Verification and Implementation of Call Sequence Analysing Algorithm

Master of Science in Technology Thesis
University of Turku
Department of Information Technology
Communication Systems
May 2016
Santeri Toikka

Supervisors:
Seppo Virtanen
Petri Sainio

TURUN YLIOPISTO
Informaatioteknologian laitos

SANTERI TOIKKA: Verification and Implementation of Call Sequence Analysing Algorithm

Diplomityö, 67 sivua
Tietoliikennetekniikka
Toukokuu 2016

Työssä arvioidaan ja verifioidaan puheluiden luokitteluun suunniteltu Call Sequence Analysing Algorithm (CSA-algoritmi). Algoritmin tavoitteena on luokitella riittävän samankaltaiset puhelut ryhmiksi tarkempaa vika-analyysia varten.

Työssä esitellään eri koneoppimisalgoritmien pääluokitukset ja niiden tyypilliset eroavaisuudet, eri luokitteluprosesseille ominaiset datatyypit, sekä toimintaympäristöt, joissa kyseinen toteutus on suunniteltu toimivaksi.

CSA-algoritmille syötetään verkon ylläpitoviesteistä koostuvia viestisarjoja, joiden sisällön perusteella samankaltaiset sarjat ryhmitellään kokonaisuuksiksi. Algoritmin suorituskykyä arvioidaan 94 käsin luokitellun verrokkisarjan avulla. Sarjat on kerätty toimivasta 3G-verkon kontrollerista. Kahta sarjaa vertailemalla sarjaparille muodostetaan keskinäinen tunnusluku: sarjojen samanlaisuutta kuvaava etäisyys. Tässä työssä keskitytään erityisesti Hamming-etäisyyteen.

Etäisyyden avulla sarjat koostetaan ryhmiksi. Muuttamalla hyväksyttävää maksimietäisyyttä, jonka perusteella kaksi sarjaa lasketaan kuuluvaksi samaan ryhmään, saadaan aikaiseksi alaryhmiä, joihin kuuluu ainoastaan samankaltaisia sarjoja. Hyväksyttävän etäisyyden kasvaessa, myös virheluokitusten määrä kasvaa.

Oikeiden lajittelutulosten vertailukohteena toimii käsin luokiteltu ryhmittely. CSA-algoritmin luokittelutuloksen tarkkuus esitetään prosentuaalisena osuutena tavoiteryhmittelystä maksimietäisyyden funktiona.

Työssä osoitetaan, miten etäisyysattribuutiksi valittu Hamming-etäisyys ei sovellu tämän datan luokitteluun. Työn lopussa ehdotetaan menetelmää ja työkalua, joiden avulla useampaa eri lajittelija-algoritmia voidaan testata nopealla kehityssyklillä.

Asiasanat: 3G, koneoppiminen, protokolla-analyysi, muokkausetäisyys

UNIVERSITY OF TURKU
Department of Information Technology

SANTERI TOIKKA: Verification and Implementation of Call Sequence Analysing Algorithm

Master of Science in Technology Thesis, 67 pages
Communication Systems
May 2016

In this thesis, a Call Sequence Analysing algorithm is analysed and verified (CSA-algorithm). The goal of the algorithm is to label similar calls to groups for more accurate error analysis.

Thesis presents different machine learning main categories and common differences between studies. It also presents different related data types and environments, where CSA-algorithm is designed to operate.

CSA reads network management data series as input and groups similar series together. Algorithm performance is evaluated with the help of 94 manually sorted sample series. Series data is collected from live 3G Radio Network Controller. Two series are compared and a describing figure distance is calculated. This thesis focuses specifically to Hamming-distance.

Basing on the describing figure distance, series are grouped together. Adjusting the maximum allowed distance, which defines the limit how dis-similar two series can be before they are considered to belong into a same group, a cluster is formed. Increment of the maximum allowed distance also increases the number of false positives.

Clustering composition is evaluated against manually sorted reference clustering. Performance is presented as a function for maximum allowed distance.

Thesis shows, how selected attribute edit distance, does not perform as a distance metric, and the cluster composition does not reach accepted level. As a further study, as toolset and working methodology is suggested for faster prototyping.

Keywords: 3G, machine learning, protocol analysis, edit distance

Contents

Chapter 1

Introduction

Internet has become a standard home appliance and changed many aspects of life. It has changed how we communicate with each other and what kind of information we share to people. One of the factors, which have enabled this revolution of internet, is mobile phones and the availability of services.

Copper network has been too old for current bandwidth demands, and fibre networks are still under construction. In sparsely populated areas mobile network is used to deliver high-speed connections to households too far from modern network infrastructure.

3G has a rich development background. It has many flavours and the composition of different features changes from one operators network to another. New networks are still built in developing countries, but the 3G-protocol development itself has already halted. There will be no releases that would refine the fundamental operations. Any new idea or redefining change is introduced as 4G, or in the near future, as 5G; both being later iterations from the same family of definitions.

3G is done, but by no means perfect or vanishing into the mists of history away from our daily lives. The booming years of 3G-network building and mobile phone advancements have left us with a huge installation base of control cabinets and link towers.

These machines can generate an overwhelming amount of log data, which describes every step of the network operation in great detail. The logging data flow is so vast that it

is more convenient to teach another machine how to read the data for you.

Advancements in data handling, and general availability of different data sources, have helped form a modern study of machine learning. The idea is to give general instructions how to analyse the incoming data, but not tell any definitive answers. The algorithm should be able to make those discoveries by itself, and maybe find something new and exciting in the process.

One of those algorithms is analysed in this thesis. It is performing a simple function: Read in 3G signalling data and form groups of phone calls that follow a similar pattern. The Call Sequence Analyser (CSA) algorithm is analysed in great detail and an understanding on why it fails to execute this task is built over the following chapters.

1.1 Objectives and motivation

Operators have started to sell mobile broadband as an option to hardwired network as coverage to target location is easier to achieve. In addition, popularity of video services and smartphone usage has together increased the bandwidth demand.

Where there is 4G coverage, there is also 3G still present simultaneously. 3G is still the main carrier for voice calls. 3G and even 2G are still used if the more desirable 4G data connection fails. Operators are not removing the old infrastructure and established networks any time soon.

It is still profitable for network infrastructure companies to develop new products for 3G. For example the recently released Multicontroller Platform replaces six old full-sized 3G Radio Network Controllers (RNC) cabinets with a compact 4U rack machine. Protocol development has long since advanced to 5G, but new 3G-networks are still being deployed in more remote areas of the world. The protocol might be in maintenance mode, but it still makes sense to optimise various upkeep tasks.

Current network monitoring is centred on single failure analysis. Operators follow

closely a set of Key Performance Indicators (KPI), which are collected from various sources. KPI numbers are meant to describe the status of the network and act as a performance measurement. They are intended for people who are not interested in operation details, but do want to know what the overall status is.

When the time to solve problems comes, the support engineer reads the full message exchange logs collected from the target environment. These logs contain all messages sent between various inter-protocol interfaces, and are quite hard to follow without deep understanding of the exchange. It would be a lot easier, if the workflow could be changed from log reading to information discovery. There should be a system, in the background, that would remove the labour-intensive log parsing process and would provide suggestions about actual reasoning behind the discovered failure.

Analysing errors is always partially guesswork. From network perspective, a cell phone that just ran out of battery is behaving similarly than a phone without adequate signal. In both cases message exchange ends unexpectedly, but the reason for it varies. The difference can be guessed from the battery status reports, if available.

The ability to maintain and provide reliability throughout the lifespan of a deployment is as important as the technology behind a working network. And as it seems currently, the lifespan of 3G-networks is already expanding over the initial intent. Cloud thinking transfers the daily reliability from operators towards product provider as the monitoring and reacting to failure situations becomes more automated.

As operators are constantly searching options to improve operation efficiency, and competition in the market is volatile, subscription prices are low and the profit margins are thin. Every little bit that helps to shave down expensive maintenance costs are welcomed.

This thesis aims to verify a machine learning algorithm developed by a third party. The Call Sequence Analysing (CSA) algorithm is implemented with C# for initial testing purposes within existing log analysing platform Emil, re-written in Python for statistics and graph generation and core parts are presented in original C++ format. The usage of

distance metric in cluster labelling sets up an optimisation problem, which defines the balance between the number of clusters found in the example data versus the composition accuracy. A manually sorted data set is used as reference when the validity of a single cluster is examined.

The objective of graphed data is twofold. Primally, it shows that the clustering starts to happen and the algorithm is grouping example data when more different series are accepted into a single cluster. This is shown as a decrease in clusters found in the same data set. Secondly, the data shows how the clustering accuracy changes when different series are grouped more generously.

1.2 Outline of the thesis

The structure of the following chapters follows loosely the CRISP-DM process, or Cross Industry Standard Process for Data Mining [1]. In process terminology, work starts from business understanding followed by data understanding. If the data suits the problem found in the first step, process continues to data preparation. Preparation step can generally be very time consuming as it involves manual labour in the form of scripting and media conversion and interpretation. After preparation, a model is built and evaluated. If the model fits the needs, project can be deployed. If it does not fit, a necessary number of process steps are iterated over until the results are satisfactory.

The thesis begins to build domain and project knowledge by presenting background information in the first chapter titled ”Introduction”. A brief look at the development history and general network architecture is analysed in the second chapter, ”3G-network history and architecture”.

In chapter 3, ”Machine learning”, the thesis focuses more to project specifics by introducing and describing different types of learning techniques. In chapter 4, ”Edit distances in application context”, a general analysis of the data and what possibilities have to be

considered is presented. In this chapter, a key attribute distance is introduced with key design choices

Proposed solution to the problem is presented in chapter 5, ”Call sequence analysing algorithm”. The most important part of the algorithm verification process is to understand how the distance metric in CSA algorithm operates and how it is calculated from the data. Therefore the Weighted Hamming distance is analysed thoroughly and presented in comparison to the similar Wagner-Fisher distance metric. The Wagner-Fisher distance is used later in data analysis, when clustering quality is discussed.

In this chapter, the application of Weighted Hamming distance to the signalling data is also presented. Key parts of the CSA algorithm are shown and analysed. The examples are written in C++, but are not too complicated to follow, if the reader has moderate understanding of programming. If not, functions are analysed on a higher level in the related chapters around the code, and the examples themselves can be left alone. The last a part of this important chapter introduces a new term, ”flat clustering”, a term that will be clarified later, when examples are analysed.

Chapter 6, ”Verification of the CSA algorithm”, starts with a general overview of the tools used for analysis and verification work. The testing plan is discussed and results of the analysed data are displayed with a varied commentary related to the topic.

Last chapter, ”Conclusions”, summarises the discovered aspects of the verification work. As a part of the preparations work is already done and the end goal is to verify the provided model, the CRISP-DM iteration step of the process is not applied.

1.3 Related work

There is quite a little published work about 3G-network diagnostic. This is might be due to the fact, that the network equipment is not accessible to a great audience. The main customers, large ISP’s, usually use the help and knowledge of the equipment provider.

Therefore, the know-how is vendor-specific and commonly sold as supporting a service.

The only published article doing similar work in the same environment as this thesis is focusing on a specific Iub interface. In the article ”Signalling analysis of Iub interface” [2], the raw binary stream is decoded with Wireshark and the protocol information read out from it for further processing. The article focuses on decoding information in Frame Protocol (FP) in UMTS Terrestrial Radio Access Network (UTRAN) signalling used between Iur and Iub interfaces. A Protocol Data Unit (PDU) is discovered in the stream, but the article does not go further on analysing the contents of the extracted data.

The algorithm analysed in this thesis uses a novel weighting method as a base for decision-making. In literature, there are no direct references to this subject, as the chosen methods are usually highly application-dependent. There are similar handlings implemented in two of the following algorithms, WhRank and WHam-method.

Weighted Hamming distance ranking algorithm (WhRank) [3] has a similar idea of giving different weights to different positions in the windowed series, but it focuses only on binary data. WhRank focuses on pattern recognition and image processing, and therefore the weight is selected based on pixel position rather than placement in a series.

The WHam-method [4] is similar to WhRank, but instead of predefined constants, the weights are calculated from a training set. The training set is used to determine the importance of different binary codes, and later, as weights in hamming mask.

A different approach to the problem of monitoring is to analyse 3G-network error conditions from the client side. A Danish university did just that with the help of an Android application. The field measurement [5] was done by driving around the Aalborg University campus area while measuring different parameters visible to the User Equipment (UE). Signal parameters like strength and bit error rate were measured along side with packet access throughput. The results were collected centrally into a server and the locations of the call terminations were rendered onto a map. In the results analysis, they discover that 3G calls drop more often than the 2G counterparts done during the same

drive.

Another common user plane analysis can be done by measuring IP traffic over the network [6]. In the paper, researchers use a client application and a corresponding server to measure network throughput when the 3G link is in different speeds. They noticed, that when the network is switching between different rates, or during a handover, data throughput is zero during several seconds. This behaviour is acknowledged and fixed on Radio Network Controller (RNC) side.

There is one commercial solution that offers mobile network optimisation and troubleshooting functionalities called Xeus Pro. It is able to parse Nokia-specific monitoring files and outputs diagnostic information about the particular network portion. The application is the result of advanced reverse engineering, as it decodes messages from an encrypted file which has a publicly unknown binary structure. Taken the nature of the solution and the high cost of licensing, the product is not discussed in this thesis any further.

Chapter 2

3G-network history and architecture

In 1991, the first 2G network was launched publicly and ETSI had already started to standardise the next iteration. The work started with the name of Universal Mobile Telecommunication System (UMTS). At the same time, The European Commission was funding its own programs: RACE I, RACE II, and ACTS. Individual countries also had their own programs aiming for the next generation solution: US, Japan, and Korea to name the most notable ones [7].

The final specification began to take shape in 1996, when Japanese Association of Radio Industries and Businesses (ARIB), and later ETSI in 1997, decided to go forward with Wideband Code Division Multiple Access (WCDMA).

To combat research effort diversification, the biggest companies joined forces and formed 3GPP to coordinate standardization work. Initial participants were British Telecom, France Telecom, Telecom Italia and Nortel Networks, but they were soon joined by Lucent, Ericsson, Nokia and others.

There are multiple proposals for 3G-network specification. The proposition from ETSI and ARIB became the most popular one, as it was based on an already deployed GSM MAP network and could provide all the old services the current users were using. This WCDMA-based system was renamed to Universal Terrestrial Radio Access Network (UTRAN). At the same time, USA and Korea decided to go forward with CMDA2000

family. The most important factor was, that the planned spectrum in the proposal was better suited for the spectrum usage in the US. As it was compatible with the IS-95 standard used in the USA, it could coexist with it in the same spectrum.

The 3GPP Release 99 is the first standard that is considered to be 3G. It describes a CDMA network with voice, video, SMS, WAP, e-mail, web, MMS, and streaming services. Each of these services is handled as a separate feature in the standard and has support in message protocols. Although most of the users currently use only voice, SMS, and raw packet transport features of the specification, to be fully protocol-compliant, one has to implement all features.

Since Release 99, there have been several revisions to the standard, some of which are marketed as 3G, some as 3.5G, and others are sold by the most distinctive feature in the particular release like WiMax.

After Release 99, the numbering tied up to the planned release year, was replaced with canonical numbering. Next version was Release 4, which was released in the second quarter of the year 2001. Originally named Release 2000, the most important feature is an all-IP core network.

A year after, Release 5 introduced IP Multimedia Subsystem (IMS) and High-Speed Downlink Packet Access (HSDPA). IMS itself is, interesting bundle of specifications and interoperability. There are several concepts, which were planned to change how mobiles are used in the future. Keeping in mind that this specification was released in 2002, when having a colour display in a cell phone was considered an upgrade, IMS was groundbreaking.

IMS includes concepts like IP Multimedia Private Identity, Globally Routable User Agent URI, Application Servers, and Media Servers. The core idea of this complex structure is to identify a user by its unique identifier in combination with the UE identifier. It is, in its own way, an alteration to traditional TCP/IP network abstraction layers and encapsulation model. Instead of routing packets to an IP address, IMS would deliver media

content for the user. Instead of transporting YouTube videos, IMS would transit video content and possibly take a transit tax with the help of supporting a charging infrastructure. Instead of keeping track of a user session with HTTP Cookies or some alternate method, the user would always connect to the internet through their own home network, hene reusing the user identifier, ISP media services, and data cap counters. IMS did not really take off on a larger scale partially because HSDPA enabled all the previous without the need for additional complexity.

HSDPA was created to address the growing demand for downlink speeds in User Equipment (UE). As mobile internet grew to be a thing with this release, HSDPA gained multiple revisions and improvements later in the succeeding releases. HSPA+ and Dual-Cell HSDPA are both incremental improvements. HSPA+ was introduced with Release 7, five years after the original. DC-HSDPA came a year after with Release 8.

While the names are market-driven, and designed to sell the new technology to the consumer, the basic idea behind upgrades has stayed the same: Add more information into the modulation and utilise a greater number of carriers simultaneously.

The first version of HSDPA operated with five to fifteen channel codes within a single cell. It is modulated with a 16-QAM (Quadrature Amplitude Modulation) or optionally with Quadrature Phase Shift Keying (QPSK). Available channels result in maximal data rates from 0.9 Mbit/s to 14.0 Mbit/s. The lowest one being a combination of five channel codes using QPSK, and the highest 16-QAM with fifteen codes. These are theoretical maximum speeds that can be achieved in a physical layer.

Although Release 5 is dated to 2002, it took several years before HSPDA was deployed in operator networks. It took five years before the next iteration of data protocol was released. With Release 7 in 2007, 3GPP added options for 64-QAM and introduced Multiple Inputs & Multiple Outputs (MIMO) techniques in data transfer. MIMO takes advantage of multipath propagation and enables simultaneous communication through multiple channels within a single cell. Although it is possible to use it only with 16-

QAM within the specification, it offers more bandwidth for the UE. Release 7 set the best possible theoretical rate to 28 Mbit/s.

Release 8 followed a year later, adding tighter channel definitions. Its greatest change for mobile data was the introduction of two-cell simultaneous communication named Dual-Cell HSDPA. With this release, the maximum bitrate was pushed to 42.2 Mbit/s. Release 8 ended the pure 3G specifications and transitioned to more market-hype-driven era, as it is the first Long Term Evolution (LTE) release, also known as 3.5G. Releases 7 and 8 paved the road for the mobile boom. Based on the Ericsson mobility report from 2012, mobile data traffic doubled each year between 2009 and 2012.

Releases are not, by any means, mandatory nor implemented in fully. As can be concluded from Janne Suominen's thesis [8], there are Finnish operators which are running 3G and 4G networks in parallel, and are still using circuit switched networks in the background. Reason for this is partially due to Finnish emergency call reliability requirements, and partially due to an operator trying to push old technology as far it can.

2.1 Architecture

The complexity of the 3G architecture and protocol itself is a sum of multiple variables. One factor has been the goal to create independent markets for NodeB devices (base stations) and for Radio Network Controllers (RNC). The second of the limiting factors in how the 3G specifications were designed, was the technical capabilities available at that time. It has had to evolve when new, more efficient codings, and more capable equipment are under development.

The air interface has the most variance throughout the development history of the standard. It can be divided into four major categories based on the channel access method: Time Division Multiple Access (TDMA), Frequency Division Multiple Access (FDMA), and Code Division Multiple Access (CDMA).

Time division shares the frequency by giving out short timeslots to all users, and each UE transmits right after each other. This was already used in 2G. In 3G, TDMA can be combined with Frequency Division, where multiple channels are used in similar manner. User transmission hops between frequencies and timeslots in a predetermined pattern. A lesser number of timeslots and used channels serves a greater audience but lowers the throughput in return.

Code Division is one of the more interesting features in use. Instead of physical division, the signal is spread with code, and user equipment is transmitting at the same time. Code division uses a pseudorandom spreading code, which is generated with higher rate than the transmitting message symbol. For example, the single binary 1 can be presented with multiple bits like 101101, and the receiving base station knows that the previous series was 1, because it uses the same pseudorandom generator in sync. The next data bit 1 would be encoded differently, but the base station would still be able to receive it, because it has also moved forward in the pseudorandom sequence. CDMA is the most popular technique in North America and in Europe.

Duplicity defines which device transmits and when. If UE and NodeB share the single frequency, and both are using it for transmitting messages, simultaneous transmission will cause collision. The problem can be solved with Time Division Duplex (TDD), a half-duplex where UE and NodeB take turns on transmission. Another common method is to use Frequency Division Duplex (FDD), which uses different frequencies for uplink and downlink. This enables full-duplex communication between UE and NodeB.

Two major deployments development branches, Universal Mobile Telecommunication System (UMTS) and CDMA2000, share some of the techniques, but not all. UMTS uses HSPA for data, where as CDMA2000 relies on Evolution-Data Optimized (EV-DO) originally developed by Qualcomm. CDMA2000 duplexity is frequency-based (FDD), as is one of sub-branch techniques of UMTS, named Wideband CDMA (W-CDMA). The rest of the two sub-braches of UMTS, Time Division CDMA (TD-CDMA) and Time

Division Synchronous CDMA (TD-SCDMA), are both using TDD.

Both UMTS and CDMA2000 are code division based (CDMA), but differ in architectural choices. UMTS is asynchronous, where as CDMA2000 is synchronous, and requires exact clock synchronisation between different NodeB's. CMDA2000 was clearly loosing the popularity contests for a number of potential customers, but the importance of both countries adopting it could not be ignored. It is still used in parts of Africa, Northern US, Canada, and for example in India.

The variety inside 3G starts to pile up when all different factors are summarized. Which Release does it follow, what duplexity, modulation and frequency allocation are being used? Does the operator still rely on circuit switched network, what interface is used in data layer, does it even use code division multiplexing? Does the architecture support Multimedia Messaging Services (MMS), does it support Dual-Carrier HSUPA, which is defined in post-LTE release 9?

Answer to a simple question "Is it 3G?" depends on who is answering. A young engineer who pays respect to the standard would answer "It is complicated.", where a seasoned veteran who does not care that much anymore, would answer "Yes. It is.".

Fortunately, there is a major difference between 3G and 4G architectural design. In 3G, behind every NodeB is a Radio Network Controller (RNC), which does all the heavy lifting in terms of network management. It is a gateway between ISP core network and cellular mobile network level. RNC holds all the status information of the phone and makes decisions, which NodeB should handle the communication. In 4G, NodeB and RNC functionalities are merged together to a single unit known as eNodeB.

There are reasons why 3G architecture is as described. In analogy to AM receiver design, where receiver is simple and cheap to produce, the sending station is more complex and expensive. Similarly, in cellular network, the UE is the dumbest element from the view of network management. The only authorative choice the UE can make, is to choose which network orders it chooses to follow.

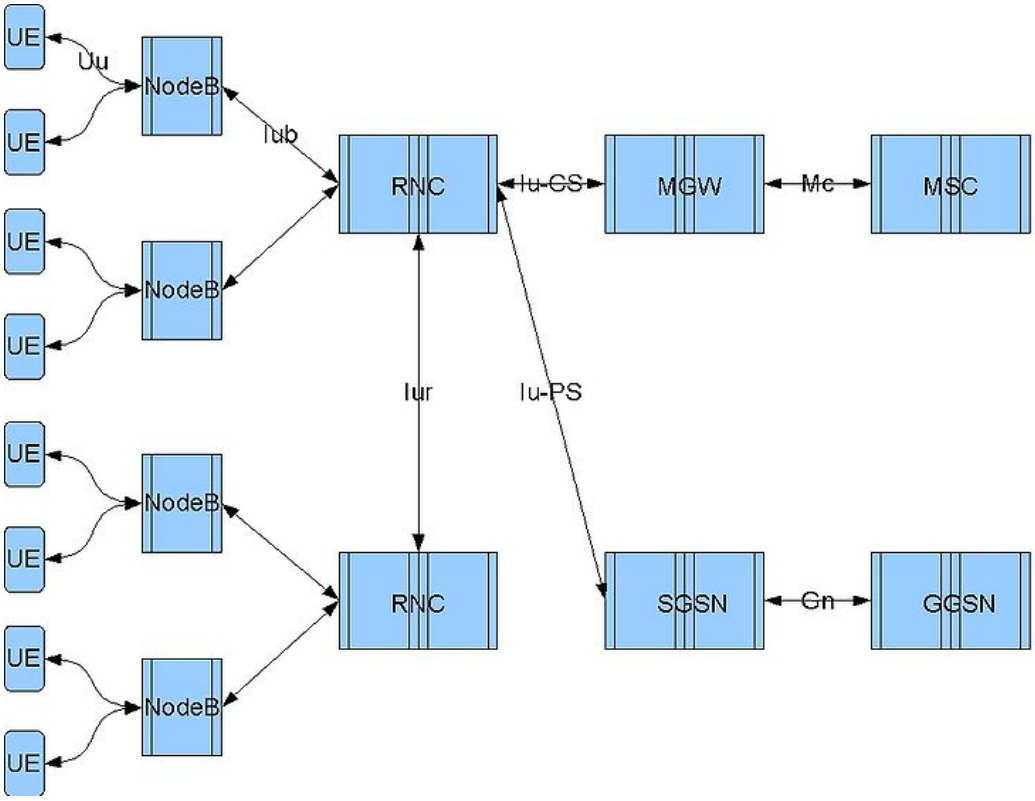

Figure 2.1: 3G RNC network architecture and protocol interfaces

Second dumbest element in the network is the NodeB that handles the air interface. One station is not aware of its surroundings and communicates only with a single RNC. NodeB receives frequency settings, and for example, updates local subscriber information to HLR (Home Location Register) through a specified interface.

The director of this orchestra is the RNC, which for example handles UE handovers between two NodeB elements. If the subscriber, the UE, crosses over the cell boarder, the RNC hands the call management to another RNC. Therefore, controllers from different vendors need to be interoperable through a common interface.

RNC is the gateway between Carrier Network (CN) and the UE. Due to backward compatibility and technical environment present when 3G was launched, the standard supports also legacy circuit switched CN. This separation is visible in the interface standard between CN and RNC, but also through out the whole specification. Only later Release 4 of the specification defined support to All-IP networks [9].

One RNC and multiple NodeB's form a single Radio Network Subsystem (RNS). Two subsystems communicate through Iur interface, which is used to handover UE control when UE moves out from reach of a single RNC. Roughly 1,500 control messages in different interface layers are required to set up and upkeep a single phone call in a 3G-based network.

Advantages over 2G are numerous, although the division between 2.5G and 3G is as blurry as is the division between a 3G and a 4G network. The W-CDMA system can be described through a number of key attributes that are not available in the previous standards [10]: Advanced power control and handovers. If all NodeB clients transmit with the same power, independent of their position, the signal can mask other weaker signals. It shortens the device battery life when excessive power is used in transmission. Also, too weak transmission power results to poor channel performance. Therefore, W-CDMA, and others, updates the power levels multiple times within a second, to keep it in optimal level.

Soft and softer handovers gives the handset ability to communicate with two or more cells at the same time. Neighbouring cells can be governed by single RNC, or they can be under control of two RNC cabinets. Different manufacturers can produce these RNC's. The specified Iur-interface is used to transfer the responsibility of the UE to other controller. At the background, RNC's can negotiate the planned jump, and transfer the access to for example circuit switched call on the fly, without user noticing any interruptions. RNC can even control the device, as of its own, through the Iur interface, if the handover has not taken place yet. This enables highly mobile movement within network.

One of the cross-standard aspects in the European 3G, UMTS-FDD specifically, being based on the preceding GSM MAP architecture, it can hand over the calls to a legacy GSM network. As Ericsson speculated in 2001 [10], this would be an important feature of network design. Operators can start the deployment by rolling out networks in the high-density area first, and still have full coverage around the operation area. This fall

back feature is still in use and its operators are hesitant to let it go as proven recent by research efforts [8].

Chapter 3

Machine learning

In the modern world, the amount of data available in a machine-readable form is increasing rapidly. Learning algorithms can be used to discover new properties from the seemingly endless sets of data, which cannot be handled by a limited human mind. It can be used to learn personal behaviour models, which can be used in interface building and customization. Learning techniques helps a capable operator to provide more meaningful content for the end user. The process is automatic; it is objective, and based on set of rules made to interpret the inputted data.

Machine learning is a broad term defined by a variety of sources. Generally, it is a set of computational methods used to enhance performance or to generate predictions from a given data. Variability and degree of uncertainty is always present with these methods. If we could have an absolute classifier, that can define if the sample belongs to a certain group, or if the outcome is known, we would not be using any methods that incorporate a degree of probability within. In this thesis, algorithms are used to study data set so huge, that it would not make sense to read it, for example.

Machine learning is a set of algorithms that are performing predications. Predictions are made based on the different attributes found in the dataset with the applied knowledge of the operation domain. For example, a classic training exercise classifies 150 observations of a flower to three subtypes based on the different attributes found in one

observation set. Attributes are sepal length, sepal width, petal length, and width. By comparing just the size of the flower petals to the converging sepals, one can divide the observations to three groups. Later, the grouping accuracy can be confirmed by observing the colour of the petals.

As an analogy, a person could sort the box full of flowers, eyes closed, just by feeling the shape of the flower, into three baskets. When the sorting is done, one could open their eyes and confirm the sorting result visually.

There will be errors though, if the relative size of the flower petals is overlapping between two subtypes. Without the luxury of observing the colour of the petals, it is hard to separate the types by hand and with statistical algorithms.

Feeling the flower shapes in your hands could lead to discovering new attributes, like the weight of the flower, the feel of the covering sepals or the stiffness of the rod, which would tip the decision towards one of the two groups. This gained knowledge could be used to enhance the accuracy of the prediction and could be taught to the algorithm as well. In both cases, machine prediction and human touch, it would increase the complexity of the decision process.

In the area of computer sciences, these machine algorithms are measured through space and complexity. How much memory it takes to run on a computer, how much resources and CPU cycles does it consume? In addition, also the data complexity of the observations has to be noted when comparing different aspects of machine algorithms together.

Common examples of the use of machine algorithms include:

- Text or document classifications, where program tries to determine if the incoming email is spam or not

- Optical character recognition, where based on the closeness of pixels in binary data, an algorithm tries to guess what letter the sample presents

- Recommendation systems in search engines and in shopping portals
- Anomaly search in bank account usage

Different learning scenarios are a good way to separate the field of algorithms. These scenarios differ in what type of data is available for the process and how it is used. There are multiple definitions for the general classification, but the most popular one is the three-fold categorization represented here.

Unsupervised learning, where the algorithm receives only unlabelled data and tries to make predictions about unseen data points based on the domain knowledge; supervised learning, where the learner receives pre labelled data, and makes predictions; and semi-supervised learning, which includes all the other scenarios that does not fit into the first two categories.

Semi-supervised learning can be fed with labelled data in the initialization process, and it can change its behaviour based on the incoming data. It is, by far, the most application-dependent category of these three.

3.1 Unsupervised learning

Unsupervised learning can be used to discover unknown patterns in the input data. There is no feedback mechanism that would guide the decision process towards a desired end result. The unsupervised learning algorithm tries to explain and summarise key features of the data.

The method for the explanation process varies based on the application, but it is always set beforehand. It can be, for example, a density estimation function, that assumes similar properties within a sample set that are close to each other in a given area when comparing a given set of attributes. Hierarchical clustering and k-means clustering are good examples of unsupervised learning techniques.

Hierarchical clustering can start from the top, where all the samples are in one cluster, and continue towards the bottom, where the desired separation is achieved. This behaviour is divisive. Opposite of divisive is agglomerative approach where, at the start, each sample is its own cluster and through out the recursion, the clusters are combined if the attributes fall within the specification.

This specification, which separates the two comparison points, is called a distance. It is not a metrical number representing the shortest path between two locations, but rather an abstract figure describing the similarity or dissimilarity of two data points. It is a key latent variable that is calculated from the two data points and describes their relation. Distance can be Euclidean, but it is not restricted to that. The definition and the use of distance is revisited many times in the following chapters, and is left without an in-depth explanation for now.

Another common learning strategy is to use clustering with a k-means [11] vector quantisation. Its process can be described in three operational steps: Centroid selection, partition, and iteration. Centroid is a geometric centre, or a mean, on a selected plane. It is a value that represents a group of other values around it. Between each centroid, and any other data point in the given set, is a distance.

As a real-world analogy, these data points could be cities on a map, and centroids would be the regional capitals. In an abstract graph world, centroids do not have to be fixed to two dimensions. There can be multiple variables defining the cluster composition.

In k-means clustering, the first centroids are randomly selected from all the available data points. After the initial selection, each data point is compared and grouped to the nearest centroid. This divides the data points into partitions or clusters. New mean centroids are selected to represent a selected group. The process is iterated over until convergence has been reached.

How many centroids should be chosen is application-dependent and should always be evaluated as a part of the analysis. Too few centroids produce groups with high variance,

and too many will obscure the findings. The goal, and the end result of unsupervised learning is to have the best possible groupings of data.

To understand when unsupervised learning should be used, or for what applications it is suitable, a good way is to study its algorithmic complexity. Agglomerative clustering has a complexity of $\mathcal{O}(n^3)$ and divisive has a complexity of $\mathcal{O}(2^n)$, meaning that the divisive method is the worse of the two. Both methods have a complexity, which increases fast when the incoming dataset increases, making hierarchical methods unsuitable for processing great amounts of data.

The other example, k-means clustering, can be NP-hard even with two clusters, but in many cases its complexity can be reduced to more tolerable levels. If the both dimensions are fixed, k-means has a complexity of $\mathcal{O}(n^{d+1}log(n))$, which makes it a bit more suitable for continuous data processing.

3.2 Supervised learning

In supervised learning, the training data must be labelled in a way that the program can discover the desired function from the dataset [12]. Labelled data can be either a manually sorted reference, or output from previous iterations of a possibly unsupervised algorithm, of which the output is verified. Training examples can be used to restore the known good state, or they can be used in algorithm tweaking over multiple iterations.

The goal of supervised learning is inverse compared unsupervised learning. Instead of finding the best possible sets of data points, the goal is to discover the function leading to the answer [13]. Ideally, the function derived from a small training example set would correctly label all the new instances of unseen data.

There is always an inaccuracy factor involved, when working with learning data examples. In supervised learning, the bias-variance trade-off is present in each decomposed function. It is very hard to produce accurate divider functions that partitions the sample

set exactly to a desired number of portions. There are always data points that are hard to fit into any of the main groups. Odd data points can be inaccuracies or just a natural phenomenon. These points prevent algorithms generalising the training set.

Bias in the learning algorithm produces underfitting. Bias is an erroneous assumption made from the example data, that later obscure the findings made from new data. It sets the decision boundary off from the actual level.

Variance can cause overfitting. If variance is high, the data is noisy, the discovered function possibly adapts to the noisiness of the data instead of the trend of data points. The overfitted function tries to be too accurate when a more generalised approach would be more desired.

The bias-variance trade-off is always present in supervised learning algorithms as it is typically impossible to optimise both variables at the same time [14]. Methods for having high variance can be more accurate in representing the training set, but are more inaccurate in analysing new data. Methods with low bias can be more complex, and hence, more time consuming.

The amount of training data also affects how and what the algorithm learns from it. The amount of data should match the complexity of the problem. More difficult problems should be solved with a higher amount of training data. For simpler problems, a lesser amount of data is sufficient, if it can be ensured that the function does not become overfit. The correct amount of data is therefore application-dependent.

High dimensionality in training data can make the function too complex. In the previous flower example, one observation of the flower contains different attributes seen in the petal and sepal shapes. Each of these attributes is a one dimension.

For verification purposes, it is convenient to display two of these dimensions together at the same time; Sepal width as a function of the sepal height would produce a scatter plot showing similarly shaped sepals close to each other. This would indicate that they are of the same type. But this might not hold true, if some dimension containing the distinctive

feature, like sepal thickness, is ignored.

Where the human perception is limited to visualising over three dimensions, machine learning algorithms often are not. There is a trade-off using high dimensionality in training data. Using too many features could lead to a high variance and overfitting. It also creates unnecessary complexity. Which attributes are more relevant for the learning task can be discovered with the help of feature selection methods or by using domain-knowledge.

There are two different approaches to find the function based on the training set: Empirical risk minimization, which is a principle of statistical learning theory, and structural risk minimization [15]. Both of these approaches uses statistics as a base and derives a probabilistic function out of the training set. Details of these approaches are outside of the scope of this thesis as the method under verification is not strictly supervised and does not rely on probabilistic behaviour.

3.3 Semi-supervised learning

Semi-supervised learning is a combination of both of the previous learning methods. It uses both labelled data and unlabelled data together without restricting to a single type. Usually, a small amount of labelled data is used to initialise a known good state of the decision algorithm. The process does not aim to discover the best possible function for partitioning, nor best possible grouping of the data. Instead semi-supervised algorithms often try to be self-learning. A combination of small amount of labelled data with learning heuristics is found to provide more accurate classification results [16] [17].

Self-learning algorithms can be either inductive or transductive. Inductive learning can be, for example, constantly improving the decision process that classifies input data. The goal of the inductive algorithm is to create general rules from observations and apply them to the rest of the data. There are indications that inductive learning can lead to false results when the analysing data [18], and caution should be used in the development.

Transductive learning tries to solve a single problem limited to a certain set, where as inductive learning tries to solve a group of problems. Transductive learning is often more complex and requires all sample points to be analysed if one is added to the set. While computationally expensive, it can give out better estimations with a lesser number of examples. Its reasoning is based on observed changes.

Transductive algorithms can be used in verification purposes, or in data the exploration phase. It is considered to be the lesser of the two, as it does not generalize answers to form that could be reused later. In contrast, transduction solves the immediate problem right away and does not try to generalize the answer.

In a semi-learning situation, where there are multiple points, but only part of them are labelled, inductive approach would evaluate only the labelled ones, and make assumptions based on only small number of examples. Transductive algorithm would evaluate them all. This gives somewhat more protection against biased labelling.

There are generally two different types of transductive algorithms. Algorithms that try to solve the problem by seeking regression within the data, or algorithms trying to predict labels for individual data points based on the various rules.

Continuous labelling algorithms are often extended manifold learning algorithms with added supervision. Discrete labelling algorithms are often created by extending a clustering algorithm with moderate level of supervision [19].

The labelling of the clusters can be achieved either by starting from the top or the bottom. When starting from the top, all samples are considered to be in same cluster and following iterations of the algorithm divides the cluster into smaller clusters. This is called partitioning transduction.

The bottom-up method starts from discrete examples, where there is only a single member in each cluster, and combines samples to bigger groups. This is called agglomerative transduction and describes closely the idea behind the Call Sequence Analysing (CSA) algorithm analysed in this thesis.

The most important prequisite for semi-supervised learning is that the example data used in the initialisation must be relevant to the classification problem. A set of assumptions has to hold true in the unlabelled data: Semi-supervised smoothness assumption, cluster assumption, and manifold assumption.

Smoothness assumes that if the two points x_1 and x_2 are close in a high-density area, then so should also be points y_1 and y_2. The clustering assumption defines that if two points are in the same cluster, they are likely to have the same label. Manifold assumption is for high dimensional data, which is usually projected to lower dimensions. It reduces complexity, but also reduces accuracy. The assumption is that the lower-dimension data is roughly representative of the original. Manifold assumption is not needed in this thesis, as the data is only one-dimensional.

Chapter 4

Edit distances in application context

There are a vast number of different failure counters for different situations. The problem with the current environment is that different scenarios of failures updates the same failure counter, and the actual analysis of the conditions and situations are left to the operator. Counters are only showing symptoms, not the actual cause behind. For example, an increment of a general link failure counter can be a result of multiple different situations, like intercommunication failure, bad reception, or the UE battery running flat. Some causes can be ruled out with logic or, for example by monitoring client battery life. All possible situations are directly visible and the analysis of different failure scenarios requires deep understanding of the protocol.

The goal of the algorithm, analysed in this work, is to achieve the same results in error diagnostics as the network operator would conclude by inspecting the log messages manually. As a relaxed target, the algorithm should be good enough to be able to group similar error conditions together, and distinguish new cases from previously collected and analysed samples.

The first obstacle is the 3G-protocol itself. Protocol behaviour can be described as chatty. To truly understand design choices made during the development of the CSA algorithm, we first have to understand the nature of the operation environment.

4.1 Operation environment

Considering architecture discussed in chapter 2, everything interesting related to network status either happens in RNC or can be observed through signalling visible to RNC. It is important to remember, that when we speak generally about 3G-networks, there are actually years of different networks with different configurations available.

Early design documents write about the T1 line (1.544 Mbit/s) and the connections to Circuit Switched networks, while a bit more modern documents prefer Ethernet, fibre in long distances, and packet switched network. These are only propositions, it has to be noted. It is hard to put a figure on how much overlapping logic RNC has to implement to be able to support varying deployment scenarios and advancements in protocol over the years.

Logging RNC operations bit cumbersome. Messages themselves are short and do not carry much state information. In order to be able to analyse what has possibly gone wrong during the UE operation, a full message log is needed. RNC functions are similar to a complicated state machine, which reacts to different messages accordingly. Very little data is kept in memory about a single on-going operation.

According to a senior engineer, post-analysing error situations did not get enough attention in the design phase of the standard, and compromises were made. RNC cabinet does not have enough memory, or enough storage space, to save call data extensively. It can push the data over Ethernet to an analysis server, but there are limitations.

When the 3G-networks were originally deployed, a RNC cabinet installation reused as much of the existing infrastructures as possible. Often this meant using multiple subscriber lines originally reserved for circuit-switched telephones and ADSL households, for example. Even if the site received dedicated communication lines, the available capacity was designed around the expected needs of the RNC cabinet. At the time of the deployment, transferring double the amount of log data out from the site in addition to the regular traffic, was not considered.

Detailed log analysis is more of an afterthought than a designed feature. Following the principles similar in MapReduce, data has to be analysed close to the source. A single server installation is used to collect data from one RNC. Log collecting servers are usually deployed with one system disk and a RAID5 array of disks for storing the filtered call sequence messages.

The RAID-array is cyclic. The storage is always full, and constantly updating as new data is written over the oldest data. Depending on the RNC activity, the disk array can hold from five minutes to two hours of call data.

Data is stored in an encrypted binary format. It is complicated to decrypt and hard to reverse engineer without binary data descriptors. A large part of the data is not directly related messages defined in a protocol, but internal messaging required to make everything else happen that is not explicitly defined or described in the protocol.

The processing of stored protocol messages is selective. An network engineer can define the conditions or flag certain messages that mark the message sequence. Only the ones matching the predefined filter definition can be stored aside. A traditional use case is an investigation request from an ISP who has seen an unusual amount of error messages of some type, and would like to conclude a deeper investigation. An engineer could log in to the log collector of the troubled RNC and could set the collection filters. This is also known as ”going fishing”.

It is not guaranteed, that all the sample logs are from the beginning of the exchange. If the fishing filter condition is met late in the exchange, the start of the exchange might already be rolled out of the log buffer. After enough data is collected and a sufficient amount of calls match the fishing filter, the logs are de-composed and analysed in a tool called Emil.

Emil takes the binary chunks of RNC signalling, decrypts them, and groups all call sequences found in the sample. The software can process multiple sample blobs, and combines sequences to a whole. The exchange sequences are spread over multiple raw

log files. The processing speed of the messages is most often throttled by IO operations, or in rare cases by CPU if the filtering process is complicated.

In the Emil software, binary messages are translated into human-read strings and values. This is done with the help of a binary profile, which is fetched from a central server based on the header information of the file. Each released product, and its firmware version has it's own description profile. Two profiles cannot be mixed in one analysis session, as one call sequence can occur in only one type of a system at a time.

At least at this time, Emil cannot handle intersystem handovers in one session. Handovers between 4G and 3G, for example, can be analysed only from the perspective of on protocol.

In general, the analysis task is consuming and manual. It includes transferring large amounts of data over a network, or by physically visiting the installation locations.

4.2 Call data classification

There are multiple algorithms for data analysis from which to choose. To be able to use such algorithms, one first has to identify the nature of the data. What type it is? Is it only binary and can it be decoded in to a natural form? Is it relational, or transactional, or both? Does the data have a state that is not explicitly written into it? Identifying the nature of data has been a great obstacle to overcome in this thesis from the very beginning of the verification work.

4.2.1 Similarity and distance

Similarity, distance, and dissimilarity commonly refer to the same thing. They are all different expressions for a variable between 0 and 1 [20]. The wording is chosen based on the context, and usually emphasises the desired end of the scale: One, denoting absolute similarity, and zero, expressing absolute dissimilarity.

Distance type	**Example**
Scaled	Height
Boolean	Day/night
Categorical	{Apples, oranges, bananas}
Ordinal	Military rank

Table 4.1: Distance types and examples

Distance is interpreted in inverse. It focuses on the lower end, where numbers close to zero are more favourable, and numbers close to the higher end, one, are not that interesting as the comparable objects have little or no similarities at all. Therefore they are distant from one another.

Table 4.1 describes common types of different distances. Distance types that already have a scale are easier to handle than ordinal data. The dataset behind the figure usually sets boundaries for what are rational limits of that number.

For example, the height of a human can not be negative, and the upper bound can be set to five meters. When comparing two objects, humans in the case, the height similarity is the height difference of these two people divided by the sum of their heights:

$$d(h_1, h_2) = \frac{|h_1 - h_2|}{|h_{min} - h_{max}|} \tag{4.1}$$

Here, two equally tall humans would have a height distance of a zero, while two persons from both ends of the spectrum would have the highest distance of one.

The ordinal system, like the military, can be sorted in a linear fashion. Interpreting ranking is a bit more complicated than that, though. If one counts the amount of ranks in a command chain between the high command, and the lowest possible foot soldier, the result varies between different factions. In Finland, the navy has ranks that have no counterpart in the ground forces. How this should be interpreted depends on what the distance metric is describing, and how it is used in the analysis. Within the military

ranking, there are linearities, but exceptions break the rule as noted.

Categorical systems are under special interests in this thesis, as they are the closest matches to the nature of the data presented here, although the analysed data has not just categorisation, but also a sequence, which is even more important, as later discussed.

4.2.2 Comparing CSA to common clustering approaches

Common clustering methods include: Partitioning, which divides the set in to partitions and in later iterations adjusts the boundaries; hierarchical ordering of a set by a given criterion; density analysis based on graph connectivity and distances; grid-based systems with multiple levels of granularity; and lastly, model based structures which are based on hypothesis.

Clustering the data by partitions requires the data set to have dimensionality. Meaning, you can build a xy-graph representing the data, or at least sort the set within one dimension from smallest to greatest. Sorting requires that two discrete samples have an order. Order implies that the two samples belong to a system.

Hierarchical ordering is close to the data set analysed in this thesis, to an extent, if the hierarchy is defined from a protocol perspective. A call establishment message cannot be sent before an initialisation command, ACK cannot be sent if there is no action before that requires it, and so forth. This type of error analysis can search deviations in a hierarchy and hence find invalid actions against protocol. Taken the complexity of 3G specifications, establishing the protocol hierarchy that can be used in distance calculation is a research topic of its own.

Grid-based systems are not a good match, because finding even that one comparable attribute, the distance between two call sequences, has been a struggle all in itself. Displaying data in a grid format would require multiple attributes that have a meaningful relation.

The closest match is the density-based approach, because the only used attribute is

the distance between two objects, but as we later discover, the density is never calculated. The similarity between density-based clustering and the algorithm in this thesis is that in both cases graph theory can be utilised.

Another approach is to classify the data based on its source. As classified in *Data Mining - concepts and techniques* [13], there are four types of data: Database data or tabular data, warehouse data, transactional data, and other types of data. This is clearly a very practical classification and does not fit in well into theoretical discussion, but it raises the idea of transactions.

The word transaction in itself is closely related to buying or selling, but the interesting part is that the actions are usually analysed in sets. Transaction can be a patch of actions that are grouped together by a single purchase. These sets are commonly analysed in data mining and produces answers to questions like: ”Customers who have bought this item, also bought...” and ”Customers who viewed this, also viewed...”. This field is thoroughly researched and there are proven algorithms ready for use.

4.3 Performance-oriented network design

In its time, when 3GPP started to push out specifications for the next generation network to follow up the widely adopted GSM, computing power was not that impressive. 3GPP Release 99 came out in 2000, same time as PlayStation 2 and Pentium 4 were released. The 3G standard family is based on work started in the early 1980s, and the development took 15 years. Taken in consideration that developing standards take quite some time, the technology, which it was built on, is already out-dated.

The first RNC cabinets were the equivalent to full-size fridges and freezers and two of those were required for full operation. The current multiplatform product handles ten times the traffic and fits into a space of only four rack units.

Release 99 introduced a new revolutionary radio interface, which utilised CDMA

modulation. This enabled more concurrent connections through one base station, but also introduced new challenges. Few of them were network management and cell division control.

Performance requirements and available technology led to two-step architecture. Abstraction layers ensure that handling is layered. Firstly, the UE can do multiple handovers without ever knowing what happens in network side. In practice this means that UE can jump between cells, or even between RNC control regions, without realising. Secondly, the core network does not make any assumptions on the radio interface. This leads to an architecture, where the responsibility of the RNC is to provide and maintain connections to circuit-switched networks while juggling physical radio attributes and the RNC control regions at the same time [7].

Deploying a 3G-network is partially political [21], which has its issues and implications for standard development. The current radio spectrum usage limits what frequencies can be used, but also dictates how the standard should also adapt to different regulators. It is beneficial for an ISP to have a monopoly status, if that can be leveraged, but it is also harmful if the device manufacturer has the same overtake on an ISP position.

3GPP took initiative through standardisation, and began to open up competition in base station market. Having only standardised messages between network elements would enable, at least in theory, competition in network gear manufacturing.

4.4 Intent to automatise error handling

For a network operator, 3G signalling analysis is a tedious task. It requires domain knowledge that is hard to come by and cannot be taught on lectures. Automating the analysis is an appealing development sector, as it would free up the demand for specialised engineers.

This is becoming even more appealing, when the age structure of the original developers is considered. Those, who have developed the standard and worked with it in nu-

merous deployment scenarios, are beginning to reach the retirement age and move away from work life.

Current metrics from the 3G-network operating status are extensively complicated. A summary of network status, which can be glanced with one look means that the view should hide most of the details. Purpose of the view is to provide status information for employee who is not well versed in network engineering. Key Performance Indicators or KPIs, were defined, to fill the void Some of them are straightforward like R̈adio Access Bearer network drop rate in a circuit-switched network, displayed in percentages(R̈AB Drop Rate CS %), but others are more abstract. The HSDPA Accessibility Success Rate (%), for example, already requires a definition for user experience. In which circumstances is a user assumed to have HSDPA available? How is it logged, if the user gets access, but soon loses it due to an expected change in environment?

There is no standard for KPI's, but through an ISP's influence on networking companies, and partially through practicality, similar KPI figures are used between different companies. Comparing KPI definitions between Ericsson, NSN and Huawei, shows huge differences in tracked variables and how they are used in KPI calculation forms. Usually Ericsson has already required attributes ready for a simple percentage display, while Nokia's figure has to be computed from multiple discrete counters. In the worst case, attribute calculation requires conditional logic to avoid division by zero in undefined edge cases. After reading through a company-specific KPI definition, one fortunately realises quite fast that two KPI figures from two different vendors cannot be compared the same as apples to apples. Variance in internal RNC implementations is so large.

One reason for this desire to automatise error handling arises from the composition of these KPIs. They do not describe the error nor show the cause for the behaviour. KPI figures can only show that symptoms have risen, as the reason can be virtually anywhere in the network.

Second problem, at least with the NSN RNC implementation, is that raised failure

flags are not accurate. It is easy to throw out an error when the link drops, but determining what caused the drop in the first place is left for the engineer to figure out.

Chapter 5

Call sequence analysing algorithm

The proposed algorithm uses serialised protocol messages and evaluates the distance between the different sequences. The process is somewhat similar to the k-means clustering discussed in chapter 3, ”Machine learning”, but instead of selecting random centroids, a new core is elected if the compared sample is dissimilar enough from previously recorded centroids.

To limit the amount of data that is processed, and to comply with hamming requirements, a fixed window [22] of 101 messages is used. To keep partial sequences comparable, the window is padded with empty elements, if needed.

The window is centred to a trigger message, which is being investigated. As there are usually multiple trigger messages in one failed call sequence, the window is re-aligned before the next analysis.

Distance is calculated using weighted hamming where each binary message is given a bit-level weight based on its position in the window. Weight constants are predefined and have a resemblance to a normal distribution as shown in figure 5.1. Messages closer to the centred trigger message have more weight in the evaluation.

Evaluation is done by comparing two messages with the same placement to each other. Similarly to Hamming distance calculation, distance is increased if the elements are inequal, and no action is taken, if the elements are equal.

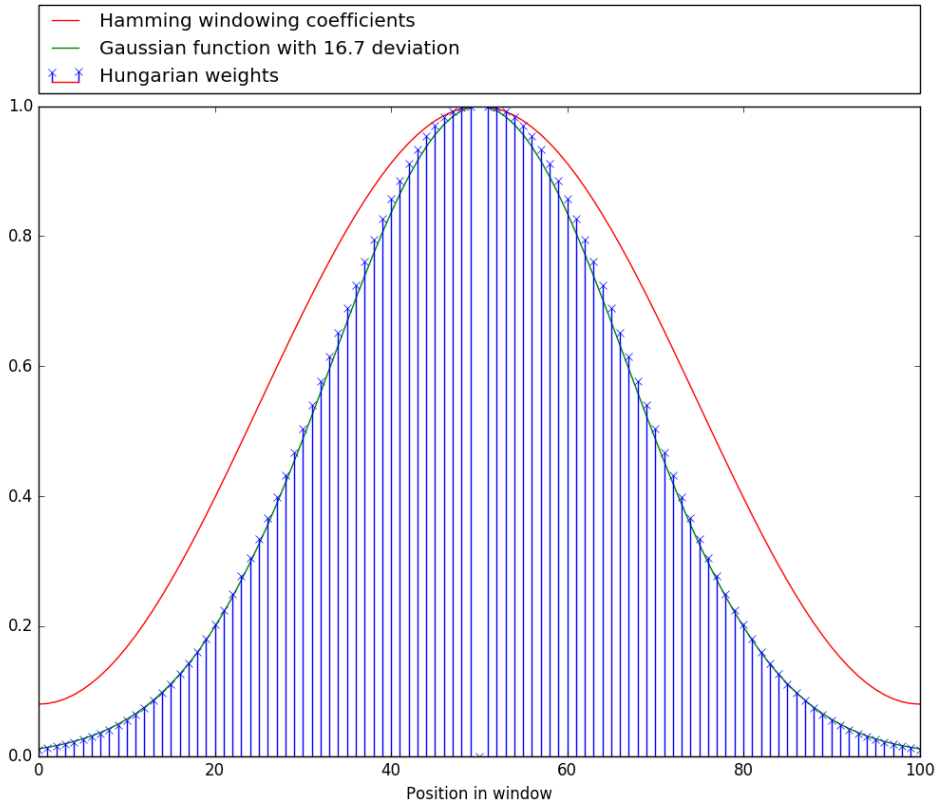

Figure 5.1: Constants used as weights in distance calculation. For computational reasons, the trigger message has zero weight. Hamming windowing coefficients are added as reference and displayed with the red legend.

If either of the messages is a placeholder, or an empty message, distance is increased with only a fraction of the weight. The default penalty for empty element comparison is half of the weight.

The sum of all weight constants is calculated and used to normalise the distance output to a range of $0 \leq weight \leq 10$. Taking the empty elements into account, the distance of 5 is uninteresting because that is the output of any comparison, if one of the series is empty. All distances close to zero are highly similar, and all distances close to 10 are highly dissimilar.

Based on the distance, call sequences are sorted into clusters, or bins, based on the

trigger message the window is centred around. A new cluster kernel is added to the bin, if the distances to other kernels exceed the set limit. A match is found if the distance is less than the desired constant.

5.1 Novelty of approach

As discussed in chapter 1.3, similar work is not publicly documented. There are examples of using weighting in Hamming windowing. Although the calculation of the weights differ, using a normal distribution can not be described as profoundly innovative, it is merely a standard tool in statistical analysis.

The question that remains is if the practise itself of, using machine learning techniques in call analysis is worth protecting. Or if it is even suited for patenting procedures. The research on current patent applications and standing patents is beyond the scope of this thesis.

5.2 Edit distances

To measure how similar, or dissimilar two strings are, an edit distance is used. Usually the figure measures how many changes need to be done to the base string, to transform it into the other string. The maximum number means that the two strings shares nothing in common and the number is equal to the highest string length in comparison.

The three common modifications that are allowed for a string, are: Insertion, deletion, and substitution. All strings can be changed into the other string using a combination of these three atomic operations. The minimal number of changes defines the string distance where each required operation increases the unit.

If the two strings are the same length, the Hamming distance sets the maximum discoverable length. It is the maximum, because the score can be lower if the transposition, a

common typing error, is allowed. Allowing transposition search in a string is application-dependent, and complicates the distance discovery algorithm.

Using the set of three operations, the metric is called Levenshtein distance. When a transposition is added to the following operations, the metric is called Damerau-Levenshtein distance.

5.2.1 Weighted Hamming distance

Hamming weight is often confusingly mixed with the distance metric, which is more specific in terms of its definition. Naming becomes even more confusing when weights are added to the distance calculation. So therefore it is good to recap the different use of terms in this context.

Hamming weight [23] is the number of bits that are different from the reference. In a binary domain, the used reference is usually zero. If the symbols are for example letters, the reference is a space character. Hamming weight does not compare two symbol sequences to each other. It merely indicates how many significant symbols are present in a given sequence.

Weight attribute has its own niche, and has few specific uses in information theory and cryptography, but it is not widely used. If people are talking about hamming in general, they are more likely referring to the use of hamming distance.

Hamming distance [24] is formed when two equal length symbol series are compared to each other. It is convenient to calculate XOR between these two series and compare the elements with equality operation. The resulting bit string is a Hamming weight series, where logical true represents a differentiating point in the two series. Counting the number of inequalities in the resulting series with a natural number, which is the Hamming distance.

In a binary domain, Hamming distance has more uses, and the XOR can be used to reconstruct the other series. Reconstruction is possible, because binary can hold only two values. In a character domain, XOR between strings shows only the positions where the strings are unequal.

Hamming distance is an edit distance. It can be used in string search functions, for example. It is always a natural number describing how many replacements has to be done to one of the compared strings in order to have the other. In terms of edit distances, hamming distance only allows substitutions.

Hamming window coefficients are special set of numbers that resemble a bell curve. Coefficients are displayed in graph 5.1 in comparison to values the used in the CSA algorithm. The generator function used with the weights in the CSA algorithm resembles a Gaussian function and yields values closer to zero when moving towards the window edge. Hamming window coefficients can be calculated with the following formula: $w(n) = \alpha - \beta cos(\frac{2\pi n}{N-1})$ using fixed values of $\alpha = 0.54$ and $\beta = 1 - \alpha$.

Weighted Hamming distance used with the CSA algorithm is a hamming distance with the symbols combined with the weights calculated from the function resembling a Gaussian function.

Summing predefined constants together forms the distance. Constants describe how important that position is in reference to other positions. As the hamming distance series have the same length, a common lookup table can be used in implementation. The numerical values of the lookup table are not relevant, as they can be normalised to a desired space. They can represent a logistical function for example, if the trailing is given an emphasis, or a normal distribution if the middle of the series is in focus.

The CSA algorithm distance is a value between zero and ten. The two series without any similarities gets a distance of ten, where exactly similar series get a distance of zero.

5.2.2 Wagner-Fischer edit distance

Wagner-Fischer was chosen to be a comparison target for the hamming distance used with the CSA algorithm, as it allows for more degrees of freedom. The algorithm itself has appeared multiple times and the origin of it is controversial. First appearance is from 1968, but the current name is based on the authors from 1974.

In comparison to hamming, which only allows substitutions, Wagner-Fischer allows substitution, insertion, and deletion. In text-based search applications, this would mean, that Wagner-Fischer does not effect the similarity score that much, if the user has forgotten to type a letter or is using only the beginning of the word as a search term.

Unlike the hamming distance, Wagner-Fischer allows comparison of unequal length series. Therefore, one does not have to pad collected log data with empty elements in distance calculation like is done with the CSA algorithm. The effect of the empty elements is elaborated and analysed later in section 6.4.1.

5.3 Algorithm implementation

The algorithm presented here has not had any specific name during its development. It has been referred to as a part of the whole tool chain named CSA, or call sequence analyser. In later stages of the development, the team decided to refer to it as the Hungarian algorithm, as its core was developed by a research group based in Hungary. Throughout this thesis it is referred to as Call Sequence Analyser algorithm.

The algorithmic idea revolves around hotspots. While 3G signalling is vast, and a number of interface messages in one call reaches over 3,000 in average, the algorithm concentrates only on the surroundings of failure counter increment.

The call alignment utility shown in listing 1 is a key part of the hotspot approach. The function assumes a full log of call sequence messages as input and limits the area of interest to the 50 previous and 50 following messages in the domain of the failure counter.

It is possible, and quite likely, that after the first failure counter there will also be others. Therefore the comparison window is centred on one counter at a time, and other counters are treated just like regular elements in the call sequence element series.

If the original sequence is incomplete from either end, the data structure is padded with empty call messages. The empty messages have a special negative ID of -1, whereas regular messages are a combination of a binary ID and attached value.

If, and when, the sequence contains more than one failure counter, all occurrences are analysed. Hence, a single phone call, that has raised an error counter, might get multiple labels.

Label-naming is a combination of the original failure name, and the identification string of the call sequence that was used to form the cluster centroid. The failure names, or generated identification strings, are not relevant in this context, as they are displayed only to the user who has access to the internal RNC binary log definitions.

The resulting set of messages from listing 1 is compared to other calls with the same failure counter. The importance of a message depends on the distance of how far the message is from the counter increment message. The proposed algorithm uses values close to the Gaussian distribution to give the highest multipliers to the messages, which are the closest to the counter and the lowest to the 50. message in the sequence counted from the centre of 101 messages.

The method takes a pointer to the CallSequence object, which is a C++ vector type, and fills it with copies of CallSequenceElements from the complete sequence. When the vector has 50 elements, the CallSequenceFCounter element is added to the vector. After the copy, the remaining 50 elements are read from the original sequence. If there are no more messages left in the original sequence, the rest of the elements are marked as empty with the special -1 identification number.

Listing 2 shows how the class variables *gaussianError* and *maxPenalty* are initialized. The number table is calculated once on object creation, and reused during the operation.

```
#include "csaengine.h"
void CSAEngine::alignCall(CallSequence* cseq,
        const CallSequence& originalseq, int counterPos) {
    int startFrom, elementsToInsert;
    // Insert dummy elements if neccessary
    if (counterPos < CSAEngine::baseHalfSize) {
        for ( int i = 0; i < CSAEngine::callSize - counterPos; i++) {
            CallSequenceElement* e = new CallSequenceElement();
            e->id = "-1";
            cseq->seq.push_back( e );
        }
    }
    if (counterPos < CSAEngine::baseHalfSize) {
        startFrom = cseq->seq.size();
    } else {
        startFrom = counterPos - CSAEngine::baseHalfSize;
    }
    // Copy the first half from the original sequence
    for ( int i = startFrom; i < counterPos; ++i) {
        cseq->seq.push_back( new CallSequenceElement( originalseq.seq.at(i)));
    }
    // Copy the found failure counter to the middle of the sequence
    cseq->seq.push_back(new CallSequenceFCounter(originalseq.seq.at(counterPos)));
    // Determine how many messages are left in original sequence
    if (originalseq.seq.size() - counterPos <= CSAEngine::baseHalfSize) {
        elementsToInsert = originalseq.seq.size() - counterPos;
    } else {
        elementsToInsert = counterPos + CSAEngine::baseHalfSize + 1;
    }
    for ( int i = counterPos + 1; i < elementsToInsert; ++i) {
        cseq->seq.push_back( new CallSequenceElement( originalseq.seq.at(i)));
    }
    // Insert dummy elements if needed
    for ( int i = cseq->seq.size(); i < CSAEngine::callSize; ++i) {
        CallSequenceElement* e = new CallSequenceElement();
        e->id = "-1";
        cseq->seq.push_back( e );
    }
    return;
}
```

Listing 1: Call alignment function

Instead of using a sampled Gaussian kernel, or a discrete Gaussian kernel, the function calculates an approximation of the Gaussian slope. This is done by incrementing the loop counter, starting from 0, with one 2.18/50 fraction at a time. The loop counter is raised to the second power and its negative is used as a power of Euler's number. A total of 50 numbers are calculated and added to the error vector.

A zero is pushed to the 51. spot in the vector. It corresponds to the CallSequenceF-Counter element in the CallSequence vector object and simplifies the distance calculation in later stages. The rest of the constants are simply the first 50 numbers, but in reverse order.

```
#include "csaengine.h"
#include <cmath>
CSAEngine::CSAEngine() : gaussianError( initGaussianError() ),
    maxPenalty( initMaxPenalty() ) {
}
std::vector<double> CSAEngine::initGaussianError() {
    std::vector<double> error;
    for ( double i = 0; i < 2.18;
        i = i + 2.18 / (double) CSAEngine::baseHalfSize ) {
        error.push_back( ( exp( - pow( i, 2))));
    }
    return error;
}
double CSAEngine::initMaxPenalty() {
    double maxPenalty = 0;
    for (size_t i = 0; i < CSAEngine::baseHalfSize; ++i ) {
        maxPenalty += CSAEngine::gapConst * gaussianError[i];
    }
    return maxPenalty * 2;
}
```

Listing 2: Constants calculation

The class variable maxPenalty is needed when the constant table is later translated to a more user-friendly form. It is the sum of all numbers in the gaussianError vector. In figure 5.1, the constants are normalised to the scale of 0 to 1, where as during the analysis work a scale of 0 to 10 was used.

The distance between the two sequences is calculated by comparing the preceding messages one by one, from the middle to the first one as described in listing 3. If the messages are the same, distance is not incremented and next message is evaluated. If the messages are different, the number presenting weight, the importance of that position, is added to the overall distance counter. If either of the sequences contains an empty element at a given point, the weight of the position is reduced to half of what it would have been in a mismatch situation.

The estimation procedure does not take in consideration the messages around the point of interest, but compares only that particular message to the other in the reference sequence. This limitation keeps the algorithm simple, effective and protocol-agnostic. Using edit distance taxonomy, this distance calculation operation allows only substitution between two series.

A single call sequence can contain multiple failure messages. For each failure, the

```
#include "csaengine.h"
double CSAEngine::weightedHamming(CallSequence cseq1, CallSequence cseq2, int counterPos) {
    double dist = 0.0;
    double treat_gap = 1.0;
    int weightIndex = 0;
    for ( int i = counterPos - 1; ( i >= counterPos - CSAEngine::baseHalfSize )
            && i >= 0; --i ) {
        if ( cseq1.seq.at(i) == cseq2.seq.at(i) ) {
            continue;
        } else if ( seqEmpty( cseq1.seq.at(i) ) || seqEmpty( cseq2.seq.at(i) )) {
            treat_gap = CSAEngine::emptyPenalty;
        } else {
            treat_gap = CSAEngine::gapConst;
        }
        dist += treat_gap * gaussianError[weightIndex];
        weightIndex++;
    }
    // Perform same operation to opposite direction in sequence
    weightIndex = 0;
    for ( int i = counterPos + 1; i < cseq1.seq.size() &&
            i < counterPos + CSAEngine::baseHalfSize + 1; ++i ) {
        if ( cseq1.seq.at(i) == cseq2.seq.at(i) ) {
            continue;
        } else if ( seqEmpty( cseq1.seq.at(i) ) || seqEmpty( cseq2.seq.at(i) )) {
            treat_gap = CSAEngine::emptyPenalty;
        } else {
            treat_gap = CSAEngine::gapConst;
        }
        dist += treat_gap * gaussianError[weightIndex];
        weightIndex++;
    }
    return dist;
}
```

Listing 3: Distance calculation

windowing and padding is adjusted accordingly, and each repositioned window is analysed, and its distance is calculated in reference to previously found kernels. If there are no previous kernels in memory, or the current sequence does not fit the set distance limits, it becomes a new kernel.

Kernels are similar to centroids in the k-means clustering, as they are representatives of a group of data points. Difference is, that once a kernel is elected, it sticks. In the reference implementation, there was no mechanism in place which would re-evaluate elected kernels against the group it gathers around itself.

After all the windows are analysed, the sequence has multiple labels. Some are from previously found kernels and in others a sequence can be the kernel itself. This leads to a situation where first call the initialising an empty cluster group already receives multiple labels.

As a side note, 3G calls do not have a unique identifier; to give a distinct name for the kernel, an identifier is composed from the call attributes and concatted to the failure counter name. What those attributes were, or what counters were used, is not relevant in this research.

5.4 Alternative algorithm implementations

Taken the original idea and operation the domain, variations to the distance calculation and data analysis were developed. The CSA algorithm groups call sequences-based failure messages, forming a new dimension for each failure. In the most simplistic situation, where there is only one call sequence under analysis, the algorithm will create a number of clusters equal to the distinct failure messages.

I decided to simplify this approach by only focusing on the surroundings of the first failure occurred. The same weighting constants were used, but instead of repeating the analysis and re-positioning the fixed sequence window, only one pass was made. The comparisons of elements were similar and the failure elements were treated as elements in the sequence.

As a second comparison variable, I decided to use Wagner-Fisher distance instead of hamming distance. Wagner-Fisher was chosen because it allows for more freedom in edit operations, and could possibly perform more robustly. The possible impact on computation time was not a concern on the verification phase, when only validity and result quality were the main optimisation targets.

5.4.1 Flat clustering

The CSA clustering algorithm focuses on failure messages, and their surroundings. Each failure is analysed separately resulting in multiple passes on a single call sequence. Each pass re-aligns the fixed window to the surrounding of that failure before distance evalua-

tion.

The flat clustering approach does not create multiple clusters, or multiple dimensions for one call sequence, hence the name. It only takes the first occurred failure counter as its fixed point of evaluation. Other possible failure messages in the same sequence are treated as regular messages. When evaluating the distance between two calls, if two messages in the same spot are actually the same failure message, distance is not increased.

Chapter 6

Verification of the CSA algorithm

During the algorithm development, the only information that was available was the output id string of the cluster to which the call was sorted. The package was designed in such away that it would be running continuously on a Linux server and the data would be processed for display in the later stages.

The original algorithm implementation was meant to just run on a server and observe change in the patterns. Changing the parameters or rerunning the analysis ended up producing different results, as the implementation is heavily dependent on the order the data arrives to input. Changing the order of the analysed sample files, or just by using a different threading execution order during the process, would change the output. The quality and the quantity of the resulting clusters changed, which led to difficulties in establishing a clear view of the output.

This behaviour is known to happen with self-learning algorithms. As the naming of the clusters is dependent on the first error message having a distance higher than the set limit, the cluster-naming is not stable even though the compositions were to be the same. The cluster identification strings were different as they were formed based on the id string of the first call sequence forming the cluster.

6.1 Used tools and different analysis procedures

As briefly discussed in section 4.1, a tool called Emil is a platform where all post-analysis work is done. The tool collects all signals belonging to one call and displays them in a readable format.

The Emil software started as a project done by engineers, who needed a tool for data analysation and visualisation. The main view of the software looks like an overcrowded spreadsheet document. It is a tool designed by engineers, to be used by engineers. Emil reads the binary-formatted log messages and presents them in a readable format. Using the descriptors stored in the central archive, the program is able to decode all the different versions of the used binary formats. As those formats changes often between different firmware releases, and between products, the binary format is strictly bound to firmware version numbering.

The log data stored in one specific format cannot be analysed with data from another binary format. This is a restriction set by the software design. Usually, two different binary formats means that the data is from different networks and different operator environments.

The Emil profile environment is a scripting interface for the program itself. There are a handful of predefined hooks that gets called when the binary log reading and translation process takes action. By using the hooks, unwanted call sequences can be filtered before further analysis, for example.

Data analysis was done with Emil program itself as a profile extension. The first version was implemented with a C++ library provided by the Hungarian research group. As Emil is written with C#, and there is a convention to call C++ objects from C#, a library interface was used. One of the major differences between these two languages is the memory management. C++ has a more C-style approach, and freeing up unused memory is eventually the users' responsibility. C#, however, uses garbage collection. One cannot allocate memory in C# side, and pass that for the library, as C++ is separated into

its own memory region. Therefore, you cannot simply create a list and pass that as an argument for a function in the C++ library even if the binary structure were the same.

A C-style compatibility layer was created, as it was suggested as a work-around for this limitation. A full call sequence was transferred from C# to C++ side by calling atomically the list operations on the compatibility layer, transferring the data in function arguments. This seemed to be the only viable solution at that time, when the Hungarian research group was still developing the code package.

This implementation led to numerous errors that were difficult to debug. Some were generated by misconception of how the co-operation between these two languages work, some by naive assumptions about the binary compatibility of the different variables of these two languages. Eventually, when the library was functioning correctly, the work laptop ran out of memory.

Emil is already quite a demanding application for resources, but great care has been taken that it functions correctly. Copying most of the analysis data to an unmanaged section, inside the C++ library, ended up consuming too much of the available memory.

At the same time, as the phase of new releases slowed down to almost a complete halt, the core logic of the library was re-implemented with C# inside the Emil profile. This was the first view of results that were connected to the original calls. It was discovered that an Excel-style listing of calls, where multiple group identification strings are attached to one call, does not give out enough information about the grouping.

The Excel-like interface was improved with automatic grouping variables. One could select a single cluster identification string, and the rows having the same id would change colour. One could also view, the addition attributes of the cluster like composition and variance distances summed in various experimental ways. An arbitrary clusters composition could be selected from interface and the same analysis metrics are displayed to the user as are used in the CSA decision process.

6.2 Analysed data

All the analysed messages are visible to the RNC. There are, for example, IUR messages sent between two RNCs and radio interface control messages. The exact composition of the analysed data cannot be analysed in this thesis, as it is considered to be a trade secret.

By this point, the reader has hopefully already understood that the data content itself is not important for the CSA algorithm. Only the indication about if two messages are an exact match is enough. Neither CSA nor the developed alternatives use any hierarchical or scaled ordering.

Therefore unique identifiers can replace all the actual messages. The dataset used to build the graphs for this thesis was converted with a one-way hash function to prevent any information leakage.

6.3 Testing plan and empirical data analysis

The devised testing plan was to deploy the CSA algorithm as a beta feature of the signalling analysis software Emil. As the analysing work is intense manual labour, which requires specific knowledge of 3G signalling environments, there are not many individuals who are capable of verifying the general behaviour of the algorithm. Fortunately, most of those who are able to do the failure analysis are already users of the Emil software. This target group includes Nokia Networks employees, and a varying number of technical support engineers working for a local ISP.

While planning the testing, doubts were raised if the two samples with the same failure message and the same fault origin would fit within the same maximum distance given for a cluster, if the samples are from two different environments. Therefore the scope of the testing was limited within a single network deployment and singular version of the hardware. This would also allow verifying and inspecting the clustering results that have been run in a larger scale in a live environment with additional features written for Emil.

The original C++ version of the library is both version and protocol-agnostic, as it receives only decomposed messages. As Emil can be run on a laptop in patch mode, it can also be ran on Windows Server in continuous mode with live input. Server hardware is required due to high I/O loads.

In a production setup, the CSA engine would accept a continuous message flow from the Emil server instance, and would output time-series data for later processing. The testing plan was not put forward as empirical testing with the devised Emil features showed that the algorithm is unreliable at the best.

6.4 Analysing clustering quality

Similarities in data sets, or in any set that contains at least one similar object, can be calculated with a Jaccard distance:

$$d(i_1, i_2) = 1 - \frac{\text{number of 1's in the set of i AND j}}{\text{number of 1's in the set of i OR j}} \tag{6.1}$$

This, how ever, is not usable with result analysis. As Jaccard index is the intersection divided by the union of two sets, it does not tell if the cluster in a set is chosen correctly. In the analysis, these two sets would be the manually sorted reference cluster, and the varying number of discovered clusters. These two sets do not have natural intersection as the clusters created by different algorithms have no connections to the reference set. Therefore, instead of using Jaccard index, it is more meaningful to use a customized scoring method while analysing the clusters.

A cluster, which has only one call reference, is considered to be an orphan. When the analysis is on-going, in a production environment or during patch-processing, orphan elements are stored, and have potential value. In patch-processing, or when generating summary for the end user the orphan clusters does not represent any additional information. In fact, they become more of a noise factor in clustering results. Therefore the orphan elements are ignored in the following figure 6.2.

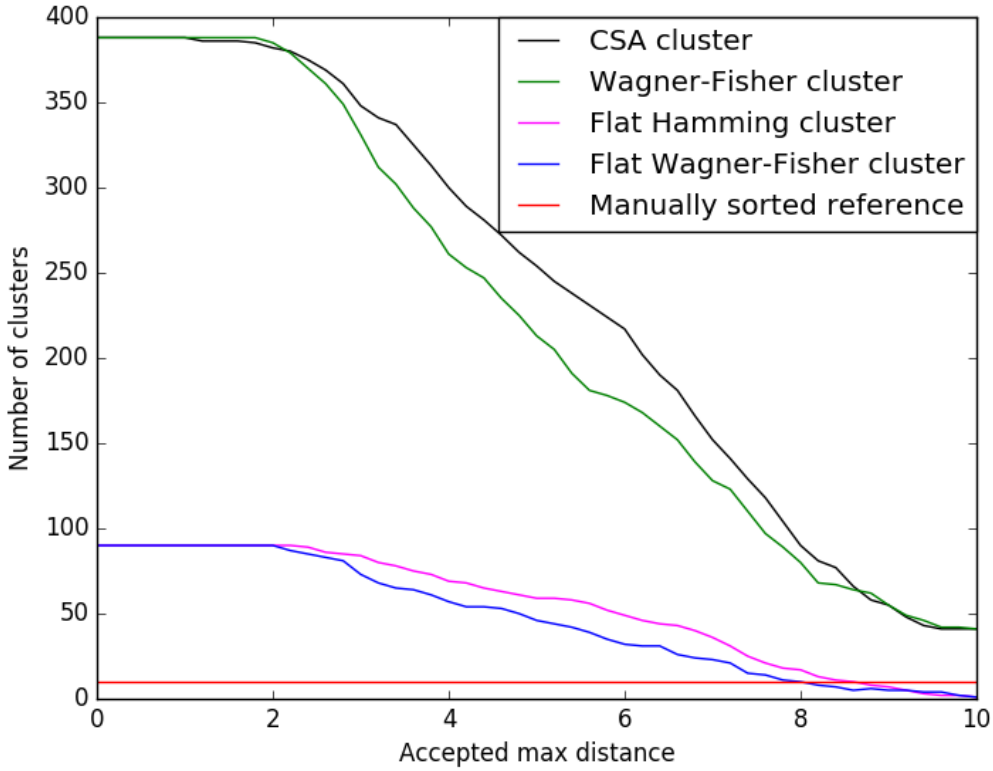

Figure 6.1: Total number of clusters within a given max distance.

Customized scoring follows the data model structure between the clusters and the calls. Both objects have references to each other, and it is possible to end up in the same call object through its clusters. This condition is always checked.

The following procedure is iterated over all the calls in given distance dimensions and repeated with all the algorithms under inspection:

1. Take one call object
2. Collect all distinct call objects that share the one or more similar clusters, ignore the orphans
3. Compare the manually sorted cluster references
4. Return the counts of hits and misses

After hits and misses are analysed for all the calls, we can calculate the percentage of hits against the total number of clustering results simply by $hits/(hits + misses)$.

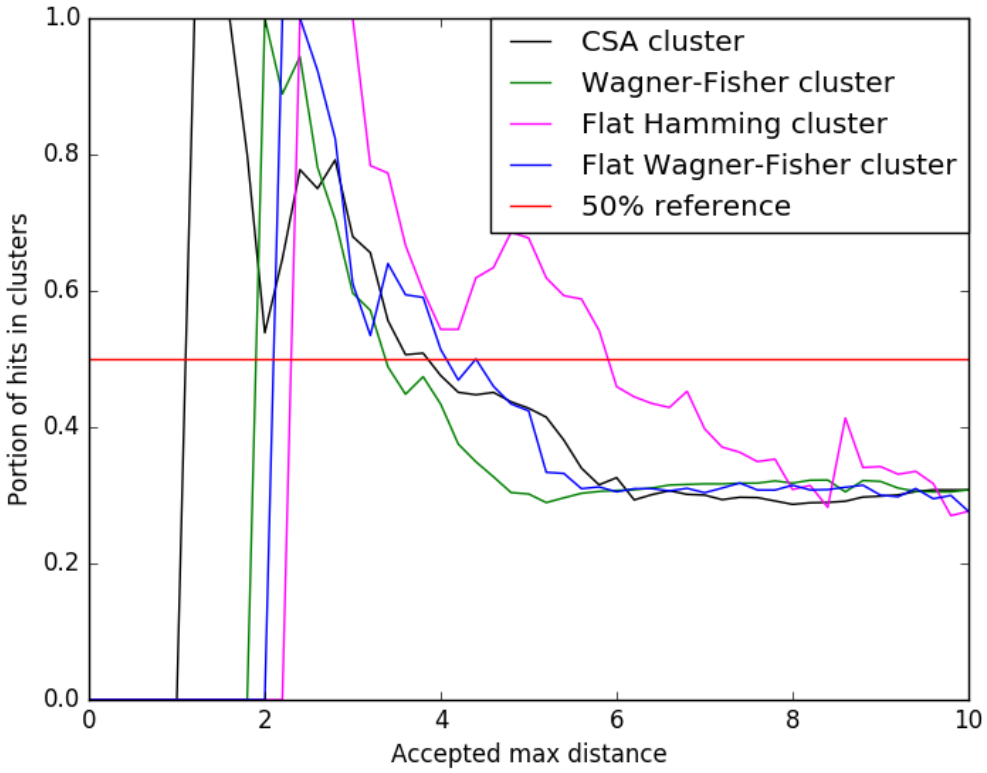

Figure 6.2: Proportion of the positive classifications when referenced to a manually sorted set. Averages under 50% line contains more false positives.

Unlike the raw amount of discovered clusters presented in figure 6.1, figure 6.2 ignores the orphan clusters.

Results below 50% indicate that there are more miss classifications in the cluster sets.

6.4.1 Effect of empty elements and algorithm stability

In figure 6.2 there is an anomaly limited to the distance region between 4 and 6 in the otherwise regressive curve. This is a result of the empty elements in the series, which were added during the window alignment process. Centring the window to the analysed

Description	Count	Percentage
Has over 38 empty elements after the first failure counter	67	75,4%
Has empty elements before first failure counter	4	4,4%
Has 1 to 38 empty elements	4	4,4%
Zero empty elements	19	21,1%

Table 6.1: Number of call sequences that fit the description. Total count exceeds the sample set as there are sequences with multiple properties.

failure message requires added padding, usually at the very end of the window.

As the comparison window is fixed in size, to 101 call elements, there are calls in the sample set where the message exchange stops before this boundary condition is met. The analysis window size is 101, but there are usually multiple failure messages in one sequence.

Therefore the window is filled with empty elements, which is summed to the total distance figure with a fixed weight of 0.5. This variable has been left as is through out this work, and its effect has not been analysed in regard to other variables.

The number of empty elements is quite high in the data set. Call sets have either zero empty elements in the sequence, or they have 38 to 46 empty elements. To be exact, 75.4% of the calls in the sample set have more than 38 empty elements at the very end.

Table 6.1 shows how many padding elements the algorithm adds to the sample dataset. The total number of different calls in the sample set is 90. There is only one sequence in the set, where there is padding before, but not after. Three of the calls required padding both before, and after the first trigger message to fit into the desired window of 101 messages. With the given set of samples, the average of 33% of all the elements used in distance calculation are empty.

Whether to use empty elements as filler is debatable. Nevertheless, in the algorithm performance analysis accepting the elements shows as an anomaly around the region of maximum distance of 5. This trend can be seen in the CSA algorithm, but can be observed

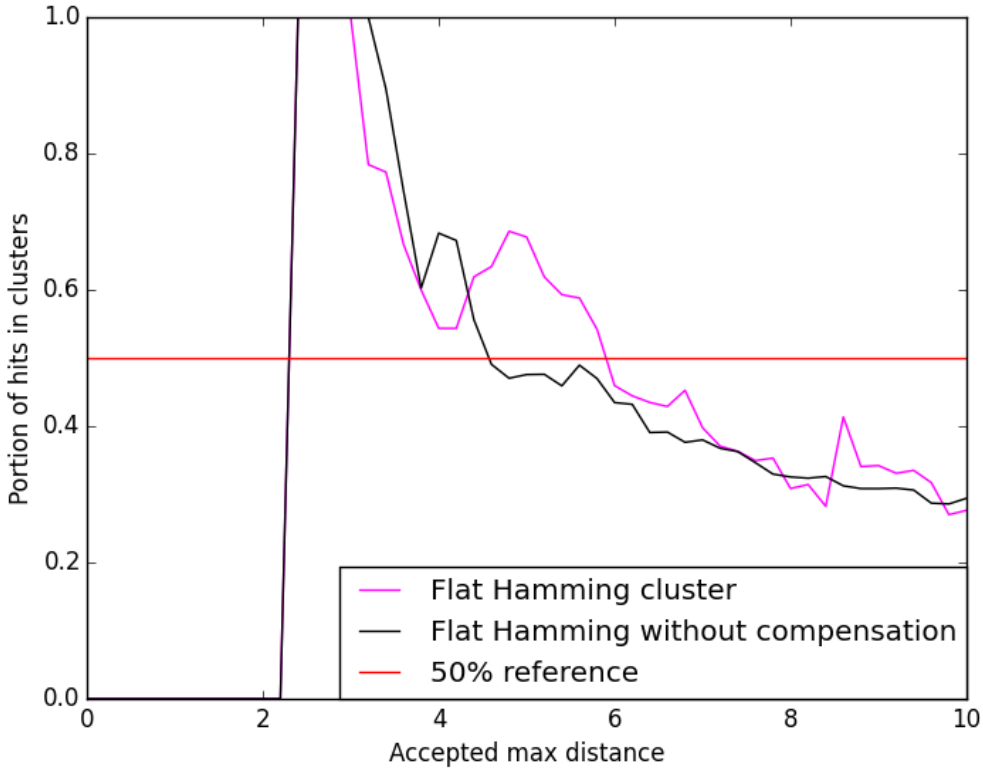

Figure 6.3: Flat Hamming analysis repeated without empty element compensation in comparison to the original results.

best from its flat counterpart.

Figure 6.3 presents the difference between accepting empty elements with half of the score to the overall distance against ignoring them totally. The CSA algorithm adds the current weight multiplied with 0.5 to the total distance between the two call sequences, if either of the elements in comparison is empty. It increases the dissimilarity.

There is a theoretical situation that comes from the algorithm implementation and from the fact that empty elements are added to the overall distance. If the analysing engine is initialised with a cluster, that contains only empty elements, each and every distance comparison will result in the score of 5. The worst possible call sequence that could initialise the engine would contain the most common error message, surrounded by most common messages and with a high number of empty elements added by the

algorithm.

6.5 Results

While using a fixed number of samples in the patch analysis, clustering is considered effective when the total number of clusters is starting to drop. Figure 6.1 shows that the clustering starts to take effect when the allowed maximum distance is increased to 2. This proves, that the distance calculation is actually working, and one ore more phone calls share at least one common label.

Logically, the slopes in figure 6.1 are more or less constant, as the maximum distance increases and more and more different calls are grouped together. This trend continues until the very end, when a maximum distance of 10 is reached. With the maximum possible distance, only the structures built into the algorithm itself are increasing the number of clusters. With CSA clustering and Wagner-Fisher clustering, this minimum number is the number of the distinct trigger messages in the analysed data. In the analysed sample set, the number is 47, but can vary with different samples. With flat versions of the previous algorithms, the minimum number is logically one.

Figure 6.1 shows that the selected comparison target, which allows more freedom in distance calculation, proves to be more effective as it produces less clusters. The difference is not much, but it is present also in the flat versions of the examined algorithm. This suggests that allowing more freedoms in edit distance calculation could lead to more accurate results.

Comparing the number of clusters between the grouping by a trigger message and the flat versions of the algorithms does not reveal any notable improvements. The number of clusters is obviously smaller, but if all sets are normalised to the same minimum-maximum range, the flat approach does not produce any advantage over having multiple labels in one call. It is only somewhat more intuitive for the researcher and for the end

user to have fewer labels.

Analysing the quality of the results proved to be a difficult task, as there was no targeted case ready where the CSA algorithm would be used. At least by the time of writing this thesis, the only intended usage was to provide metrics about the number of new clusters found during general analysis, without caring what the cluster composition actually is. This general metric would answer to questions like, ”What is the status of current errors in a network?”, and later, ”Did the general error profile change after I did this configuration change?”. Both are very vague uses of the classification.

For general overview on how the CSA algorithm is performing, a general accuracy requirement is needed. A study researching classifier accuracy improvement through autonomous or supervised learning techniques [16] defines that 75% to 80% is low and above 90% is high accuracy.

When examining figure 6.2, one can notice, that accuracy of all the classifiers is dropped under 50% when the accepted maximum distance is increased to 4. This includes the flat hamming clustering, when the distorting effect of empty elements is taken in account as presented in figure 6.3.

This gives us an acceptable maximum distance range of 2 to 4, where the clustering is actually happening and the composition of clusters is not totally random. From the first figure 6.1, one can see that between 2 and 4, the number of clusters is decreased only a little. When the algorithm has the defined high accuracy (90% to 100%), hardly any of the calls are actually grouped together. Similarly, when the desired grouping is happening the most, at the maximum distance of 4, the composition of clusters starts to be irrelevant and random.

Taking into account how the empty elements affect the distance calculated between calls, the area of 5 is the most unreliable one. If the first algorithm is run accepting the maximum distance of 5, and the first analysed call is totally empty, it will match all the following ones also.

Accepting empty elements to the distance calculation results in a situation where the series with only empty elements gets a score of 5 regardless of the content of the other series. If the other series has any empty elements of its own, the distance is even lower than 5. The empty elements are also a partial reason for the high amount of fluctuation in the perceived output of the CSA algorithm.

Chapter 7

Conclusions

In this thesis, a Call Sequence Analysing algorithm was analysed and verified. Based on the tests results presented in Figure 6.2, I can conclude that the implementation does not work. It produces cluster labels, the compositions of which do not reach acceptable accuracy levels. Increasing freedoms in the binary match process does not increase clustering quality. Therefore, the chosen distance metric does not seem to be fit for analysing this type of data.

Establishing a reference level, something to compare the results to, turned out to be difficult with the limitations set by the CSA algorithm design. As there is no previous work published about the subject, or similar studies under work within the company at the time of development that we knew about, it proved to be quite challenging to make conclusions about the overall performance. The initial study was therefore started from algorithm mechanics, from what parts it is composed and what makes it function.

The algorithm is using an edit distance metric commonly found in text searches and indexing applications. The behaviour is exactly the same as Hamming distance comparison, where the symbols are replaced with network messages. In analogy, the network messages are characters in a language that has its syntax and corpus defined by an underlying 3G-protocol. Each Ping expects a Pong in return. If Pong is not received, the rest of the network messages attached to the call sequence starts to diverge dramatically from

the expected flow seen in the standard operation of the network. The CSA algorithm tries to enlighten those misty situations where the missing Pong as a symptom is a result from various preconditions.

The processing of the network messages is limited to the set window. This window is set by the network engineers, educated guess, and by the fact that the call sequence is hardly ever complete due to logging and performance restrictions.

Considering the limitations of transferring vast amounts of log messages from the RNC installation site, the analysis can be preferably done with a single laptop. As there are RNC installation pockets where even the core network might not be on par with the actual requirements, the investigating engineer will have to physically travel to the destination and fish out partial logs to external hard drives. The most logical place to implement labelled cluster view of the data is the tool used for raw data parsing. Therefore Emil was used as a platform. It is a familiar tool to those who understand the differences in message sequences, and the profile scripting features proved to be flexible for fast mock-upping.

After the first actual visualisations, the involved engineers were not pleased with the outcome, as it was not exact. Expectations were set higher than the performance seemed to be after the first visualisations, but nobody had an idea how to put a number to that discovery. It became apparent for the engineering team that machine learning always includes inaccuracies. It might strike as an apparent feature to overlook, but nobody had actually prepared for this fact. The team, me included, had little or none previous experience or knowledge about this field of study at the beginning of the project. The work was ordered from outside and it was set to be a deliverable that can be deployed to production after it is complete.

Taking in consideration the statelessness of the algorithm and the incapability to perform reruns with the same dataset, currently production seems to be the only feasible environment for CSA algorithm testing. It might prove to be useful with different type of data than the data analysed in this thesis.

The amount of analysed data is not enough to be representative of all the different sequences. It is also impossible to process that amount of information with a single laptop or even with a few servers. Covering all grounds requires a full distributed deployment, preferably to a test environment that receives live data from production. Therefore a more theoretical approach was chosen during the later stages of the algorithm verification.

The general label for the CSA algorithm was discovered only in the late stages of writing this thesis. A general perspective of different machine learning scenarios and data structures has revealed that CSA is a semi-supervised transductive algorithm. It has the same behaviour as other transductive algorithms: Each change in the variables, adjusting the window size, changing the accepted maximum distance, or changing data handling, requires re-clustering of the whole example set. This can be classified as being general behaviour of transductive algorithms, which are also suffering from unstable output. To combat this, an analysis was done within a single thread, which reads the input always in the same order. This is a requirement that is not suitable for production, but had to be implemented in order to produce stable graphing data for this thesis.

The CRISP-DM method described in chapter 1.2, ”Outline of the thesis”, is the workflow preferred by the academic approach. The workflow executed with the CSA development is more close to skunk works, where advanced projects and concepts are developed and tested in secrecy. If the project had followed CRIPS-DM, it could have gone through multiple algorithm design options. There already would be a set metric, which defines the success or the performance rate of the algorithm, and there would be no need to develop one out from scratch. Most likely, the project would have cycled through transductive algorithms and even used the edit distance metrics in distance calculation and have come into the same conclusion.

A key result found in the thesis is that the CSA algorithm does not work. Around the interesting region of maximum distance from 3 to 4, the algorithm has barely started combining sequences to clusters. After the distance of 4, the cluster composition becomes

troublesome; as a coin flip would provide more accurate results.

The reference implementations developed to act as a comparison base, had only marginal improvements. The flat version of CSA algorithm started as a prospect for major improvement, but was later discovered to be affected the most by the handling of empty elements.

The reason why empty elements improve the results is counter-intuitive, but it is a result of two design factors. Firstly, when there is more empty elements in a call sequence, an increasing amount of elements is evaluated softly. Instead of an exact match, where the distance increment is zero, matches against empty elements get only half of the designed distance penalty. Secondly, the bell-curve, or the hamming weight coefficients, prioritises the messages in the centre of the window rather than surroundings. With these two combined, the classification decision focus moves towards a lesser amount of messages, and the original triggering failure message becomes a defining factor. Therefore, if labelling based on a single known trigger message is more accurate than using edit distance and complicated log parsing, why to bother?

The algorithm should be powerful enough to be able to label subgroups of a single trigger message with at least a 90% success rate. As shown in this thesis, this target can not be achieved.

7.1 Future work

Model-based clustering is something that has been discussed over the evaluation of CSA clustering, but has not taken a concrete form. It has proven to be hard to find a common error mechanism, that could describe all the different error situations found in networks. Models could be built with the help of evolutional programming and a huge set of ”known to be good”-sequences of messages. These could be used to form a comparison level for the failing ones. Searching for outliers in data sets could be more accurate than the methods used with the CSA algorithm.

One of the ideas hatched during the research phase was to try out forming decision trees. As human designs the protocol, there are patterns within. The most simplistic example is the challenge - acknowledgement pattern common in networks. A more complex example would be a looping power control that adjusts the transmission variables. Measurement data gets sent over the channel, and occasionally it is replied to with a new set of parameters. It would be interesting to try out an algorithm, which takes into consideration the protocol stage under execution, and predicts based on the next few messages the outcome of the sequence. What is the probability that a call ends up being successful, if this one deviation point is followed by this set of messages? Has this behaviour been seen before and should this whole sequence be stored in a log?

Either way, the current approach for message analysis needs more variables. Having only one single distance metric that is not directly comparable to the other distance metric outside having a common cluster, appears to be insufficient. After changing the approach and increasing the dimensionality, something like Manifold regularization [25] could be tested with the data.

The most promising platform for testing different styles of clustering is the scikit-learn project. It is a python framework that already has most common literature example algorithms implemented and ready to use. Many of the different metrics were explored while drawing the graphs in this thesis; for example the Jaccard distance was originally tested with the help of the framework. The scikit-learn[1] project is comprehensive and should be used as the base when starting a new project, if possible.

[1] http://scikit-learn.org/

References

[1] Colin Shearer. The CRISP-DM Model: The New Blueprint for Data Mining. *Journal of Data Warehousing*, 5(4), Fall 2000.

[2] Dong Min, Bai Chengming, Wei Ruiping, Ye Ziliang, Qiu Rongcai, and Bi Sheng. Signaling Analysis of Iub Interface in TD-SCDMA. In *WiCOM, the 8th International Conference on Wireless Communications, Networking and Mobile Computing*, pages 1–4, September 2012.

[3] Lei Zhang, Yongdong Zhang, Jinhu Tang, Ke Lu, and Qi Tian. Binary Code Ranking with Weighted Hamming Distance. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1586–1593, June 2013.

[4] Bin Fan, Qingqun Kong, Xiaotong Yuan, Zhiheng Wang, and Chunhong Pan. Learning weighted Hamming distance for binary descriptors. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 2395–2399, May 2013.

[5] Alessandro Messina, Gabriel Caragea, Pol Torres Compta, Frank H. P. Fitzek, and Stephan A. Rein. Investigating Call Drops with Field Measurements on Commercial Mobile Phones. In *IEEE 77th Vehicular Technology Conference (VTC Spring)*, pages 1–5, June 2013.

[6] Wee Lum Tan and Onching Yue. Measurement-based Performance Model of IP Traffic over 3G Networks. In *TENCON 2005 - 2005 IEEE Region 10 Conference*, pages 1–5, November 2005.

[7] Juha Korhonen. *Introduction to 3G mobile communications*. Artech House, Norwood, Massachusetts, 2003.

[8] Janne Suominen. Voice call establishment analysis in LTE user equipment. Master's thesis, University of Turku, 2014.

[9] Harri Holma and Antti Toskala. *WCDMA for UMTS: HSPA Evolution and LTE*. Wiley Online Library, Spaceship Earth, 2010.

[10] Ericsson Radio Systems. Basic Concepts of WCDMA Radio Access Network. Booklet, 2001.

[11] James MacQueen. Some methods for classification and analysis of multivariate observations. In *Berkeley Symposium on Mathematical Statistics and Probability*, pages 281 – 297, Berkeley, California, 1967. University of California Press.

[12] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of machine learning*. MIT press, Cambridge, Massachusetts, 2012.

[13] Han Jiawei and Micheline Kamber. *Data mining: concepts and techniques*, volume 5. Morgan Kaufmann, Waltham, Massachusetts, 2013.

[14] Stuart Geman, Elie Bienenstock, and René Doursat. Neural networks and the bias/-variance dilemma. *Neural computation*, 4(1):1 – 58, January 1992.

[15] Vladimir Naumovich Vapnik and Alexey Yakovlevich Chervonenkis. On the Uniform Convergence of Relative Frequencies of Events to Their Probabilities. *Theory of Probability & Its Applications*, 16(2):264 – 280, January 1971.

[16] Brent Longstaff, Sasank Reddy, and Deborah Estrin. Improving activity classification for health applications on mobile devices using active and semi-supervised learning. In *4th International Conference on Pervasive Computing Technologies for Healthcare*, pages 1–7, March 2010.

[17] Pavan Kumar Mallapragada, Rong Jin, Anil K. Jain, and Yi Liu. SemiBoost: Boosting for Semi-Supervised Learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(11):2000–2014, November 2009.

[18] Yu-Feng Li and Zhi-Hua Zhou. Towards Making Unlabeled Data Never Hurt. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(1):175–188, January 2015.

[19] Oliver Chapelle, Bernhard Schölkopf, and Alexander Zien. *Semi-Supervised learning*. The MIT press, Cambridge, Massachusetts, 2006.

[20] Leonard Kaufman and Peter Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*, volume 344. John Wiley & Sons, Hoboken, New Jersey, 2009.

[21] Joo Seong Park, Mintaig Kim, and Hyo Jun Lee. Analysis of European 3G markets and advanced strategies for 3G development. In *The 7th International Conference on Advanced Communication Technology*, volume 1, pages 428 – 431. IEEE, 2005.

[22] Lawrence Rabiner and Bernard Gold. *Theory and application of digital signal processing*. Prentice-Hall, Upper Saddle River, New Jersey, 1975.

[23] Irwing Reed. A class of multiple-error-correcting codes and the decoding scheme. *Transactions of the IRE Professional Group on Information Theory*, 4(4):38–49, September 1954.

[24] Richard Wesley Hamming. Error detecting and error correcting codes. *The Bell System Technical Journal*, 29(2):147–160, April 1950.

[25] Mikhail Belkin, Partha Niyogi, and Vikas Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *The Journal of Machine Learning Research*, 7:2399 – 2434, January 2006.