



**University of
Zurich** ^{UZH}

Towards a BitTorrent Weather Forecast

*Stefan Bublitz
Zurich, Switzerland
Student ID: 10-739-530*

Supervisor: Andri Lareida, Thomas Bocek
Date of Submission: January 08, 2017

Abstract

Over the last years, Neural Networks gained popularity in artificial intelligence as a tool capable of finding complex relationships in data. Originally used to examine biological processes in the human brain, researchers analyzed the application of Neural Networks on classification and regression problems. This thesis applies Neural Networks on BitTorrent usage data in order to predict the popularity of files in BitTorrent networks. Multiple research questions formulated at the beginning of this thesis aiming at the implementation and quality of utilized networks are discussed, tested and evaluated.

Zusammenfassung

Neurale Netzwerke wurden in den letzten Jahren vermehrt für Regressions- und Klassifikationsanalysen verwendet. Dieses gestiegene Interesse beruht unter anderem auf der Allgegenwärtigkeit von Big Data. Ausserdem lassen sich mit Hilfe von Neuralem Netzwerken grosse Datenmengen auswerten und darauf Vorhersagen treffen.

Im Verlauf dieser Arbeit wird eine enorme Datensammlung betreffend dem Verhalten von Benutzern im BitTorrent Netzwerk ausgewertet. Gleichzeitig werden verschiedene Neurale Netzwerkstrukturen analysiert mit dem Ziel die Bekanntheit von Daten im Torrent Netzwerk vorherzusagen. Zusätzlich zu dieser weltweiten Beliebtheit steht die Zerlegung dieser weltweiten in örtliche Bekanntheit im Mittelpunkt. Des weiteren ist die Berechnung der Kosten, die für das Weiterleiten der Transaktionen erzeugt werden, zusammen mit ihrer Ortung ein Anliegen.

Während der Analyse bezüglich der Netzwerkstrukturen fiel auf, dass die Anzahl der Ergebnisvariablen einen entscheidenden Einfluss auf die Komplexität des Netzwerkes aufweist. Die Berechnung der globalen Bekanntheit mit Hilfe Neuraler Netzwerke kann bewältigt werden. Da jedoch die örtlich bezogene Beliebtheit über 238 verschiedene Funktionen verfügt, steht die Vereinfachung dieser Problemstellung im Zentrum.

An Stelle einer Regression stellt sich die Anwendung einer Klassifikationsanalyse als mögliche Alternative heraus. Die Auswertung zeigt, dass das gelernte Modell bessere Klassifikationen ausführt als ein zufälliger Klassifizierer.

Acknowledgments

I want to thank Professor Stiller for providing the opportunity to learn more about BitTorrent, big data and machine learning.

Moreover, I am grateful for the support I received from Andri Lareida whenever I had a question regarding this thesis. Without him, this thesis would have been much more tedious and cumbersome.

Furthermore, I want to thank Andri for providing me with a computer, which I could use to program this thesis. Without that machine, this whole project would not have been possible.

Contents

Abstract	i
Zusammenfassung	iii
Acknowledgments	v
1 Introduction	1
1.1 Motivation	1
1.2 Description of Work	1
1.3 Thesis Outline	2
2 Related Work	3
2.1 Peer - to - peer (P2P) architecture	3
2.2 Machine Learning algorithm	4
3 Weather Forecasting Architecture	7
3.1 Data Processing	7
3.1.1 Remove duplicate data	8
3.1.2 Simplify the time specification	8
3.1.3 Adding torrent specific time	8
3.1.4 Calculate a torrent's popularity	9
3.1.5 Calculate a torrent's expected routing effort	9
3.1.6 Convert text into a number	13
3.2 The Multilayer Perceptron	14

3.2.1	Overall Structure	15
3.2.2	Propagation Process	16
3.2.3	Choosing an activation function	17
3.2.4	Learning a Network	17
3.2.5	Early Stopping Rules	18
3.2.6	Prevent Overfitting	19
3.2.7	Moving Window	20
3.3	Evaluation	21
3.3.1	Natural Language Processing	21
3.3.2	Neural Networks	21
4	Implementation	25
4.1	Overall architecture	25
4.1.1	Data Storage	25
4.1.2	Data Creation	27
4.1.3	Network Analysis	28
4.1.4	Data Retrieval	29
4.2	Algorithm Implementation	30
4.2.1	Data Processing	30
4.3	Learning the Neural Network	32
4.4	Evaluation	32
4.4.1	Console Plots	33
4.4.2	Interactive Application	33

5	Evaluation	37
5.1	Data Quality	37
5.2	Routing Algorithm	37
5.3	Evaluate Word2Vec	38
5.4	Evaluate Regression	38
5.4.1	Early Stopping Criteria	39
5.4.2	Moving Window	39
5.5	Evaluate Classification	40
6	Further Work	47
6.1	Merge Similar Movies	47
6.2	Learn Neural Networks on a Cluster	47
6.3	Network Analysis	48
7	Conclusion	51
7.1	Summary	51
7.2	Research Questions	51
7.2.1	How can the popularity of individual movies over time be determined?	51
7.2.2	Given a certain routing protocol, how can the costs for transporting traffic be calculated?	52
7.2.3	How can Machine Learning algorithms predict popularity over time?	52
7.2.4	What accuracy can be achieved by the chosen Machine Learning algorithm?	52
7.3	Final Statement	53
	Acronyms	59
	List of Figures	60
	List of Tables	63
A	Databases	67

B Plots	69
----------------	-----------

C Contents of the CD	71
-----------------------------	-----------

Chapter 1

Introduction

Over the past years, the Internet as an instrument to exchange information increased according to the latest analysis conducted by Cisco in 2016 [22]. Furthermore, the writers predict an increase in annual global Internet Protocol (IP) traffic from 1.1 zetabytes in 2016 to 2.3 zetabytes by 2020. As a major source creating these data streams, the authors identify P2P applications. P2P applications utilize direct connections between users working with the program to exchange and spread information. These direct connections are less likely to suffer from the traditional bottleneck problem introduced by the server - client architecture [4]. However, since connections between peers are independent from the physical network required to direct the information, expenses due to routing can increase noticeably. Therefore, analyzing these traffic patterns and even trying to predict these can provide useful information to optimize the underlying routing protocols. As a reference architecture, the content distribution application BitTorrent which utilizes direct P2P connections to share files among participating individuals, is being analyzed.

1.1 Motivation

The effort Internet Service Providers (ISPs) spend on trying to reduce cross ISP P2P traffic increased over the past years. This development is the reason why analyzing and visualizing the gap between the revenue ISPs receive from customers together with the costs they have to spend to route traffic can reveal more suitable routes.

Furthermore, predicting a file's popularity over time can give valuable insights on the IP traffic which will have to be transmitted by responsible ISPs.

1.2 Description of Work

This Master Thesis provides a framework to create, manipulate, process and visualize data from various sources. The following work covers handling records from the Video Consumption in Overlay Networks (VIOLA) data set and proposes various algorithms to make

these entries compatible with Machine Learning algorithms such as Neural Networks[30]. Moreover, it analyzes different approaches to predict user behavior within the BitTorrent network. Applying these methods, this work strives to forecast countries contributing the majority of peers together with those nations having to route the arising traffic. In order to achieve this objective, enriching the dataset with additional information to allow for a more in depth network analysis is part of this essay. At last, visualizing the results from this analysis together with an evaluation of the suggested mechanisms is included. Specifically, in relation to the .torrent network, the following questions shall be answered in this thesis:

- How can the popularity of individual movies over time be determined?
- Given a certain routing protocol, how can the costs for transporting traffic be calculated?
- How can Machine Learning algorithms predict localized popularity over time?
- What accuracy can be achieved by the chosen Machine Learning algorithm?

The next section expands on the structure of this thesis, elaborating on the content each article is going to handle.

1.3 Thesis Outline

This work is divided into four chapters. The succeeding chapter is going to analyze and reflect on recent papers about the P2P architecture and its influence on the global IP traffic. Moreover, the origin and selection method of the different data sets, allowing the analysis and prediction of behavioral patterns are explained. Furthermore, Neural Networks are identified as the main device to predict user behavior. Further papers give insights into the wide range of parameters and influencing factors, which have to be taken into account when calculating predictions using Neural Networks. At last, evaluation methods to assess the Neural Network's potential to make applicable prognoses are being studied.

The ensuing section elaborates on the approach used to apply Neural Networks on the VIOLA data set. After explaining and analyzing the different methods necessary to preprocess the data, design decisions regarding the structure of the utilized Neural Networks are explained. Furthermore, the performance and quality of various Network structures are assessed.

The fourth chapter explains the architecture of the program used to calculate the results of this thesis. Besides going into detail on how to set the application up to run on other computers, it clarifies the benefit of individual components and elaborates on how to use them independently. Finally, it explains how to expand the application to include more parameters like other routing protocols or to influence the quality of the prediction.

Chapter 2

Related Work

This section covers useful literature for understanding content of research areas comprised in this thesis. After providing the fundamentals regarding P2P architecture, the benefit of discovering heavily used routing paths is more closely elucidated. Afterwards, the process of collecting data useful for determining mentioned paths is analyzed. At last, important machine learning algorithms are described and their usefulness in relation to predicting statistics analyzed.

2.1 P2P architecture

In recent years, many publications addressed the architecture of P2P applications and its performance compared to the traditional server - client architecture. P2P applications have their origin in the USENET application, which was developed in 1979 [8]. However, P2P applications began to become popular not until 1999 with Napster, a music sharing network [46]. While the range of tasks P2P applications can address extended over time including implementations addressing Distributed Hashtable [45, 21, 38], grid computing [2], memory [19] and Voice over IP (VoIP) [5], file sharing remains popular among P2P applications [7].

With this growth in usage, ISPs have to cope with increased costs for routing the resulting traffic [12, 36]. [20] mentioned, that more than half of the ISP backbone traffic is related to content delivery. On the one hand, the popularity of Content Distribution Networks (CDNs) is viable for ISPs because participants in an interview stated that using CDNs is one of the main reasons for upgrading the download bandwidth [34]. However, increase in usage also creates additional challenges e.g. possible bypassing of load balancing algorithms or the increased difficulty to estimate traffic matrices [41]. Hence, various authors addressed the reduction of tier-1 traffic while keeping the performance P2P delivery offers. [31] showed that cross ISP traffic can be reduced by using the content locality.

To retrieve a specific file when using a P2P file sharing application, the user has to retrieve a .torrent file, which provides information on the target file to be loaded and other peers currently sharing the file. The user can then decide which P2P application

to use [49]. While the individual applications use the same network to share files, the applied strategy to achieve the exchange of files slightly differs from each other. Fields in which they diverge include characteristics to decide which user to unchoke and the total number of peers to directly connect to [25, 40, 51]. Pursuing the rarest piece first policy together with randomly unchoking peers, the BitTorrent client imposes the most general strategy and is therefore considered to be the reference client for the remainder of this thesis.

By introducing peer to peer connections, the length of the routing path as well as the individual participants responsible for transmitting the messages can vary widely. This transition is based on the independence of the P2P network from the underlying physical grid. Aforementioned separation makes room for a wide array of routing protocols often trying to improve performance like the quality of the connection or the time required for searching [54, 18, 37].

With growing use of P2P applications, ISPs determined that up to 60% of the backbone traffic was P2P related [31]. In order to reduce this amount, ISPs decided to block or rate - limit P2P related packets, leading to an arms race with individual developers trying to evade these restrictions [10, 11]. This circumstance lead to the research for “ISP - friendly” P2P designs, which are an integral part of this thesis.

In general, instead of randomly unchoking peers, “ISP - friendly” protocols use locality information to decide, which peers to unchoke in order to reduce cross - ISP traffic. One such protocol is the biased neighbor selection, which mainly unchokes peers within the same ISP. In order to achieve this, a modified client, tracker or P2P traffic shaping devices have to be employed [6]. Another option is the Ono system, which compares the routing behavior of individual peers to determine whether users are physically close to one another [9]. At last, the Proactive Provider Participation for P2P (P4P) approach allows ISPs to implement their own tracker, called *iTrackers*, which communicate with both participants - trackers and peers. These *iTrackers* provide the distance between peers and other capabilities of the network provider [27].

Further support can be offered by predicting the number peers currently sharing a file together with their location. This information can then be used to identify routes, which have a high expected congestion and provide alternative routes instead. The next section focuses on algorithms suitable for predicting functions.

2.2 Machine Learning algorithm

Machine Learning is a broad statistical field with the purpose to make prediction based on existing data. These algorithms can be divided into supervised and unsupervised. Machine learning algorithms can be divided into supervised and unsupervised algorithms, differing in the data they require for learning.

A mechanism for regression or classification tasks which has been developed in 1943 are Neural Networks [50]. Neural Networks have already been applied to a wide range of applications like data mining for rule extraction ([32]) or stock market price prediction

([35]). In recent years, however, these networks became popular because of the growing importance of big data. Successfully applying Neural Networks on big data has been the content of many papers, e.g., Natural Language Processing (NLP) [26] or deep learning [24].

Unlike the aforementioned applications combining big data and Neural Networks, this thesis is going to learn and evaluate the networks on a single machine. The architecture of the program responsible for executing this algorithm is explained in the subsequent chapter.

Chapter 3

Weather Forecasting Architecture

This chapter explains the algorithms utilized to answer research questions stated in the introduction. After giving a broad overview over the application’s architecture, tools used to prepare available records are described. Afterwards, design considerations regarding the implementation of Neural Networks are reviewed in detail.

The Weather Forecasting algorithm can be divided into the data processing, the machine learning and the visualization element. The processing component is responsible for retrieving data from external sources and improving its quality. The learning segment uses this data to build an algorithm capable to predict popularity for movies. At last, the visualizing interface uses records from both processing and learning components to illustrate them for the end user. The succeeding section describes the design of each component before chapter 4 takes a closer look at the implementation of these features.

3.1 Data Processing

Any machine learning algorithm requires data to train an algorithm. More precisely, predicting the popularity of content within a file sharing network requires information on attributes identifying shared files together with statistics to be predicted for new inputs.

The VIOLA project recorded user download and upload behavior over the course of three months. The records were collected by repeatedly requesting different trackers for peers sharing files in 20 minute intervals. By submitting multiple requests in a row, the probability to identify all users within a swarm was increased. The resulting data of these requests including the individual IP addresses, which file was shared, the current timestamp together with additional locality information retrieved by analyzing the IP address were stored in a database.

In order to provide more details on movies shared within the individual swarms, additional data was crawled from websites like Internet Movie Database (IMDB) or Rotten Tomatoes [23, 14]. The complete structure of the VIOLA data set is shown in figure A.1 which can be found in the appendix.

$$\text{IntervalID} = \left\lfloor \frac{\text{minutes}}{20} \right\rfloor \quad (3.1)$$

Figure 3.1: Calculating the intervalID given a partial timestamp.

The subsequent sections elaborate on the various data processing methods applied on these records necessary to utilize them for learning Neural Networks.

3.1.1 Remove duplicate data

Due to repeatedly retrieving peers from trackers, the aforementioned announce table can contain duplicate IP address entries within a 20 minute interval. Since this data only confirms that the user is still sharing the file together with the relaxation, that 20 minute intervals are sufficient for this purpose, these entries can be removed.

3.1.2 Simplify the time specification

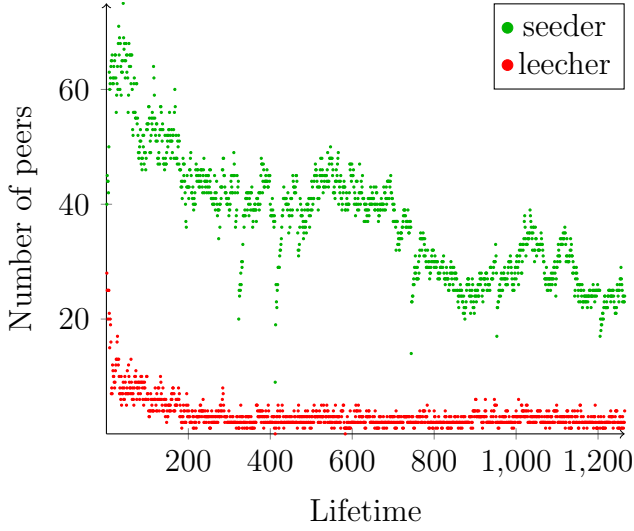
The provided timestamp is an 8-byte number identifying a date exactly onto the millisecond. However, for the purpose of analyzing the popularity of files throughout time, it is sufficient to know to which 20 minute interval each record belongs. In order to assign each record to a request, the timestamp is initially increased by 5 minutes. Since the first request started at the 01.05.2016 at 00:05:02 o'clock, reducing the timestamp by this amount aligns requests with the full hour, necessary for the following approach. Continuing to transform the number into a timestamp and using the *minutes* fraction of the result, formula 3.1 depicts how the *intervalid* describing the 20 minute range within an hour can be calculated.

Using the combination of the attributes *year*, *month*, *day*, *hour* along with the newly created *intervalid* provides an identifier for the request for each record in the data set.

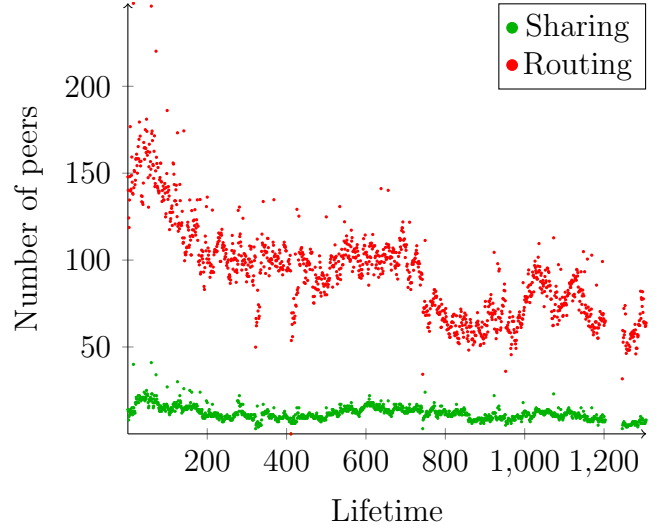
3.1.3 Adding torrent specific time

While the absolute time is useful for identifying seasonal patterns or retrieving a swarm for a specific movie at a particular moment, a file specific time is required. This figure is an index starting at 1 describing the first request where a peer shared this file. Each additional request increments this statistic by one, whether the request contained a user distributing this file or not.

Retrieving the maximum value for this newly created attribute *time* and counting the appearances of each of these values gives an insight on the lifetime of torrents in this CDN. This search reveals, that the shortest running time only contains 1 request, while the longest spans 6623 requests. Analyzing the distribution shows, that a large share of torrents observed by VIOLA span the entire observed time. In order to ensure, that the



(a) The movie's popularity from a global perspective.



(b) The movie's popularity for all AS located in the United States.

Figure 3.2: While both plots illustrate the popularity of a specific torrent, figure 3.2a depicts the global popularity using the *seeder* and *leecher* statistic. 3.2b details the localized popularity of the same torrent for the United States.

whole lifetime of a movie is monitored, all files recorded on the first and last day are not included in this analysis.

3.1.4 Calculate a torrent's popularity

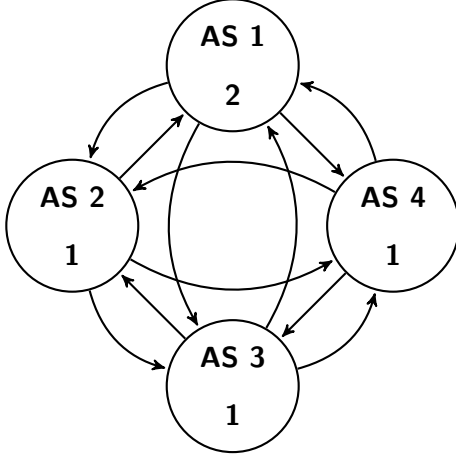
Popularity within a CDN is bound to a specific file and can change over time. The global popularity can be calculated by retrieving the *seeder* and *leecher* statistic contained in the VIOLA data set. In order to get an insight on the popularity of a movie for each Autonomous System (AS), counting the distinct *PeerUID* addresses within each AS for each request and movie is necessary. In order to analyze the locality of the popularity, all peers from an AS within a country are added up. This leads to a localized popularity statistic for each country having at least one peer belonging to an AS.

Figure 3.2 shows two plots depicting the resulting popularity for the torrent titled “The Dead Files S08E08 You Will Be Mine HDTV x264-W4F[rartv]” over its lifetime.

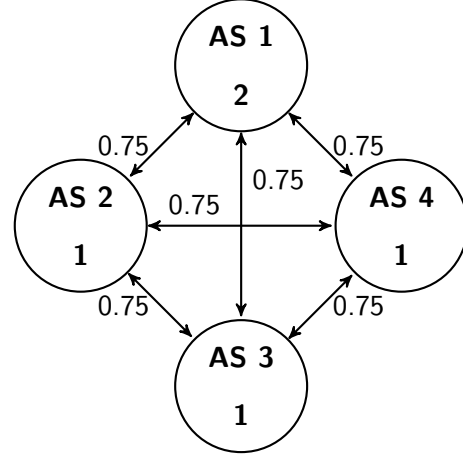
The localized popularity statistic depicted in 3.2b also contains the routing effort shouldered by the United States. The next sections explain the approach for its calculation.

3.1.5 Calculate a torrent's expected routing effort

Calculating the expected routing costs caused by the popularity is slightly more involved and is dependent on the client used for retrieving and sharing the file. As already mentioned, different torrent applications implement various strategies followed when sharing



(a) Explicit notation of connections between peers. A peer's decision which other peer to unchoke is independent among each other.



(b) Same scenario as in 3.5a, now including probabilities that a connection is unchoked.

Figure 3.3: Network showing connections between ASes sharing a file. Additionally to the AS identifier the number of IP addresses from that AS are denoted, i.e. two users belong to AS 1, while ASes 2, 3 and 4 have one peer each. All other ASes do not have a peer sharing this file and are therefore not in this graph.

a file. Important when considering the routing effort is the number of connections a client establishes for uploading content. BitTorrent allocates four uploads slot, where three peers are chosen according to the highest bandwidth they offer in return, while the last one is allocated randomly. In order to simplify assigning these channels to users, all four are assigned randomly.

In order to calculate the expected costs for routing, the probability of a specific connection being unchoked is required. The network in figure 3.5a displays a simplified version of a swarm sharing a file. Each node identifies an AS, while an edge between two ASes signifies that both ASes have at least one user belonging to this swarm. Since this graph only contains ASes from the same swarm, the resulting network is a Meshed network.

Calculating the probability of a relation being unchoked can be achieved by first computing its complement. In order to theoretically represent a graph, the set of m peers $n_1, n_2, \dots, n_m$ belonging to the same swarm is denoted with N . Furthermore, the set of links $l_{12}, l_{13}, \dots, l_{1m}, l_{21}, \dots, l_{nm}$ connecting peers within a swarm is depicted with L . l_{ij} indicates the edge between node i and j . Since a swarm creates a Meshed network, node n_i can choose from $m - 1$ links when choosing to upload to another peer. The probability, that an edge stays choked in a sequence of three unchoking decisions is therefore:

$$\begin{aligned} Pr(\text{Not Unchoke}) &= \left(\frac{n-1}{n} \cdot \frac{n-2}{n-1} \cdot \frac{n-3}{n-2} \right) \\ &= \left(\frac{n-3}{n} \right) \end{aligned} \tag{3.2}$$

Using the previous calculated result, the probability that an edge is unchoked within three unchoking decisions is therefore:

$$\begin{aligned}
 Pr(\text{Unchoke}) &= 1 - Pr(\text{Not Unchoke}) \\
 &= 1 - \left(\frac{n-3}{n} \right) \\
 &= \frac{3}{n}
 \end{aligned} \tag{3.3}$$

In order to be able to include other Torrent clients than BitTorrent, the generalization of the aforementioned equation is helpful. Considering o successive decisions, the following formula can be utilized:

$$\begin{aligned}
 Pr(\text{Unchoke}) &= 1 - Pr(\text{Not Unchoke}) \\
 &= 1 - \left(\frac{n-1}{n} \cdot \frac{n-2}{n-1} \cdot \dots \cdot \frac{n-o}{n-o-1} \right) \\
 &= 1 - \left(\frac{n-o}{n} \right) \\
 &= \frac{o}{n}
 \end{aligned} \tag{3.4}$$

Returning to the exemplary swarm network from figure 3.5a, the probability a connection is being unchoked, given $n = 4$ nodes in the swarm is the following:

$$\begin{aligned}
 Pr(\text{Unchoke}) &= 1 - Pr(\text{Not Unchoke}) \\
 &= 1 - \left(\frac{4-3}{4} \right) \\
 &= \frac{3}{4}
 \end{aligned} \tag{3.5}$$

After calculating these probabilities, the actual routing paths have to be identified in order to be able to draw conclusions on heavily used links. This statistic requires additional data as these relationships cannot be inferred using records from the VIOLA data set.

The Center for Applied Internet Data Analysis (CAIDA) is a department at the University of California's San Diego Supercomputer Center. They collect and analyze topological data regarding the Internet. One data set of interest to this thesis is the identification of AS relationships [15]. Updated monthly, it contains relations between AS, including the classification of their contact. AS links are either labeled as P2P, in which case both ASes assure to route the traffic of the other party without any monetary returns. If the relation is not P2P, then a contract defines this path to be a Customer - to - Provider (C2P) connection. In this case, the provider routs traffic from the customer AS, where the customer has to pay a fee for each transaction. As an examples, Figure 3.4 shows a possible hierarchical order of ASes.

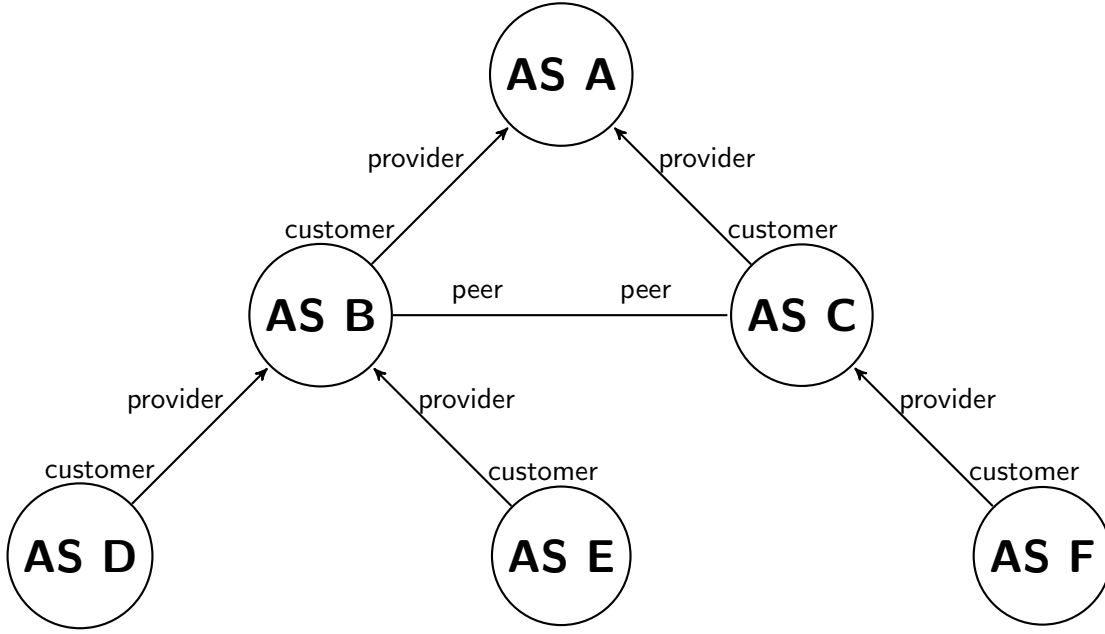


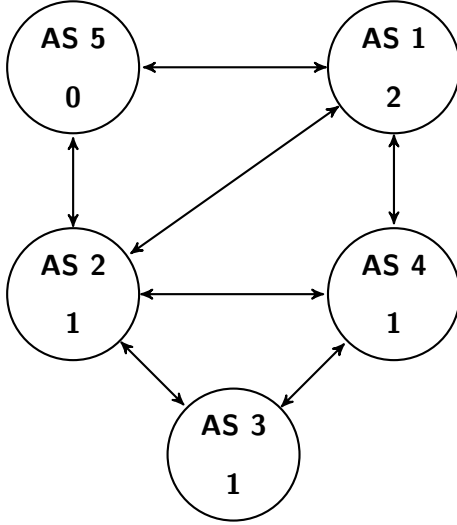
Figure 3.4: Relations between different ASes. Each edge identifies a cashflow between connected nodes, where the customer pays the provider for routing traffic. Peers on the same hierarchical level do not exchange money for transmitting information. (based on [15])

The CAIDA data collection contains 77290 different ASes, with 464475 connections linking them where 35444 of these relations are P2P connections. A.2 illustrates a full Entity Relationship Diagram (ERD) depicting tables used from the CAIDA data set including relationships among them.

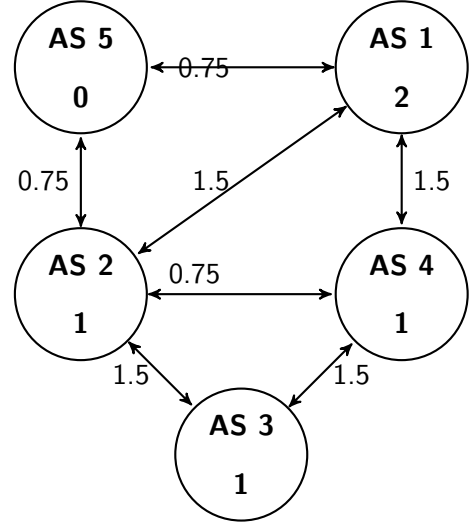
Since both data sets, VIOLA and the aforementioned CAIDA records use the same identifier to describe ASes, combining both reveals new paths between individual peers. In order to simplify finding admissible routing paths, the directed cashflow edges are assumed to be possible routing paths. Further relaxing this characteristic, edges indicate that traffic can be routed in both directions instead of only from consumer to provider. Furthermore, P2P connections are only utilized, if source and target of a transmission share the P2P link.

Having access to transmitting paths through the Internet, ASes no longer have to share a direct connection. Furthermore, there can exist multiple possible routes between two ASes. Deciding which path to consider when identifying ASes responsible for transmitting traffic requires applying a routing algorithm.

This thesis uses the shortest path algorithm to identify routes between peers [43]. Given a network G with vertices V and edges E , the shortest path algorithm can be applied to identify the path with the shortest distance between any two nodes within V . In a network with unweighted edges, the distance between two nodes is equal to the number of nodes which have to be traversed when going along the path. Using the shortest path to simulate routing paths within the networks constitutes the best case scenario as the costs for routing in the network are minimized. In case multiple shortest paths exist, the expected cost is distributed evenly among all included ASes.



(a) Network showing AS relationships. AS 5 is an additional AS from the CAIDA records necessary, since it lies on the shortest path between AS 1 and 2. All other ASes share a direct link.



(b) Same network as in 3.5a. The statistics denoted at the edges imply the expected load created by sharing a file of size 1.

Figure 3.5: Networks presenting the relationship between ASes from the CAIDA and the VIOLA data set.

The size of a file to be shared is assumed to be one, allowing the summation of probabilities, if an edge is used by multiple shortest paths. Figure 3.3 gives an example of the resulting costs for routing.

3.1.6 Convert text into a number

Machine Learning algorithms are implemented using a set of mathematical operations to achieve a predefined goal. On this account, these algorithms require the transformation of textual data into numerical form. While target variables describing popularity of a file are all numerical, some of the attributes characterizing these files are only available in text.

Various algorithms have been developed addressing the conversion of textual features into a numeric representation. In the following sections, the Bag of words as well as the Word2Vec algorithm are described in more detail.

Bag of words: [53] Before applying this algorithm, determining the vocabulary of the considered data set is necessary. Denoting the set of verses to be represented numerically with W , the individual words used in W build the vocabulary V . Each output generated by the Bag of words algorithm is now an integer vector of length $|V|$. The entries in the output vector correspond to the vocabulary in the order they appear in the underlying verses. Entries are 0, if a given word does not appear in the associated text. Otherwise, the item is equal to the number of times the word appears in the verse. Refer to table 3.1 for an example of this algorithm.

Verse	Bag of Words
John likes to watch movies. Mary likes movies too.	[1, 2, 1, 1, 2, 0, 0, 0, 1, 1]
John also likes to watch football games.	[1, 1, 1, 1, 0, 1, 1, 1, 0, 0]

Table 3.1: An example of applying the Bag of words algorithms to a set of verses. The second entry of the resulting vector identifies the word "likes", which appears once in the first and twice in the second sentence respectively.

This vector can then be used as an input for Machine Learning algorithms.

Word2Vec: [48] The Word2Vec algorithm is a Natural Language Processing algorithm which allows to represent a sentence with a numerical vector of length n . Unlike the aforementioned Bag of words method, the dimension of the resulting vector does not have to be equal to the size of the utilized vocabulary, but can adopt an arbitrary size. The algorithm uses a Neural Network, where the Bag of words model is used to calculate values of the input layers. Moreover, it implements 2 hidden layers, adding the feature to account for order of words within sentences. By using an output layer of length n , the outcome of the algorithm can be compressed to any length.

The drawback of this algorithm is, that it does not account for the frequency with which words are used. In order to account for duplicate words within a phrase as well as to affect a vector depending on a word's frequency over all phrases, Term Frequency (TF) / Inverse Document Frequency (TF) are being calculated. Both are defined as follows:

$$\begin{aligned}
 tf(t, d) &= f_{t,d} \\
 idf(t, D) &= \log \left(\frac{N}{|\{d \in D : t \in d\}|} \right) \\
 \text{where} & \\
 f_{t,d} &: \text{raw frequency of a term } t \text{ in document } d \\
 N &: \text{Total number of texts in the data set } N = |D| \\
 |\{d \in D : t \in d\}| &: \text{Number of documents where } t \text{ appears}
 \end{aligned} \tag{3.6}$$

Multiplying both statistics with the result achieved by running the word2vec algorithm results in a vector of the same length, now incorporating TF and TF.

The next section expands on applying Neural Networks on a data set in order to approximate existing and predict novel functions.

3.2 The Multilayer Perceptron

The key benefit provided by Neural Networks is the capability to approximate complex functions which can then be used for further prediction. The Multilayer Perceptron (MLP) is a popular Neural Network for regression and classification tasks [13]. In the following

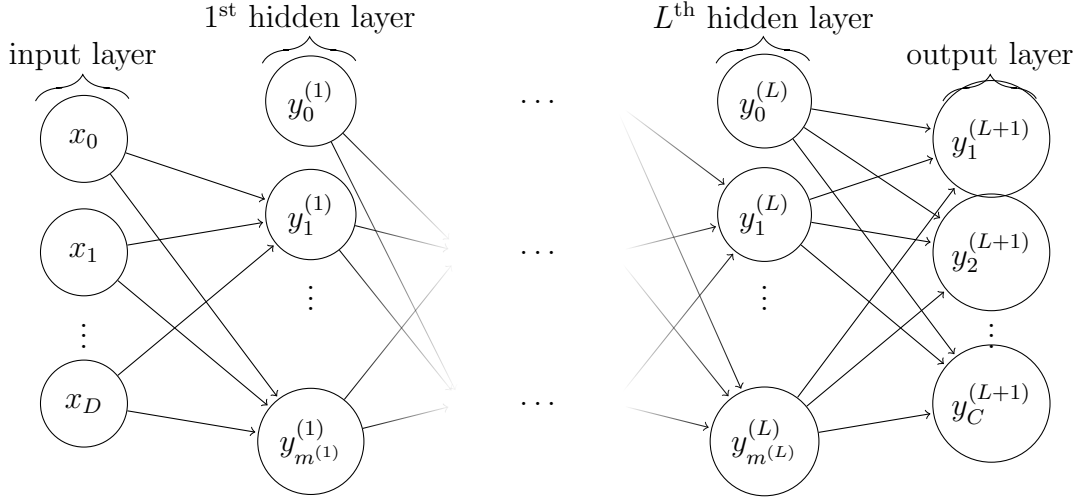


Figure 3.6: Network graph of a $(L + 1)$ -layer perceptron with D input units and C output units. The l^{th} hidden layer contains $m^{(l)}$ hidden units.

section, the overall structure of the MLP is described. In subsequent paragraphs, additional considerations which have to be taken into account when building Neural Networks are analyzed. Moreover, the process of learning and applying Neural Networks is explained.

Over the past years, different types of Neural Networks have been developed. While all networks have in common to estimate a function, they differ in their objective for prediction. In the next sections two kinds of Networks are described in more detail: The MLP [13] and the Long short-term memory Recurrent Neural Network [44].

3.2.1 Overall Structure

The MLP is structured in layers, comprising at least input and output layers. Layers are aligned successively. While nodes within layers are independent, vertices of consecutive layers are fully connected. Layers between the input and output layer are called hidden layers. Implementing 2 or more hidden layers enable the network to calculate deep learning problems as this network structure facilitates detecting complex dependencies between input and output nodes. Consult figure 3.6 which shows the structure of a MLP including up to L hidden layers.

The inputs of any MLP denoted with x_i , where $i \in [0, D]$ are numbers in an unbound range $[-\infty, \infty]$ and identify properties of statistics to be predicted. Values for the C nodes within the output layer have to be calculated. Necessary for calculating output values, each edge in the network has an associated weight $w \in [0, 1]$, which is initialized randomly when creating the network.

Considering the structure of Neural Networks, hidden layers are not mandatory as long as the output is linearly dependent on the input. Otherwise, the network will not be able to find a suitable approximation. Implementing one hidden layer allows the function to

approximate function of dimension 2 and even under dimension 3 under specific circumstances [16]. In order to be able to approximate any polynomial, [1] showed that two hidden layers suffice. More than 2 layers are generally required when Neural Networks are required to classify images.

Since popularity statistics over time are polynomials, two layers are enough to approximate this function. However, the number of nodes within both layers remains to be defined. A popular approach sets linearly decreasing nodes from input to output layers. Afterwards, nodes are evenly increased until the quality of the network does not improve.

This thesis implements the backward propagation algorithm, which updates edge weights according to the occurred error between calculated prediction by the forward propagation and true output values. The next section describes the propagation process used when learning a Neural Network.

3.2.2 Propagation Process

The propagation process can be divided into two phases: The procedure starts with the Forward propagation which uses provided inputs to calculate one output value for each node in the output layer. In order to improve the quality of the networks, the following backward propagation stage updates edge weights in the network depending on the quality of the outputs calculated by the forward propagation. Hereinafter, both stages are being described.

Forward Propagation: This process contains actions to calculate values for nodes in the output layer for provided inputs. Forward propagation is an iterative approach starting with the first hidden layer going through the output layer. Nodes in each layer calculate the weighted average of values from the preceding layer given the weights of the connecting edges. Afterwards, an activation function is used to normalize the value, which is then considered by nodes within the succeeding layer. Examples for activation functions include the Sigmoid and the Softplus function which are defined as follows:

$$\begin{aligned} \text{Sigmoid: } f(x) &= \frac{1}{1 + e^{-x}} \\ \text{Softplus: } f(x) &= \ln(1 + e^x) \end{aligned} \tag{3.7}$$

Figure 3.7 illustrates this process graphically.

Backward propagation: This procedure starts with calculating the deviation between the calculated output by the network and the real statistics defined by the data. Knowing this difference, the backward propagation algorithm relies on running the Forward propagation algorithm in reverse. The assumption is, that calculating the contribution each edge weight provides to the discrepancy and updating it accordingly, the quality of the overall network improves. In order to be able to execute this algorithm, the derivations of the aforementioned activation functions have to be known:

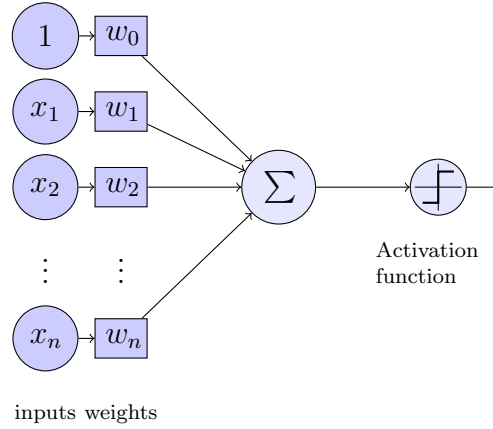


Figure 3.7: Process depicting the operation an individual node within a Neural Network executes.

$$\begin{aligned} \text{Sigmoid: } f'(x) &= f(x) \cdot (1 - f(x)) \\ \text{Softplus: } f'(x) &= \frac{1}{1 + e^{-x}} \end{aligned} \tag{3.8}$$

After dividing current edge weights with the result retrieved by running the derivative, the loop between forward and backward propagation starts again. Since edge weights have been updated, the calculated output is likely to differ from the last iteration, leading to new improvements regarding the weights.

3.2.3 Choosing an activation function

When deciding which activation to employ in a Neural Network, the range of the calculated output has to be considered. While the aforementioned Softplus activation function maintains the range of $[-\infty, \infty]$ for output values, the Sigmoid function binds the output between $[0, 1]$. While this property has no negative effect when applied in hidden layers, selecting an appropriate activation function for the output layer is more crucial. Calculating the achieved error between the determined output and real data requires both statistics to lie in the same area. Therefore, if attributes to be predicted have a range other than $[0, 1]$, the calculate output has either to be scale accordingly or a different activation function like Softplus has to be applied.

3.2.4 Learning a Network

As already mentioned, learning a network requires data describing the functions to be approximated. In order to be able to assess the quality of the model during and after the learning process, the records are divided into 3 sets: The training, evaluation and test set. While the training data set is used to learn the model, the evaluation tuples can be utilized as an indicator regarding the quality. The testing records are not involved in

learning or evaluating the model and therefore are well suited for giving an insight into the forecast quality provided by the model [52].

Updating weights using forward and backward propagation using a subset of records of the training data set is called an iteration. The actual number of tuples to be used per iteration is defined by the batch size and can range from 1 to the number of tuples in the training data set. While using the whole data set in one iteration is the fastest solution, it also requires the most amount of Random Access Memory (RAM) and the time required to reach convergence increases [33]. This thesis therefore chooses to compare the resulting network using different batch sizes to find a good trade off.

After a specific record is used for training, it cannot be used again before the model has been trained on all other training tuples. After propagating the whole data set through the network, one *epoch* has been performed. Depending on the number of elapsed epochs as well as the quality of the model, the learning process can be stopped and tested. To prevent over priced running times, this theses performs 100.000 epochs. Even if converges has not been reached until then, the learning process is stopped. Since a large number of epoch can also lead to overfitting the Neural Network, the next section goes into detail on early stopping algorithms before the succeeding paragraph analyzes other mechanism to prohibit the Network from overfitting.

3.2.5 Early Stopping Rules

Early stopping refers to stop the learning phase of a model because it starts to overfit. Overfitting refers to the case where the training error decreases with consecutive epochs, while the validation error increases. This development signalizes, that the function provided by the data has been approximated too well, because data from the validation set does not confirm the learned function. The reason training and evaluation data depict different functions is the presence of noise in the data, meaning that observations in the data set do not have to lie on the true function to be approximated.

Multiple early stopping criteria help to identify the moment the Neural Network starts to overfit, suggesting to stop further learning [29]. One example is called the *training progress* and is defined by training and validation error.

Before this criteria can be calculated, the process of learning a network is slightly adapted. Instead of considering each epoch separately, strips of epochs of length k are introduced. Comparing training and validation error to terminate the learning phase are only calculated after the end of a strip.

In order to be able to compare the training with the validation error, three helping statistics have to be defined. The first one is the lowest validation error obtained up to epoch t and is defined as follows:

$$E_{opt}(t) = \min_{t' \leq t} E_{va}(t') \quad (3.9)$$

Using $E_{opt}(t)$, the generalization loss can be defined as the relative increase in the validation error compared to the minimum obtained so far Formula 3.10 depicts its definition.

$$GL_{opt}(t) = 100 \cdot \left(\frac{E_{va}(t)}{E_{opt}(t)} - 1 \right) \quad (3.10)$$

The third statistic is the *training progress* and the only one to consider the training error instead of the validation error. Defined on the entire strip of length k , this variable is defined as follows:

$$P_k(t) = 1000 \cdot \left(\frac{\sum_{t'=t-k+1}^t E_{tr}(t')}{k \cdot \min_{t'=t-k+1}^t E_{tr}(t')} - 1 \right) \quad (3.11)$$

This first stopping criteria is the quotient of generalization loss and training progress and defined like this:

$$PQ_\alpha = \frac{GL_{opt}(t)}{P_k(t)} > \alpha \quad (3.12)$$

This stopping criteria facilitates stopping very early as soon as the validation error increases. While this is suitable for cases, where data is easily overfitted, robust data may recover from this development and still converge. For these cases, this second type of stopping criteria is useful. Only considering the validation error achieved in the last epoch of a strip, this criteria stops, when the validation error increases over s successive strips. While consecutive epochs might exhibit increasing validation errors, it is less likely, that successive end of strip epoch do the same. Therefore, if this occurs, overfitting is assumed.

In order to determine, whether overfitting is an issue with regards to VIOLA records, this thesis compares various stopping criteria to determine a fitting one.

Besides early stopping, there exist other tools to prevent a Neural Network from overfitting. The next section goes into detail on two alternatives: crossfold validation and the drop out method.

3.2.6 Prevent Overfitting

Crossfold validation is a tool often used to prevent a Machine Learning algorithm from overfitting [42]. Crossfold validation implements a new strategy when splitting a data set into learning and evaluation proportion. While the testing data part is still defined in the same fashion as before, the remaining data is divided into k folds about the same size. The Machine Learning algorithm is then executed k times, where $k - 1$ folds are used for training and the k^{th} fold for validation. By varying the fold used for validation each iteration, the whole data set can be used to verify the quality of the algorithm.

Applying crossfold validation on Neural Networks can be achieved in a similar way. However, the already lengthy process of learning a network is increased k times. [17] proposed an approach where instead of a single PC, a cluster of computers is used to learn the network. After splitting the data set into folds, each computer uses a different combination of folds for learning and evaluation, similar to the aforementioned crossfold validation. At the end of an epoch strip, learning and evaluation errors over all PCs are averaged and early stopping criteria applied.

This thesis applies this approach in a slightly adjusted implementation. Instead of utilizing a whole cluster, records are distributed evenly on multiple cores available to a single machine. This scenario is compared to applying all cores to learning the whole data set. While this alternative omits crossfold validation, the planning process coming with parallelizing is also dropped. The goal behind this comparison is to find a good trade off between the time necessary for learning a networks and the quality of the achieved result.

[39] found another approach to reduce overfitting. The dropout method changes the structure of the Neural Network randomly to facilitate including nodes not noticed beforehand. By either dropping out nodes by setting weights of incoming edges to 0 or by scaling incoming weights by a constant factor $w \in [0, 1)$, dropout can be implemented without having to change the propagation algorithm.

While applying the aforementioned tools against overfitting aim to increase the quality of the resulting estimator, other algorithms exists also aiming at this improvement. The next section explains applying a moving window to enhance the value of the model.

3.2.7 Moving Window

Besides applying Neural Networks to approximate a function and predict on the same range, Neural Networks can be used to predict the future course of a function. The requirement for this algorithm to be able to make useful prediction is that the function exhibits periodic behavior. Moreover, the target variables have to be dependent on their past window.

Considering popularity of movies, while periodic behavior is not predominantly available, the intuition suggests that movies which were popular in the recent past will also be popular in the near future. Regarding the size of the window, i.e., the number of past requests considered as input variables is going to be content of the evaluation.

In order to be able to compare various Neural Network configurations, evaluation of the resulting models is necessary. The succeeding section is going to explain different algorithms to assess the quality of Machine Learning algorithms, more precisely of Neural Networks.

3.3 Evaluation

The evaluation phase can be divided into several levels. While the natural language processing algorithms need to be evaluated according to the informative value when comparing multiple inputs, visualizing routing paths give an overview over heavily used routes. At last, evaluating Neural Network is necessary to be able to quantify the quality of a learned model.

3.3.1 Natural Language Processing

The goal of Natural Language Processing algorithms applied in this thesis is to turn textual data into numerical form. Moreover, the similarity between different should be maintained.

Calculating similarity between vectors can be achieved by using the Cosine similarity, defined as follows [3]:

$$\begin{aligned}\cos(\delta) &= \frac{A \cdot B}{||A|| \cdot ||B||} \\ &= \frac{\sum_{i=1}^n A_i B_i}{\sqrt{\sum_{i=1}^n A_i^2} \cdot \sqrt{\sum_{i=1}^n B_i^2}}\end{aligned}\tag{3.13}$$

where

A_i, B_i : Components of vector A and B respectively

While computing the similarity between two vectors provides a statistic, visualizing to achieve a broad overview is cumbersome. Therefore, an alternative is using the dimensionality reduction algorithm t - Stochastic Neighbor Embedding (t-SNE) [28]. The algorithm produces a Scatter plot where similar texts are likely to be near to each other, while dissimilar texts are far apart.

After providing two tools to assess the quality of applied Natural Language processing algorithms, the next paragraph focuses on evaluating Neural Networks.

3.3.2 Neural Networks

In the previous section, dividing a data set into training, validation and test set was stated. Evaluating an estimator utilizes the unused records within the test set. This gives an insight in the predictability of the learned model.

When evaluating Neural Networks, the type of prediction this model is going to be used for has to be taken into account. While Neural Networks in general approximate functions defined by the provided data, the calculated outputs can be interpreted in various ways.

Only considering the numerical output implies that the Neural Network has been used to calculate a regression. In this case, the quality of the end result is defined by the deviation of predicted values from the true outcome. There exist different algorithms to calculate this error, including the L1 or absolute distance. This thesis is going to use the mean squared error, which is the predominantly used statistic to assess the quality of Neural Networks. Its definition is shown in formula 3.14. When comparing multiple models, a smaller mean squared error is preferred.

$$MSE = \frac{1}{n} \cdot \sum_{i=1}^n (\hat{Y}_i - Y_i)^2$$

where

$$\begin{aligned} n : & \quad \text{Number of records in the data set} \\ \hat{Y}_i : & \quad \text{Predicted value} \\ Y_i : & \quad \text{True value} \end{aligned} \tag{3.14}$$

An alternative lies within using the Neural Network as a classifier. This requires, that target statistics lie in the range $[0, 1]$. This outcome can then be interpreted as follows: Each target variable assigns the inputs to a specific class. If an output $w \geq 0.5$, then this category applies to the record used for prediction.

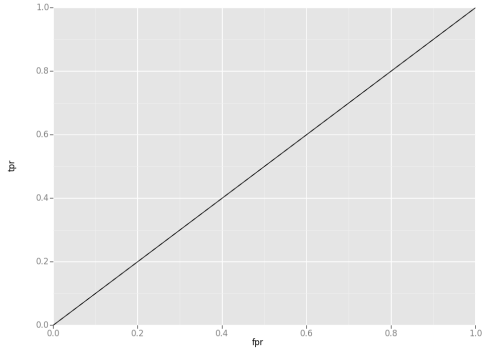
Evaluating a classifier can be achieved by using a Receiver operating characteristic (ROC) [47]. The quality of a classification can be defined using sensitivity and specificity, better known as True Positive Rate (TPR) and True Negative Rate (TNR). While the first statistic describes the relative number of successful classifications, the second variable explains the share where a class has not been chosen correctly. Their definition is depicted in formula 3.15.

$$\begin{aligned} TPR &= \frac{TP}{TP + FN} \\ TNR &= \frac{TN}{FP + TN} \end{aligned}$$

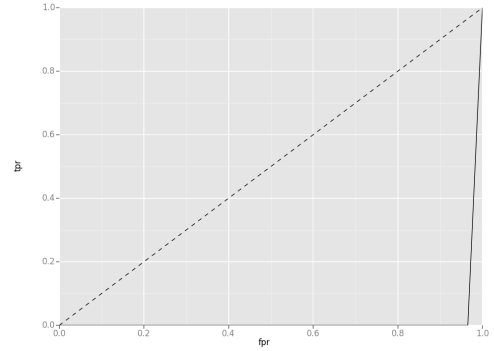
where

$$\begin{aligned} TP : & \quad \text{Number True Positives} \\ FP : & \quad \text{Number False Positives} \\ TN : & \quad \text{Number True Negatives} \\ FN : & \quad \text{Number False Negatives} \end{aligned} \tag{3.15}$$

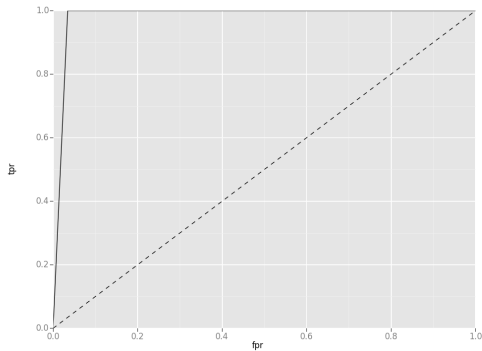
The aforementioned variable w specifies, what values an output has to take in order for that class to be predicted, e.g., $w \geq 0.5$. Varying this threshold leads to different values for TPR and TNR. Figure B.2a shows plots depicting the quality of various random classifier. In order to compare multiple classifiers, the Area Under the Curve (AUC) has to be computed, lying in the range $[0, 1]$. While larger values are preferred, classifiers having an AUC below 0.5 predict worse outcomes than a random classifier, meaning that the classifier does not work properly.



(a) Performance of a random classifier. This is considered as the baseline, since randomly choosing a class is better than a bad and therefore no classifier. The AUC is 0.5.



(b) Performance of a bad classifier. Regardless of the threshold, the False Positives always outweigh the True Positives. The AUC is 0.0563.



(c) Performance of a good classifier. The AUC is 0.9472.

Figure 3.8: All plots show the quality of classifiers. Only looking at the plots, the farther the curve approaches the upper - right corner, the better the classifier is. Comparing two curves can be achieved with the AUC.

After explaining the theoretical background behind this thesis, the next chapter goes into detail on the implementation of the individual algorithms.

Chapter 4

Implementation

In order to apply algorithms for data and natural language processing as well as constructing and using Neural Networks, implementing an interactive application is necessary. The succeeding chapter explains the motives underlying the architectural decisions when implementing the BitTorrent Weatherforecast program. Moreover it goes into detail on the requirements this implementation has to fulfill regarding the usability along with maintaining the possibility to add extensions. While the first section expands on the overall architecture chosen for the implementation, subsequent segments elaborate on the components forming the basis for this application.

4.1 Overall architecture

The resulting application can be divided into six different components containing operations ranging from creating data to facilitating the initial development process over retrieving and processing records from existing sources. Further tasks include creating graphs based on data for analysis as well as training and evaluating Neural Networks. Figure 4.1 presents a broad overview over all components, including dependencies among each other.

Hereinafter the implementation of the individual components are explained in more detail, starting with data creation, followed by the integration of external data sources. The next part describes learning and evaluating Neural Networks while the last section goes into detail on the visualization element of this application.

4.1.1 Data Storage

Storing and retrieving data is a crucial part of this algorithm. The package *Database* comprises implementations regarding saving data locally and inserting these record either into a local or a remote database. Split into three bundles, the *Files* package include the model used for representing a table in this application. Furthermore, the packages *Local*

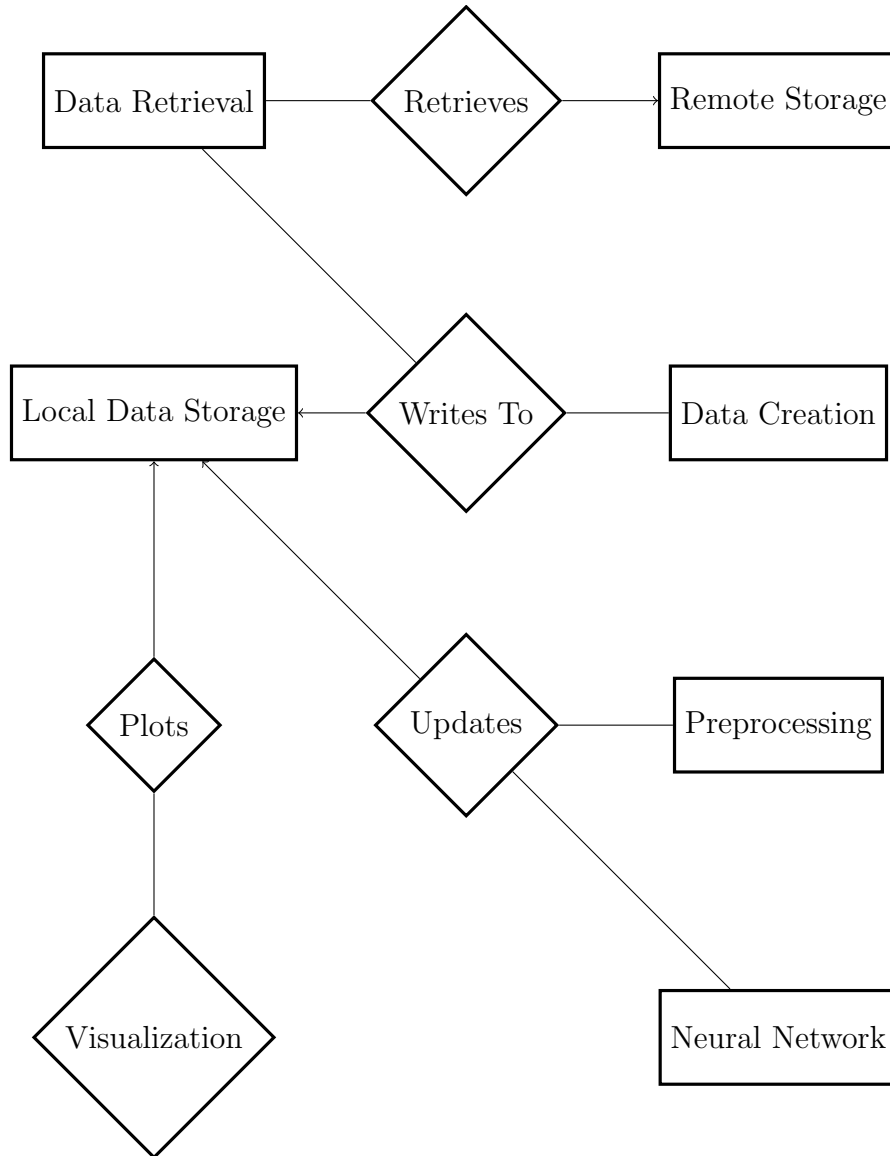


Figure 4.1: Main components assembling the BitTorrent Weatherforecast application.

and *Remote* contain implementations to handle database operations locally or remotely respectively. The next paragraphs go into detail on the individual packages.

Files

This package contains two classes including the module *CsvFile* describing file used to store data. This module encompasses three variables: The *filename* describing the name used when storing this file on the hard disk. The variable *tablename* defines either the table from which the encompassed data has been retrieved or into which table the included data has to be inserted into. At last, the variable *columns* contains a list of tuples, each containing a combination of name and type, defining the structure of the table specified in the variable *tablename*. The order of the columns match the arrangement of properties in the Comma-Separated Values (CSV) file. This module is used extensively when defining queries for interaction with databases, as it abstracts the data and allows a general query definition.

The second module *Controller* provides various constructors for creating *CsvFile* modules including explicitly creating the file given a table structure or inferring a file object given an existing *CsvFile*. Other functions include text processing operations like creating, removing or appending data to a file. Reading and processing data from a file can be achieved by using either a sequential or parallel approach to increase the performance when handling large amounts of data.

Local Database

Implementing a local database on the machine running this program was necessary, because some aspects included in the preprocessing algorithm could not be achieved by the remote database management system. Moreover, implementing a database instead of handling the raw CSV files explained in the preceding section simplifies processing the contained data.

This package is divided into three modules. The *Adapter* class is an interface defining queries to be implemented by a local database adapter. These queries include creating or dropping a table, inserting data into a table and processing queries like calculating the inverse document frequency or dividing a data set into buckets for cross - fold validation. Moreover, it contains the credentials necessary to connect to the database. The Bit-Torrent Weatherforecast implementation provides an adapter for a PostgreSQL database management system implementing these queries.

At last, the *Controller* module uses the specific adapter implementation and applies the queries to the underlying CSV file module. Some queries are explained later on when going into detail on the specific implementation of the algorithms.

Remote Database

The remote database package provides modules to interact with the database implemented by the VIOLA project. The aforementioned VIOLA project is deployed on a remote cluster of computers and requires an Secure Shell (SSH) log in before issuing a query is possible. Since splitting a query onto multiple machines creates additional communication overhead, determining the computer responsible for managing the query is necessary. Figure 4.2 depicts a chart, where different processes necessary for querying the remote database are illustrated.

Executing multiple queries at once is possible by providing multiple files when starting a query. In this case, all enclosed files are copied using one command. While their execution is divided on to the individual machines within the cluster, RAM shortage prevents a more frequent use of this functionality.

4.1.2 Data Creation

In the initial phases of this project, data creation was used extensively to provide Neural Networks records used for learning, validating and testing. The objective of this package is to create data using a mathematical function. This approach enables a varying complexity of the used data set as well as a variable extent of the resulting data size. Furthermore,

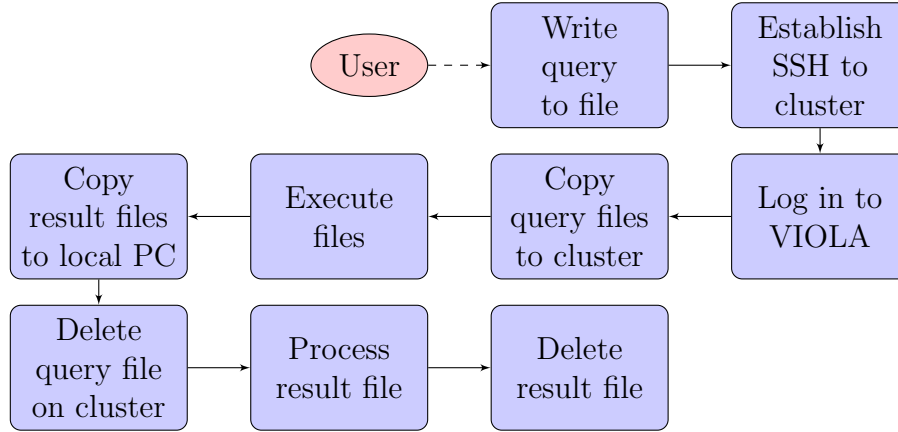


Figure 4.2: Simplified process flow chart when querying the remote database.

generated data using this approach is free from noise. Albeit distant from reality, correct data simplifies developing a functioning Neural Network. This package has dependencies with the files package which is used to create a CSV file and insert the generated records.

The package *DataGeneration* implementing the aforementioned features is organized as follows: This bundle contains four classes, including one abstract class *DataGenerator* defining the logic to create and store data by using three abstract functions. Each generator has to implement this abstract class by providing the mathematical function used for data generation as well as the number of variables included in this function together with a name employed for storing the data. As an example, one generator uses the Sinus as a function to produce data.

In order to create data, one generator has to be initialized. The optional parameter *seed* is used for initializing the Random Number Generator (RNG) utilized for creating the inputs for the mathematical function, ensuring the reproducibility of this algorithm. If omitted, the current timestamp is selected as a seed. After initialization, the tertiary function *generateData* can be used to create a data set. The only mandatory argument *number_tuples* defines the number of records to be created. The optional arguments *min* and *max* determine the range of input values created by the RNG. As default values, the aforementioned arguments have 0 and 1 respectively. As an output, the function returns a *file* object, describing the CSV containing the created data.

4.1.3 Network Analysis

Network analysis became necessary to find routing paths between peers of a swarm. Since additional data is necessary to find connections between ASes, including information provided by CAIDA is required. Therefore, easily updating the imported data set to keep paths as updated as possible as well as changing the routing protocol used to identify routes between peers is one goal of this package.

The Network Analysis package only contains one module: The *RelationController* is responsible for loading the *csv* files containing the CAIDA records into the local database.

Afterwards, joining these tuples and running the path finding algorithm creates two tables, which are then inserted into the remote database. There, both are utilized by the Neural Network preprocessing algorithm to calculate the routing costs. The structure of these tables is depicted in figure 4.3. Implementing different routing algorithms is possible by only exchanging the table *ASPath*. Updating *ASCountryMap* is only necessary, if additional ASes are present in this new table.

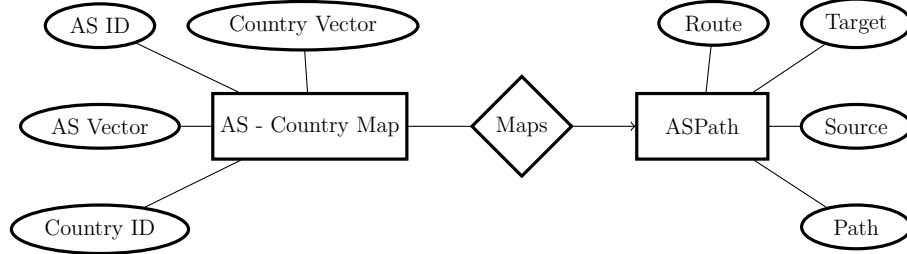


Figure 4.3: ERD of tables created by the BitTorrent Weatherforecast application when calculating routing costs.

After building this routing network, VIOLA records have to be retrieved and prepared to be able to serve as inputs for Neural Networks. The *DataRetrieval* package is responsible for implementing these procedures.

4.1.4 Data Retrieval

As previously mentioned, this package retrieves data from the VIOLA cluster as well as prepares these tuples for Neural Networks. This package also contains only 1 module, having dependencies to the *Remote* database package to query records from the cluster.

Queries executed on the cluster are performed by six computers. This enables a faster computation compared to a single Central Processing Unit (CPU) machine. However, RAM is a sparse resource on remote servers. This scarcity can lead to exceptions when either other programmers execute expensive queries or a single complex query is being performed. In both cases, already started queries are cancelled, returning an exception. The database's state is rolled back to the initial situation, when the query was launched. In order to prevent this exception from happening, splitting large queries into smaller ones is necessary. Figure 4.4 shows how to decide on splitting larger queries, while figure 4.6 depicts an application.

```
1 select distinct year, month from l_dataset;
```

Figure 4.4: Query to determine individual batches to split a complex query into multiple smaller ones.

Even if a subset of queries fails due to RAM scarcity, rerunning the query shown in figure 4.4 collects missing tuples which enables repeating the second query.

After describing major packages used in the BitTorrent Weatherforecast application, the next chapter explains the actual implementation of algorithms characterized in chapter 3.

```

1  select    a.time ,
2           a.quality ,
3           a.movieindex ,
4           a.seeder ,
5           a.leecher ,
6           a.asnumber ,
7           b.asnumber routes ,
8           a.title
9  from      l_dataset a
10 left join l_tmp_route_announce b on
11          a.year = b.year and
12          a.month = b.month and
13          a.day = b.day and
14          a.hour = b.hour and
15          a.intervalid = b.intervalid and
16          a.infohash = b.infohash
17 join l_torrent_title_short c on
18          a.title = c.index
19 where     a.year = {0} and
20          a.month = {1} and
21          exists (
22              select *
23              from    usedmovies
24              where   usedmovies.infohash = a.infohash)

```

Figure 4.5: Query to retrieve data from the cluster after preprocessing it. The query computes two Joins, requiring more RAM than is currently available. {0} / {1} contain the number for year and month retrieved in the aforementioned query respectively.

4.2 Algorithm Implementation

After describing the overall structure of the BitTorrent Weatherforecast application, this section explains the implementation of various algorithm including specifying the module responsible for calculating it. Moreover, available parameters to influence the outcome are explained.

The first explained algorithm is the data processing algorithm, followed by the Neural Network learning module. Afterwards, two procedures capable of evaluating these networks are analyzed. At last, different approaches to plot these results are described.

4.2.1 Data Processing

The data processing unit encompasses a huge amount of different parameters. One such parameter indicates whether the AS relations should be updated in the beginning. Others define the files, from which these relations have to be taken from. The main parameter

is the definition which columns are going to be preprocessed including defining its type. This allows even numerical values to be used as input for the word2vec algorithm. This might make sense, if the numerical values are interpreted as categorial values.

Update Routing Network

This module is separated into two parts: At first, the CAIDA data set is used to update the network, if the corresponding parameter is activated. After revising the routing network, this module begins with processing attributes describing individual movies from the table *torrents*. These attribute can be set via a parameter.

Convert text into a number

Depending on the attributes provided for learning, this module retrieves and starts to process the records. While columns of numerical types are directly copied into the final data set, textual data is processed using the word2vec algorithm as described in the last chapter.

Calculate global popularity

Another decision has to be made when calculating the global popularity of a torrent. Up to three different *trackerURIs* exist, each reporting different seeder and leecher statistics. In order to calculate a single statistic for seeder and leecher respectively, the maxima of both over all three reports were chosen.

Calculate local popularity

Afterwards, distinct values for the column *PeerUID* are being counted for each AS to retrieve the local popularity. In order to prevent having to store a long array, where most of the 77220 AS are zero, a sparse representation is being implemented. Since Impala does not support sparse arrays, the following notation is being used to refer to a sparse array:

1 $\{\{77220, 1\}, \{0, 1\}\}$

Figure 4.6: String denoting a sparse array. The length of this array is 77220. All entries are zero, except the first one which is equal to one. Indices start with zero. While the first entry has to denote the size of the array, the order of the remaining enries is irrelevant, important for subsequent group by queries.

Using the table *ASCountryMap*, for any given AS, the corresponding sparse array identifying that AS can be retrieved. Furthermore, *countryID* identifies the country, the organization running this AS belongs to. The last column, *countryvector* describes, similar to the *asvector* column, a sparse vector identifying the country of that record.

The *countryID* is used to group ASes within the same country. Albeit cumbersome, within the same query, the *asvector* column is also concatenated to result in a sparse vector identifying all ASes within a country.

Calculating the routing cost includes the *aspath* table. This table has four columns: Source, Target, Path and Route. While source and target column identify the respective AS, the path identifies a single AS on that route. If multiple ASes have to be traversed when going from source to target, multiple entries describing the same source - target combination are stored within this table, differing in the *path* column. The last attribute identifies the direction of this edge. Since the directed network structure has been relaxed at the beginning of this thesis, this column is 0.

The most expensive operation when calculating routing costs is the partial self - join on the *intervalannounce* table. This step is necessary in order to retrieve all source - target AS combinations within a swarm. This result can then be joined with the *aspath* table on source and target to retrieve all *path* ASes for the data set. Summing over the distinct *PeerUIDs* for each *infohash* for each *path* gives the expected cost for routing.

4.3 Learning the Neural Network

In order to learn a Neural Network, a suitable data set is required. The structure expected by this module is depicted in figure 4.7. The data retrieval module loading records from the VIOLA data set creates such a table. Moreover, the data creation process builds the same table.

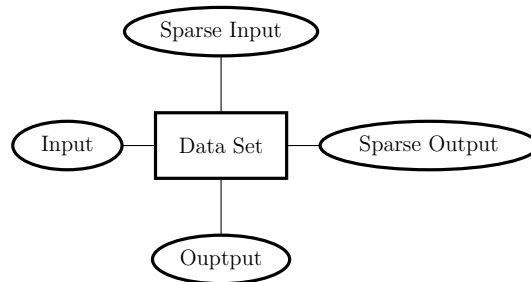


Figure 4.7: ERD of tables accepted by the Neural Network Learning module.

Besides providing a data set, other interesting parameters are for example the number of cores on which the learning task should be divided on to. Defining the number of layers implemented as well as the number of nodes within each layer is achieved by using a list. The length of the list defines the number of layers in the network, while each entry defines the number of nodes in that layer. This enables fast and easy modifying the network.

4.4 Evaluation

The application contains several modules allowing for a wide range of different plots. In general, these modules can be divided into two categories: the console programs can be started with various parameters and display plots, which can then in turn be saved as a PNG file. Another module draws a map, which shows the localized popularity over individual countries together with the global popularity.

4.4.1 Console Plots

The modules which can be used to generate plots contain between [0,2] arguments. Some modules require a specific query to get the data, for example the *PlotASPath* module. This program retrieves its information directly from VIOLA by executing a query, before displaying it with a plot.

Other examples include the *PlotNeuralNetwork*, which after providing a model containing a Neural Network together with window sizes for *seeder* and *leecher* attributes, shows a comparison between real data and predicted outcome.

The application *PlotROC* requires a model of a Neural Network together with the data set used for learning this classifier. The program then retrieves the testing data to generate a ROC plot. Afterwards, the AUC score is being printed to the console.

The *PlotNNError* program requires specifying a table, where the results of the learning process of a Neural Network are being stored. Using this table, three plots depicting the training, validation and finally the required time for training are being created.

Plotting similarity between movies using the t-SNE can be achieved by executing *PlotTSNE*. This plot colors all titles, which contain a defined word red, while the default color is blue. The defined phrase does not have to be a single word but can be a whole phrase.

These are the applications which are capable of displaying a few plots. The next section goes into detail on the interactive application.

4.4.2 Interactive Application

The application has been implemented to give a fast overview over the popularity of movies, both globally and locally.

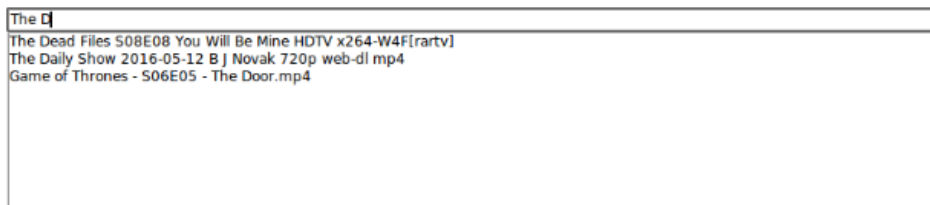
When logging in, a movie has to be selected. In order to prevent having to go through a long list, a autofill textfield has been implemented, as can be seen in figure 4.8.

After proceeding to the next screen, a world map is being displayed. The program chooses one country at random, which then defines the local popularity being plotted into the world map. The darker the green gets, the more popular the torrent is. Moreover, while grey and white both signalize that no user is currently sharing this file, countries colored in grey have never had a single peer sharing this torrent. Below the map, two plots depicting the local and global popularity are displayed giving an insight on the development of the statistics.

Every few milliseconds, the content of the world map updates, displaying the local popularity of the succeeding request. A red vertical line depicted in the global popularity plot indicates the current request, giving a hint on the current global popularity. In order to visualize small changes in popularity, a log - scale has been used before turning the numbers into colors.

Towards a BitTorrent Weather Forecast

Choose a movie:



The

- The Dead Files S08E08 You Will Be Mine HDTV x264-W4F[rartv]
- The Daily Show 2016-05-12 B J Novak 720p web-dl mp4
- Game of Thrones - S06E05 - The Door.mp4

Figure 4.8: Implementation of an Autofill textfield. Typing in opens the dropdown menu and shows search results for the typed letters. By clicking on one of them, the result is selected and filled into the textfield. The button to move on to the next screen is not pressed automatically.

Including the costs for routing, the world map changes its color to red. In the case depicted in figure ?? the difference between popularity and costs for routing has been calculated and plotted. Since negative result are depicted using a red color scheme, routing costs seem to be higher in most parts of the world.

Visualizing the similarity between various movies together with the top n

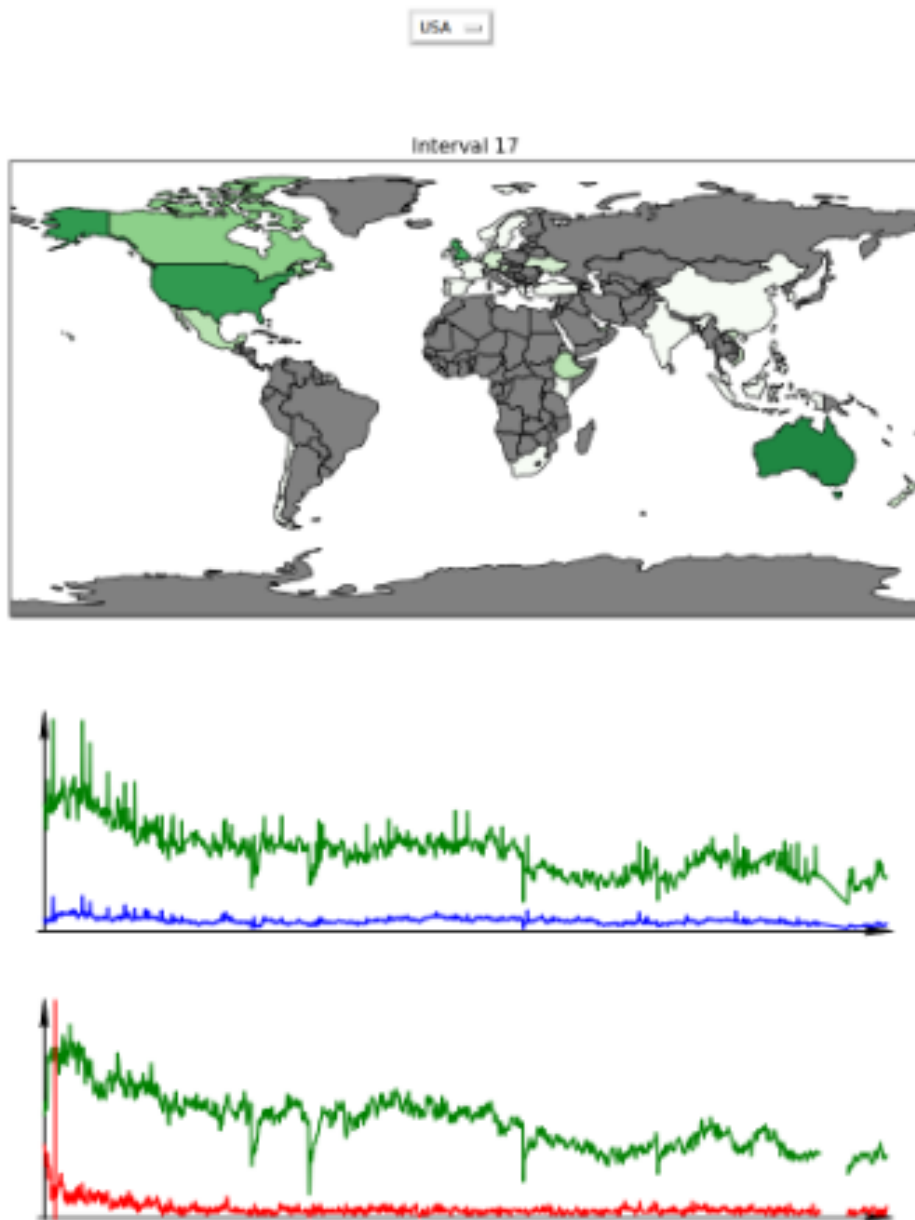


Figure 4.9: Interactive Plotting application. The upper part shows the world map where different colors indicate local popularity.

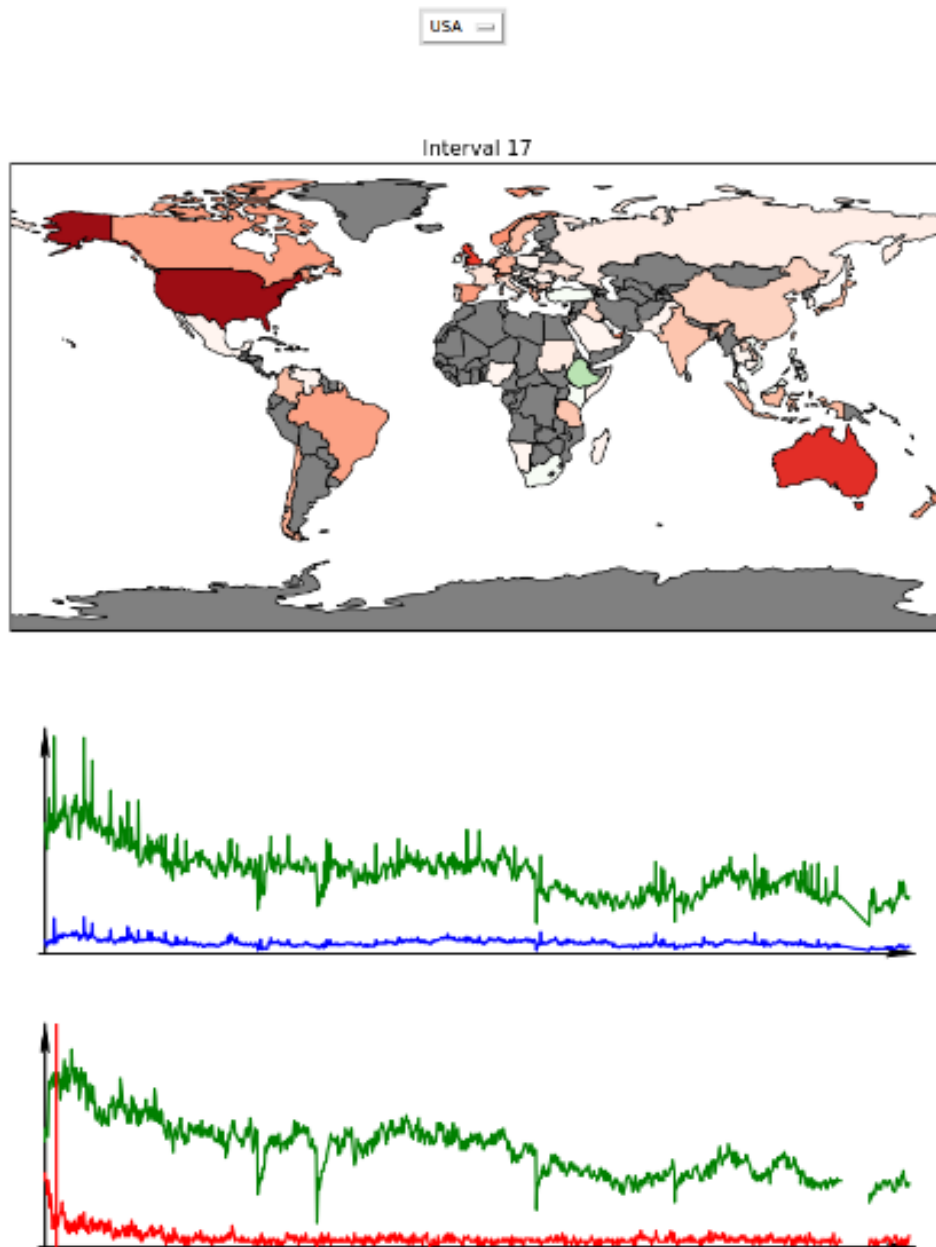


Figure 4.10: Interactive Plotting application. Here, the cost for routing are included as well as the popularity.

Chapter 5

Evaluation

The evaluation encompasses various algorithms applied in this thesis. The first area being evaluated is the data which is considered for learning. Afterwards, the algorithm applied to identify routes between peers is being analyzed. Subsequent

5.1 Data Quality

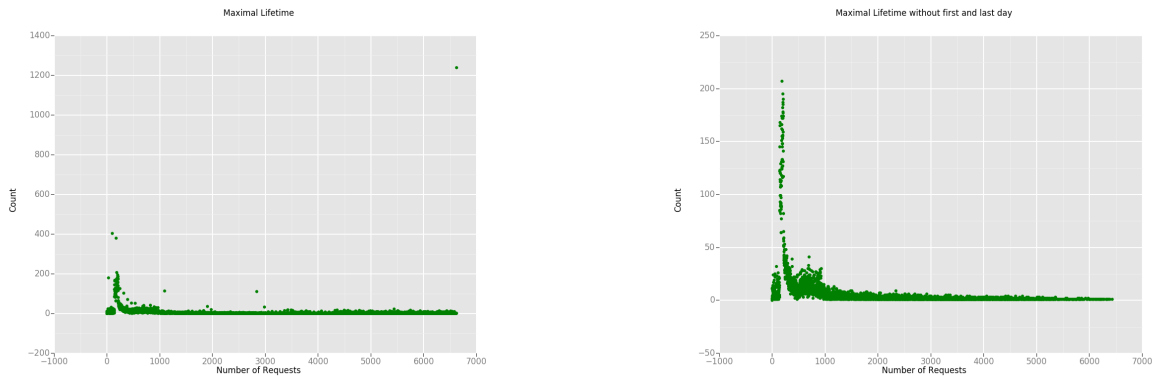
VIOLA collected data regarding BitTorrent sharing behavior over three months. Taking a closer look at individual torrents, analyzing its lifetime is interesting. That statistic describes the duration at least one peer shares this torrent. Figure 5.1a depict the distribution over the range of lifetimes observed within the data set. Indicated by the spike at the upper end of the range, it appears that many torrents are shared over a longer period than three months. In order to ensure that the whole lifetime has been captured by VIOLA, torrents recorded on the first and last day of collecting data have not been included into the data analysis part. This restriction reduces the number of included movies from 38351 to 29513. The resulting distribution is depicted in figure 5.1b.

Additional precautions include the removal of torrent records, which have been deleted beforehand and therefore contain invalid data.

5.2 Routing Algorithm

As explained in earlier chapters, this thesis implements the shortest path algorithm in order to determine routes between different peers in the network. This section aims to evaluate the quality of the shortest path algorithm to identify routing paths.

When looking at the nationality of AS lying on shortest paths, it becomes clear that a huge amount of AS on shortest paths belong to the USA. Figure B.1 illustrates the distribution of those shortest - path - AS on countries.



(a) Lifetime of torrents recorded within the VIOLA data set

(b) Lifetime of torrents recorded within the VIOLA data set

Figure 5.1: Errors of the PQ_3 stopping criteria.

Another statistic counts the number of requests a movie has been documented by VIOLA. Figure 5.1a shows its distribution, having a peak at the highest number of requests. This suggests, that a large portion of movies in this file sharing network is being shared over a long time. In order to ensure, that the whole lifetime of a movie is being recorded, only movies not captured on the first and last day are being used for this analysis.

Removing these mentioned movies reveals that most files are shared using torrent for about 200 requests

5.3 Evaluate Word2Vec

In order to be able to compare multiple vectors produced by the Word2Vec algorithm, the t-SNE has to be applied. The plot in figure 5.2 is a point cloud. Each point identifies a phrase. If points are close to each other, they are similar. An interesting observable fact is, that the similarity also depends on the number of words in the phrase. Figure 5.2 has points colored which contain the phrase “5.1”. While this word appears often within the phrases, the phrases containing this word are not as close to each other as those phrases in the second example: Figure 5.3 has the same initial situation however it colored phrases containing “Game of Thrones” blue. That shows, that the longer the title is, the more likely it is, that movies with similar title are next to each other.

After evaluating the processing algorithm, training a Neural Networks used for Regression is going to be evaluated.

5.4 Evaluate Regression

When comparing different Neural Networks, the goal is to reduce the test error in order to approximate the underlying function best. This evaluation is going to start with comparing the performance of various early stopping criteria.

5.4.1 Early Stopping Criteria

In the following, the two major stopping criteria are going to be compared. Starting with PQ_1 , plots 5.4 and 5.5 show, that both training and evaluation error seem decreased nicely. However, the evaluation error makes a dip leading to PQ_1 to stop the learning phase. Studying 5.6 shows that the function slowly approximated the mean, but was not able to continue.

Moving on to the next stopping criteria, PQ_3 , the situation looks similar to the aforementioned PQ_1 . criteria. Where PQ_3 was able to check off a few more epoch strips than PQ_1 , the network could still not converge. Figure 5.7 shows that the function learned by PQ_3 moved a bit further in the direction of the average, strengthen the assumption that Neural Networks start converging like this. The corresponding training and validation errors have been plotted in the Appendix because the respective errors of PQ_1 were very similar.

The last criteria is the UP_8 criteria. Figure 5.10 shows a successful convergence.

The training and evaluation error observed in UP_8 support the fact, that convergence has been achieved. This means, that improvements only happen incremental. While increasing the size of the network is one way to further decrease the deviation from the true value, introducing a moving window is another alternative.

5.4.2 Moving Window

Two different windows are being compared: The first one introduces a window of length three leading to an approximated function depicted in 5.10.

Analyzing this function shows that the function course has a slight seasonal effect. By further increasing this window size, this seasonal effect also extends. Moreover, the improved approximation is intuitive because the past and current popularity of a movie has an influence on the popularity within the next iteration.

Using another Neural Network with the same internal specifications but utilizing a larger window size confirms the idea behind the seasonal effect.

The successful estimation of this model evinced the best fit reachable on a single machine. Other setups either converged in a worse fit or where too complex and had therefore to be stopped.

Moving on to the the localized popularity for individual countries. Since 238 countries have been recorded in the VIOLA data set, this module has to predict as many functions. Starting a test run with 1500 nodes in the first and 750 nodes in the second layer took too long to converge. Therefore, a classification has been considered:

is in order to predict popularity for movies, a wider estimator, implementing more functions is necessary.

5.5 Evaluate Classification

Since the complexity was too large when including 238 target functions for estimation, calculating a classifier might be a better solution. Therefore, multiple new Neural Networks are set up. One uses 400 nodes in the first and 100 nodes in the second, while the other utilizes 500 nodes in the first and 300 in the second layer. Moreover, both use the UP_8 early stopping rule.

After learning the module, looking at the training and validation error of both classifications, it became clear, that this classification had converged. While not stopped by the stopping criteria, the error test error to get an intuition after running the training algorithm was 0.00345. Albeit slightly higher than the validation error with 0.00331, the real test is being



Figure 5.2: t - SNE. Blue points contain “5.1”, red not



Figure 5.3: t - SNE. Blue points contain “Game of Thrones”, red not

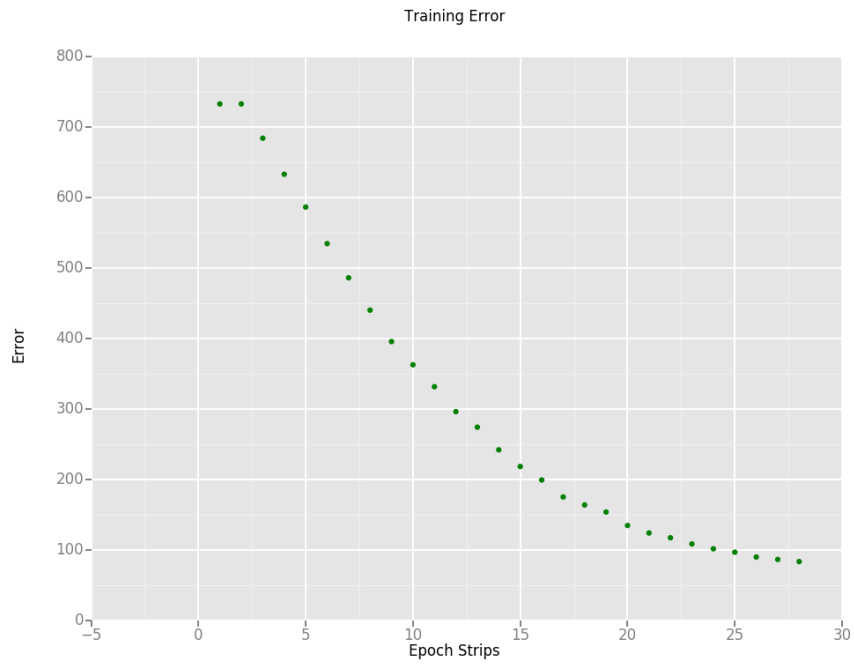


Figure 5.4: Development of the training error. Early stopping PQ_1 has been applied.

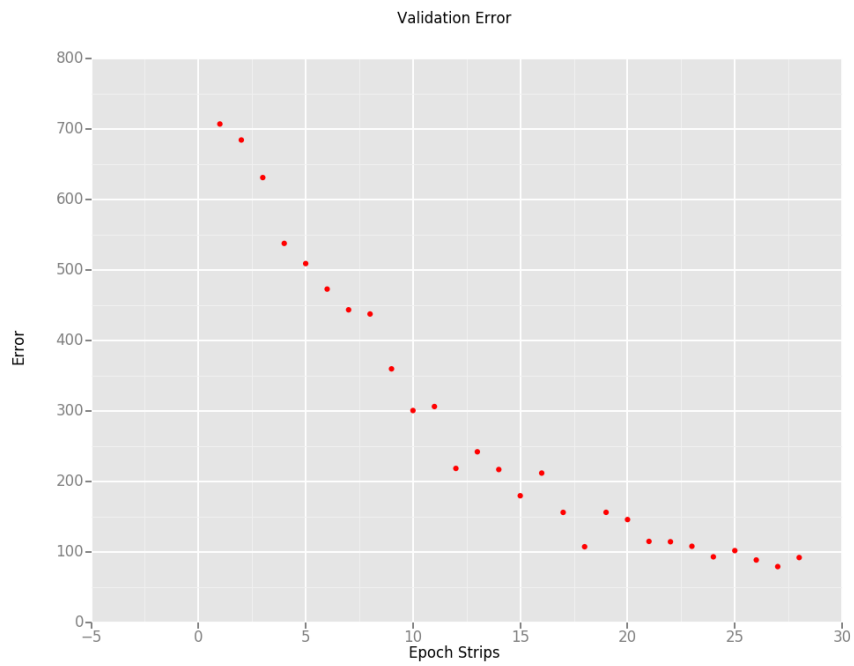


Figure 5.5: Development of the validation error. Early stopping PQ_1 has been applied.

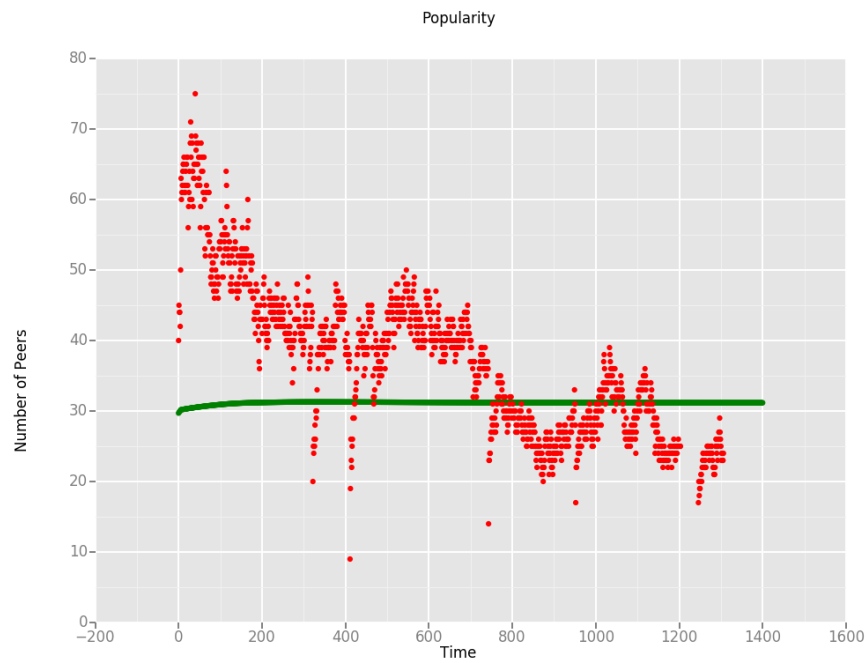


Figure 5.6: Final result of the regression using PQ_1 as Early Stopping.

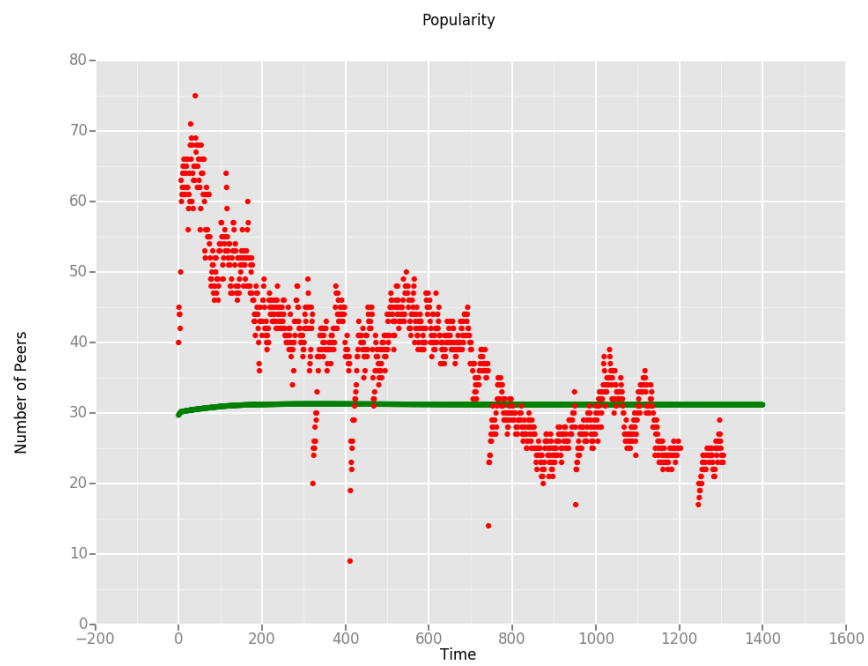


Figure 5.7: Function learned by the PQ_3 module together with original data

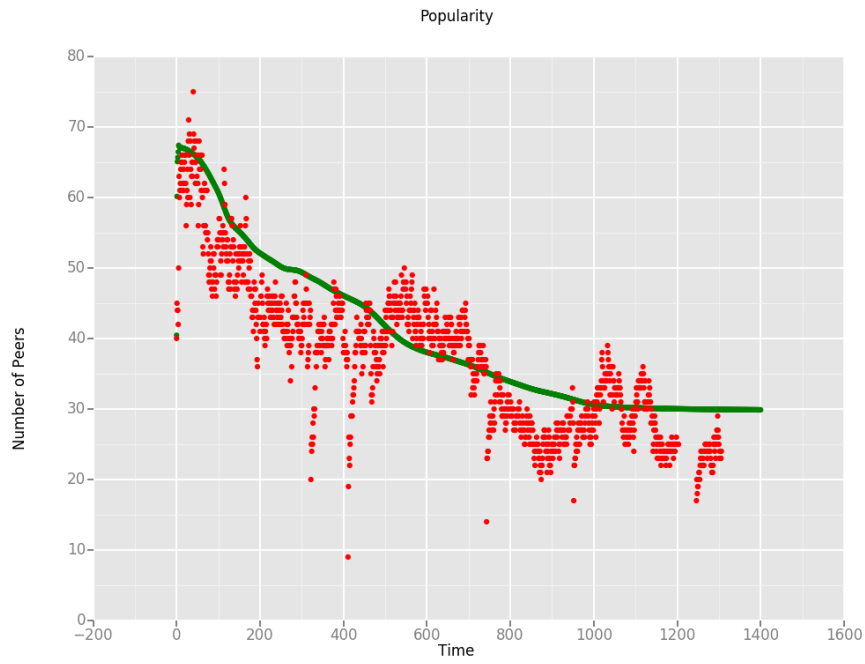


Figure 5.8: Function learned by the UP_8 module together with original data

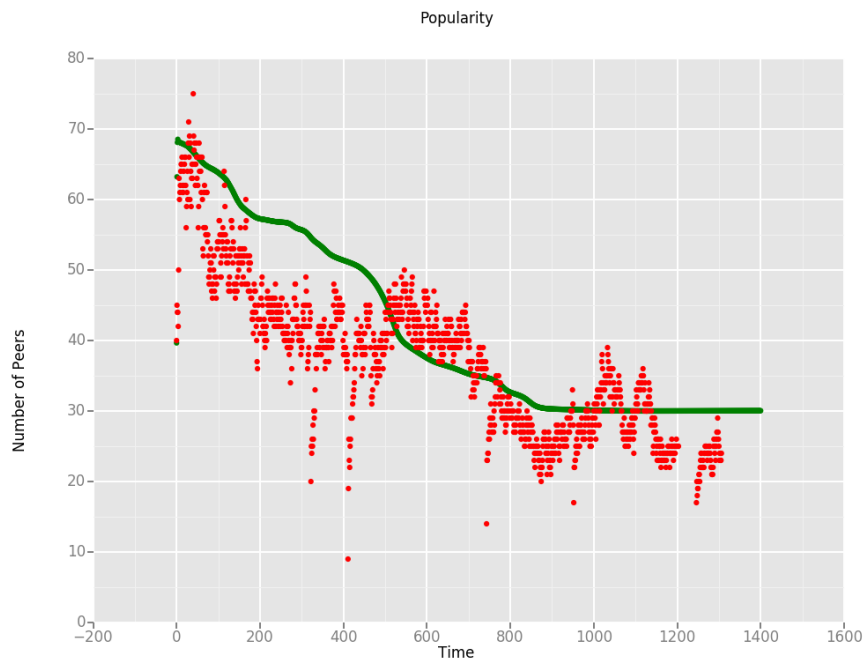


Figure 5.9: Function learned with the usage of a Moving Window

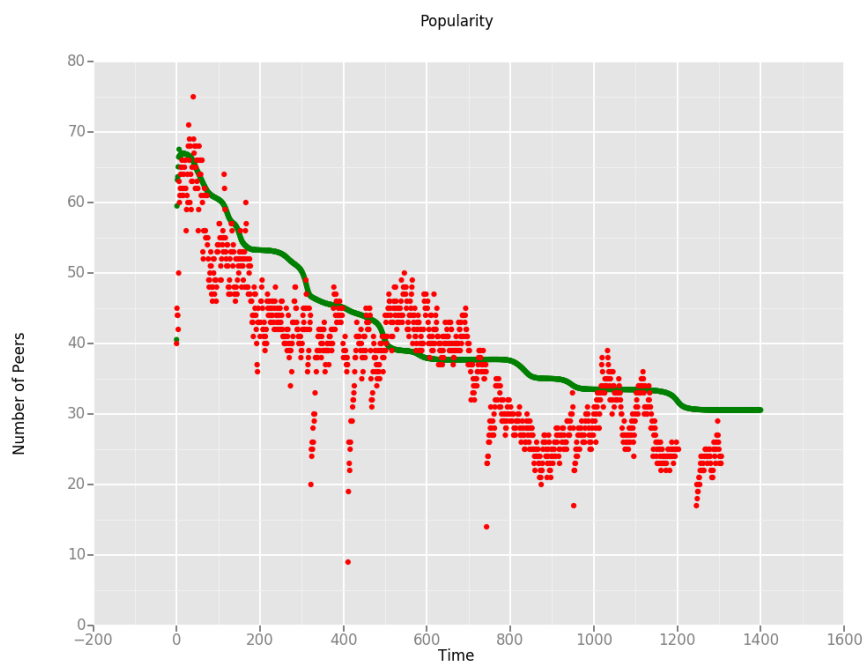


Figure 5.10: Function learned with the usage of a Moving Window of length 7

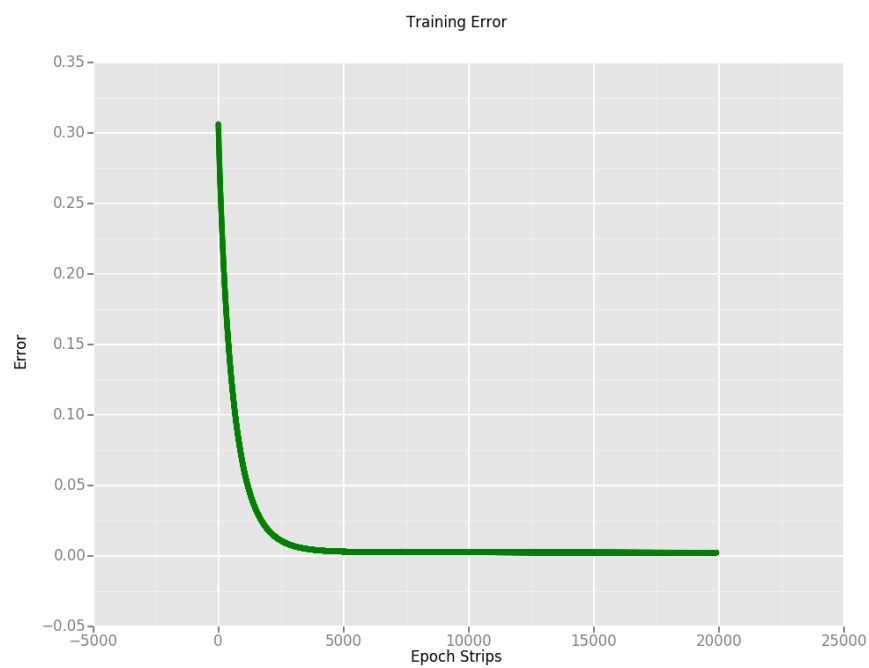


Figure 5.11: Lifetime of torrents recorded within the VIOLA data set

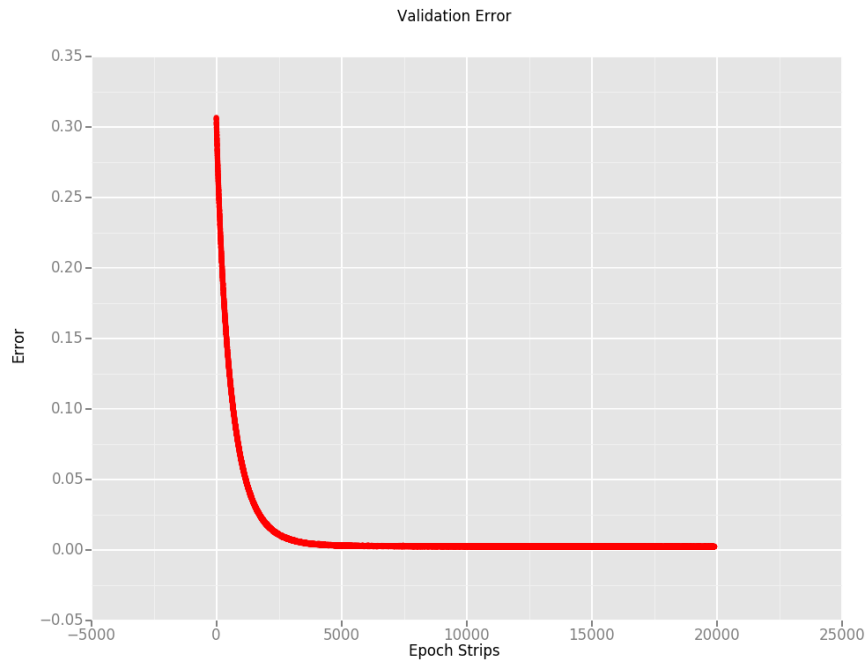


Figure 5.12: Lifetime of torrents recorded within the VIOLA data set

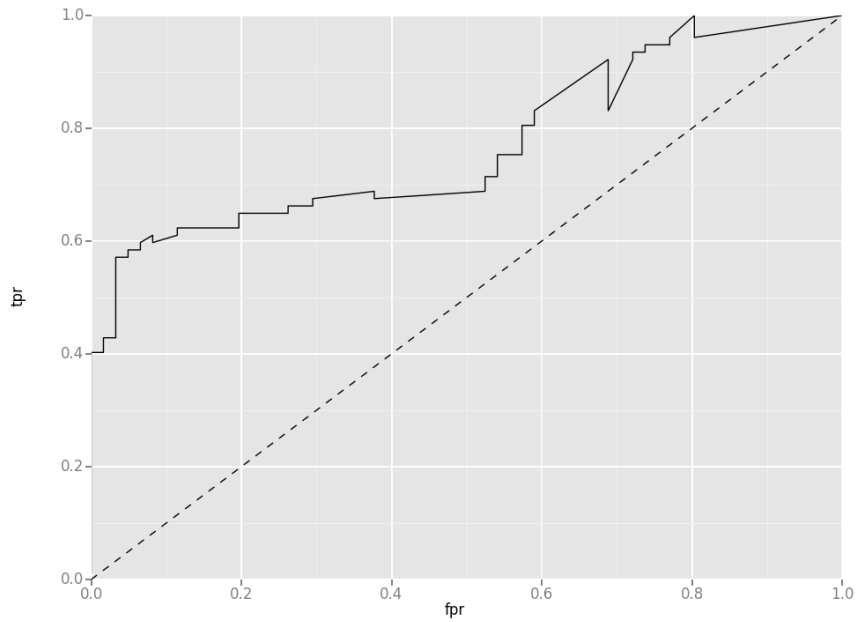


Figure 5.13: ROC plot validating the quality of this classifier. The area under the curve is 0.77.

Chapter 6

Further Work

In this section, additional ideas extending the applied algorithms are being stated. The goal is to improve either outcomes produced by algorithms or the data used to learn the Networks.

6.1 Merge Similar Movies

The first idea revolves around the vector representations of movie titles generated by the *word2vec* algorithm. The main advantage, as already mentioned in the evaluation, is that similar titles share a high similarity regarding their vector. Figure 6.1 illustrates the similarity. It can be seen, that a high number of movies are very similar to the movie in mind.

One opportunity to reduce the complexity of the data set is to merge announce records of similar movies. Looking at table 6.1, the top 13 torrents contain video about “Better Call Saul”. As a title, they could choose the part from the current title, which all have in common. In the given case, this were “Better Call Saul”. While joining announce records is possible, joining the corresponding records from the torrent table is more involved. Since these tuples are important for learning Neural Network and cannot be omitted, this algorithm has not been implemented.

6.2 Learn Neural Networks on a Cluster

The evaluation showed that learning the Neural Network is more efficient using all cores for matrix multiplication than dividing data onto several cores - at least in the single computer case. Learning Neural Networks on the VIOLA data using a cluster, where each PC has its own processor and RAM to utilize, might exhibit other results.

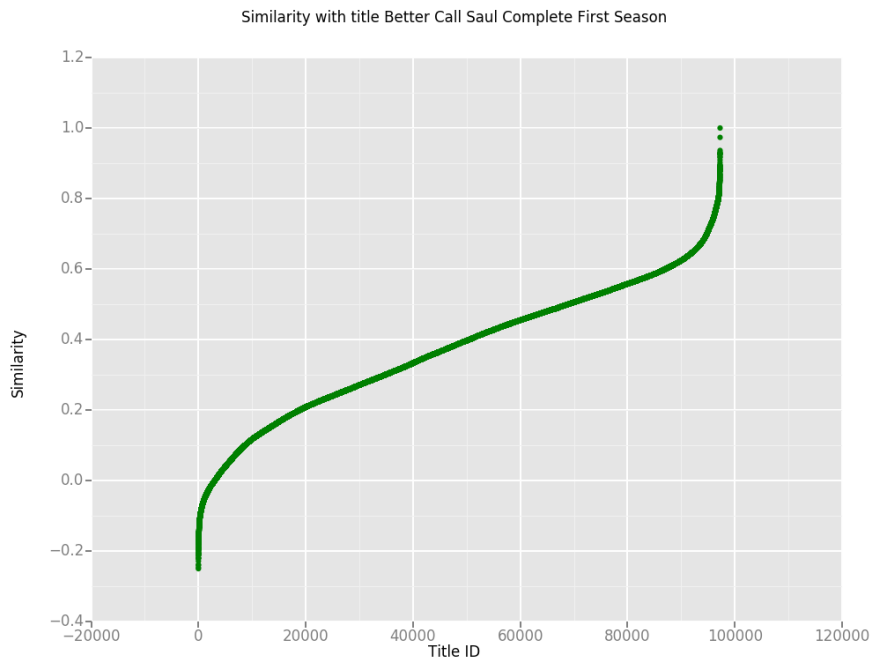


Figure 6.1: Cosine Similarity between the torrent titled “Better Call Saul Complete First Season” to all other movies in the data set.

6.3 Network Analysis

The last section deals with the network analysis component. Whenever network analysis algorithms had to be applied, simplifications were put in place in order to be able to concentrate on learning and evaluating Neural Networks.

As an example, this theses assumed, that the traffic, which is sent when a movie is uploaded, is equal to one. As long as only a single swarm is being analyzed, this has no negative effect. However, this does not support summing over all movies in the data set in order to get a congestion graph for the whole file sharing network. This can only be achieved, if the actual file size is included as the traffic sent instead of one.

Lastly, implementing another routing protocol might better reflect realistic routing behavior. While the shortest path algorithm seems to be the optimal solution, it is not guaranteed that this scenario is realistic. Looking at figure ??, the shortest path algorithm create routing paths, which are not possible in reality. Considering the connection between ASes A and Z. The shortest path between both is along AS B. This is an unlikely example, because while B has to route the traffic from A to Z, the outgoing edges signalize, that B has to pay A and Z to do that. This problem can be solved by sorting all paths between source and target according to their length and determine the shortest realistic path as the routing path. However, even in this case a contingency plan has to be defined, when no realistic path exist. Therefore, normal shortest path algorithm has been implemented.

Movie	Similarity
Better Call Saul Complete First Season	0.9999999999999998
Better Call Saul (Season 1) Complete	0.9734038670367342
Better.Call.Saul - Series.02x10 x264 SM0KIN	0.9362950125874016
Better Call Saul Season 2 S02E03 720p Web dl x264 MrLss	0.9310092805646067
Better Call Saul Season 2 S02E02 720p Web dl x264 MrLss	0.9281143032572814
Better Call Saul Season 2 S02E01 720p Web dl x264 MrLss	0.9279070056617542
Better Call Saul Season 2 S02E04 720p Web dl x264 MrLss	0.9274199172796699
Better Call Saul Season 2 S02E06 720p Web dl x264 MrLss	0.9259704882942109
Better Call Saul Season 2 S02E05 720p Web dl x264 MrLss	0.9242582469606385
BETTER CALL SAUL SEASON 1 COMPLETE HDTV 720 264 cenobites	0.9178648333759962
BETTER CALL SAUL SEASON 2 COMPLETE HDTV 720 264 cenobites	0.9172592406512815
Better Call Saul -Season 2 - Ep 1 -10 [720p HDTV] x265	0.9092371536376405
Better Call Saul [Season 1 and Season 2] (2015-2016) WEB-DLRip ExTr3M3nC0DeS	0.901692825514393
Agent X (US) S01 Season 1 Complete 720p HDTV x264 [TV2NITE]	0.8953514713758334
Call The Midwife S05 E04 SD.mp4	0.8943170952603671
Martels Agent Carter (2015) S01 Season 1 Complete 720p HDTV x264 [TV2NITE]	0.8931616066825581
Call The Midwife S05 E05 SD.mp4	0.8926442026846211
CALL THE MIDWIFE SERIES FOUR	0.8901642876831326
Better Call Saul S02E03 1080p HDTV x265-KRAVE	0.8900874678598585
DareDevil Season 2 (2016) S02E03-E06 400p small mp4s	0.8887880466222017

Table 6.1: Top 20 movies from the VIOLA data set which have the highest similarity with “Better Call Saul Complete First Season”

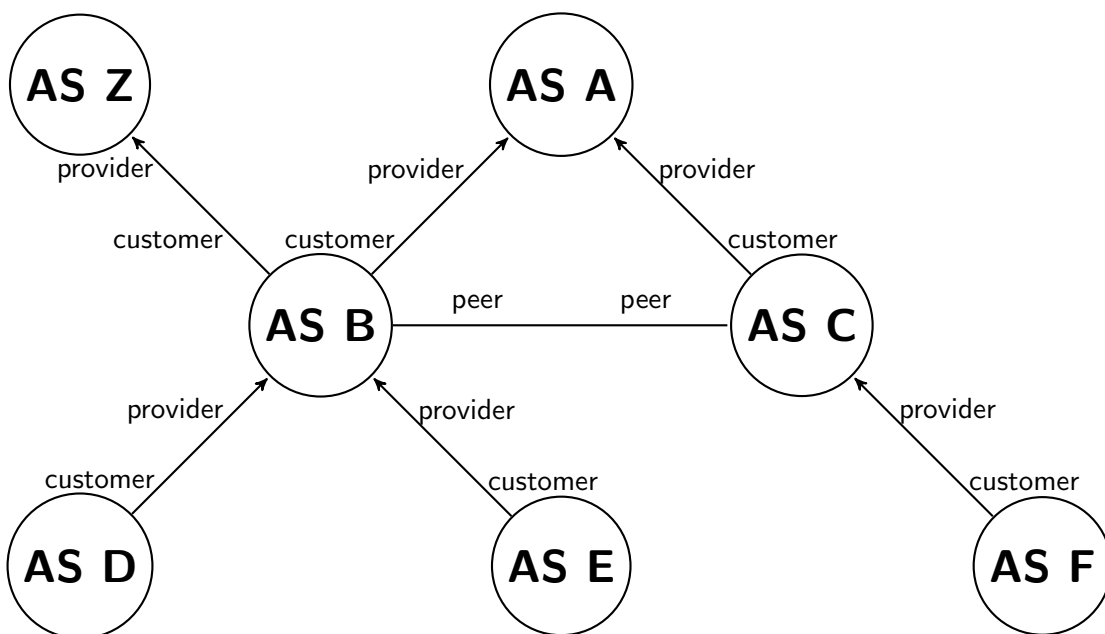


Figure 6.2: Example of a network, where using the shortest path algorithm produces an invalid path.

Chapter 7

Conclusion

This chapter serve concluding this thesis by shortly summarizing its content and the results. After giving a brief overview over the contributions each chapter added to the overall outcome, findings regarding the individual research questions are stated. At last, a final statement is completes this thesis.

7.1 Summary

This work was divided into seven chapters and started with giving a short introduction explaining the motivation behind calculating popularity of files in a CDN. Moreover, research questions aiming at analyzing big data and applying Machine Learning algorithms were stated. The pursued objective was to evaluate the practicality of Neural Network when handling big data. The third chapter explained theoretical algorithms enabling the implementation of the aforementioned algorithms. The architecture and realization was part of chapter four, before the subsequent chapter evaluated the individual algorithms. Chapter 5 The following chapter established a basis for

7.2 Research Questions

In this section, the declared research questions are repeated and answered by reflecting on the result achieved in the preceded evaluation. After answering these research questions, a final statement concludes this thesis.

7.2.1 How can the popularity of individual movies over time be determined?

Calculating the popularity of movies was a problem which seemed to have a straightforward solution. Popularity in general could be determined by considering the number of seeder and leechers within a swarm.

On the other hand, calculating locality based popularity required defining a level of abstraction to which locality should be considered. While ASes offer a fine - grained solution highly suitable for further calculation, both visualizing and learning Neural Networks become cumbersome with such a high number of outcomes. The other extreme, merging ASes into continents, while simplifying the learning and visualization process, the amount of lost information due to averaging over the had to be prevented. As a trade - off, the demand was calculated for each AS, while Neural Networks as well as the visualization element sum over countries to simplify these algorithms.

7.2.2 Given a certain routing protocol, how can the costs for transporting traffic be calculated?

Including created transportation costs by uploading content required adding additional data to the VIOLA data set. By introducing AS relationships detected by CAIDA, applying routing protocols became necessary as source and target might be connected by multiple paths.

After determining relevant paths, calculating the expected traffic on the network was dependent on the torrent - client utilized by the seeder. Sharing the occurred traffic evenly on unchoked connection gives an estimation on the costs arisen from sharing the file.

7.2.3 How can Machine Learning algorithms predict popularity over time?

Neural Networks use a gradient descent approach to approximate functions defined by data provided for training. Popularity in general has been implemented by predicting two functions: one describing the amount of seeder and one for leecher respectively over time. The neural network is then used to approximate this function. Due to the assumption, that new inputs lie on the same curve, Neural Network are capable of making predictions just by approximating a function.

In term of localized popularity, the number of countries to be predicted by the algorithm were too high. Therefore, reducing the complexity by using a classifier instead of an estimator enabled the Neural Network to make applicable predictions.

7.2.4 What accuracy can be achieved by the chosen Machine Learning algorithm?

In order to maximize the accuracy of the prediction made by the Neural Network, overfitting has to be reduced. Therefore, employing an early stopping rule to increase the quality of the estimator is recommended. By comparing multiple stopping rules, the promising

approach can be selected. While the PQ criteria stopped early, suggesting that using VIOLA data overfitting is less likely when using Neural Networks, the UP_8 criteria offered a good tradeoff between required time for learning and prediction quality.

While crossfold - validation only seemed increase the time necessary for learning, introducing a moving window of size seven for each of the target variables led to a better convergence. In summary, it can be stated that the final Neural Network achieved a deviation from 2.133.

Concentrating on localized demand, a Neural Network utilizing two layers where the first layer employed 1500, while the seconds used 750 nodes to predict 238 functions was not able to converge. The problem underlying this optimization problem was the wide array of functions to be considered. Testing with an even larger network would have been a lengthy process because in order to enable a better outcome, a larger network has to be implemented. Since the last failed version took more than a week to finish, trying a second even longer trial was not an option.

An alternative generalizes this problem. Instead of calculating an estimator, a classifier determining the country having currently the largest among of peers. While the resulting model was not a perfect classifier, its predictions were clearly better than the random classifier. The AUC of the best classifier was 0.77.

7.3 Final Statement

This work looked at how Neural Networks can be applied on big data in order to make applicable predictions. After enriching the VIOLA data set with records from the CAIDA set, the possibilities offered by this new VIOLA data set were huge. Therefore, the decision to also include costs for routing traffic through the network were included. Thereupon multiple tools to prevent overfitting had been implemented. When comparing the stopping criteria, it became clear that due to the huge amount of records, overfitting is less of a problem. However, the complexity introduced by the local popularity estimator had to be addressed. Deciding to implement a classifier instead of an estimator yielded a good outcome.

The Neural Network development process can be cumbersome when having to wait for results not even knowing whether the algorithm is going to converge. However, finding applicable networks for complex cases is a good explanation for the popularity of Neural Networks.

Bibliography

- [1] Adoni A, Panigrahy R, Valiant G, and Zhang L. Learning Polynomials with Neural Networks. 2014.
- [2] Luther A, Buyya R, Ranjan R, and Venugopal S. Peer-to-Peer Grid Computing and a .NET-based Alchemi Framework. 2005.
- [3] Singhal A. Modern Information Retrieval: A Brief Overview. 2001.
- [4] F Baccelli, F Mathieu, I Norros, and R Varloot. Can P2P Networks be Super-Scalable? 2013.
- [5] S Baset and H Schulzrinne. An Analysis of the Skype Peer-to-Peer Internet Telephony Protocol. 2004.
- [6] R Bindal, P Cao, and W Chan. Improving Traffic Locality in BitTorrent via Biased Neighbor Selection. 2006.
- [7] Huang C, Wang A, Li J, and Ross K. Measuring and Evaluating Large-Scale CDNs. 2008.
- [8] Lueg C and D Fisher. From Usenet to CoWebs: Interacting with Social Information Spaces. 2011.
- [9] D Choffnes and F Bustamante. Taming the Torrent. 2008.
- [10] M Dischinger, A Mislove, A Haeberlen, and K Gummadi. Detecting BitTorrent Blocking. 2008.
- [11] P Eckersley. Packet forgery by isps: A report on the comcast affair. <http://www.eff.org/wp/packet-forgery-isps-report-comcast-affair>, 2007. Accessed: 2016-12-15.
- [12] Eweek. The hidden cost of p2p. <http://www.eweek.com/c/a/Enterprise-Networking/The-Hidden-Cost-of-P2P>, 2001. Accessed: 2016-10-03.
- [13] Rosenblatt F. Principles of Neurodynamics. Perceptrons and the theory of Brain Mechanisms. 1961.
- [14] Fandango and Flixter. Rotten tomatoes: Movies,tv shows, movie trailers, reviews. <https://www.rottentomatoes.com/>. Accessed: 2016-10-03.

- [15] Center for Applied Internet Data Analysis. The caida as relationships dataset. <http://www.caida.org/data/as-relationships/>, 2016. Accessed: 2016-12-15.
- [16] Brightwell G, Kenyon C, and Paugam Moisy H. Multilayer neural networks: one or two hidden layers? . 1996.
- [17] Dahl G, McAvinney A, and Newhall T. Parallelizing Neural Network Training for Cluster Systems. 2008.
- [18] S Ghosh, A Bejan, and A Bhattacharya. Altruistic Routing in Structured P2P Networks. 2006.
- [19] Osipkov I, Wang P, Hopper N, and Kim Y. Robust Accounting in Decentralized P2P Storage Systems. 2005.
- [20] Poese I, Frank B, Ager B, Smaragdakis G, and Feldmann A. Improving Content Delivery Using Provider-aided Distance Information. 2010.
- [21] Stoica I, Morris R, Liben Nowell D, Karger D, Kaashoek M, Dabek F, and Balakrishnan H. Chord: A Scalable Peer-to-peer Lookup Protocol for Internet Applications. 2001.
- [22] Cisco Inc. White paper: Cisco VNI Forecast and Methodology, 2015-2020. 2016.
- [23] IMDb.com Inc. Imdb movies, tv and celebrities. <http://www.imdb.com/>. Accessed: 2016-10-03.
- [24] Dean J, Corrado G, Monga R, and Chen K. Large Scale Distributed Deep Networks. 2012.
- [25] J Johnsen, L Karlsen, and S Birkeland. Peer-to-peer networking with BitTorrent. 2005.
- [26] Gimpel K, Das D, and Smith N. Distributed Asynchronous Online Learning for Natural Language Processing. 2010.
- [27] A Krishnamurthy, Y Liu, and Silberschatz A. P4P: Provider Portal for Applications. 2008.
- [28] Maaten L and Hinton G. Visualizing Data using t-SNE. 2000.
- [29] Prechelt L. Early Stopping | but when? 2012.
- [30] A Lareida and et al. Viola - video consumption in overlay networks. <http://www.csg.uzh.ch/research/viola.html>. Accessed: 2016-10-03.
- [31] M Lin, J Lui, and D Chiu. Design and Analysis of ISP-friendly File Distribution Protocols. 2009.
- [32] Craven M and Shavlik J. Using Neural Networks for Data Mining. 1997.
- [33] Li M, Zhang T, Chen Y, and Smola A. Efficient Mini-batch Training for Stochastic Optimization. 2014.

- [34] Mennecke M. Dsl broadband providers perform balancing act. <http://www.slyck.com/news.php?story=973>, 2005. Accessed: 2016-12-19.
- [35] Patel M and Yalamalle S. Stock Price Prediction Using Artificial Neural Network. 2014.
- [36] J Maitland. P2p plagues service providers. <http://www.lightreading.com/ethernet-ip/p2p-plagues-service-providers/d/d-id/591062>, 2003. Accessed: 2016-10-03.
- [37] S Marti, P Ganesan, and H Garcia-Molina. SPROUT: P2P Routing with Social Networks. 2004.
- [38] P Maymounkov and D Mazieres. Kademlia: A Peer-to-peer Information System Based on the XOR Metric. *IPTPS '01 Revised Papers from the First International Workshop on Peer-to-Peer Systems*, pages 53–65, 2002.
- [39] Srivastava N, Hinton G, Krizhevsky A, Sutskever I, and Salakhutdinov R. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. 2014.
- [40] M Piatek, T Isdal, T Anderson, A Krishnamurthy, and Venkataramani A . Do incentives build robustness in BitTorrent? 2007.
- [41] Keralapura R, Taft N, Chuah C, and Iannaccone G. Can ISPs Take the Heat from Overlay Networks? 2004.
- [42] Kohavi R. A Study of CrossValidation and Bootstrap for Accuracy Estimation and Model Selection. 1995.
- [43] Dreyfus S. An Appraisal of Some Shortest Path Algorithms. 1967.
- [44] Hochreiter S and Schmidhuber J. Long Short - Term Memory. 1997.
- [45] Ratnasamy S, Francis P, Handley M, and Karp R. A Scalable Content-Addressable Network. 2002.
- [46] R Sit, M Semanko, R Kim, and E Basic. The Future of the Decentralized Model of P2P File-Sharing. 2000.
- [47] Fawcett T. ROC Graphs: Notes and Practical Considerations for Researchers. 2004.
- [48] Mikolov T, Chen K, Corrado G, and Dean J. Efficient Estimation of Word Representations in Vector Space. 2013.
- [49] R Toole. BitTorrent Architecture and Protocol. 2006.
- [50] McCulloch W and Pitts W. A Logical Calculus of the Ideas Immanent in Nervous Activity. 1943.
- [51] R Wattenhofer. Extending the Functionality of BitThief. 2008.
- [52] Bengio Y. Practical Recommendations for Gradient-Based Training of Deep Architectures. 2012.

- [53] Harris Z. Distributional Structure. 1954.
- [54] Y Zhou. On the Performance of P2P Network: An Assortment Method. 2011.

Acronyms

AS Autonomous System. 9, 10, 11, 12, 13, 28, 30, 31, 32, 37, 38, 48, 51, 52, 61

AUC Area Under the Curve. 22, 33, 53, 61, 69

C2P Customer - to - Provider. 11

CAIDA Center for Applied Internet Data Analysis. 11, 12, 13, 28, 31, 52, 53, 61, 63, 67

CDN Content Distribution Network. 3, 8, 9, 51

CPU Central Processing Unit. 29

CSV Comma-Separated Values. 26, 27, 28

ERD Entity Relationship Diagram. 11, 29, 32, 61, 62, 63, 67

IMDB Internet Movie Database. 7

IP Internet Protocol. 1, 2, 7, 8, 10, 61

ISP Internet Service Provider. 1, 3, 4

MLP Multilayer Perceptron. 14, 15

NLP Natural Language Processing. 4

P2P Peer - to - peer. vii, 1, 2, 3, 4, 11, 12

P4P Proactive Provider Participation for P2P. 4

RAM Random Access Memory. 18, 27, 29, 47, 61

RNG Random Number Generator. 28

ROC Receiver operating characteristic. 22, 33

SSH Secure Shell. 27

TNR True Negative Rate. 22

TPR True Positive Rate. 22

t-SNE t - Stochastic Neighbor Embedding. 21, 33, 38

TF Term Frequency. 14

TF Inverse Document Frequency. 14

VIOLA Video Consumption in Overlay Networks. 1, 2, 7, 8, 9, 11, 12, 13, 19, 23, 27, 29, 32, 37, 38, 37, 39, 40, 47, 52, 53, 61, 62, 63, 65, 67

VoIP Voice over IP. 3

List of Figures

3.1	Calculating the intervalID given a partial timestamp.	8
3.2	While both plots illustrate the popularity of a specific torrent, figure 3.2a depicts the global popularity using the <i>seeder</i> and <i>leecher</i> statistic. 3.2b details the localized popularity of the same torrent for the United States. .	9
3.3	Network showing connections between ASes sharing a file. Additionally to the AS identifier the number of IP addresses from that AS are denoted, i.e. two users belong to AS 1, while ASes 2, 3 and 4 have one peer each. All other ASes do not have a peer sharing this file and are therefore not in this graph.	10
3.4	Relations between different ASes. Each edge identifies a cashflow between connected nodes, where the customer pays the provider for routing traffic. Peers on the same hierarchical level do not exchange money for transmitting information. (based on [15])	12
3.5	Networks presenting the relationship between ASes from the CAIDA and the VIOLA data set.	13
3.6	Network graph for a $(L + 1)$ -layer perceptron.	15
3.7	Process depicting the operation an individual node within a Neural Network executes.	17
3.8	All plots show the quality of classifiers. Only looking at the plots, the farther the curve approaches the upper - right corner, the better the qualifier is. Comparing two curves can be achieved with the AUC.	23
4.1	Main components assembling the BitTorrent Weatherforecast application. .	26
4.2	Simplified process flow chart when querying the remote database.	28
4.3	ERD of tables created by the BitTorrent Weatherforecast application when calculating routing costs.	29
4.4	Query to determine individual batches to split a complex query into multiple smaller ones.	29

4.5	Query to retrieve data from the cluster after preprocessing it. The query computes two Joins, requiring more RAM than is currently available. $\{0\}$ / $\{1\}$ contain the number for year and month retrieved in the aforementioned query respectively.	30
4.6	String denoting a sparse array. The length of this array is 77220. All entries are zero, except the first one which is equal to one. Indices start with zero. While the first entry has to denote the size of the array, the order of the remaining enries is irrelevant, important for subsequent group by queries.	31
4.7	ERD of tables accepted by the Neural Network Learning module.	32
4.8	Implementation of an Autofill textfield. Typing in opens the dropdown menu and shows search results for the typed letters. By clicking on one of them, the result is selected and filled into the textfield. The button to move on to the next screen is not pressed automatically.	34
4.9	Interactive Plotting application. The upper part shows the world map where different colors indicate local popularity.	35
4.10	Interactive Plotting application. Here, the cost for routing are included as well as the popularity.	36
5.1	Errors of the PQ_3 stopping criteria.	38
5.2	t - SNE. Blue points contain “5.1”, red not	41
5.3	t - SNE. Blue points contain “Game of Thrones”, red not	41
5.4	Development of the training error. Early stopping PQ_1 has been applied.	42
5.5	Development of the validation error. Early stopping PQ_1 has been applied.	42
5.6	Final result of the regression using PQ_1 as Early Stopping.	43
5.7	Function learned by the PQ_3 module together with original data	43
5.8	Function learned by the UP_8 module together with original data	44
5.9	Function learned with the usage of a Moving Window	44
5.10	Function learned with the usage of a Moving Window of length 7	45
5.11	Lifetime of torrents recorded within the VIOLA data set	45
5.12	Lifetime of torrents recorded within the VIOLA data set	46
5.13	ROC plot validating the quality of this classifier. The area under the curve is 0.77.	46
6.1	Cosine Similarity between the torrent titled “Better Call Saul Complete First Season” to all other movies in the data set.	48

6.2	Example of a network, where using the shortest path algorithm produces an invalid path.	50
A.1	ERD depicting the structure of the VIOLA database.	67
A.2	ERD depicting the structure of the CAIDA database.	68
B.1	Relative number of nodes which are on shortest paths belonging to a country.	70
B.2	Errors of the PQ_3 stopping criteria.	70

List of Tables

3.1	An example of applying the Bag of words algorithms to a set of verses. The second entry of the resulting vector identifies the word "likes", which appears once in the first and twice in the second sentence respectively. . . .	14
6.1	Top 20 movies from the VIOLA data set which have the highest similarity with "Better Call Saul Complete First Season"	49

Appendix A

Databases

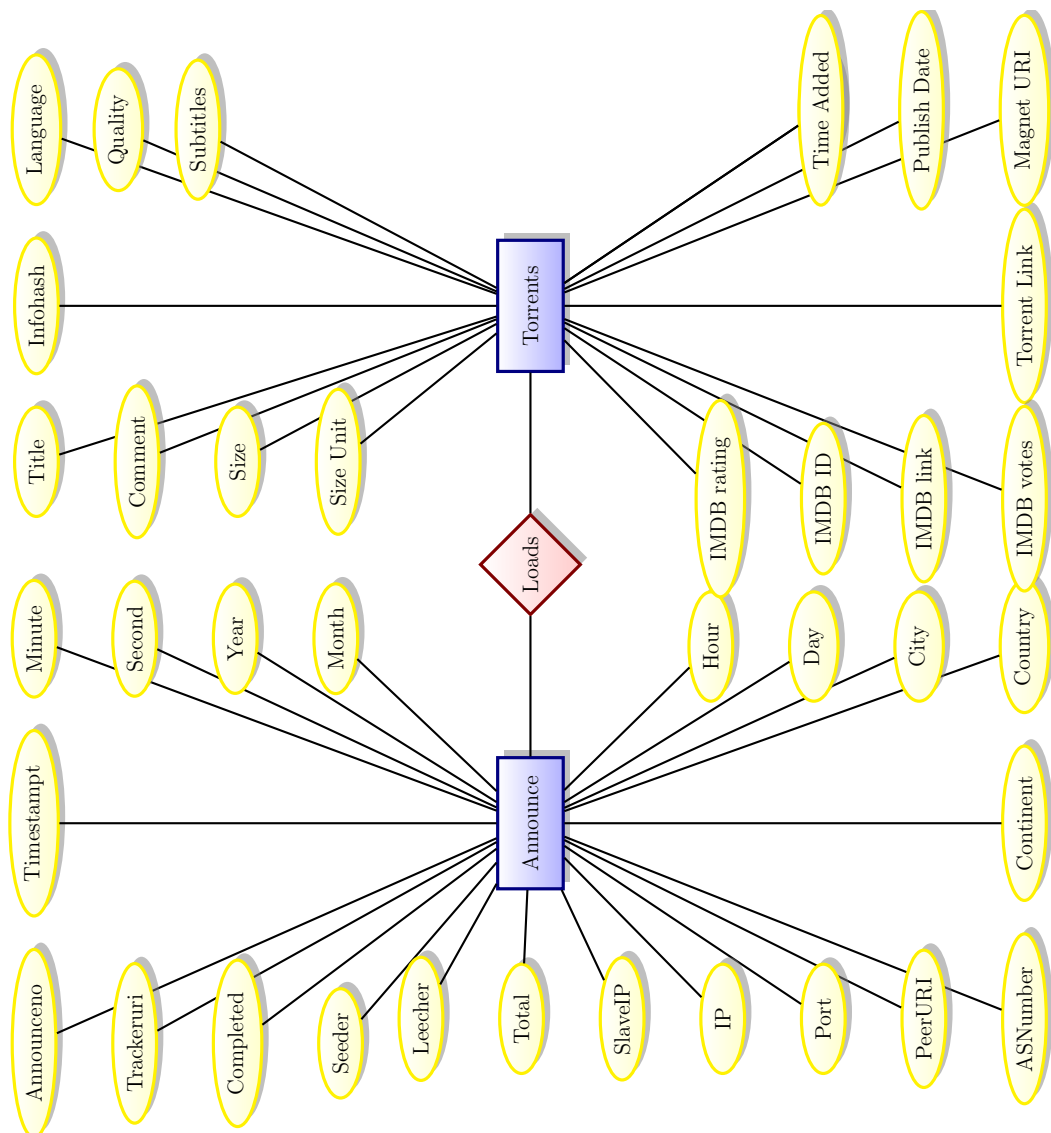


Figure A.1: ERD depicting the structure of the VIOLA database.

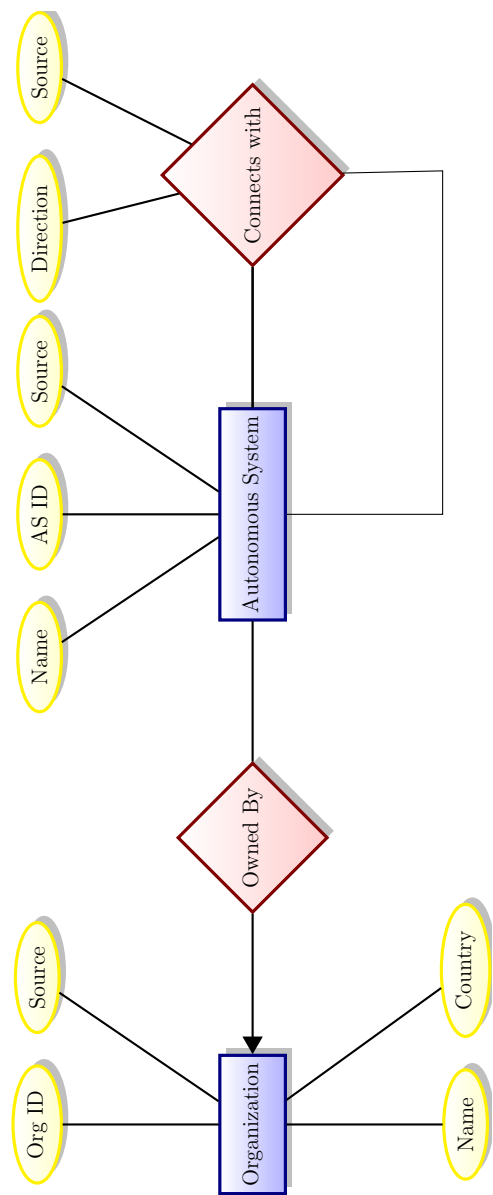


Figure A.2: ERD depicting the structure of the CAIDA database.

Appendix B

Plots

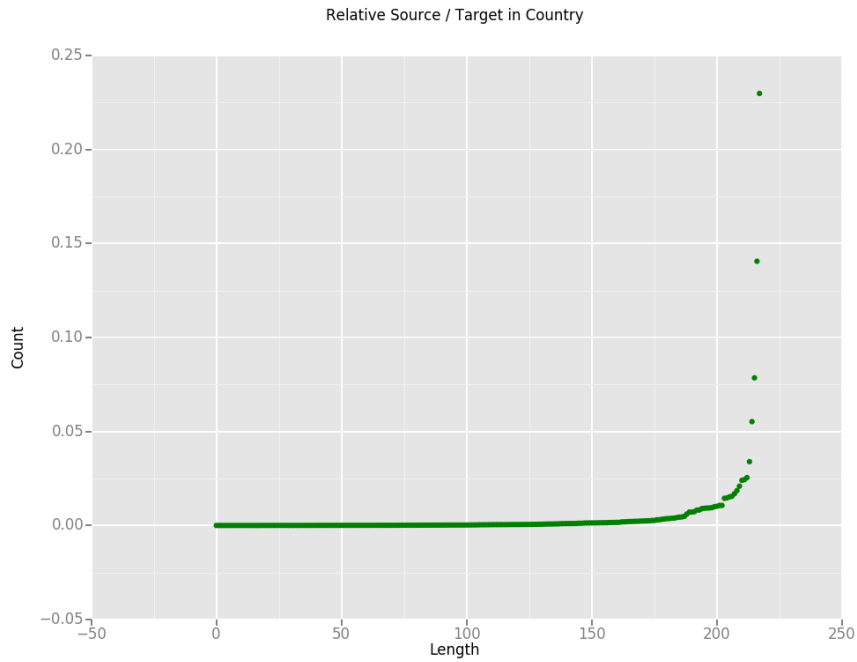
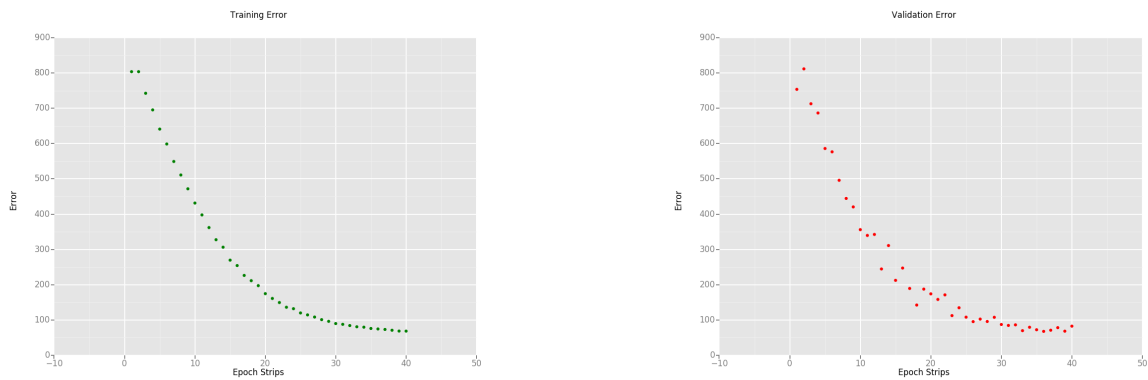


Figure B.1: Relative number of nodes which are on shortest paths belonging to a country.



(a) Performance of a random classifier. This is considered as the baseline, since randomly choosing a class is better than a bad and therefore no classifier. The AUC is 0.5.

(b) Performance of a bad classifier. Regardless of the threshold, the False Positives always outweigh the True Positives. The AUC is 0.0563.

Figure B.2: Errors of the PQ_3 stopping criteria.

Appendix C

Contents of the CD

- Abstract.txt
- Zusfs.txt
- Masterarbeit.pdf
- Python Code
- Latex - Files
- Literature
- Midterm Präsentation
- Database Dump