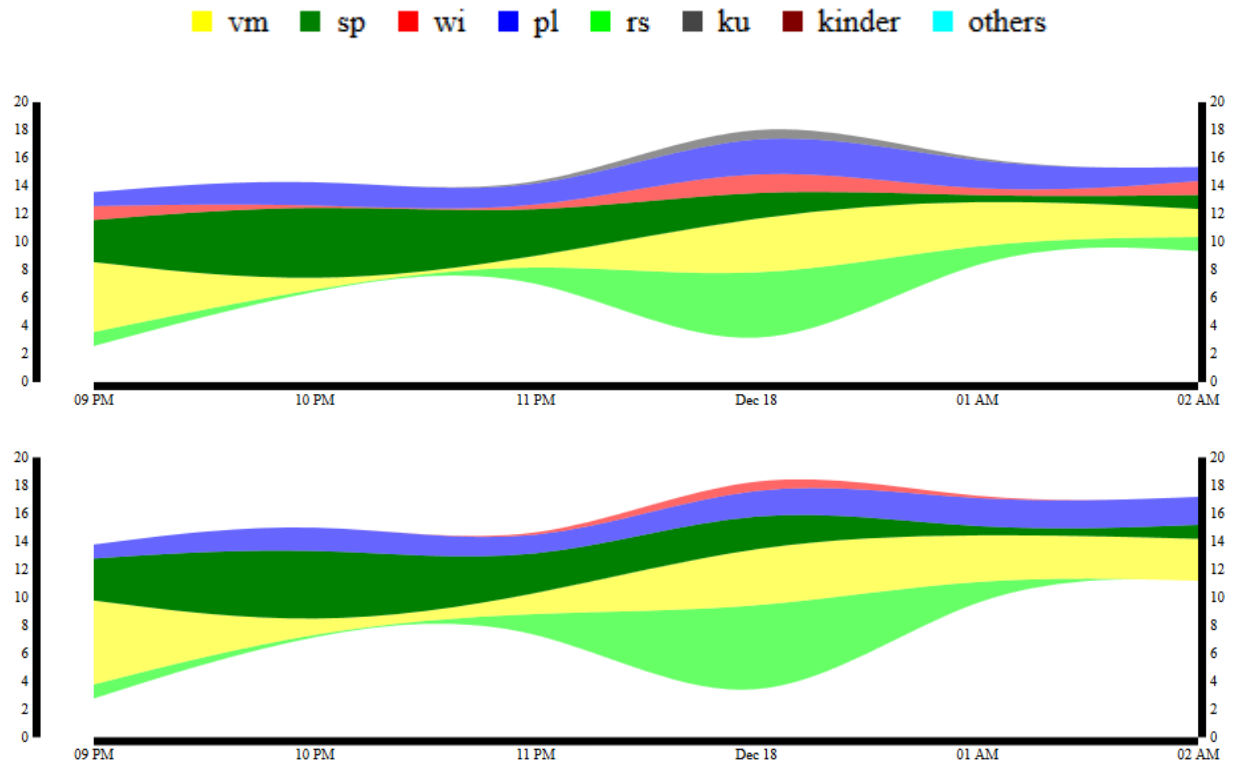


Figure 5-1: A screenshot of the continuous Streamgraphs showing their general behavior (case study 1). The upper Streamgraph corresponds to the values of actual “ressorts” and the lower Streamgraph corresponds to the “predicted ressorts”. Usually, the two Streamgraphs were found to be similar to each other when the classifier output from the model trained on 15,000 records of DPA data was used.

To justify the similarities, an explorative analysis was also performed by zooming in the portions of the Streamgraphs with similar behaviors. These sections of the graphs were observed statically, and the values of the “ressorts” (actual and predicted) were compared for a few hours. This involved selecting a few portions of the continuous Streamgraphs using the slider implemented in Section 4, through which motionless graphs can be obtained. For each hour in the stationary graphs, the values of all the ribbons corresponding to “ressort” were compared. It was found that in most of the cases the values were either equal or somewhat similar to each other. For example, a portion of the Streamgraphs in Figure 5-1 was selected and observed. When the yellow ribbon at 12 a.m. was clicked for inspection, the values of “ressort” and their counts in the two graphs were found equal as shown in Figure 5-2.

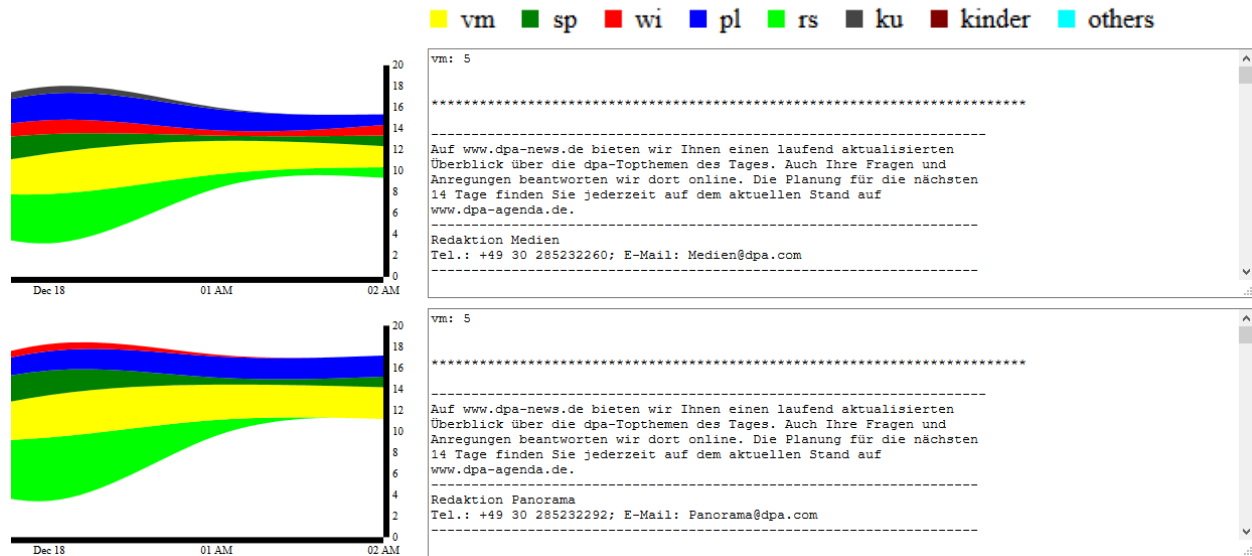


Figure 5-2: Static observation of the Streamgraphs shown in Figure 5-1

It demonstrates similarity in the values and count of "ressort". On clicking the yellow ribbons at 12 a.m. in the two graphs, "ressort" was found to be "pl" with the count being "5" in both the graphs as seen in the text-areas adjacent to each graph. The text-areas also show the corresponding "textForUi" for that instance of time.

Analyses similar to Figure 5-2 were done for each hour, and the corresponding obtained values were found to be approximately equal for both the graphs, as shown in Table 5.1.

Time	Actual values and count of "ressort"	Predicted values and count of "ressort"
9 p.m.	"rs": "1" "vm": "6" "sp": "3" "pl": "1"	"rs": "1" "vm": "5" "sp": "3" "pl": "1" "wi": "1"
10 p.m.	"sp": "6" "pl": "2"	"sp": "6" "pl": "2"
11 p.m.	"sp": "3" "pl": "1"	"sp": "2" "pl": "1" "vm": "1"
12 a.m.	"rs": "7" "vm": "5" "sp": "2" "wi": "2" "pl": "3" "ku": "1"	"rs": "9" "vm": "5" "sp": "3" "wi": "1" "pl": "2"
1 a.m.	"vm": "3" "pl": "2"	"vm": "3" "pl": "2"

Table 5.1: Analysis for Figure 5-2 showing values and count of "ressort"

On clicking, the application performed the LDA Analysis on the dataset (“textForUi” for the static portion of the Streamgraph). A circle-packed graph for the “textForUi” corresponding to graphs in Figure 5-1 is shown in Figure 5-3.

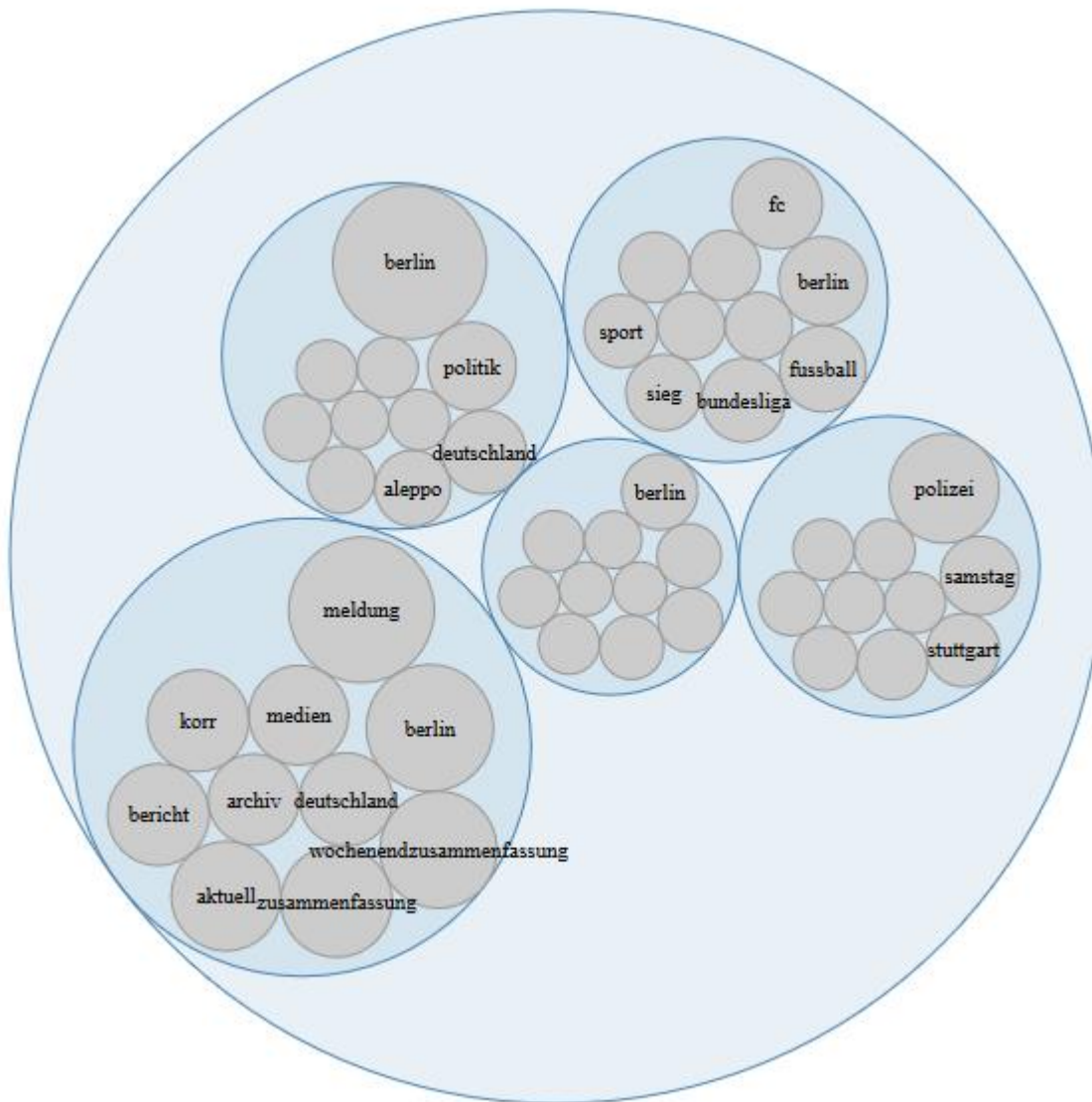


Figure 5-3 : Circle-packed graph for “textForUi” corresponding to Figure 5-1.

LDA topic modelling is done to get 5 topics and 10 top words in each topic. 5 circles represent the topics and the inner-most circles represent the top words in each topic. The sizes of 5 circles correspond to the relative importance of each topic and the same is applicable for the circles representing top words in each topic.

5.1.3 Deviations observed in the behavior of classifier and visual analytics

Some deviations were observed in the Streamgraphs where the two Streamgraphs were not consistent with each other and were quite dissimilar. This required probing the visualizations and inspection of the behavior to figure out if there is something wrong with the performance of the classifier or if there are abrupt changes in the type of data fed to the classifier.

Sudden changes had occurred in the visualization output for the data corresponding to midnight of 20th December 2016. After seeing the abrupt deviations in the continuous Streamgraphs, a portion of the Streamgraphs between 1 a.m. to 6 a.m. was selected for examination. A snapshot of the Streamgraphs is shown in Figure 5-4.

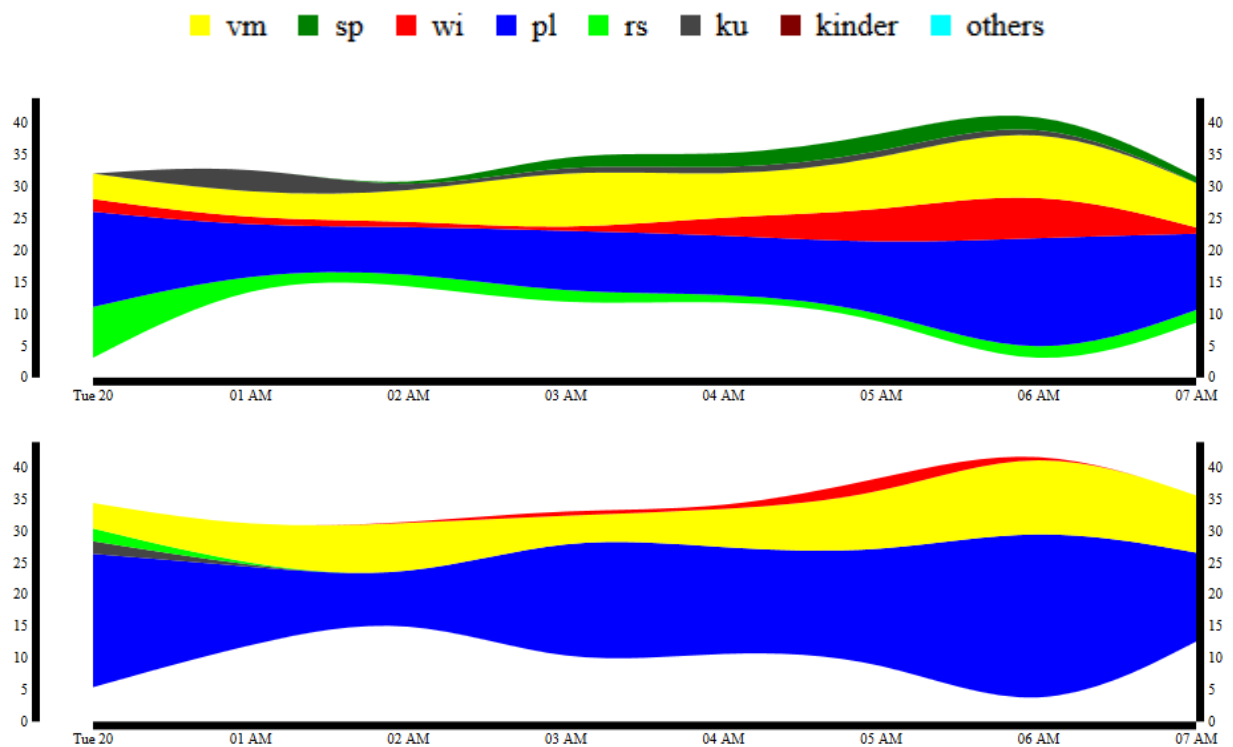


Figure 5-4: Selected portion of continuous Streamgraphs showing abrupt deviations where actual values and count of “ressort” differ from predicted values and count in case study 1. The user can easily recognize the difference between the two graphs, which are not consistent with each other.

As shown in Figure 5-4, the two Streamgraphs are quite different from each other. To explore the associated data, a mouse-click was done at 3 a.m. The observed corresponding data is shown in Figure 5-5.

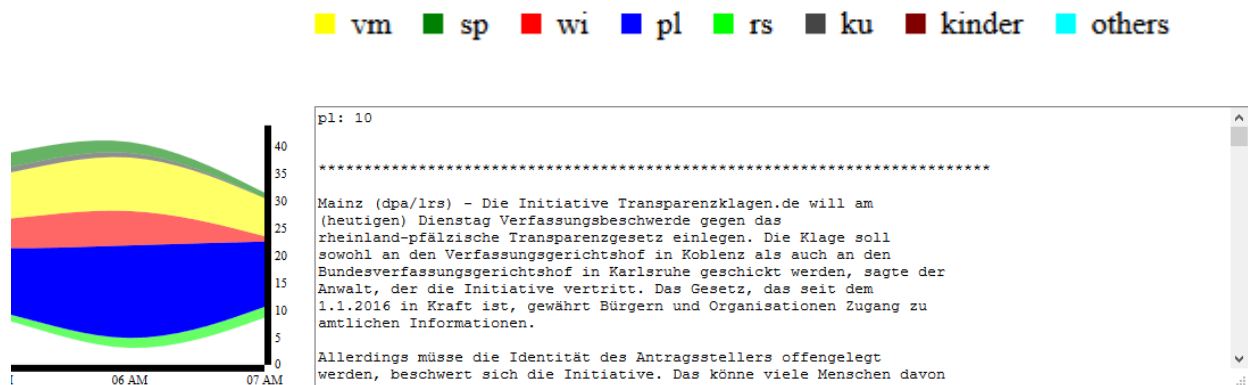


Figure 5-5: Actual data corresponding to 3a.m. for the blue ribbon.

Actual “ressort” is “pl” and the “textForUi” for that hour is shown in the same text-area. The user can read this text-area to find the news related to the category which he/she had clicked.

Then, the mouse-click was done in the blue ribbon to find the data at 3a.m. for the predicted value of “ressort”. A screenshot of the same is given in Figure 5-6.

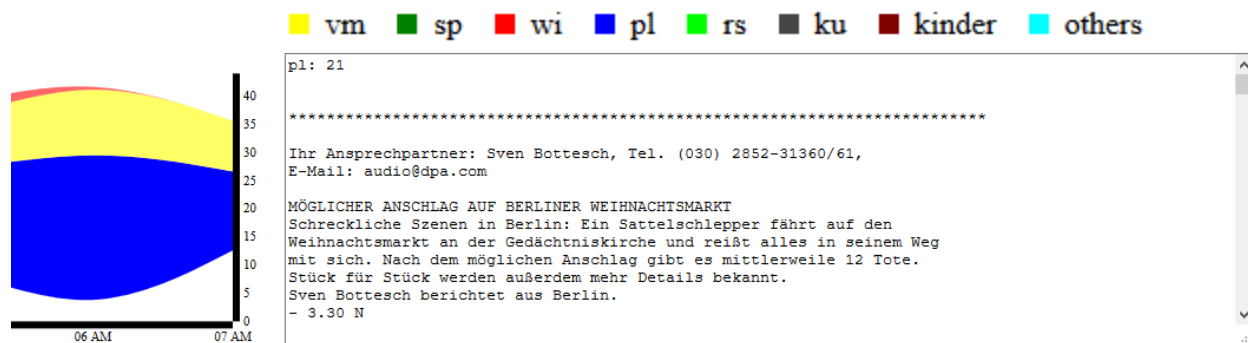


Figure 5-6: Predicted data corresponding to 3a.m. for the blue ribbon.

Predicted “ressort” is “pl” and the “textForUi” for that hour is shown in the same text-area. The associated “textForUi” is a description of the terrorist attack in Berlin.

Now, to dive deeper and inspect the connected “textForUi” for the selected portion of the Streamgraphs, topic modelling was done and the output was visualized in the form of a circle-packed graph. The parameters like “number of topics” and “no. of words for each topic” were given from the front-end. “Number of topics” was chosen to be 5 and the “number of top-words for each topic” was given as 10. A screenshot of the packed-circle graph is shown in Figure 5-7.

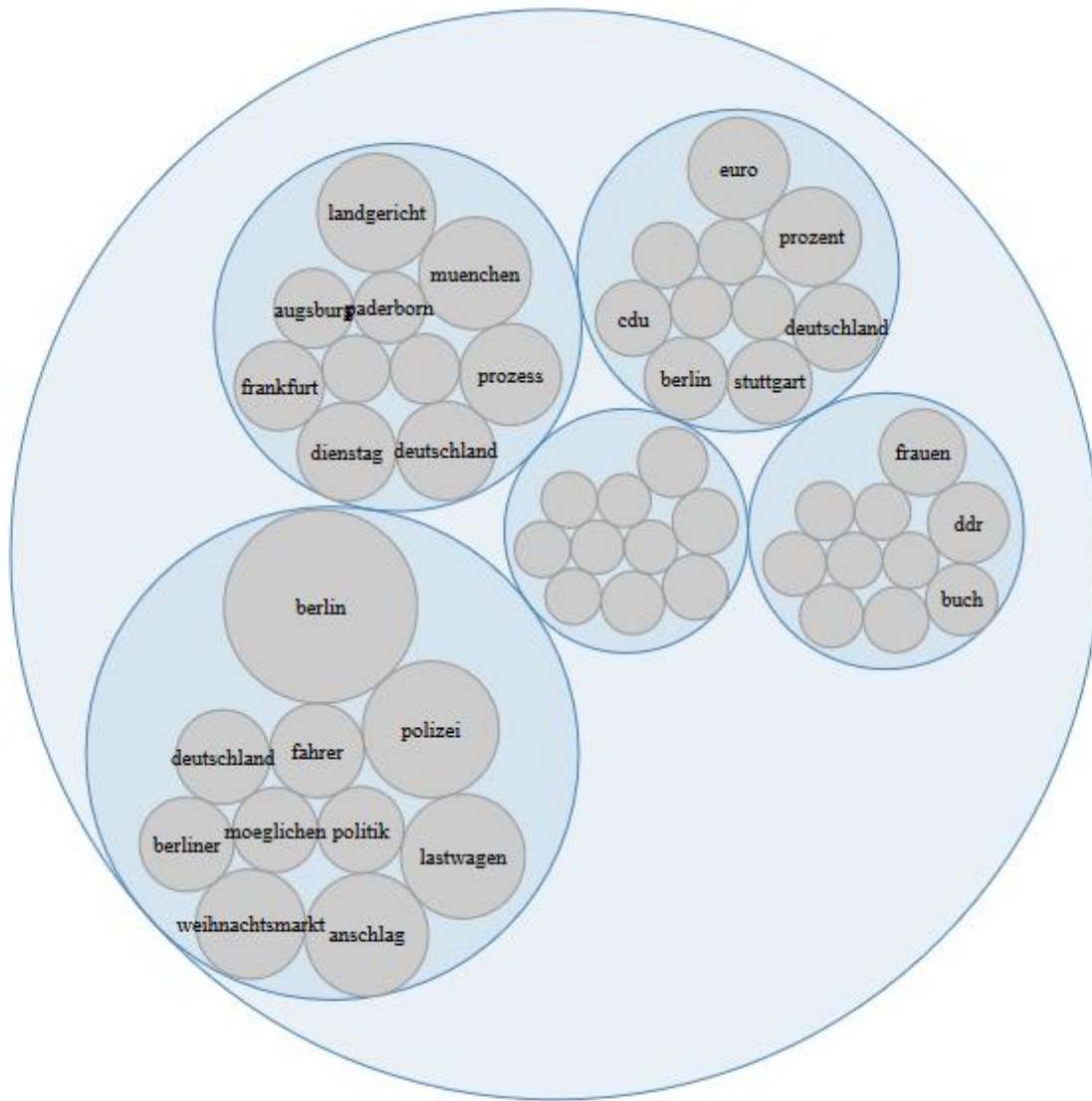


Figure 5-7: Circle-packed graph

obtained after topic modelling of “textForUi” for the Streamgraph of Figure 5-4. LDA is done to retrieve 5 topics and top 10 words for each topic. 5 inner circles represent the topics and the innermost circles represent the top words for each topic. The size of the circles are a representation of the weights of the corresponding topics and words.

As the biggest inner circle of Figure 5-7 has the highest weight, it means that the topic associated with this circle was the most important for the selected time interval. So, this circle was zoomed in to observe the details of the topic minutely. When this circle was clicked, it got focused and updated the text-area below the circle-packed graph with the “textForUi” which contained the words present in this circle. The

idea of this visual tool was to give a visual explorative aid to examine the topic responsible for the abrupt change in the visualizations. A zoomed-in view of the biggest inner circle is shown in Figure 5-8.

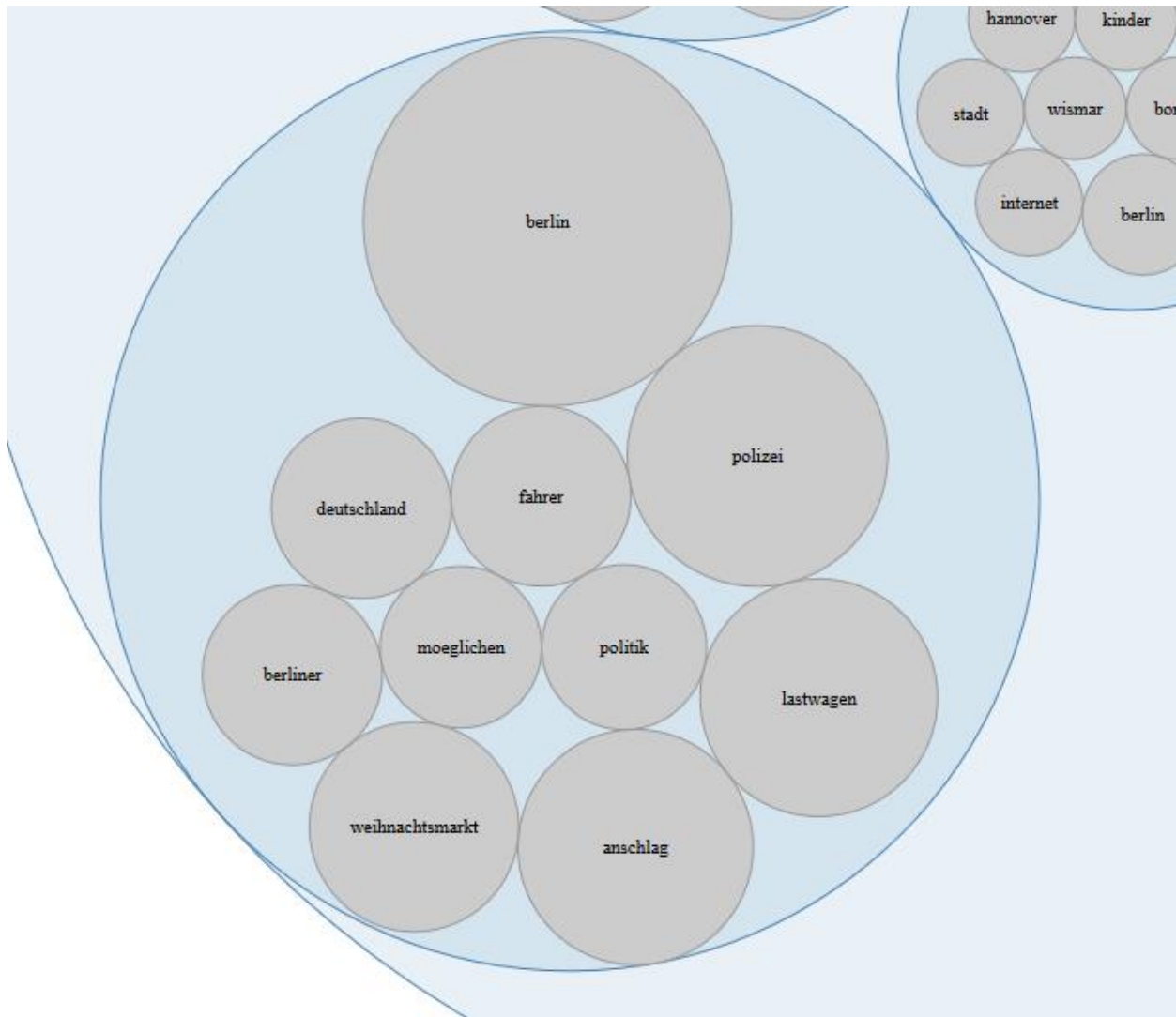


Figure 5-8: A zoomed-in view of the biggest inner circle of Figure 5-7.

This image shows the top words related to the corresponding topic. The user can gain an idea of the underlying document by merely seeing this graph. The words like “polizei”, “anschlag”, “lastwagen”, etc. indicate an attack in the area.

From Figure 5-8, the 10 most significant words related to the topic can be viewed. The words in the bigger innermost circles are extremely noteworthy as they have tremendous importance in the entire dataset, being part of the biggest circle. Taking a note of Figure 5-8, the most considerable words are “berlin”, “polizei”, “anschlag”, “lastwagen”, “weihnachtsmarkt”, “fahrer”, and so on. On seeing these words, a human can easily make a conjecture that this data refers to an attack by a truck-driver at the Christmas market in Berlin, which might have caused a police-intervention in that area.

Case study

Now, the report in updated text-box below the circle-packed graph is read to gain an insight of the “textForUi” associated to the circle of Figure 5-8. A screenshot of this text-area is shown in Figure 5-9. The data inside this text-area described the event of an attack in the Christmas market, near the Kaiser Wilhelm Memorial Church in Berlin, which resulted in some people being injured and few casualties. The attack took place on 19th December 2016 at 20:02 CET and was assumed to be a terrorist attack, after investigation.

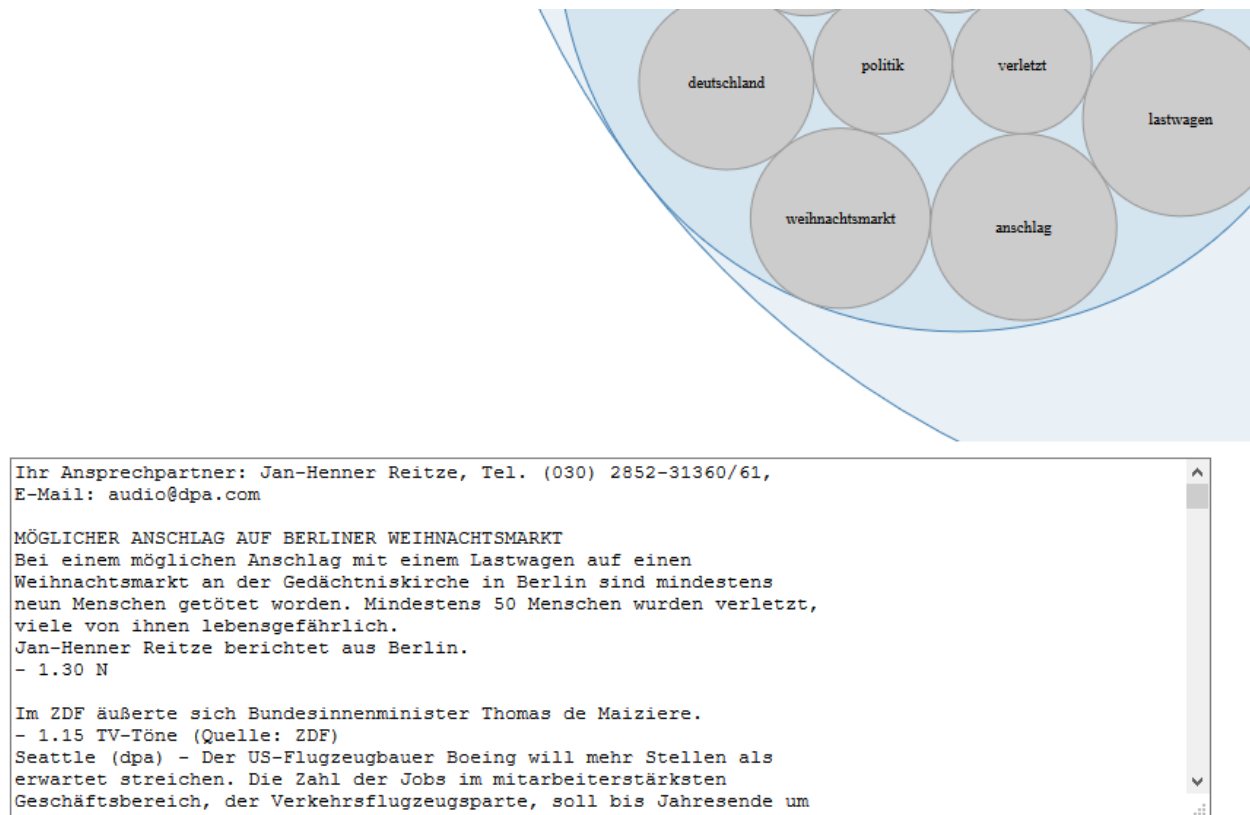


Figure 5-9: “textsForUi” which contain the words from Figure 5-8.

The text-area mostly contains the news-data of an attack in the Christmas market, near the Kaiser Wilhelm Memorial Church in Berlin, which resulted in some people being injured and even few casualties.

Thus, explorative analysis of the visualizations shows the occurrence of an abnormal incident on 19th December 2016. Therefore, the classifier was unable to make the right predictions about the “ressort” for “textForUi” corresponding to the news-data which started flowing in around this time. So, it can be concluded that the deviation in the behavior of Streamgraphs in Figure 5-4 was due to an unexpected reported incident, which resulted in unanticipated data input to the classifier. Hence, this deviation in the behavior of classifier can be considered normal as it was the outcome of a scarce and unexpected data.

5.1.4 Discussion on this case study

Explorations similar to the previous sections were performed on the visualization obtained in this case study. The predictions were found to be in line with the actual values, except for a few cases (which

generally represented rare incidents). Hence, it was deduced that the machine learning model used in this case study was performing well, other than exceptional events, which caused it to digress from the normal behavior.

5.2 Classifier trained on 150 records of data

In the second case study, an SVM classifier model was trained on 150 documents of news dataset obtained by the VIS Department of the University of Stuttgart from the German news agency Deutsche Presse-Agentur (DPA), by the approach discussed in Section 3. Next, this model was used to make predictions about “ressort” for “textForUi” for the entire dataset. Average classification accuracy was found to be around 55% when evaluation was done by 10-fold cross-validation. Real-time monitoring and analysis of this machine learning application was conducted to infer whether the model was good enough or required retraining.

5.2.1 Setup

First of all, the SVM classifier was trained on 150 records of random data taken from 100,000 sets of DPA data and the model was then saved to be used by the application. Now, the database and RServe were started. The application was then started and the model trained on this data was loaded by the Java application for predictions. The Streamgraphs for the actual and predicted values of “ressort” were observed.

5.2.2 Monitoring and analysis of general behavior of the classifier

It was observed that the two Streamgraphs appeared to be of different shapes very often. An example screenshot showing the common behavior of Streamgraphs is shown in Figure 5-10. This dataset corresponds to the “textForUi” from 2p.m. to 7p.m. for 17th December, 2017.

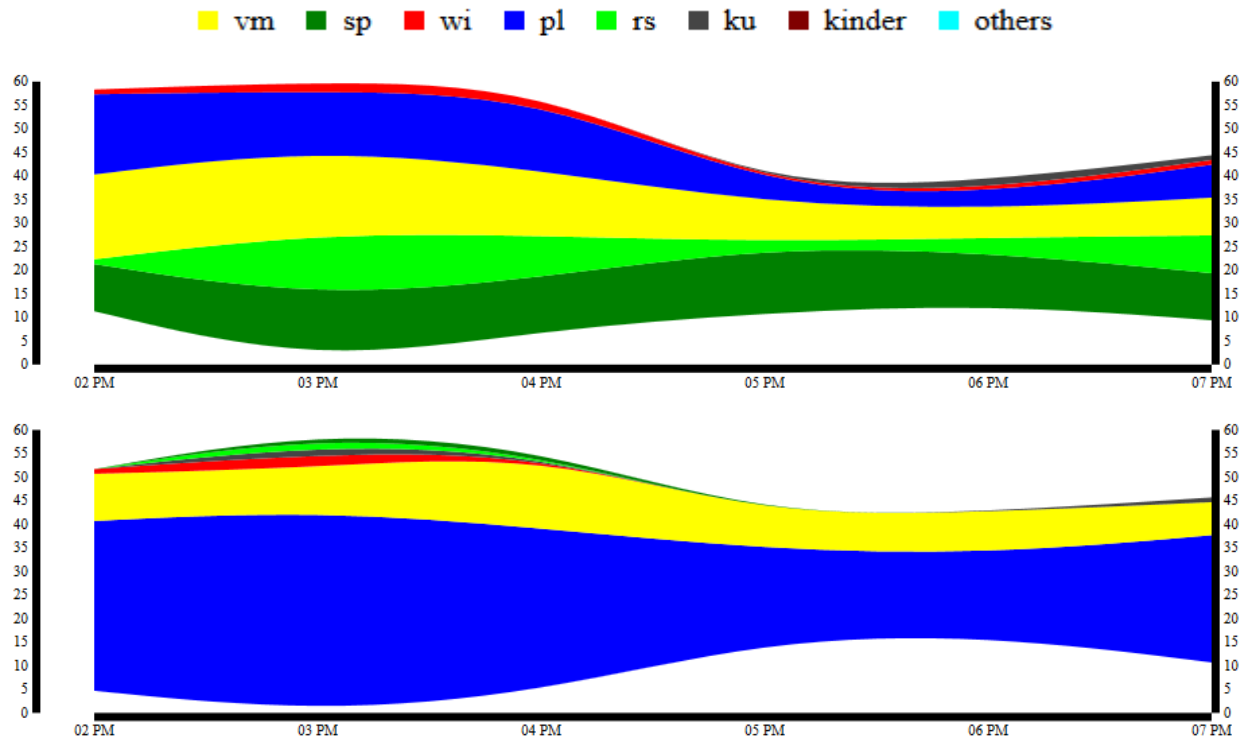


Figure 5-10: A screenshot of the Streamgraphs showing their general behavior when the classifier was trained on 150 records of data. The upper Streamgraph corresponds to the values of actual “ressort” and the lower Streamgraph corresponds to the “predicted ressort”. Usually, the two Streamgraphs were extremely inconsistent with each other.

The two Streamgraphs were non-identical in most of the cases. As seen from Figure 5-10, the two Streamgraphs show an extreme form of deviation. The predictions of the classifier are mostly inclined to one value of “ressort”. So, it was required to investigate the related “textForUi” to understand the reason for deviation.

When a click was done on the Streamgraph for the actual values of “ressort”, LDA topic modelling ran in the backend and the resultant data in the form of circle-packed graph appeared on the HTML page. Topic modelling was done to obtain 6 topics and top 10 words associated to each topic. These values could be chosen by the user. Figure 5-11 shows the resultant image of the circle-packed graph.

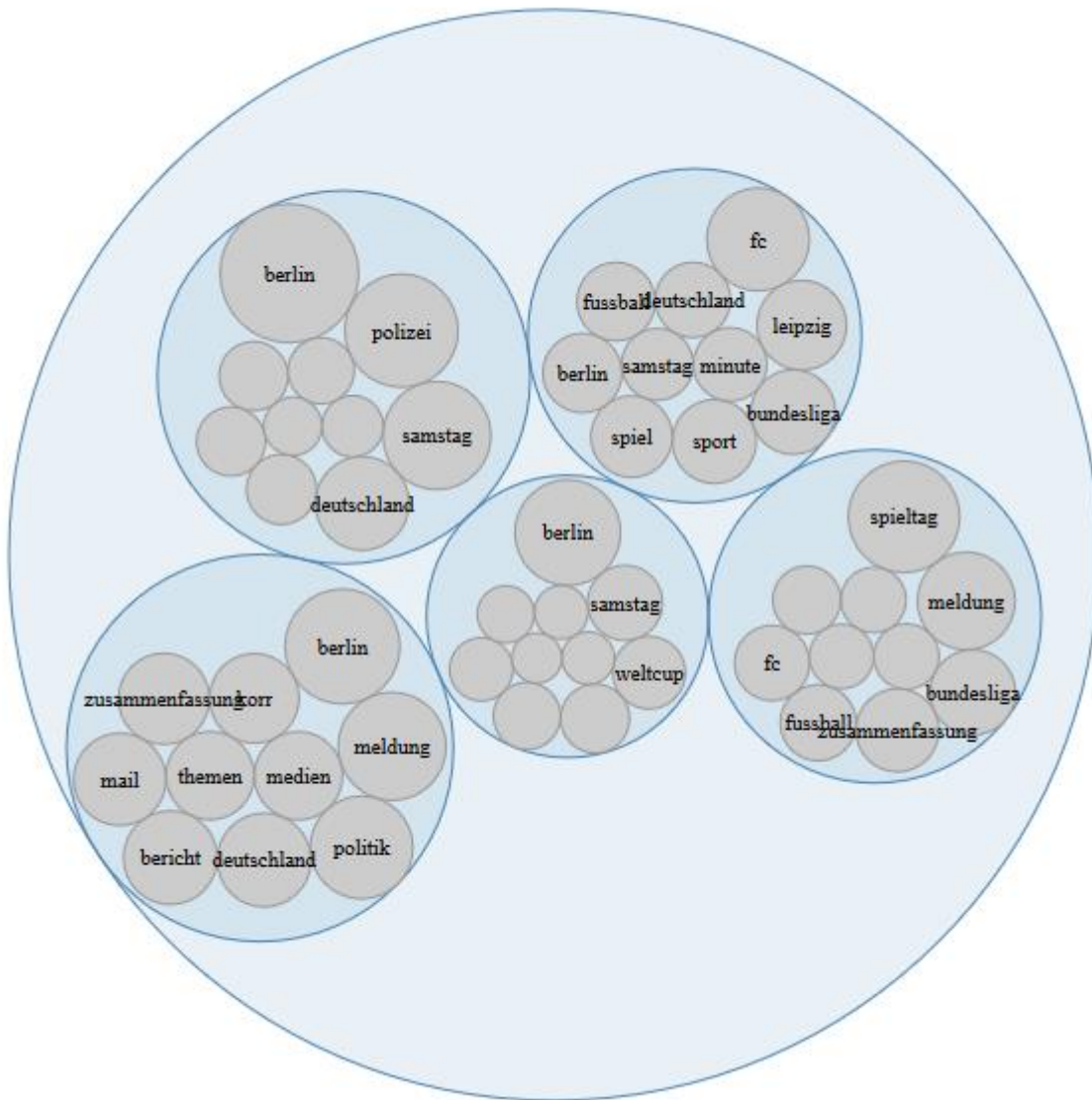


Figure 5-11: Circle-packed graph for “textForUi” corresponding to Figure 5-10.

LDA topic modelling is done to get 5 topics and 10 top words in each topic. 5 circles represent the topics and the inner-most circles represent the top words in each topic. The sizes of 5 circles correspond to the relative importance of each topic and the same is applicable for the circles representing top words in each topic.

When the biggest circle in Figure 5-11 is zoomed in by clicking at it, the corresponding text-area gets updated with the “textsForUi” containing the words in this circle. A snapshot of this circle is shown in Figure 5-12, and the corresponding text-area can be seen in Figure 5-13.

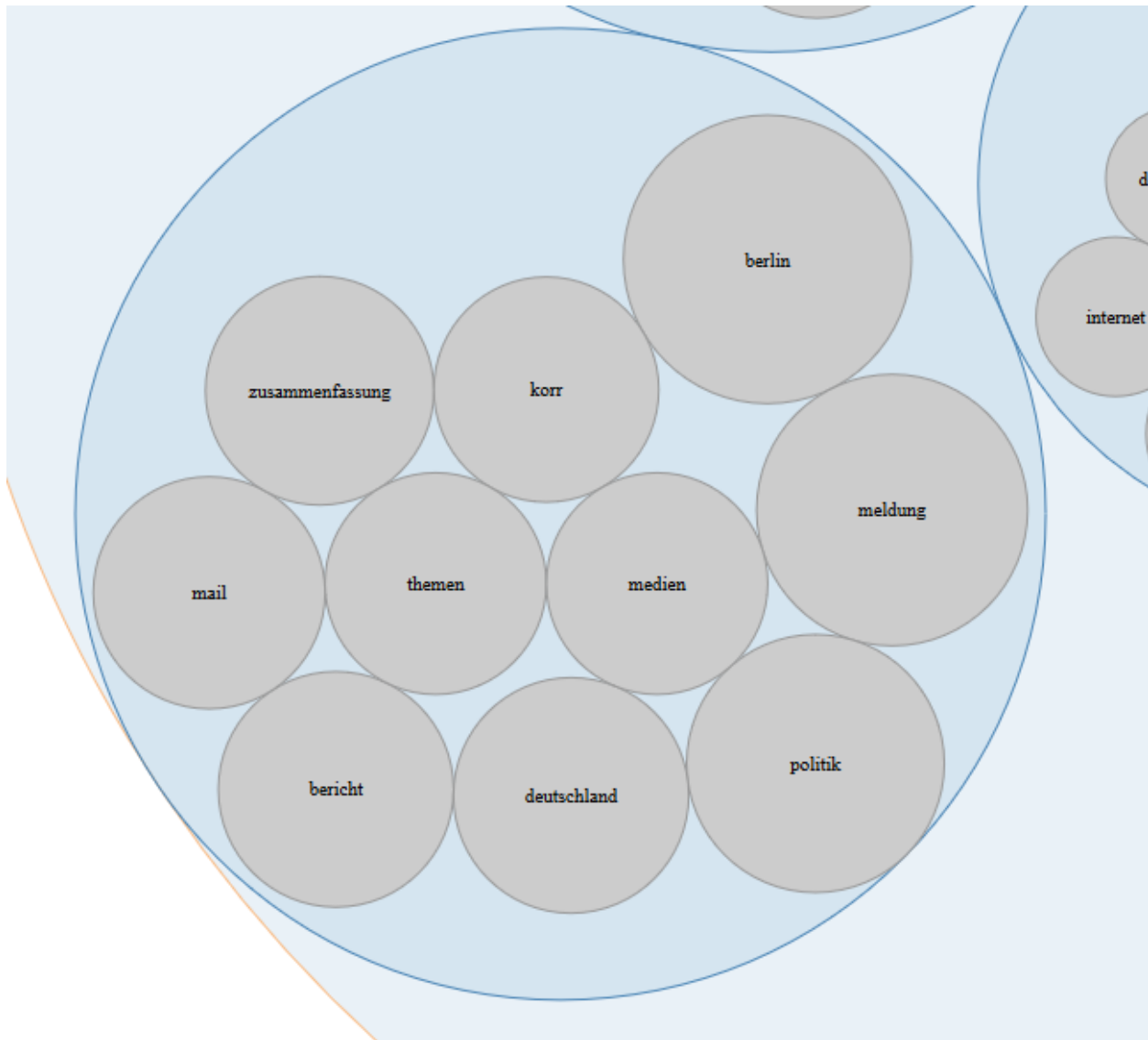


Figure 5-12: A zoomed-in view of the biggest circle inside Figure 5-11.

The top 10 words of the connected topic can be seen clearly. The user cannot understand the complete context of the topic, merely by looking at the figure. He/she may have to read the report related to this topic to probe further. The words like “mail”, “politik”, “meldung”, etc. don’t clearly indicate the related event, but it doesn’t look like an abnormal event.

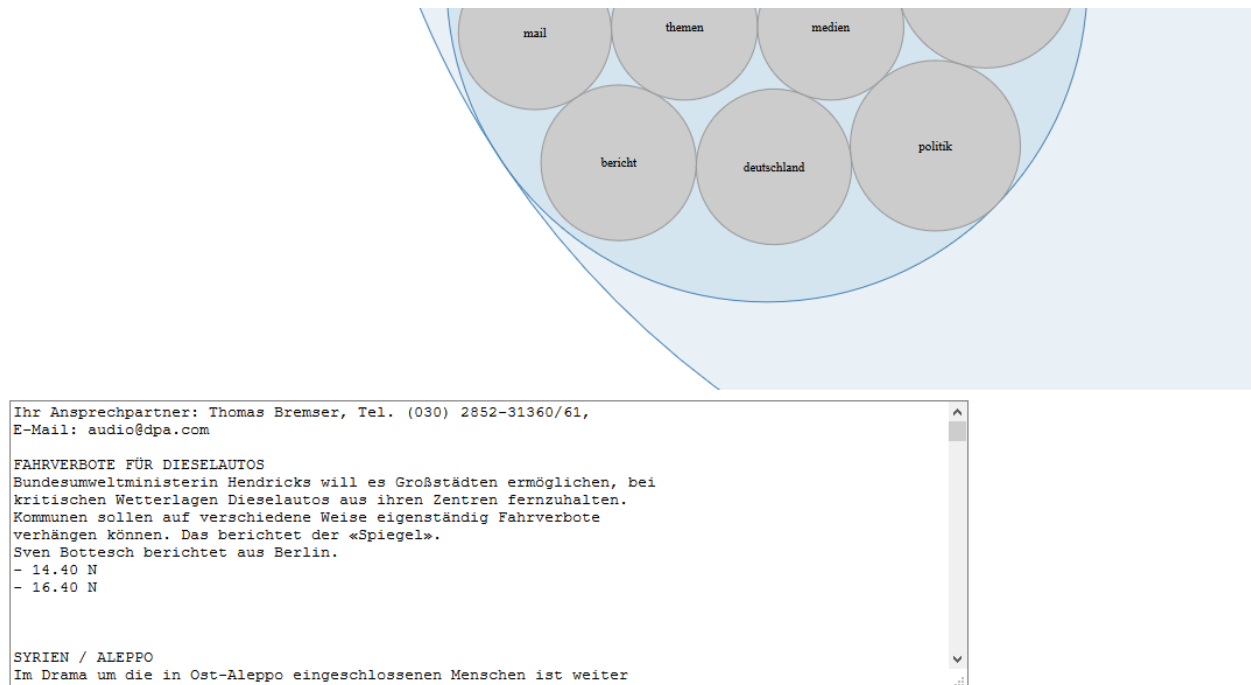


Figure 5-13: Text-area for “textForUi” containing words of the circle shown in Figure 5-12. After reading this area, it was found that the news-data represents the day-to-day politics of Germany.

After probing into Figure 5-11, it was perceived that the words do not constitute an abnormal event. To justify the speculation, the updated text-area was read and it was found that the news belonged to the day-to-day politics and general events of Germany. Other topics in Figure 5-11 were also scrutinized. For example, the 2nd largest circle represented the data corresponding to Sports news (mainly football) of Germany as seen in Figure 5-14. The words “spiel”, “weltcup”, “bundesliga”, “sport”, “fussball”, etc. are indications of Sports-related news. The corresponding news reports in the text-area further confirmed this hypothesis.

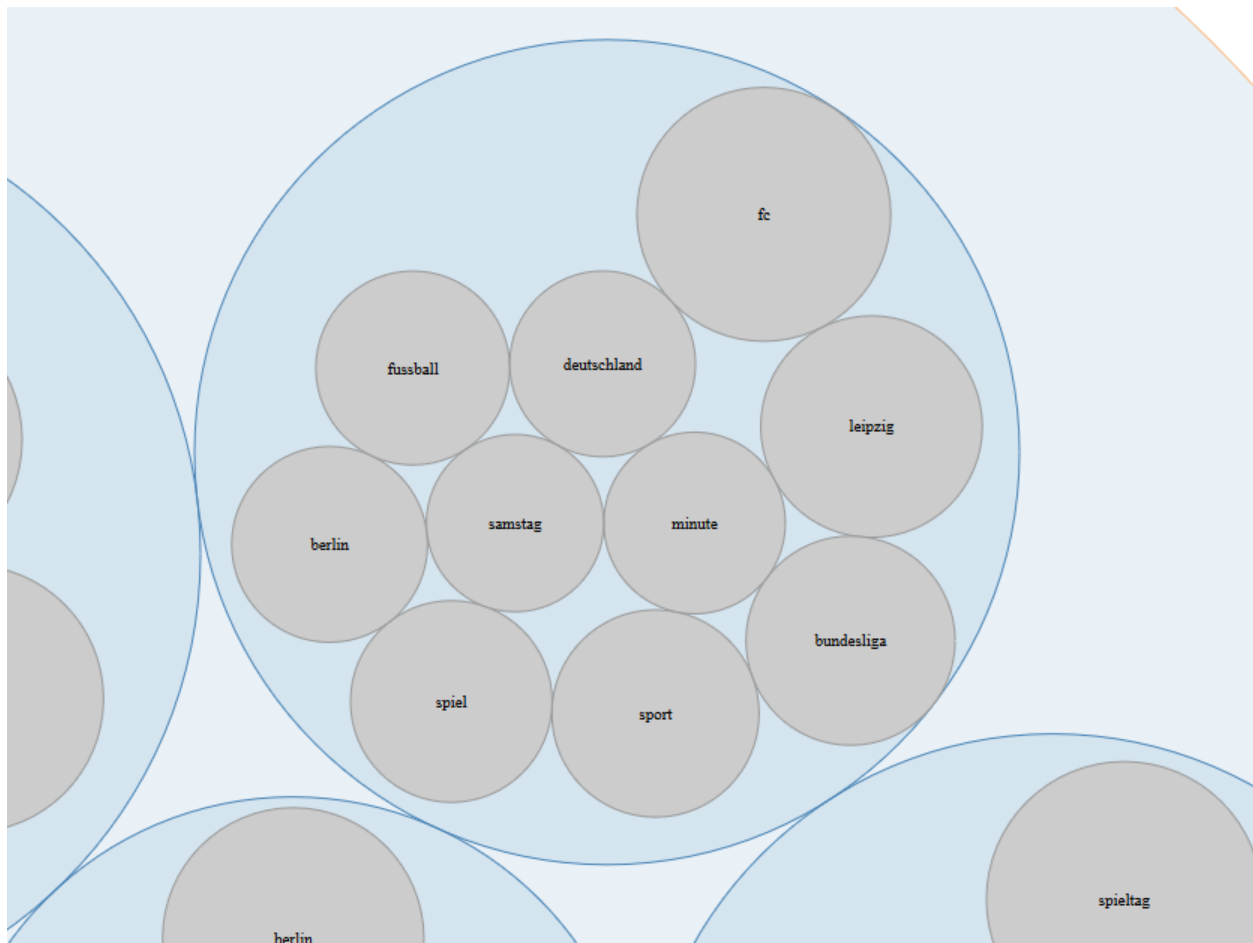


Figure 5-14: A zoomed-in view of the 2nd largest circle inside Figure 5-11.

The top 10 words of the connected topic like “spiel”, “sport”, “bundesliga”, etc. can be seen clearly to assume that the data belongs to Sports-related news.

5.2.3 Discussion on this case study

Similar explorations were carried out for Streamgraphs observed in this case study. Many deviations were observed in the visualizations, i.e., the predicted values of “ressort” were very different from the actual values of “ressort”, even in case of general news data. Hence, it can be concluded that the machine-learning model was not efficient enough to handle the common news data. So, real-time monitoring and visual analysis of the algorithm shows that retraining of the model is needed.

6. Conclusion

In this thesis, an approach for real-time monitoring and analysis of machine learning applications has been presented. Contributions in the areas of Information Visualization, Visual Analytics, Machine Learning, Topic Modelling, etc. have been used to deal with the problem. The research question of understanding the behavior of machine learning models to find the point of time when retraining of the model is needed has been addressed.

6.1 Summary

To deal with the problem statement, real-time simulation of static dataset is performed and visualizations of real and predicted labels are done. These visualizations corresponding to actual and predicted labels are comparable by humans. To aid the comparison between visualizations, several information visualization and visual analytics methods have been provided. For instance, users can select portions of visualizations to obtain static graphs, retrieve information of data related to the visualizations, and perform topic modelling to obtain most significant topics and their related words, and gain an insight of data associated with topics. In case of abnormal deviations of classifier output in contrast to real labels, users can perform explorative analysis of data at those points of time to determine whether the data is exceptional or the classifier is not performing well. Thus, users have been provided with visual aid to ascertain the correctness of machine learning application so that they can decide the correctness of the classifier model merely by observation and exploration, rather than digging into the model for its evaluation. This solves the problem of those users for whom the machine learning application is a black-box.

The software developed to demonstrate the approach is composed of backend and frontend. This software simulates a set of real-time data and sends it continuously to the frontend for visualization. Backend of the software also sends this data to the classifier, which predicts the associated labels. These labels are then sent to the frontend for visualization. Users of this software can thus, continuously monitor the output of the machine learning application, and observe discrepancies in the behavior of the two visualizations. Streamgraphs have been used for visualizations of the two labels (actual and predicted). In case of inconsistencies between the Streamgraphs, users can probe into the data which has generated the labels. These investigations can be done directly in the frontend. Users can also perform latent Dirichlet allocation (LDA) topic modelling and obtain the results in the form of circle-packed graphs. They can also decide on the number of topics and words for LDA. Users have been provided with the option of viewing the data related to a particular topic acquired from application of LDA algorithm.

Furthermore, two case studies are conducted to assess the approach and software developed in this work. In the first case, a classifier model with good accuracy score is used and in the second case, the one with

Conclusion

considerably low accuracy score is taken. The performance of these classifiers is verified using the visual analytics tools developed in this thesis.

Hence, the objective of real-time monitoring and providing judgment regarding classifier performance using visual analytics has been achieved in this thesis. There is a possibility to do so without having access to details of the machine learning model.

6.2 Outlook

This work can be extended to incorporate more and more features of visual analytics and ability to deal with the actual real-time data. For instance, further extension may involve giving several options to the user for choosing the classifier model and then obtaining the visualizations, so that he/she can compare the predictions of different models and choose the best one. Also, the user could be provided with the alternative to obtain visualization of past data by mere movements and dragging of mouse buttons, i.e., there could be possibilities of giving the user more control over the visualizations via mouse-clicks and scroll buttons. Another area of improvement could be integrating other options to visualize topic model such as word-clouds and content-lens.

Furthermore, there is a scope of improvement in the software which could be attained by modularizing the backend more by including a messaging service at the backend, so that the software could perform well with the actual real-time data instead of simulations of real-time data. In the future, a version of this software could be developed which could handle the actual real-time time data and then, monitor and analyze the same.

7. Bibliography

- [AAA14] Kholoud Alsmearat, Mahmoud Al-Ayyoub, and Riyad Al-Shalabi, "An extensive study of the bag-of-words approach for gender identification of arabic articles", *11th International Conference on Computer Systems and Applications (AICCSA), IEEE, 2014*, pages 601–608, 2014.
- [BCF+06] M. Baena-Garcia, J. del Campo-Ávila, R. Fidalgo, A. Bifet, R. Gavaldá, R. Morales-Bueno, "Early drift detection method", *Fourth International Workshop on Knowledge Discovery from Data Streams*, 2006.
- [BDH+11] Mike Bostock, Jason Davies, Jeffrey Heer, Vadim Ogievetsky, and community, <https://d3js.org/> , v2.0 August, 2011.
- [BL96] C. Brodley, T. Lane, "Creating and exploiting coverage and Diversity", *AAAI-96 Workshop on Integrating Multiple Learned Models, Portland, OR, 1996*, pages 8-14.
- [Ble12] David M. Blei., "Probabilistic Topic Models", *Communications of the ACM, April 2012*, Vol. 55 No. 4, pages 77-84.
- [BNJ03] David M. Blei, Andrew Y. Ng , Michael I. Jordan, "Latent dirichlet allocation", *The Journal of Machine Learning Research*, 2003, pages993-1022.
- [BTH+13] H. Bosch, D. Thom, F. Heimerl, E. Puttmann, S. Koch, R. Kruger, M. Worner, T. Ertl, "ScatterBlogs2: Real-Time Monitoring of Microblog Messages through User-Guided Filtering", *IEEE Transactions on Visualization & Computer Graphics 19.12 (Dec. 2013)*, pages 2022–2031.
- [BW08] L. Byron and M. Wattenberg, "Stacked graphs—geometry & aesthetics", *Visualization and Computer Graphics, IEEE Transactions*, vol. 14, no. 6, 2008, pages 1245–1252, 2008.
- [CK02] McCallum and Andrew Kachites. "MALLET: A Machine Learning for Language Toolkit", 2002, <http://mallet.cs.umass.edu/> , accessed on 26th July 2018
- [CLK10] J. Choo, H. Lee, J. Kihm, H. Park., "iVisClassifier: An interactive visual analytics system for classification based on supervised dimension reduction", *IEEE Conference on Visual Analytics Science and Technology (VAST)*, pages 27-34, 2010.
- [Cod15] Codophile, <http://codophile.com/2015/05/02/how-to-integrate-r-with-java-using-rserve/> , accessed on 26th July, 2018
- [CRA17] CRAN, <https://cran.r-project.org/web/packages/e1071/index.html>, accessed on 26th July 2018
- [CV95] C. Cortes and V. Vapnik, "Support-vector networks", *Machine Learning*, vol. 20, pages 273-297, 1995
- [GMC+04] J. Gama, P. Medas, G. Castillo, P. Rodrigues, "Learning with drift detection", *Proceedings of the advances in artificial intelligence SBIA 2004: 17th Brazilian symposium on artificial intelligence (Vol. 3171)*, pages 286-295, 2004

- [GZB+14] J. Gama, I. Zliobaite, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation", *ACM Computing Surveys (CSUR)*, 46 (2014), pages 1-37, 2014
- [HHN00] S. Havre, B. Hetzler and L. Nowell. "ThemeRiver: Visualizing Theme Changes over Time", *IEEE Symposium on Information Visualization*, pages 115-123, 2000
- [HKB+12] F. Heimerl, S. Koch, H. Bosch, T. Ertl, "Visual classifier training for text document retrieval." *IEEE Transactions on Visualization and Computer Graphics* 18.12 (2012), pages 2839–2848, 2012
- [HLS13] E. Haddi, X. Liu, Y. Shi, "The role of text pre-processing in sentiment analysis", *Procedia Computer Science*, 17 (2013), pages 26-32, 2013
- [KBT+13] R. Krüger, H. Bosch, D. Thom, E. Püttmann, Q. Han, S. Koch, F. Heimerl, T. Ertl, "Prolix-Visual Prediction Analysis for Box Office Success", 2013
- [KKE+10] D. A. Keim, J. Kohlhammer, G. P. Ellis, and F. Mansmann, "Mastering the Information Age - Solving Problems with Visual Analytics", *Eurographics Association, Goslar*, 2010.
- [KMS+08] D.A. Keim, F. Mansmann, J. Schneidewind, J. Thomas, H.Ziegler, "Visual Analytics: Scope and Challenges", *Springer*, 2008
- [LWL+17] S. Liu, X. Wang, M. Liu, J. Zhu, "Towards better analysis of machine learning models: A visual analytics perspective", *Visual Informatics*, pages 48-56, 2017
- [MBB+11] A. Marcus, M. S. Bernstein, O. Badar, D. R. Karger, S. Madden, R. C. Miller. "Twitinfo: aggregating and visualizing microblogs for event Exploration", *Proc. SIGCHI Conf. Human Factors in Computing Systems, CHI '11, New York, NY, USA. ACM*, pages 227–236, 2011
- [MCS06] Jonathan Milgram, Mohamed Cheriet, Robert Sabourin, "'One Against One' or 'One Against All': Which One is Better for Handwriting Recognition with SVMs?", *Tenth International Workshop on Frontiers in Handwriting Recognition, Oct 2006, La Baule (France), Suvisoft*, 2006.
- [MRG+91] J.D. Mackinlay, G.G. Robertson, S.K. Card, "Perspective wall: Detail and context smoothly integrated" , *Proceedings of SIGCHI'91, 1991*, pages 173-179, 1991
- [PSP+15] J. G. S. Paiva, W. R. Schwartz, H. Pedrini, and R. Minghim, "An approach to supporting incremental visual data classification." *IEEE Transactions on Visualization and Computer Graphics*, vol. 21, no. 1, 2015, pages 4–17, 2015
- [Rib12] Severino Ribeca,
https://datavizcatalogue.com/methods/circle_packing.html , visited on 28th July, 2018
- [SAM82] R. Spence, M. Apperley, "Data base navigation: An office environment for the professional", *Behaviour & Information Technology*, vol. 1, no. 1, pages 43-54, 1982.
- [SB14] S. Shalev-Shwartz, S. Ben-David, "Understanding Machine Learning: From theory to algorithms", Cambridge: Cambridge University Press. Pages 22-24
- [Set11] B. Settles. "Closing the loop: Fast, interactive semi-supervised annotation with queries on features and instances", *Proc. of the Conf. on Empirical Methods in Natural Language Processing, EMNLP'11*, pages 1467–1478, 2011.

-
- [Shn96] Ben Shneiderman, "The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations", *Proceedings of the 1996 IEEE Symposium on Visual Languages*, Page 336, September 03-06, 1996.
- [SOF18] Pivotal Software, <https://spring.io/guides/gs/rest-service/>, accessed on 26th July 2018
- [SPG14] C. D. Stolper, A. Perer, D. Gotz, "Progressive VisualAnalytics: User-Driven Visual Exploration of In-Progress Analytics", *TVCG*, 20(12), 2014.
- [SSG10] C. Seifert, V. Sabol, M. Granitzer, "Classifier hypothesis generation using visual analysis methods", *NDT* (1), pages 98–111, 2010.
- [TC05] James J. Thomas, Kristin A. Cook, "Illuminating the Path: The Research and Development Agenda for Visual Analytics", 2005.
- [TPC+06] A. Tsymbal, M. Pechenizkiy, P. Cunningham, S. Puuronen, "Handling local concept drift with dynamic integration of classifiers: Domain of antibiotic resistance in nosocomial infections", *Proc. 19th IEEE Int. Symp. Comput.-Based Med. Syst.*, pages 679-684, 2006.
- [Urb03] Simon Urbanek, "A Fast Way to Provide R Functionality to Applications"
- [VGC17] A. Vadehra, M.R. Grossman, G.V. Cormack, "Impact of Feature Selection on Micro-Text Classification", *arXiv preprint arXiv:1708.08123* (2017) (cit. on p. 28), 2017.
- [Wai17] University of Waikato, <https://www.cs.waikato.ac.nz/ml/weka/> v3.8.2 (stable), December 22, 2017, accessed on 25th July, 2018.
- [Wel14] Wellecks, <https://wellecks.wordpress.com/2014/09/03/these-are-your-tweets-on-lda-part-i/>, 2014.
- [ZPG16] I. Zliobaite, M. Pechenizkiy, J. Gama, "An overview of concept drift applications", *Big Data Analysis: New Algorithms for a New Society*, Springer (2016), pages 91-114, 2016

Acknowledgement

I am sincerely thankful to my supervisors Dr. Dennis Thom from the University of Stuttgart and Dr. Sebastian Klenk (5Analytics) for their support and guidance during all the phases of my master thesis. I would also like to thank Prof. Dr. Thomas Ertl for giving me this wonderful opportunity to do my master thesis at the Institute of Visualization and Interactive Systems, University of Stuttgart (VISUS).

I am also thankful to my family and friends for their moral support during the tenure of my master thesis.

Shaista Sultana

Declaration

I hereby declare that the work presented in this thesis is entirely my own and that I did not use any other sources and references than the listed ones. I have marked all direct or indirect statements from other sources contained therein as quotations. Neither this work nor significant parts of it were part of another examination procedure. I have not published this work in whole or in part before. The electronic copy is consistent with all submitted copies.

place, date, signature