

Imperial College of Science, Technology and Medicine
Department of Computing

eTRIKS Analytical Environment: A Practical Platform for Medical Big Data Analysis

Axel Oehmichen

Submitted in part fulfilment of the requirements for the degree of
Doctor of Philosophy in Computing of Imperial College London
December 2018

Abstract

Personalised medicine and translational research have become sciences driven by Big Data. Healthcare and medical research are generating more and more complex data, encompassing clinical investigations, 'omics, imaging, pharmacokinetics, Next Generation Sequencing and beyond. In addition to traditional collection methods, economical and numerous information sensing IoT devices such as mobile devices, smart sensors, cameras or connected medical devices have created a deluge of data that research institutes and hospitals have difficulties to deal with. While the collection of data is greatly accelerating, improving patient care by developing personalised therapies and new drugs depends increasingly on an organization's ability to rapidly and intelligently leverage complex molecular and clinical data from that variety of large-scale heterogeneous data sources. As a result, the analysis of these datasets has become increasingly computationally expensive and has laid bare the limitations of current systems. From the patient perspective, the advent of electronic medical records coupled with so much personal data being collected have raised concerns about privacy. Many countries have introduced laws to protect people's privacy, however, many of these laws have proven to be less effective in practice. Therefore, along with the capacity to process the humongous amount of medical data, the addition of privacy preserving features to protect patients' privacy has become a necessity.

In this thesis, our first contribution is the development a new platform called the eTRIKS Analytical Environment (eAE) as an answer to those needs of analysing and exploring massive amounts of medical data in a privacy preserving fashion with the constraint of enabling the broadest audience, ranging from medical doctors to advanced coders, to easily and intuitively exploit this new resource. We will present the use of location data in the context of public health research, the work done in the context of data privacy for location data and the extension of the eAE to support privacy preserving analytics. Our second contribution is the implementation of new workflows for tranSMART that leverage the eAE and the support of novel life science approaches for features extraction using deep learning models in the context of sleep research. Finally, we demonstrate the universality and extensibility of the architecture to other research domains by proposing a model aiming at the identification of relevant features for characterizing political deception on Twitter.

Copyright Declaration

The copyright of this thesis rests with the author. Unless otherwise indicated, its contents are licensed under a Creative Commons Attribution-NonCommercial 4.0 International Licence (CC BY-NC).

Under this licence, you may copy and redistribute the material in any medium or format. You may also create and distribute modified versions of the work. This is on the condition that: you credit the author and do not use it, or any derivative works, for a commercial purpose.

When reusing or sharing this work, ensure you make the licence terms clear to others by naming the licence and linking to the licence text. Where a work has been adapted, you should indicate that the work has been changed and describe those changes.

Please seek permission from the copyright holder for uses of this work that are not included in this licence or permitted under UK Copyright Law.

Acknowledgements

I would like to take this opportunity to express my thanks to all of those who have always been by my side and supported me through that adventure.

Firstly, I must thank my supervisor, Professor Yi-ke Guo, without whom none of this would have been possible. I am deeply grateful for his professional guidance and sharing his wisdom.

I would like to give a special thanks to Florian Guitton who has been a close collaborator and friend from whom I have learned and shared so much.

I am thankful to Dr Heinis and Dr de Montjoye for their invaluable support and guidance.

All my friends and colleagues in Imperial College London, Diana O'Malley, Kai Sun, Miguel Molina-Solana, Shubham Jain, Arnaud Tournier, Florimond Houssiau, Akara Supratak, Ioannis Pandis, Lei Nie, Hao Dong, Paul Agapow, Susan Mulcahy, Juan Gómez Romero, Jean Grizet, Kevin Hua, Julio Amador Díaz López, Pierre Richemond, Ali Farzaneh, David Akroyd, Shicai Wang, Chao Wu, Bertan Kavuncu and Ibrahim Emam.

I would like to thank Cédric Wahl who encouraged me to follow this path.

I would like to express my gratitude to the eTRIKS and OPAL projects for supporting this work.

Finally, I would like to give my deepest thanks to Cécile and my mother for their constant support, patience, love and encouragement.

Dedication

To my mother

Vi Veri Veniversum Vivus Vici

Contents

Abstract	i
Copyright Declaration	iii
Acknowledgements	v
1 Introduction	1
1.1 Motivation and objectives	1
1.2 Contributions	2
1.3 Impact and adoption of the research	3
1.4 Thesis organisation	4
1.5 Statement of Originality	5
1.6 Publications	5
2 Background	11
2.1 Towards large scale data analysis in Life Science	11
2.1.1 A deluge of data	12
2.1.2 Moving away from a pure symptom-based medicine	13
2.1.3 Complexity of computing infrastructures in Life Science	18

2.2	Scalability in distributed systems	19
2.2.1	Introduction	19
2.2.2	Scheduling and management scalability	21
2.2.3	Storage scalability	23
2.2.4	Computational scalability	26
2.3	Architectures to support machine intelligence	28
2.3.1	Machine Learning	29
2.3.2	Deep Learning	30
2.3.3	Hardware acceleration for AI research	32
2.4	Compliance and security in distributed systems	34
2.4.1	GDPR and privacy of patient data	34
2.4.2	Security of the data	37
2.4.3	Privacy of companies	38
2.5	General-purpose analytical platforms for Life Science	39
2.5.1	Introduction	40
2.5.2	Existing architectures	41
2.5.3	Conclusion	45
3	eTRIKS Analytical Environment: Design Principles and Core Concepts	46
3.1	Introduction and users' needs	46
3.2	Existing knowledge management platforms and their limitations	49
3.3	eTRIKS Analytical Environment	52
3.3.1	Introduction	52

3.3.2	General Environment	53
3.3.3	Endpoints Layer	54
3.3.4	Storage Layer	56
3.3.5	Management Layer	57
3.3.6	Computation Layer	60
3.3.7	Interaction between Layers	62
3.3.8	Security of the architecture	63
4	Implementation of the eTRIKS Analytical Environment	65
4.1	Implementation	65
4.1.1	General Environment	65
4.1.2	Endpoints layer	66
4.1.3	Storage Layer	71
4.1.4	Management layer	73
4.1.5	Computation Layer	74
4.2	Benchmarking and Scalability	75
4.2.1	Resource usage	75
4.2.2	Scheduler	76
4.2.3	Compute Scalability	77
4.2.4	Storage Scalability	79
4.2.5	Summary	81
4.3	TensorDB: Database Infrastructure for Continuous Machine Learning	82
4.3.1	Introduction	83

4.3.2	Related work	84
4.3.3	Architecture	85
4.3.4	Application Evaluation	89
4.3.5	Conclusion	90
5	eTRIKS Analytical Environment with Privacy	91
5.1	Building Privacy capabilities	91
5.1.1	Location data as a support for public health	91
5.1.2	Attempts at sharing location data	95
5.1.3	Sensitivity of location data	96
5.2	Privacy preserving eTRIKS Analytical Environment	98
5.2.1	New services and features	98
5.2.2	Scalability of the platform	104
5.2.3	Privacy of the platform	108
5.2.4	Algorithms on the platform	111
5.2.5	Privacy module for density	113
5.2.6	Related work	119
5.3	Discussion and future work	121
6	Analytics Developed using the eTRIKS Analytical Environment	122
6.1	Analytics for tranSMART	122
6.1.1	Iterative Model Generation and Cross-validation Pipeline	123
6.1.2	General statistics	126
6.1.3	Pathway Enrichment	128

6.2	DeepSleepNet	130
6.2.1	Introduction	130
6.2.2	Tackle class imbalances	132
6.2.3	Results	132
6.3	Characterizing Political Deception On Twitter	137
6.3.1	Background	138
6.3.2	Data and Methodology	141
6.3.3	Feature Selection	147
6.3.4	Fake news classification	158
6.3.5	Conclusion	164
7	eTRIKS Analytical Environment supporting Open Science	166
7.1	Sustainability of the platform	166
7.1.1	Hosting of the project and supporting the users	166
7.1.2	Continuous integration and system deployment	170
7.1.3	Agile methods	170
7.2	Future of the platform	172
7.2.1	Adopters	172
7.2.2	Community building	173
8	Conclusion	174
8.1	Summary of Thesis Achievements	174
8.2	Future Work	176
	Bibliography	177

List of Tables

2.1	Feature comparisons between eTRIKS Analytical Environment, IBM Platform Conductor, Arvados, BOINC and Petuum.	43
5.1	Structure of a Call Detail Record.	102
5.2	Comparison of core operations for the four potential solutions considered for the database.	105
6.1	Confusion matrix from Supratak et al. [SDWG17] obtained from the cross-validation on the F4-EOG channel from the MASS dataset	136
6.2	Confusion matrix from Supratak et al. [SDWG17] obtained from the cross-validation on the Fpz-Cz channel from the Sleep-EDF dataset	136
6.3	This table of contingency reports the differences and the similarities between the labelling performed by the two teams in the used dataset.	143
6.4	Analysis of features coming from Twitter API. The results (p-value and t-stat) come from the Kolmogorov-Smirnov test [MJ51] on the distributions between the viral fake news and the other viral tweets. Rows are ordered by p-value. Variables above the line are those whose differences are considered statistically significant (p-value smaller than 0.01).	148
6.5	Features extracted for the text analysis. Again, rows are ordered by statistical significance; significant variables are above the line. It is interesting to see that those are mostly the ones associated with spelling used by bots (randomly generated to avoid collisions).	154

List of Figures

2.1	Conceptual representation of replication with eventual consistency.	24
2.2	Conceptual representation of replication with strong consistency.	25
2.3	42 Years of microprocessor trend [Kar18]	27
2.4	An example of an artificial neural network with the mathematical model [Glo] of an artificial neuron [Sta]. An input signal x_0 travels along the axons, which then interacts with dendrites of the other neuron and becomes w_0x_0 based on the synaptic strength. The synapses w_0 between the axon and the dendrite control the strength of influence of one neuron to another. The dendrites carry the weighted input signals to the cell body. The cell body accumulates the results, sums them, and then applies an element-wise function f to fire an output signal via an output axon.	31
3.1	A schematic representation of the architecture of the eTRIKS Analytical Environment.	53
4.1	A schematic representation of the eTRIKS Analytical Environment implementation.	67

- 4.3 A schematic representation of the integration of Borderline with the eAE and the integration of tranSMART as part of the eDP. Users would access the platform through  (Borderline web UI(s)) where they can select datasets via  (Borderline data-source middle-ware(s)). Data is retrieved from a target such as  (eDP External API) which in turns relies on  (eDP File Parser(s)) and  (eDP Query Executor(s)) to compile the selection. Once extracted, the data is pushed to  (Swift Object Store cluster). In addition to selecting data, users might use  (Borderline UI) to write custom analysis code and workflows. These are bundled with the data and made available to  (Borderline Cloud Connector(s)). From there, it is sent to  (eAE External API) via  (eAE File Carrier(s)). It is then dispatched by  (eAE Scheduler(s)) on the appropriate  (eAE Compute Node(s)). Once the computation has finished, results come back to  (Borderline Cloud Connector(s)) and are pushed to  (Swift Object Store cluster) to be accessible to the users. Operational items such as service health, sessions and routes are stored in  (MongoDB cluster). 69
- 4.4 Illustration of a simple submission of three jobs using the python eae package. . . 72
- 4.5 Evolution of the usage percentage aggregated per two days across all the machines during three months. We observe that, on average, the resource usage of the compute resources is significantly improved (21% on average). 76
- 4.6 The performance of a single scheduler with respect to the submission size. Each point represents the average running time of 10 experiments along with the standard deviation. 77
- 4.7 The performance of the *Management Layer* as the number of schedulers decreases. Each point is the average running time of 3 experiments along with the standard deviation. 78
- 4.8 The scalability of the eTRIKS Analytical Environment with respect to the data size. Each point represents the average running time of 30 experiments, with the error bar representing the standard deviation. 79

4.9	The compute scalability of the eTRIKS Analytical Environment with respect to the cluster size. Each point represents the average running time of 30 experiments, with the error bar representing the standard deviation.	80
4.10	The storage upload scalability of the eTRIKS Analytical Environment with respect to the data size. Each point represents the average running time of 5 experiments, with the error bar representing the standard deviation.	81
4.11	The storage download scalability of the eTRIKS Analytical Environment with respect to the data size. Each point represents the average running time of 5 experiments, with the error bar representing the standard deviation.	81
4.12	TensorDB workflow: the database query mechanism connects all the components. The work is distributed across multiple machine.	85
5.1	Sources and sinks of people and parasites from Wesolowski et al.'s [WET ⁺ 12] study. Kernel density maps showing ranked sources (red) and sinks (blue) of human travel and total parasite movement in Kenya, where each settlement was designated as a relative source or sink based on yearly estimates. (A) Travel sources and sinks. (B) Parasite sources and sinks.	93
5.2	A schematic representation of the architecture of the OPAL platform.	99
5.3	A schematic representation of the flow of the data from the raw data to the platform's output. 1. Pseudonymizing and ingesting data 2. Data Fetching for compute and creation of user specific CSVs 3. Executing Map function 4. Outputs aggregation and applying privacy mechanisms	103
5.4	The insertion performance of Timescale with amount of data inserted. We observe that the speed remains essentially stable as the amount of data being ingested increases.	106
5.5	The data fetching performance of Timescale with time interval of the query. We observe fetching overheads to be significant for smaller queries which decreases as the query interval size increases.	106

5.6	Measuring time taken for Compute and number of users for various interval range sizes with sampling parameters. Sampling Parameters used - Blue: 1%, Red: 10%, Brown: 100%.	108
6.1	Iterative model generation and cross-validation pipeline	124
6.2	Modelling of the pipeline for an unbiased approach to statistical testing of whole datasets.	127
6.3	Illustration of a KEGG disease pathway with the differentially expressed genes associated with smoking.	129
6.4	An overview architecture of DeepSleepNet from Supratak et al. [SDWG17] consisting of two main parts: representation learning and sequence residual learning. Each trainable layer is a layer containing parameters to be optimised during a training process. The specifications of the first convolutional layers of the two CNN depends on the sampling rate (Fs) of the EEG data.	131
6.5	Examples from Supratak et al. [SDWG17] of the hypnogram manually scored by a sleep expert (top) and the hypnogram automatically scored by DeepSleepNet (bottom) for Subject-1 from the MASS dataset.	136
6.6	A schematic representation of the architecture of the proposed FakeNews Platform.	146
6.7	Density distribution of the decimal logarithm of the continuous variables from Table 6.4 that are statistically significant. From the image, we can see that viral tweets not containing fake news (in blue) tend to have <i>peakier</i> distributions. . .	149
6.8	Distribution of the four significant discrete variables (user.verified, num_hashtags, num_mentions and num_media) from Table 6.4. The test for the proportion of verified account confirms an expected fact: the proportion of verified account is much weaker for viral tweets containing fake news than for other viral tweets, suggesting that fake news tend to be created by more ‘anonymous’ people. Besides, tweets with fake news have generally more hashtags and media but fewer mentions.	150
6.9	Most recurrent words in the tweets (single and bigram)	150

6.10	Frequency of appearance of most used hashtags in tweets containing fake news (red) and not containing them.	151
6.11	Correlation of the features related to the spreading of tweets. <i>rt</i> stands for retweet (e.g. <i>rt_timeto10</i> , time to get to 10 retweets), and <i>fav</i> for favourite (e.g. <i>fav_timeto10</i> , time to get to 10 favourites).	152
6.12	Distribution of the decimal logarithm of the time (in hour) to get to 1000 favourites (<i>fav_timeto1000</i>) for both tweets containing fake news and tweets not containing them. The associated p-value is 9.60e-11 which proves the significance of the propagation pattern.	153
6.13	Comparison between the different core sentiments between tweets containing fake news and tweets not containing them.	155
6.14	Evolution of the different core sentiments over the course of the four months, between tweets containing fake news and tweets not containing them.	156
6.15	Difference of the evolution of the sentiment computed by <i>word2vec</i> between tweets containing fake news and other tweets. Each point represents a tweet in the timeline of our dataset and the probability of the tweet for being positive. The blue line represents the average probability per day.	157
6.16	Most used emojis in the dataset	158
6.17	Distribution of the decimal logarithm of the number of emojis and the sentiment score of tweets with emojis for the fake news and the other tweets	159
6.18	AUC computed on all subsets of features for the different machine learning algorithms evaluated.	161
6.19	Best performances for each subset of features, and for each metric of performance.	161
6.20	Evaluation of the impact of hyperparameters on XGboost performance	163
6.21	Most important variables for the best model for the AUC and the recall	163

- 7.1 Illustration of the eAE's *Scheduling and Management service* hosted on GitHub.
The repository contains a README describing the main features of the service, the docker file to build the docker container, the YAML build file (.travis.yml) for Travis, the tests to automatically validate the build, and the code of the service. The issues tab contains all the issues reported by users and developers or future features related to the service. The pull requests tab contains all currently open pull requests to be merged into the development branch by the admins. The wiki tab contains all the necessary documentation of the service (design of the service, description of the API, comments, etc). The shields (build and dependencies) are dynamic which allows anyone to check the current status of the project. . . . 168
- 7.2 Illustration of the create job query in Postman with the type of request (POST), the URL, a description of the request, the parameters in the body of the request and a Python version example of the request. We can see as well an example response for the query with the associated code (200 in this instance). 169

Chapter 1

Introduction

1.1 Motivation and objectives

The analysis of very large and growing, multiscale, multimodal datasets (i.e., Big Data) has today become a major challenge to address in order to convert data into knowledge and achieve innovation. Particularly so in biomedical research, where scientists are more and more confronted with Big Data challenges due to the rapid advances of high-throughput biotechnologies. The complexity, diversity, rich context and size of recent biomedical data, such as Next Generation Sequencing (NGS) data, 'omics, and imaging data, have shown the limitations of current systems. The collection, management, storage, and analysis of biomedical Big Data consequently mandates the development of new methodologies and technologies.

The problems of developing systems for analysing multi-modal medical data are, on the one hand, the massive amounts of data needed for analysis and the associated need of a scalable infrastructure and, on the other hand, the quickly changing needs of those analytics, i.e., the need for new and different algorithms and tools for data processing, integration and analytics.

We developed the eTRIKS Analytical Environment (eAE) in answer to these needs of analysing and exploring massive amounts of medical data. The eTRIKS Analytical Environment is a modular framework which enables the analysis of medical data at scale. Its modular architecture

allows for the quick addition or replacement of analytics tools and modules with little overhead, thereby ensuring support of users as the data analytics needs and tools evolve. We built the eTRIKS Analytical Environment on well-accepted technologies to ensure user adoption. As we will show in the evaluation section, the system scales very well in the number of users as well as in the amount of data analyzed. Several examples where the eAE has led to successful research will be presented as well to illustrate the scope of capabilities in the context of translational research.

To demonstrate the universality of the platform to other domains, we will present two successful projects that have been carried out using the eAE. The first project aimed at creating a privacy preserving analytical platform for terabytes of location data in the context of public health research and monitoring. This project has enabled us to demonstrate the extensibility and modularity of the eAE with the implementation of new modules to extend the eAE's vanilla capabilities and meet the needs of the OPAL project. The second project aimed at identifying relevant features for assessing political deception on Twitter using statistical, machine learning and deep learning methods. This second project demonstrated the flexibility of the eAE in the context of data science research by providing a broad scope of analytical capabilities.

1.2 Contributions

This thesis presents a new architecture for high-performance distributed computation and concurrent multi-tenancy. This new architecture leverages new and old technologies which have matured in the last few years such as in-memory computation and containerization technology. We will provide several examples where the architecture has been successfully applied, the new possibilities it has enabled and its extensibility to other domains. The main contribution is designing and successfully implementing an open source version of the architecture. More specifically, this thesis makes the following contributions:

1. Define a new architecture for high-performance analysis called the eTRIKS Analytical Environment (eAE).

2. The implementation of the architecture and its evaluation.
3. The extension of the eAE to support privacy compliant analytics and the development of a privacy preserving population density algorithm.
4. A framework, called TensorDB, that fuses database infrastructure and application software to streamline the development, training, evaluation and analyzing machine learning models. This work was later incorporated within TensorLayer [DSM⁺17] to support the training of Deep Learning and Reinforcement Learning models in a seamless fashion to the users.
5. The implementation of new workflows for tranSMART that leverage the eAE and the support of novel life science approaches for features extraction using deep learning models.
6. The identification of relevant features for characterizing political deception on Twitter.

1.3 Impact and adoption of the research

The first adopters and strongest supporters have been the Data Science Institute and the eTRIKS project at Imperial College London. Thanks to that adoption, we have been able to support various internal analytical efforts (DeepSleep [SDWG17], FakeNews, Borderline [OGA⁺18]) in the context of the Innovative Medicines Initiative (IMI) and the European Union's Horizon 2020 projects such as OncoTrack [GYVdS⁺19]. The ITMAT project, which is part of the National Institute for Health Research Imperial BRC, has also adopted the eAE as their analytical environment to propel their analyses.

Another major adopter of the eAE was International Business Machines Corporation (IBM). They added it to their portfolio of supported projects, and it is advertised as part of their large scale computing platforms and POWER architecture to their clients.

The most recent adopter has been the OPen Algorithm (OPAL) project as detailed in Chapter 5. They use the eAE platform in production in the context of on-premises pilot projects with

Orange-Sonatel in Senegal and Telefónica in Colombia. There are, at the time of writing, around a dozen people (from the Senegalese government, United Nation, Agence Nationale de Statistique et de la Démographie, researchers from Orange and Telefónica among others) actively using them in each country and an additional five other actors which are evaluating the adoption of the platform in their respective countries. The project and the platform were featured at the United Nation's World Data Forum¹, Mckinsey Global Institute² and The Innovator³.

1.4 Thesis organisation

Chapter 2 introduces the background of this thesis, presenting some of the new challenges that Big Data in the context of translational medicine research introduces and thus the necessity of creating a new analytical platform to address those challenges.

Chapter 3 presents the eTRIKS Analytical Environment architecture design principles that we developed and a comparison with existing systems.

Chapter 4 describes the implementation of the proposed architecture that has been done and open-sourced.

Chapter 5 introduces the public health benefits that can be reaped from location data and the privacy issues that the use of location data introduces. We then introduce the OPAL project and the extension of the eTRIKS Analytical Environment as the solution for their secure and privacy compliant platform for location data analytics.

Chapter 6 introduces the analytics that have been developed for tranSMART using the eAE as back-end and other life science projects and analyses that have been carried out using the eAE.

¹<https://www.opalproject.org/newsfeed/2018/11/15/opal-session-and-demonstration-at-the-united-nations-world-data-forum>

²See notes in: <https://www.mckinsey.com/featured-insights/artificial-intelligence/applying-artificial-intelligence-for-social-good>

³<https://static1.squarespace.com/static/599ef170197aeac586fed53f/t/5c7c67dbe5e5f0bda52ac290/1551656928637/The+Innovator+February+2019.pdf>

Chapter 7 presents how the eTRIKS Analytical Environment takes part in the effort towards supporting Open Science.

1.5 Statement of Originality

I declare that the content of the thesis is composed by myself, and the work it presents is my own. All use of the previously published work of others has been listed in the bibliography.

1.6 Publications

In relation to this thesis:

1. **Characterizing Political Deception On Twitter: A case study on the 2016 US elections** In: *Submitted to IEEE Access.*(Oehmichen et al.)

Political fake news have become a major challenge of our time, and its successful flagging a main source of concern for publishers, governments and social media. The approach we present in this work focuses on Twitter and aims at finding characteristic features (including temporal diffusion and NLP) that can help in the process of automating the identification of tweets containing fake news. In particular, we looked into a dataset of four months-worth of tweets related with the 2016 US presidential election. Our results suggest that there are indeed some features (such as favourite and retweet counts, the distributions of followers, or the number of URLs on tweets) that can lead to successful identification of tweets containing fake news.

2. **OPAL: High-performance platform for large scale privacy-preserving location data analytics** In: *Submitted to the 28th ACM International Conference on Information and Knowledge Management (CIKM 2019).*(Oehmichen et al.)

Mobile phones and other ubiquitous technologies are generating vast amounts of high-resolution location data. This data has been shown to have a great potential for the

public good, e.g. to monitor human migration during crises or to predict the spread of epidemic diseases. Location data is, however, considered one of the most sensitive types of data, and a large body of research has shown the limits of traditional data anonymization methods for Big Data. Privacy concerns have so far strongly limited the use of location data collected by telcos, especially in developing countries. In this paper, we introduced OPAL (for OPen ALgorithms), an open-source, scalable, and privacy-preserving platform for location data. At its core, OPAL relies on an open algorithm to extract key aggregated statistics from location data for a wide range of potential use cases. We first discuss how we designed the OPAL platform, building a modular and resilient framework for efficient location analytics. We then describe the layered privacy mechanisms we put in place to protect privacy, giving formal verification for our population density algorithm. We finally evaluate the scalability and extensibility of the platform and discuss related work.

3. **A multi-tenant computational platform for translational medicine** In: *38th IEEE International Conference on Distributed Computing Systems (ICDCS 2018)*. (Oehmichen et al.)

In this paper, we presented the eTRIKS platform. It introduces three new components namely the eTRIKS Analytical Environment (eAE), the Borderline project and the eTRIKS Data Platform (eDP). Each component was built to be part of a microservice architecture. In our implementation we assume the underlining operational database to be MongoDB and the functional object store to be Swift. Borderline and the eAE were both written in JavaScript assuming Node.js as runtime environment, while the eHS was written partly in C#, partly in JavaScript. The choice of a rather uniform Node.js-based stack enables a simplified maintenance of all lightweight microservices. Likewise, all communication between the components relies on a set of similarly designed HTTP API. Finally, each individual microservice is developed to work seamlessly in Docker containers, allowing flexible and efficient orchestration of deployment and easy scale up/down of services independently.

4. **eTRIKS analytical environment: A modular high performance framework for medical data analysis** In: *Big Data 2017*. (Oehmichen et al.)

In this paper, we presented the eTRIKS Analytical Environment. It introduced a component based, distributed framework for distributed data exploration and high-performance computing. We designed the eTRIKS Analytical Environment to provide users with an analytics environment which (a) has a frontend/endpoints which are user-friendly, extensible as well as easily integrated into tools, (b) is modular and finally (c) is also scalable. At the top, interacting with users, is the endpoints layer which essentially hosts the containers which either provide the UI to users or the interface to integrate it into third party, external tools. The endpoints layer also contains the infrastructure to run smaller computations locally. Interacting with the endpoints layer is the storage layer which caches analytics results to avoid recomputation of frequent analysis, thereby making analysis more efficient. If a computation still needs to be computed, the scheduling layer will take care of it and schedule in on the computation layer once computational resources become available. The computation layer provides the capability for the distributed computation of analyses and thus enables the scalability of the environment.

5. **Characterizing Political Fake News in Twitter by its Meta-Data** In: *ArXiv.(Amador et al.)*

In this paper, we presented a preliminary approach towards characterizing political deception on Twitter through the analysis of their meta-data. In particular, we focused on more than 1.5M tweets collected on the day of the election of Donald Trump as 45th president of the United States of America. We used the meta-data embedded within those tweets in order to look for differences between tweets containing fake news and tweets not containing them.

6. **TensorDB: Database Infrastructure for Continuous Machine Learning** In: *19th International Conference on Artificial Intelligence 2017 (ICAI'17).(Liu et al.)*

In this paper, we introduced the TensorDB system, a framework that fuses database infrastructure and application software to streamline the development, training, evaluation and analyzing of machine learning models. The design principle is to track the whole model building process with database and connected different components by database query mechanism. This design produces a highly flexible framework enable each compo-

ment to be updated. The theoretical value is that it enables continuous machine learning. TensorDB is motivated by production application of machine learning model, as consolidation of many engineering practice and serve as the foundation for high level tools for machine learning application.

Other works:

1. **A computational framework for complex disease stratification from multiple large-scale datasets** In: *BMC Systems Biology 2018*.(De Meulder et al.)

In this paper, we presented a multilevel data integration for multi-’omics datasets on complex diseases. Indeed, multi-’omics datasets are becoming more readily available and there is a need to set standards and good practices for integrated analysis of biological, clinical and environmental data. In this paper, we present a framework to plan and generate single and multi-’omics signatures of disease states. The framework is divided into four major steps: dataset subsetting, feature filtering, ’omics based clustering and biomarker identification. We illustrate the usefulness of this framework by identifying potential patient clusters based on integrated multi-’omics signatures in a publicly available ovarian cystadenocarcinoma dataset. The analysis generated a higher number of stable and clinically relevant clusters than previously reported, and enabled the generation of predictive models of patient outcomes. This framework will help health researchers plan and perform multi-’omics Big Data analyses to generate hypotheses and make sense of their rich, diverse and ever-growing datasets, to enable implementation of translational P4 medicine.

2. **Visualizing Large Knowledge Graphs: A Performance Analysis** In: *Future Generation Computer Systems (FGCS) 2018*.(Gomez-Romero et al.)

In this paper, we presented the increasing importance of Knowledge Graphs as a source of data and context information in Data Science. A first step in data analysis is data exploration, in which visualization plays a key role. Currently, Semantic Web technologies are prevalent for modelling and querying knowledge graphs; however, most visualization

approaches in this area tend to be overly simplified and targeted to small-sized representations. In this work, we analyzed the performance of Big Data technologies applied to large scale knowledge graph visualization. To do so, we have implemented a graph processing pipeline in the Apache Spark framework and carried out several experiments with real-world and synthetic graphs. From our benchmarks, we conclude that distributed implementations of the graph building, metric calculation and layout stages can efficiently manage very large graphs, even without applying partitioning or incremental processing strategies.

3. **TensorLayer: A Versatile Library for Efficient Deep Learning Development**

In: *ACM on Multimedia Conference 2017 (MM'17)*.(Dong et al.)

In this paper, we introduced a new versatile Python library that aims at helping researchers and engineers efficiently develop deep learning systems. Deep learning has enabled major advances in the fields of computer vision, natural language processing, and multimedia among many others. Developing a deep learning system is arduous and complex, as it involves constructing neural network architectures, managing training/trained models, tuning optimization process, preprocessing and organizing data, etc. It offers rich abstractions for neural networks, model and data management, and parallel workflow mechanism. While boosting efficiency, TensorLayer maintains both performance and scalability. TensorLayer was released in September 2016 on GitHub, and has helped people from academia and industry develop real-world applications of deep learning.

4. **Optimising parallel R correlation matrix calculations on gene expression data using MapReduce** In: *BMC Bioinformatics, vol. 15 (2014)*.(Wang et al.)

This paper evaluated the current parallel modes for correlation calculation methods and introduces an efficient data distribution and parallel calculation algorithm based on MapReduce to optimise the correlation calculation. The performance has been studied using two gene expression benchmarks. In the micro-benchmark, the new implementation using MapReduce, based on the R package RHIPE, demonstrates a 3.26-5.83 fold increase compared to the default Snowfall and 1.56-1.64 fold increase compared to the basic RHIPE in the Euclidean, Pearson and Spearman correlations. Though vanilla R and

the optimised Snowfall outperforms the optimised RHIPE in the micro-benchmark, they do not scale well with the macro-benchmark. In the macro-benchmark the optimised RHIPE performs 2.03-16.56 times faster than vanilla R. Benefiting from the 3.30-5.13 times faster data preparation, the optimised RHIPE performs 1.22-1.71 times faster than the optimised Snowfall. Both the optimised RHIPE and the optimised Snowfall successfully perform the Kendall correlation with TCGA dataset within 7 hours. Both of them conduct more than 30 times faster than the estimated vanilla R

5. **DSIMBench: A benchmark for microarray data using R** In: *40th International Conference on Very Large Databases (VLDB 14)*. (Wang et al.)

Parallel computing in R has been widely used to analyse microarray data. We have seen various applications using various data distribution and calculation approaches. Newer data storage systems, such as MySQL Cluster and HBase, have been proposed for R data storage; while the parallel computation frameworks, including MPI and MapReduce, have been applied to R computation. Thus, it is difficult to understand the whole analysis workflows for which the tool kits are suited for a specific environment. This paper proposes DSIMBench, a benchmark containing two classic microarray analysis functions with eight different parallel R workflows, and evaluate the benchmark in the IC Cloud test-bed platform.

Chapter 2

Background

In this chapter, we will start with a presentation of the transition of life science research towards large scale data analysis. Then, we will introduce the necessary technical background on machine learning including supervised learning, unsupervised learning, and deep learning and their associated hardware accelerators, which will be leveraged in this thesis. We will follow with a brief discussion on privacy and security related issues associated with personal data. Finally, we will review the existing general-purpose analytical platforms for Life Science and highlight the necessity to develop a new platform for the efficient management and analysis of large scale medical data in a privacy preserving fashion.

2.1 Towards large scale data analysis in Life Science

Many modern scientific applications directly or indirectly depend on distributed systems running in large-scale compute clusters situated in data centres. Historically, those distributed systems have relied on monolithic supercomputers to perform the required tasks. However, the declining cost of hardware has enabled companies and universities to purchase tremendous amounts of commodity and specialised hardware, servers and saw the advent of *Cloud Computing*.

2.1.1 A deluge of data

The technical capabilities for data collection, as well as the variety of the data collected, have increased exponentially in the last few years, introducing new, unprecedented challenges to information management. The number of available data sources exploded thanks to the development of the Web 3.0 and social networks, the ubiquity of mobile devices (mobile phones, wearables, fitness trackers, etc.) and sensor networks (CCTV, air quality, machine health, etc.). Advances in gathering scientific data also contributed considerably to this development.

Our ability to collect medical data in particular is increasing even faster, which leaves us with a humongous amount of data that can only be analyzed with difficulty. That explosive growth is not expected to slow down anytime soon for several reasons. Firstly, sequencing hardware has become so affordable that entire farms of devices are sequencing DNA faster and more accurately than ever. Indeed, to increase the coverage of DNA sequencing, only more hardware needs to be used. An example from genomics illustrates the growth very well: next-generation sequencing has led to a rise in the number of human genomes sequenced every year by a factor of 10 [Mar13, Cos14], which far outpaces the data analysis capabilities. Secondly, sequencing is only one example among a very large pool. Data today is recorded with more and more devices (e.g. MRI, X-ray or other medical devices and even with activity trackers) and there are substantial efforts to move medical records to an electronic form to facilitate exchange and analysis. Thirdly, the instruments used are becoming increasingly precise, which translates into increasingly high resolutions and in a massive growth of the data output [Met10]. The size of the data, however, is not the only challenge of working with medical data. The data produced is becoming more heterogeneous (originating from different devices, software, manufacturers, etc.), complex (protein folding simulation, metabolomics, etc.) as well as dirtier (incomplete, incorrect, inaccurate, irrelevant records) at the same time and is thus, in dire need of reconciliation, integration, and cleansing. Efficient integration of the data is, nevertheless, pivotal as medical data is moving more and more at the centre of medical research to develop novel diagnostic methods and treatments [Glu05, ST12]. As solely symptom-based diagnostic methods are slowly showing their limitations, the shift to a more data-driven medicine that

makes the efficient ability to extract, transform and analyze medical data become key in the new era of data-centric medicine [OMBe12].

The resulting flood of data is difficult, if not impossible, to manage using traditional tools. On the one hand, the sheer size of the data requires massive storage arrays and massively parallel processing nodes using hundreds or thousands of computers. On the other hand, innovative methods for data analysis are necessary to extract useful information from the raw data. Big Data has four important features, so-called four Vs [IBMc, Lan01]: Volume of data, Velocity of processing the data, Variety of data sources, and Veracity of the data quality. These four earmarks need to be addressed with specific theories and technologies. Some recent large-scale medical research initiatives like the Human Brain Project [MML⁺11], the BRAIN initiative [ILC13], the Precision Medicine initiative [Ash15] or the recent BD2K [Nat] Big Data in genomics initiative, have helped considerably by proposing harmonised data formats, thus addressing some of the Big Data challenges in medical research and healthcare. Notwithstanding, many challenges remain and among them producing scalable analytics and scalable platforms to propel them.

2.1.2 Moving away from a pure symptom-based medicine

In order to understand or discover biological processes, and thanks to advances in other fields such as physics and chemistry, medical research has increased the size and variety of data collected. That abundance of data has driven medical research to leverage an extreme wealth of analytics to explore and analyze the data.

Medical analytics background

Traditional approaches to diagnosing and treating diseases based on symptoms alone are no longer sufficient in the face of increasingly complex disease patterns. Entirely dissimilar diseases with substantially different underlying causes and interactions can exhibit similar symptoms. Those overlapping apparent similarities render their diagnosis and the proper treatment based

solely on symptoms very challenging. To overcome the limitations of identification and thus, treatment of disease based on symptoms alone, medical scientists are in the process of developing new methods to understand the precise interactions (pathways and regulatory networks), at different biochemical and biophysical levels, that lead from local malfunction to disease, or how psychological, nutritional, therapeutic, or other environmental factors modulate these interactions. Given the complexity of diseases, algorithms to process and analyze medical data efficiently and in a scalable fashion are a crucial component of these new methods: data of different sources and modalities (medical records, genetics, imaging, blood tests, etc.) need to be integrated and used to discover meaningful correlations within and between etiological, diagnostic, pathogenic, treatment, and prognostic parameters. By finding patients with the same disease into constellations of parameterised biological, anatomical, physiological and clinical variables that define homogeneous populations, signatures of disease (shared attributes of patients suffering from the same disease) can be identified. Disease signatures or biomarkers are crucial to discriminate diseases for the purpose of developing more robust and reliable diagnostic methods and better understand the disease itself (with the potential to develop new treatments).

On this path towards data-centric medical research and to move beyond the classic symptom-based diagnostics and treatments, medical research is facing major challenges. First, data heterogeneity and data quality are not directly related to the underlying technology. The protocols and acquisition processes used to capture that data directly impacts the quality and dirtiness of the data captured. Detailing those processes and training medical professionals on the consequences of those choices is therefore key to ensure that the interpretation of the results is reasonable. Second, whilst the fundamental analytics algorithms may not be developed by medical professionals, it is important that life science researchers and doctors understand the underlying concepts to properly build models of diseases. Even if, it is expected that some insight can be obtained through the analysis of data alone, fundamental domain knowledge will always be needed. Hence, it is of fundamental importance that the developed models are not only informed by data but also by an understanding of the domain, i.e., models that clearly understand and connect cause and effect. Furthermore, understanding the strengths

and weaknesses of different analysis approaches by medical researchers will only strengthen the models by designing combinations for improved analysis.

Causality

Epidemiological research usually seeks to identify if any causal relationship exists between the risk factor and the disease. A traditional example that illustrates the association between risk factor and disease is the impact of smoking on lung cancer [WC54]. However, the story of that person who smoked throughout their lives and never suffered from cancer shows that epidemiological problems are not straightforward. There is an association at work, but exposure is a necessary but not sufficient condition for disease. Another problem for establishing causality in epidemiology is the necessity to reject other possible explanations for the observed association. Confounding factors may arise as well, causing a spurious association between dependent and independent variables. For example, many people who smoke heavily have low intakes of vitamins [SHM90]. The Bradford Hill criteria [HIL65] have been used to strengthen the evidence of causality in these types of studies [ANe13]. Statistical methods, as well as sophisticated and specialised methods, have been developed in R, Stata or SAS to conduct research and uncover new causing factors.

Testing

A statistical hypothesis is a hypothesis that is testable on the basis of observing a process that is modelled via a set of random variables [MA63]. Statistical analysis aims at providing statistical insights about the datasets for further research, without any prior statistical knowledge, by performing multiple statistical tests on a given data set. Statistical hypothesis testing is a fundamental technique of both Bayesian and frequentist inference, although the two types of inference have noteworthy differences. Statistical hypothesis tests establish a procedure that controls the probability of incorrectly deciding that a default position (null hypothesis) is incorrect. The procedure is based on how plausible it would be for a set of observations to take place if the null hypothesis were true.

Testing has been of crucial importance early on in biomedical research. Poor and complex signals, curse of dimensionality, computational needs of Bayesian are only a few of the problems that researchers have faced. Many techniques have been successfully applied to overcome these problems. Principal component analysis (PCA) is a frequently used signal separation technique to discover potential subgroups of the dataset [JC16]. It uses an orthogonal transformation to convert observations of correlated variables into linearly uncorrelated ones (i.e. principal components), the number of principal components is less than or equal to the smallest of the number of original variables or the number of observations thus effectively reducing the dimensionality.

Clustering

Cluster analysis or clustering is the task of grouping a set of objects in such a way that objects in the same group (called a cluster) are more similar (in some sense or another) to each other than to those in other groups.

Clustering analysis can fall into two categories taking different kinds of input: feature-based clustering and similarity-based clustering. Feature-based clustering takes a feature matrix as the input and is applicable to raw noisy datasets. Commonly, finite mixture models, such as Mixture of Gaussians Model and infinite mixture models, such as Dirichlet process mixture model are used [Bis06]. The basic idea of using mixture models is first fitting the mixture model with data and then computing the posterior probability of the data point whether it belongs to a cluster. The similarity-based clustering method, on the other hand, requires a distance matrix as the input and facilitates the domain-specific similarity.

In Bioinformatics, clustering methods can be used to group similar samples and also similar features. For example, a gene expression dataset collected from multiple patients can be represented by a matrix, in which rows can represent genes and columns represent patients. The resulting matrix can well exceed terabytes in the context of a proteomics analysis. Clustering by columns (patients) can find groups of patients resulting in a possible patient stratification or discovering a correlation between genes and conditions [KBCG03][OKHC14].

Time-series

A time series is a series of data points indexed (or listed or graphed) ordered by time. Most commonly, a time series is a sequence taken at successive equally spaced points in time and is thus a sequence of discrete-time data. Biological processes are often dynamic; thus, researchers must monitor their activity at multiple time points. The most abundant source of information regarding such dynamic activity is time-series gene expression data [BJGS12]. Not surprisingly, generating time-series expression data has become one of the most fundamental methods for querying biological processes that range from various responses during development to cyclic biological systems. Recent improvements in methods for measuring gene expression, such as high-throughput RNA sequencing, and the increased focus on clinical applications of genomics make expression studies more feasible and relevant.

Prediction

In statistics, prediction belongs to statistical inference. One specific approach to such inference is known as predictive inference, but the prediction can be undertaken within any of the different approaches to statistical inference [Cox07]. As a matter of fact, one practical description of statistics is that it serves as a means of transferring knowledge about a sample of a population to the whole population, and to other associated populations, which is not necessarily the same as prediction over time. The process is known as forecasting refers to the transfer of information across time (often to specific points in time). Prediction is usually performed on cross-sectional data, while forecasting often requires time series methods.

Statistical methods used for prediction include regression analysis such as linear regression, generalised linear models (logistic regression, Poisson regression, Probit regression, etc.). As for forecasting, vector autoregression models and autoregressive moving average models can be utilised. The KaplanMeier estimator has been extensively used for patient survival analysis. The estimate may be useful to examine recovery rates, probability of death, and effectiveness of treatment [KBe11].

Deep learning algorithms, in particular convolutional networks, have rapidly become a methodology of choice for analysing medical images and predicting patient trajectories [GP16, EKe17].

2.1.3 Complexity of computing infrastructures in Life Science

With the increasingly swift growth of data in biology and life sciences in general, we are witnessing a major evolution in the way research is conducted. Until recently, most life science researchers would carry their research on their local machine or a single dedicated bare metal server. Putting aside the poor usage of certain tools [ZEE016], that approach has been successful as long as the size of the data was reasonable (less than a few hundred of GB) and the complexity of the problem limited. However, the limits of that approach have become apparent as the research is increasingly moving from hypothesis-driven studies to data-driven simulations of whole systems to unveil more complex mechanisms. Such approaches necessitate the use of large-scale computational resources and a large panel of computational tools (different languages and packages, different interoperability needs or specialised hardware) in order to run increasingly complex analyses and achieve better data assimilation. Synergies between life sciences and architecture researchers are fundamental in moving research forward as the distribution over many heterogeneous machines is required either to keep up with a large number of user requests, to perform parallel processing in a timely manner, to support mixed workloads or to be able to tolerate faults without disruption to service.

This shift represents a fundamental challenge as the use of large-scale computational and storage resources traditionally requires extensive knowledge in parallel data processing (distributed storage, communication, orchestration, etc.) and the allocation of a large pool of computing resources to a single user would result in a massive amount of compute time being wasted. In order to expose an accessible programming interface to non-expert application programmers, and act as personalised and on-demand bioinformatics services, data processing frameworks hiding challenging aspects of distributed programming had to be developed. Examples of the details abstracted include fault tolerance, complex scheduling, data provenance, security/privacy, and message-based communication. In addition to those core needs, it is vital to make sure that

the data is not lost during computation and nobody else is accessing the data by accident or malice. Finally, the shift towards an on-demand paradigm would prevent users from corrupting the installations inadvertently and would allow architecture researchers to develop new features such as support of privacy methods or improved scaling to be used seamlessly by the users.

2.2 Scalability in distributed systems

The software architecture details in a symbolic and schematic way the different elements of one or more computer systems, their interactions, and their interrelations. Unlike the specifications produced by the functional analysis, the architecture model, produced during the design phase, does not describe what a computer system should achieve, but rather how it should be designed to meet the specifications.

2.2.1 Introduction

Scalability can be defined as the capability of a system, network, or process to handle a growing amount of work, or its potential to be enlarged to accommodate that growth [Bon00]. A system is deemed scalable if it is capable of increasing its total output under an increased load when additional resources (software and hardware) are added. In the context of electronics systems, databases, routers, and networking, a system whose performance improves after adding hardware, proportionally to the capacity added is said to be a scalable system.

An algorithm, architecture, program, networking protocol, or other system is said to scale if it is suitably efficient and practical when applied to large situations: large number of participating nodes in the case of a distributed system, large input data sets or users. If the design or system fails or slows down to the point it is not helpful to the task at hand when the input quantity increases, the system is deemed not scalable. Fundamentally, if there are a large number of causes (n) that affect scaling, then resource requirements (such as algorithmic time-complexity) must grow less than n^2 as n increases. If we take a search engine to illustrate, the search engine

will be said to be scalable not only for the number of users, but also for the number of objects it indexes. Scalability refers to the ability of a site to increase in size as demand warrants while delivering consistent performance [DRW07].

The scalability of system can be measured according to various dimensions:

1. Administrative scalability: An increasing number of organizations or users can easily share a single distributed system.
2. Functional scalability: The addition of new functionality to enhance a system can be executed at minimal effort.
3. Geographic scalability: The distribution of the system from a local area to a more distributed geographic pattern does not result in a loss in performance, usefulness, or usability.
4. Load scalability: The capacity of the system to easily expand and contract its resource pool to accommodate heavier or lighter loads or number of inputs.
5. Generation scalability: The capacity of a system to scale up by using heterogeneous components from different vendors [MMSW07] at any given time.
6. Business scalability: the capacity of a system to accept increased business volume without impacting the contribution margin ($\text{margin} = \text{revenue} - \text{variable costs}$).

Methods of adding more resources for a particular application fall into two broad categories: horizontal and vertical scaling [MMSW07].

- Horizontal scaling (scale-out): The ability to connect multiple hardware or software entities, such as servers, so that they work as a single logical unit (cluster). The addition (or removal) of a new node results in the proportionate increase (decrease) in capacity of the cluster. The decreasing cost of hardware and their underlying increase in performance have enabled “commodity” systems to be commissioned for tasks that once would have required supercomputers. System architects can configure hundreds of small computing

nodes in a cluster to obtain an aggregated computing power that is often greater than computers based on a single traditional processor. The advancement of high-performance interconnects such as InfiniBand, Gigabit Ethernet and Myrinet further encouraged this model. It is those new deployments that have encouraged developers to create new classes of distributed software and efficient management and maintenance of multiple nodes, as well as hardware such as shared data storage with much higher I/O performance.

- Vertical scaling (scale-up): The ability to resize the resources of a single node in a system, typically involving the addition of CPUs or memory to a single entity. The use of virtualization technology enables, in the context of vertical scaling of existing systems, to provide more resources for the hosted set of operating system and application modules to share. Application scalability is the ability of an application to improve performances (number of clients it can serve, computation speed, etc.) on a scaled-up version of the system [ERAEB05].

The two models are not mutually exclusive and each one presents drawbacks and advantages. A larger number of nodes means increased management complexity, concurrency issues, as well as a more complex programming models. The latency and network speed between the nodes of the mesh further complicates the development and deployment of distributed applications. Additionally, some applications do not lend themselves to a distributed computing model, thus do not benefit at all from horizontal scaling. In contrast, a single node will always be bound by the maximum amount of resources the motherboard can host excluding de facto very large-scale applications.

2.2.2 Scheduling and management scalability

In the field of massively parallel processing, job schedulers are the “operating systems” of modern Big Data architectures and supercomputing systems. Job schedulers are responsible for allocating the computing resources and administer the execution of processes on those resources. Historically, job schedulers were the domain of supercomputers, and were designed

and optimised to run massive, long-running computations over days or even weeks on homogeneous hardware for specific tasks [RBA⁺18]. One of the first task of those supercomputers were scientific computations such as modelisation or numerical prediction. Numerical weather prediction using supercomputers started in the 1960s with the CDC 6600 [IWCCtT04] and used mathematical models of the atmosphere and oceans to predict the weather based on current weather conditions.

In its simplest form, a scheduler is composed of three base elements:

1. A queue containing the tasks
2. A worker to execute the task
3. A master managing the queue and dispatching the tasks to the worker

Schedulers are often implemented in a way that puts the emphasis on keeping all computer resources busy (e.g load balancing). That priority gives way to multiple users sharing the same system resources effectively or to achieve a target quality of service defined by the operator of the system. The target quality of service can be defined according to different objectives: maximizing throughput (the total amount of work completed per time unit); minimizing wait time (time from work becoming enabled until the first point where it begins execution on resources); minimizing latency or response time (time from work started until it is finished in case of batch activity [Fei15, LL73], or until the system responds and sends back the first output to the user in case of interactive activity [SGG05]); or maximizing fairness (equal CPU time for each process or a defined minimal amount of compute time according to the priority or workload of each process). In practice, these goals often conflict (e.g. throughput versus latency) and the scheduler will implement a suitable compromise in accordance with the defined guidelines provided to it. Depending upon the user's needs and objectives, the preference can be measured by any one of the concerns mentioned above. In real-time environments, such as embedded systems, the scheduler also must ensure that processes can meet deadlines to in turn guarantee the stability of the system.

However, as data centres and applications grow more heterogeneous and complex, allocating the proper resources to various applications increasingly depends on understanding the tradeoffs

between the different allocations. Complex scheduling strategies have to be put in place to enable mixed workloads to benefit from different types of resources (GPUs, TPUs, SSDs), different generations of hardware, different types of costing to users (shared vs sole allocation), different energy constraints or cross data centres computations (hybrid clouds). Recently, a new class of Big Data data workloads consisting of many short computations taking seconds or minutes that process enormous quantities of data has emerged. Therefore, the efficiency of the job scheduler, for both supercomputers and Big Data systems, represents a fundamental limit on the efficiency of the system.

As detailed by Reuther et al. [RBA⁺18], a great deal of work has been done in the last 40 years to address those issues and two main families of schedulers emerged with the HPC family on one side and the Big Data family on the other. The HPC family can be further broken down into two sub-families: traditional and new HPC schedulers. The traditional HPC schedulers include PBS [Hen95], HTCondor [LLM88], OpenLava [JB16] and LSF [ZZWD93] among others. The new HPC schedulers include Cray ALPS [NP06] and Simple Linux Utility for Resource Management (Slurm) [YJG03]. The Big Data schedulers include Google Borg [BGO⁺16] and Omega [SKAEMW13] which gave way to the open-source scheduler Kubernetes [BGO⁺16], Apache Hadoop MapReduce [DQR12], Apache YARN [XNR14] and Apache Mesos [HKP⁺11]. Both families share common features but are optimised for the specific class of problems we have presented.

2.2.3 Storage scalability

Since the introduction of the first hard drive more than 60 years ago, storage technologies have made huge strides thanks to multiple groundbreaking technological shifts. Those shifts are responsible for the steady progression in terms of capacity, speed and implementation flexibility. The stability of the interfaces throughout those shifts have allowed continual advances in both storage devices and applications, without frequent changes to the standards [MMIGRGE03]. A key feature of the original progression was its vertical approach to scaling. Indeed, the first stage of evolution of storage technology was focused on the growth in storage capacity, speed,

and bandwidth of storage on a single unit of computing at a time. In a second stage, the growth focused on the pulling of multiple storage nodes together in unified logical storage entities and formed the first attempts at Storage Area Network (SAN). But here again, logical storage was approached with a very vertical approach to scalability, clustering pools of nodes into single file systems made available usually through the network in a Network Attached Storage (NAS) fashion.

The concept of Object storage, also known as object-based storage, only started to emerge in the early 2000s. Object storage manages data as objects, as opposed to traditional storage architectures like file systems which manage data as a file hierarchy, and block storage which manages data as blocks within sectors and tracks. Each object typically includes the data itself, some metadata, and a globally unique identifier. This conceptual change has transformed the evolution path from a vertical scale-up to a horizontal scale-out. Objects are storage containers with a file-like interface, effectively representing a convergence of the NAS and SAN architectures. Objects capture the benefits of both NAS (a high-level abstraction that enables cross-platform data sharing as well as policy-based security) and SAN (direct access and scalability of a switched fabric of devices) [MMIGRGE03].

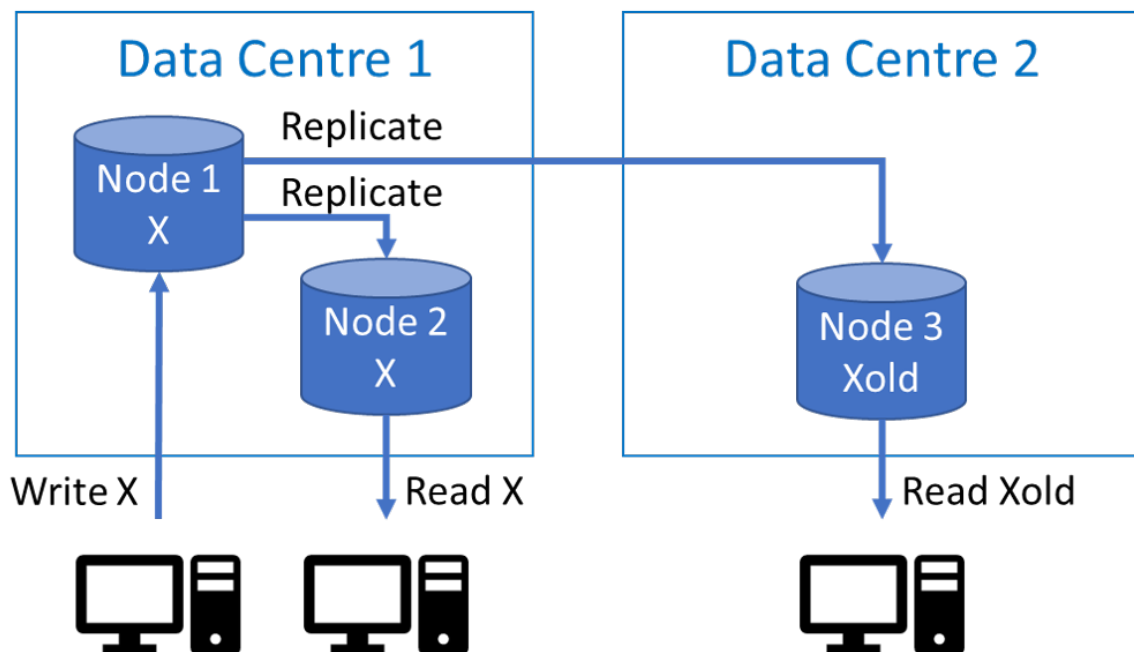


Figure 2.1: Conceptual representation of replication with eventual consistency.

Scalability, for scale-out data storage, is defined as the maximum storage capacity which guarantees full data consistency. This implies that at any given moment there is only ever one valid version of stored data in the whole cluster regardless of the number of redundant physical data copies. Clusters that provide “lazy” redundancy by updating copies in an asynchronous fashion are called “eventually consistent”, while clusters providing immediate redundancy by updating copies in a synchronous fashion are called “strongly consistent”. Figure 2.1 illustrates a scenario where new data is inserted into a cluster with eventual consistency. This type of scale-out design with eventual consistency is suitable when availability and responsiveness are more valuable than consistency, which is true for many websites (such as Facebook) or web caches (a small wait might be required to get the latest version).

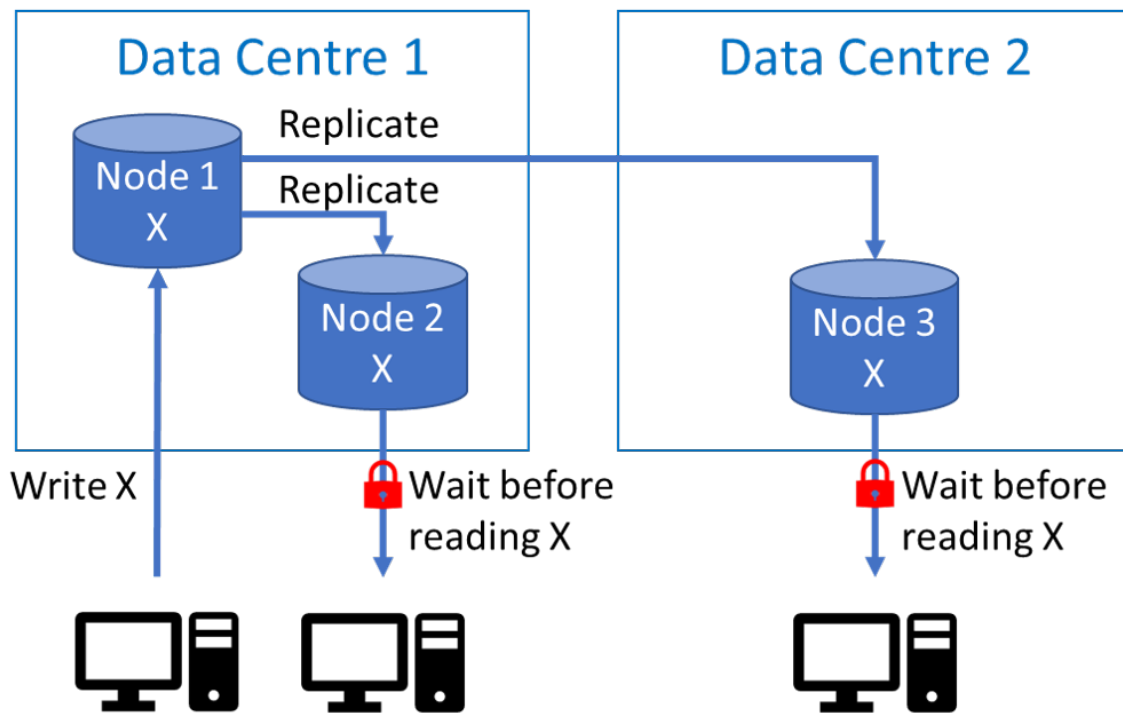


Figure 2.2: Conceptual representation of replication with strong consistency.

Figure 2.2 illustrates a scenario where new data is inserted into a cluster with strong consistency. This type of scale-out design with strong consistency is suitable for all classical transaction-oriented applications. This implies that data viewed immediately after an update will be consistent for all observers of the entity [Gooa]. This characteristic has been a fundamental assumption for many developers working with databases as it is part of the ACID (Atomicity, Consistency, Isolation, Durability) properties of database transactions. However, developers

must compromise on the scalability and performance of their application to obtain those strong consistency guarantees. In other words, data has to be locked during the replication or update processes to ensure that no other processes are updating the same data.

In order to balance strong and eventual consistency, developers and researchers have brought forward several techniques and explored different trade-offs. For example, non-relational databases let developers choose an optimal balance between strong consistency and eventual consistency for each part of the application. The indexes could be subject to a strong consistency while the referenced objects could be eventual. This approach gives more time for the data entities to be replicated across the nodes of the cluster while keeping performances of the application high, effectively combining the benefits of both worlds. But it should be noted that a query against the indexes cannot exclude the possibility of an index not yet being consistent with the associated entities at the time of the query, which may result in an entity not being retrieved at all.

2.2.4 Computational scalability

CPU traditionally refers to a processor, more precisely to its processing unit (registers and combinational logic) and control unit contained on a single chip. As Figure 2.3 shows, computer performance was tied to the clock frequency of the single core processor until the early 2000s thanks to the doubling number of transistors in each chipset. From 2005, the doubling of cores in processors every 18 months enabled manufacturers to continue to follow Moore's Law to some extent for a short while. From 2010, the exponential increases in transistor resources are wasted as they translate only in a linear gain in performance. On the other hand, the amount of data being produced and requiring to be processed is increasing at an exponential rate [YHG⁺16]. It is in this context that new processing architectures better suited to Big Data workloads started to emerge; architectures that offer a scalable, massively parallel, sea-of-cores approach.

The workloads existing processors were designed and optimised over decades to handle are vastly different than Big Data and Machine Learning workloads. Traditional software, such as

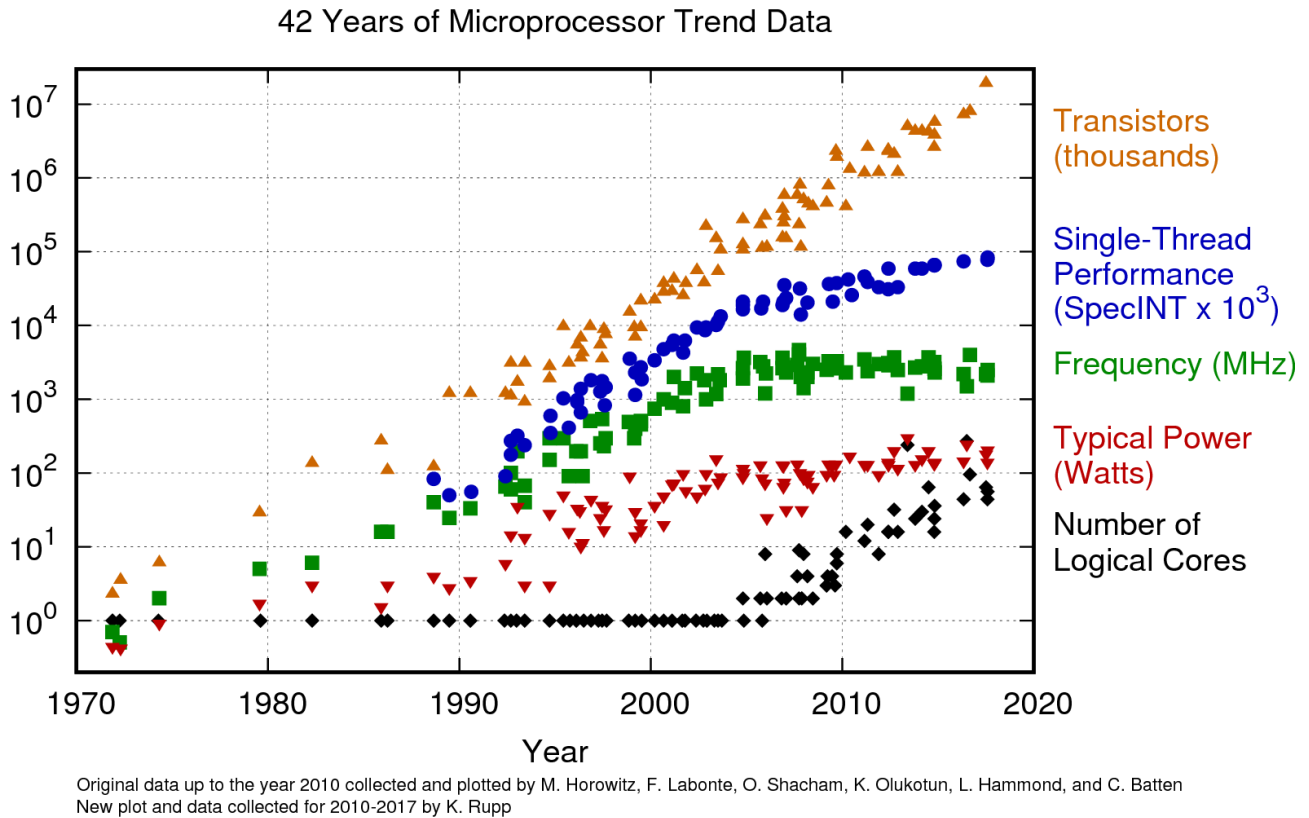


Figure 2.3: 42 Years of microprocessor trend [Kar18]

operating systems or heavy-duty backends in banks, represent millions to hundreds of millions of lines of code, while Big Data/Machine Learning code size is closer to thousands of lines of code. This difference can be explained by a shift in paradigm from a usually linear execution (for traditional software on high power single cores) to individually small computations replicated massively across many, many servers to be executed in parallel. The MapReduce paradigm [DG08a] emerged as a popular solution for processing Big Data sets in a scalable fashion with a parallel, distributed algorithm on large many-cores clusters. The Hadoop implementation, followed by Spark's a few years later, has enabled researchers to run efficiently a massive number of small tasks across a vast number of cores thanks to clever scheduling, data locality (using HDFS) and no concurrency to be handled by the user. Those newly found capabilities have enabled new discoveries in the medical field both in healthcare and translational research [AA16, CCF⁺16, Tay10, MHBe10, QEG⁺10]. Yet, the MapReduce paradigm is a restricted programming framework. MapReduce programs must be written as acyclic data-flow programs, e.g., a stateless mapper followed by a stateless reducer. This paradigm makes repeated querying of datasets difficult and imposes limitations on iterative algorithms that re-

visit a single working set multiple times (which is the norm in deep learning and frequent in machine learning).

2.3 Architectures to support machine intelligence

Machine intelligence, more commonly called Artificial Intelligence (AI), is intelligence demonstrated by machines in contrast to the natural intelligence displayed by humans. In computer science, AI research is defined as “the study and design of intelligent agents” where an intelligent agent is any autonomous entity that perceives its environment and takes actions which maximise its chances of successfully achieving its goals [RN12, PMG98]. The learning of those agents (e.g. machine learning) is a fundamental concept of AI research. While definitions of machine learning (ML) are many and have changed considerably over time, a traditional established view was provided by Mitchell [Mit97]: “Machine Learning is the study of computer algorithms that improve automatically through experience.” Some of the traditional problems or goals of artificial intelligence research include natural language processing, reasoning, knowledge, learning, planning, perception, and the ability to manipulate and move objects [Lug09, PMG98]. Many approaches have been developed in the past few decades including statistical methods, computational intelligence, and traditional symbolic AI. Artificial intelligence also greatly pulls from other scientific fields like social sciences, philosophy, computer science, mathematics, linguistics, and neuroscience. In the twenty-first century, various artificial intelligence techniques have matured and found a new popularity thanks to the recent surge in computing power, large amounts of data, theoretical understanding and, advancements in the field of Big Data. AI techniques and methods have become an everyday tool in the arsenal of researchers helping them to overcome many challenges and solve various complex problems. The most common tools used in AI are mathematical optimization, artificial neural networks, and methods based on statistics, probability and machine learning.

2.3.1 Machine Learning

In the simplest setting, and following Mitchell's definition, machine learning algorithms build a mathematical model of training data. The training data varies depending on which type of machine learning algorithm we plan on using. Machine learning is usually divided into three main types:

1. Supervised and semi-supervised learning
2. Unsupervised learning
3. Reinforcement learning

Supervised learning

Supervised learning is the machine learning task of learning a mapping from inputs x to outputs y , given a labelled set of input-output pairs $\mathcal{D} = (x_i, y_i)_{i=1}^N$ where $\mathcal{D}$ is the training set, N the number of training examples, x_i/y_i the i^{th} input/output of the training set. In the simplest setting, we aim at learning a function $g : X \rightarrow Y$ where X is the input space and Y is the output space.

Some of the most common algorithms used in supervised learning include: Support Vector Machines (SVM), regression (linear, logistic, etc.) and naive Bayes. Classification algorithms are used when the outputs are bounded to a given set of values. Regression algorithms attempt at modelling continuous outputs, e.g. they may have any value within a range.

Unsupervised learning

In the context of unsupervised learning, only output data are given without any inputs. The goal is to discover underlying structures (like grouping or clustering of data points) in a dataset that has not been labelled, classified or categorised. This method is one of the preferred approaches for knowledge discovery. A central application of unsupervised learning is in the field of density estimation in statistics, e.g., building models of the form $p(x_i, \theta)$.

In contrast to supervised learning, there is no single generic objective in unsupervised learning as it varies depending on the tasks and the nature of the dataset. It is also more widely applicable than supervised learning since it does not require a human expert to manually label the data. Some of the most common algorithms used in unsupervised learning include: clustering (hierarchical clustering, k-means, etc.), discovering latent factors (Principal Component Analysis, Expectationmaximization algorithm. etc.), anomaly detection and neural networks. Regression algorithms are named for their continuous outputs (such as temperature or weight), e.g. they may have any value within a range.

Reinforcement learning

In the context of reinforcement learning, we are concerned with how software agents learn how to act and behave in an environment so as to maximise some notion of cumulative reward (which may include a punishment concept as well). The problems at the centre of reinforcement learning are in relation with the theory of optimal control, which is mostly focused on the existence and characterization of optimal solutions, and algorithms for their exact computation, and less with learning or approximation, particularly in the absence of a mathematical model of the environment.

2.3.2 Deep Learning

Deep learning, also called deep structured learning or hierarchical learning, is a branch of machine learning that utilises multiple layers of linear and non-linear functions to transform from inputs into representations that are useful for subsequent tasks such as classification and regression. Similarly to the rest of the machine learning field, learning settings can be supervised, semi-supervised or unsupervised.

An artificial neural network is a network of simple entities called artificial neurons. Each neuron receives input signals, which in turn change their internal state (activation) according to that input, and produce output depending on the input and activation. These neurons are

commonly organised in layers as this allows us to efficiently calculate activations of all neurons in each layer with a simple matrix multiplication. A neural network can be viewed as a directed acyclic graph describing how a sequence of layers is combined to form the network. Figure 2.4 illustrates an artificial neural network with a set of fully connected layers where each circular node represents an artificial neuron and an arrow represents a connection from the output of one artificial neuron to the input of another. Artificial neurons were originally inspired from the working of a biophysical neuron with inputs and outputs but do aim at representing a biological neuron model.

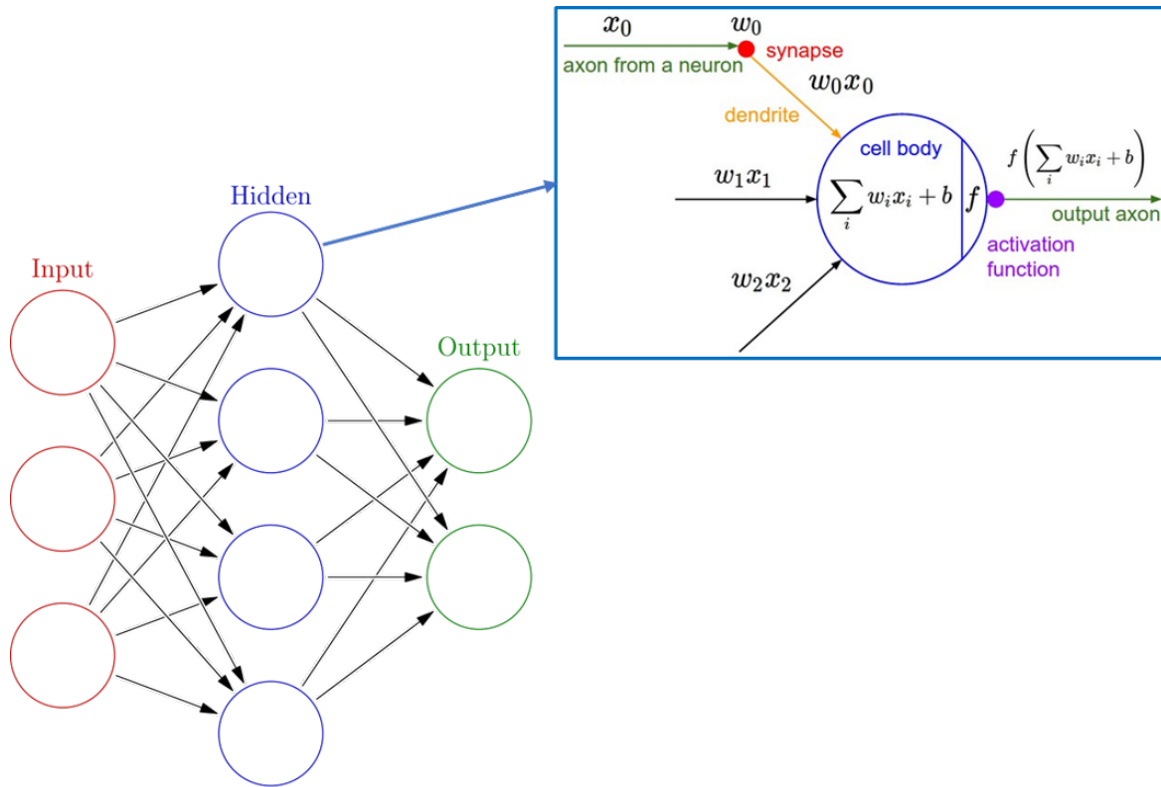


Figure 2.4: An example of an artificial neural network with the mathematical model [Glo] of an artificial neuron [Sta]. An input signal x_0 travels along the axons, which then interacts with dendrites of the other neuron and becomes w_0x_0 based on the synaptic strength. The synapses w_0 between the axon and the dendrite control the strength of influence of one neuron to another. The dendrites carry the weighted input signals to the cell body. The cell body accumulates the results, sums them, and then applies an element-wise function f to fire an output signal via an output axon.

Most modern deep learning models are based on artificial neural networks such as deep neural networks (Convolutional Neural Networks or Recurrent Neural Networks), deep belief networks and recurrent neural networks. Those networks are generally interpreted from the per-

spective of the universal approximation theorem [BH00, Hor91, Csá01] or probabilistic inference [Mur12, Ben09, DY14]. The universal approximation theorem defines the capacity of feed-forward neural networks with a single hidden layer of finite size to approximate continuous functions [Csá01, Has95]. The proof was published by George Cybenko for the sigmoid activation function [Cyb89] and generalised to feed-forward multi-layer architectures by Kurt Hornik [Hor91]. The probabilistic interpretation [Mur12] originated from machine learning research. A distinctive attribute of the probabilistic interpretation is inference [Hin09, Sch15], as well as the optimization concepts of training and testing, related to fitting and generalization. The probabilistic interpretation considers the activation non-linearity as a cumulative distribution function which led to the introduction of dropout as regularizer in neural networks [HSK⁺12].

In the field of medical research, deep learning has been successfully applied to many research problems and became the methodology of choice in some instances. Among the most notable ones, research has explored the use of deep learning to predict bio-molecular targets [DJS14, ZTHZ17], gene ontology annotations and gene-function relationships [CSB14], sleep quality [SDWG17, SJFL⁺16], predictions of health complications from electronic health record data [SNK⁺17], toxic effects of drugs and drugs discovery [LBH15, GHS16, CEW⁺18]. New application domains such as protein folding are also being actively investigated with the first results [WCZQ18, Dee18] showing promises.

2.3.3 Hardware acceleration for AI research

Hardware accelerators belong to the class of Application-Specific Integrated Circuits (ASIC). Their intended use is to execute some functions more efficiently than on a general-purpose CPU. The implementation of computing tasks directly in silicon allows for decreased latency, increased capacity and throughput is known as hardware acceleration. Many hardware technologies can be used in accelerating machine learning algorithms, such as Graphics Processing Unit (GPU), Tensor Processing Unit (TPU) and Field-Programmable Gate Array (FPGA). Any operation on the data can be computed purely in a software running on a generic CPU, custom-made

hardware, or in some mix of both. Hardware accelerators are crucial for artificial intelligence applications, especially computer vision, machine learning, and artificial neural networks. The potential applications for artificial intelligence accelerators include autonomous vehicles [NVI], robots [Mov], natural language processing [Qua] and health care among many others.

Hardware acceleration is helpful for performance and power efficiency when the functions are fixed so updates are not as needed as in software solutions. The emergence of reprogrammable logic devices, such as FPGAs, has diminished the restriction of hardware acceleration to fixed algorithms. Reprogrammable logic devices have allowed hardware acceleration to be applied to problem domains requiring to apply modifications to algorithms and processing flows. The reprogrammability has also enabled the possibility to improve the logic on existing devices remotely without changing the device altogether. The number of IoT devices becomes increasingly large due to the demands of the practical applications (such as CCTV or medical monitoring), which poses a significant challenge to construct a high-performance implementation of machine learning algorithms directly on the devices. In order to benefit from increased locality of data to execution context, thereby reducing computing latency between modules and functional units and network bandwidth consumption, register-transfer level (RTL) abstractions of hardware designs have been developed. Those abstractions have allowed emerging architectures such as in-memory computing, transport triggered architectures, and networks-on-chip to fully realise that potential.

Data centers benefited as well from those innovations as architectures evolved towards heterogeneous multi-cores composed of a mix of cores and accelerators, a machine-learning accelerator can achieve the rare combination of efficiency (due to the small number of target algorithms) and broad application scope. Unlike accelerators on IoT devices, accelerators in data centers can tackle large scale problems as space and energy are no longer a constraint. Those new architectures aim at tackling classes of problems that used to be computationally prohibitive or impossible such as ray tracing (RT cores), cryptography (Hardware Random Number Generator), multilinear algebra (TPU) and computer vision (Vision Processing Unit). The synergy between the large data sets in the cloud and the numerous computers that power it has enabled a renaissance in machine learning [JBBe17]. In recent years, deep neural networks (DNNs) have led to

breakthroughs such as reducing word error rates in speech recognition by 30% [Dea16]; cutting the error rate in an image recognition competition from 26% to 3.5% [HZRS16, SLJ⁺15, KH12]; beating the best human players at Go [SHMe16]. Those breakthroughs were made possible thanks to the broader developments and adoption of hardware accelerators (TPUs in this instance) in AI workflows for training the deep learning models. TPUs are very fast at performing dense vector and matrix computations with gains ranging from 15x to 30x faster than contemporary GPUs and CPUs [Goob]. The energy efficiency achieved by TPUs is also much better than conventional chips, with improvements between 30x and 80x in TOPS/Watt measure.

However, the introduction of custom hardware into AI workflows complexifies the design of analytical environments that efficiently leverage them.

2.4 Compliance and security in distributed systems

2.4.1 GDPR and privacy of patient data

Medical data is deeply personal as it can reveal more information about us than any other piece of information. Yet, in order to provide treatments to the patients, it is vital that health professionals get access to this information. This conundrum of balancing the protection of personal information and using the data for the advancement of medical research has been understood by society and lawmakers alike. It is in this context that a legislation protecting medical data has been voted and implemented in a growing number of countries.

The growing use of electronic health records (EHR) concomitant with ever more data being collected in general by public health systems in countries like France (Sécurité Sociale), the UK (NHS) and the US have opened new possibilities to medical researchers. By mining these masses of data, researchers and doctors can improve diagnostic techniques as well as treatments by moving from symptom-based to evidence-based medicine based on data mining. The growing interest from medical research to make data available is met with new ethical guidelines as well as legislation [HA17]. The awareness of the legal and ethical context when working with patient

data is of key importance in order to protect private patient information. The legal and ethical standards that researchers must follow typically depend on the institution or hospital where the data is collected and stored. This situation introduces a great variance between institutions and countries. Naturally, the ethics boards might have very different initiatives depending on their objectives to put an emphasis on research or security of the data. However, those guidelines are not a superset of the legal requirements: the legality of the initiative always comes before the compliance with ethics.

The General Data Protection Regulation (GDPR) [TT16] of the European Union (EU) came into law on the 25th of May 2018. GDPR is now at the heart of the EU privacy and data protection for all individuals within the European Union and the European Economic Area. Those laws aim at defining the export of personal data outside the EU, give control to individuals over their personal data and to simplify the regulatory environment by homogenizing the regulation across all EU countries. A central notion of the GDPR is the concept of *informed consent* of the patient which specifies that no personal data may be used unless the researchers have received an unambiguous affirmation of consent from individual data subjects. Providentially, the final version of the regulation includes a special set of rules for research in recognition of the benefits that research offers to society. Those rules stipulate that data minimisation is a requirement, meaning data has to be de-identified to the extent that research objectives can be achieved [BBM17]. This rule gives some leeway for research as it does not enforce a mandatory anonymisation of the data. Another rule, to facilitate the re-use of data for research, stipulates that the collected data can be freely shared between member states even if the data was collected for another purpose. The free movement of personal data (without the need for explicit consent) opens the way for masses of patient data collected in hospitals to be used for medical research.

The approach advocated for the de-identification of the data is to use the concept of pseudonymization as described in Article 4(3b) [Vol18]: “the processing of personal data in such a way that the data can no longer be attributed to a specific data subject without the use of additional information, as long as such additional information is kept separately and subject to technical and organizational measures to ensure non-attribution to an identified or identifiable individ-

ual”. The International Standardization Organization (ISO) proposed a set of guidelines for re-identification risk assessment and implementation aspects of de-identification through ISO 25237:2017 [Int]. In practical terms, pseudonimization is the replacement of identifiers (like names and social security numbers) into a new domain space of equal size or larger. One common way to carry out those transpositions is with the use of hash functions (SHA-3, AES-128) and a long cryptographic salt hash per data set. An interesting aspect of GDPR is that it represents a minimum standard from which countries can deviate from, leaving state members to work out the details for themselves. On the one side, this facilitated its adoption by the member states as it adapts more easily with the existing local rules and can be made more relevant to their own society and culture. On the other side, this flexibility has led member states to implement their own, sometimes contradictory (such as in France’s case [Tam18, 20117]), systems of safeguards and exemptions for subject data rights in the context of research. Those discrepancies have made the task of creating a cross countries data processing infrastructure for medical data a lot more arduous.

In the US, something similar has been brought forward with the Health Insurance Portability and Accountability Act Title II (HIPAA) [Rot13]. HIPAA, unlike GDPR, gives very well-defined requirements for national standards for electronic health care transactions and national identifiers for health insurers, healthcare providers and employers. HIPAA details specifically what identifiers need to be removed from the datasets for it to be considered de-identified and for the medical researcher to be in the legality. Namely those 18 identifiers are: names, geographic subdivisions smaller than states, all elements of dates (except year) for dates directly related to an individual, telephone numbers, fax numbers, E-mail addresses, social security numbers, medical record numbers, health plan numbers, account numbers, serial numbers, driving license/license plate numbers, Internet protocol addresses, web Universal Resource Locators (URLs), full face photographs, biometric identifiers (finger and voice prints) and a limited number of similar unique identifiers of a subject [UC]. The benefit of clearly defined rules for a legal, safe use of the data for research purposes facilitates the life of researchers. Besides, the HIPAAs provisions do not unduly limit research on the data but just frame its use. Nevertheless, removing fields like the exact geographic location, day and month from dates specifically

relating to people may hinder research somewhat. One substantial drawback of the HIPAA compared with GDPR is the lack of flexibility. Because the rules are so specifically tailored, any new identifiers (such as Facebook or LinkedIn profiles) is automatically left out and a gap in data protection appears, requiring a new set of laws to plug the hole. With an average of 263.57 days for a bill to pass into law in the US [Car15], this leaves people without any privacy protection for a substantial amount of time.

2.4.2 Security of the data

Data security is defined as all the protection measures taken to warrant unauthorised access to computers, databases, storage devices and websites. Data security also protects data from corruption (bit rot, malicious alterations).

Bit rot, also known as data degradation, is the gradual corruption of computer data due to an accumulation of non-critical failures in a data storage device. The data degradation is the consequence of the gradual decay of storage media over the course of long periods of time. The causes may vary by medium:

- Solid-state media may experience data decay due to imperfect insulation causing the electrical charges, which are used to store the data, to leak away. Manufacturers are handling the probabilities of bit rot by Solid-state media through the extensive use of error correcting codes (ECC).
- Magnetic media may experience data decay as bits lose their magnetic orientation. Periodic refreshing by rewriting the data can alleviate this problem. Magnetic tapes are currently the reference solution for long term (>20 years) storage in banks for example.
- Optical media may experience data decay from the chemical breakdown of the storage medium. The storage of the optical media in dark, cool, low humidity places can lessen the speed of decay.

Most disk, disk controller and higher-level systems are exposed to chances of unrecoverable failure. The growing capacity of disks and file sizes have increased the likelihood of the occurrence of data decay and the risk of other forms of uncorrected and undetected data corruption

increases. Redundant segregated copies, higher-level software systems (implementing integrity checking like checksum and self-repairing algorithms) or replicated data storage (such as object storage) may be employed to mitigate those risks.

Data privacy can contribute greatly to the data security as described in Section 2.4.1.

Data encryption is the transformation of data into another form, or code, in such a way that only people with access to a secret decryption key or password can read it. Encryption does not prevent interference but denies the attacker to get immediate access to intelligible content. In an encryption scheme, the plaintext is encrypted using a cipher (such as AES) generating ciphertext that can be read only if decrypted. Depending on the constraints and requirements, the encryption can happen at rest for storage (through hardware or software encryption) or on the fly for transmitting to a recipient (similar to HTTP over TLS for web browsing).

Network segmentation is the separation of a computer network into subnetworks, each being a network segment. The segmentation enables the segregation of different parts of the environment and the creation of data silos to act as an additional layer of protection for the data. One usual way to establish network segments is by setting up different virtual local area networks (VLANs). Those VLANs may overlap or be subordinated to one another (e.g. one VLAN may be able to access another VLAN but the inverse might not be true). This technique has been extensively used as a privacy mechanism in the context of the use of location data in the context of public health research.

2.4.3 Privacy of companies

The privacy of citizens has been highlighted extensively in the last two years with the introduction of GDPR and a large number of data breaches from companies such as AOL, Yahoo or Equifax. However, another, equally important, type of privacy has been largely left out of the public eye is the privacy of companies. To the extent of our knowledge and after extensive research on the topic, there is no official/unofficial/legal definition of the privacy of companies. In the context of this research, we define the *Privacy of Companies* as the state in which one

company is not observed or disturbed by another company or state. For example, industrial espionage understood as the covert and sometimes illegal practice of investigating competitors to gain a business advantage, falls under that umbrella. Yet, the spectrum covered by the privacy of companies is much broader and nuanced than simple industrial espionage.

The publication of certain information, by the company itself, of apparently innocuous material could turn out to be either extremely damaging or have unintended consequences. One obvious example is the publication of “anonymized” data by the Massachusetts Group Insurance Commission in the mid-1990s. The re-identification of Massachusetts Governor William Welds medical data in 1997 from the insurance dataset (which had been stripped of direct identifiers) has shown the limitations of de-identification methods. A less obvious example, in the context of the *Privacy of Companies*, is the extraction of derived knowledge or metaknowledge without direct access to the data. The automated computation and publication of population densities by telecommunication companies using Call Detail Records (CDRs) illustrates that risk. In appearance, the publication of population densities for cities or regions do not represent a direct risk for the company as it does not reveal any trade secret or information about the company. However, by observing different deltas between the official numbers and the ones computed by the company, it is possible to infer the approximate market share of the telecommunication company for a given city or region. Competitors could, in turn, use that information to focus their targeted marketing efforts only to specific regions of interest. Those actions would probably result in hurting the bottom line of the company publishing the data and thus, breaching the privacy of the company.

2.5 General-purpose analytical platforms for Life Science

Many federated and distributed systems have been developed to analyze medical data. This section explores research work aimed towards addressing the main challenges of the efficient management and flexible analysis of large-scale medical data in a privacy-preserving fashion.

2.5.1 Introduction

General-purpose analytical platform usually refers to systems that allow analysts to send many queries of different types, using a rich and flexible query language. Depending on the interoperability requirements, those platforms have taken different approaches such as APIs or clients (Python, Java, etc.) to support a flexible query language. A larger number of entry points facilitates the integration with other applications thus encouraging the adoption and expands the pool of potential adopters. An important aspect related to processing medical data is to keep track of the data used and how data products were derived and what, i.e., input data software (and versions thereof), was involved in producing it. Capturing, storing and enabling querying of the data provenance or lineage is crucial to enable open and reproducible research. Many examples in the past, such as The Reproducibility Project: Cancer Biology initiative¹, have shown that data alone was not enough to reproduce experiments, thus weakening the results and credibility of the published research.

Scientists and researchers take advantage of various tools and frameworks for data analysis, design algorithms and build computational applications. However, all those applications have similar requirements with respect to data manipulations, processing, and algorithm development. The first manipulations in most analytical pipelines are data transformations: dimensionality reduction, instance selection, and data cleaning. Dimensionality reduction aims to map high-dimensionality space onto lower-dimensionality ones without significant loss of information. A variety of means exists to reduce dimensions in the context of large-scale data but a popular one is Principal Component Analysis (PCA) [Pea01]. Instance selection refers to techniques for selecting a data subset that resembles and represents the whole dataset. While dimensionality reduction deals with wide datasets, data reduction, more specifically instance selection, aims to reduce a dataset's height. Lastly, data cleaning is another type of data manipulation; it refers to pre-processing such as noise and outlier removal. Thus, it addresses the challenges of dirty and noisy data. Those steps are crucial to ensure the robustness, flexibility, and adequacy of the inputs for the subsequent analyses.

¹<https://cos.io/rpcb/>

The second manipulations are data analysis and storage, where the term storage refers not only to physical storage on a permanent medium, but also to how data are represented in memory. In these stages, processes can be embedded to capture data provenance and therefore remedy provenance challenge independently of the category of manipulations. Those large and sophisticated workflows with dozens of compute- and data-intensive tasks that span across a plethora of heterogeneous data rely on very different stacks and technologies demanding faster and more capable infrastructure. Adding more hardware of different nature (CPU, ASIC, GPU, etc.) is not always possible or sustainable due to complexities, costs, and the risk of data and cluster sprawl. Building a single shared infrastructure for both types of workloads is both pertinent and beneficial.

As detailed in Section 2.4.1, multiple privacy laws and increased sensitisation of the public opinion to the risk caused by the use of sensitive information mandates the inclusion of strong data security capabilities in the design of any platform. Secure computation, audit, and privacy by design are some of the core features to increase overall security and comply with the latest legislation that governs the use of that data. In addition, data provenance and privacy are cornerstones for enabling open science and reproducibility of data analysis with versioned scripts and tools while securing the highest degree of ethical research. Total transparency for scientists increases the credibility of their work among peers.

Therefore, any bioinformatics framework must support production pipelines made up of both parallel and serial steps, varied software and data types, complex dependencies and security constraints, fixed and user-defined parameters and outputs.

2.5.2 Existing architectures

Among the most prominent ones (excluding the eAE), we can cite IBM Platform Conductor [IBMb], Arvados [Cur], Berkeley Open Infrastructure for Network Computing (BOINC) [And04] and Petuum [XYH⁺15]. Those platforms share some common features (scalability, scheduling, etc.) as well as specificities linked to the fact that they originally aimed at answering different targeted needs for the users. Some other projects, such as Diffix [FPEM17], aimed at solving

part of the problem by providing database querying capabilities that anonymise query results by adding noise tailored to the query as well as the underlying dataset. However, this solution would be inadequate in our context as it would not scale to large scale datasets ($>1\text{TB}$), would require preloading the cleaned-up data (thus supporting only limited data transformations) and provide only basic statistical capabilities with regards to analytics. Table 2.1 summarizes the differences in the main features provided by comparable existing systems (proprietary or not): eAE, IBM Platform Conductor, Arvados, BOINC and Petuum.

From Table 2.1, we can see that the different platforms have very different sets of supports for analytics. Those differences originate from differences in the type of problems that they try to address. Some platforms such as IBM Platform Conductor, eAE and Arvados aim at being universal at least to some degree by relying on commonly used languages for analytics such as Python and R. To the contrary, BOINC and Petuum have opted to attempt at achieving higher performances. The flip side is that they inherently limit the scope of supported analytics. In the context of fast pace research and rapidly evolving needs, the latter choice proves inadequate at supporting the needs of a broad scope of users.

BOINC was designed from the beginning to run on volunteer computing (using consumer devices) or grid computing indifferently. This requirement drove the development towards a highly decentralised architecture and high-capacity at staging and resuming tasks in the eventuality that the node goes offline. It is that idea that drove part of the development of the multi-master scheduler of the eAE to support a decentralised architecture with heterogeneous compute nodes possibly coming online and offline at any given time. IBM and Petuum took the standard production approaches to attempt at restarting the nodes and rescheduling the failed jobs on other nodes in the meantime.

The different projects have taken very different approaches with respect to storage capabilities. BOINC has no storage capabilities per se as it transfers any required data with each job request with the results being sent back to be stored as files. However, some data locality awareness has been put in place resulting in jobs being preferentially sent to clients where the input files are already available. This approach is not adequate in the context of life science research where

Platforms	eTRIKS Analytical Environment	IBM Platform Conductor	Arvados	BOINC	Petuum
Analysis support					
Spark	●	●			
Python	●	●	●		
R	●	●	●		
C/C++				●	
Fortran				●	
Java		●	●		
Go/Ruby/Perl			●		
Fixed set of ML pipelines					●
Computation types					
CPU	●	●	●	●	●
GPU	●	●		●	●
Storage capabilities					
SQL	●	●			●
NoSQL	●	●			●
Content-Addressable Storage	○	●	●		
Monitoring and scheduling capabilities					
Jobs Status	●	●	●	●	●
Clusters Status	●	●	○	●	●
Complex batch processing	●	●	○	●	●
Multi-master scheduling	●	●		○	
Workflow capabilities	○	●	●		○
Data security capabilities					
Data provenance	○	●	●		
Extensive platform audit	●	●	●	●	
Privacy	●	○			
Secure computation (sandboxing)	●	○			
Support of GDPR compliance	●	○			
Interoperability					
REST API	●	●	●		●
Distributed Clients	●	●	●	●	●
Platform support					
Installation procedures	●		●	●	●
Configuration documentation	●	●	●	●	●
Support available	●	●	●	●	●
Open Source project	●		●	●	○

● Fully supported
 ○ Partially supported

Table 2.1: Feature comparisons between eTRIKS Analytical Environment, IBM Platform Conductor, Arvados, BOINC and Petuum.

data files may be terabytes large and whole files are transferred to clients even if then they need only a sub-part of it. Arvados has taken a similar approach with content-addressable storage which suffers similar issues but with the added benefit of facilitating workflows and support for data provenance. The other projects have put in place more advanced storage capabilities with the support of SQL/NOSQL databases and content-addressable storages (Swift for eAE and Spectrum Scale technology for IBM). Those storage layers enable more fine-grain data payloads for jobs thus lowering the compute time, wasting less bandwidth and over whole better performances at the cost of increased architecture complexity and more complex loading procedures. One interesting feature unique to IBM Spectrum Scale is the availability of a storage interface allowing the data to be stored in whichever storage but accessible across all of them thanks to complex internal replication procedures capabilities. Even though it requires some manual set up and configuration, it greatly facilitates the setup of data lakes.

A crucial aspect related to processing data in general and medical data in particular, is to keep track precisely of how data products were derived and what, i.e., input data software (and versions thereof), was involved in producing it. Capturing, storing and enabling querying of the data provenance or lineage is thus crucial but is also well supported in the context of workflow systems. This is particularly well supported in Arvados and IBM Platform Conductor, while absent from BOINC and Petuum. The eAE was designed to support data provenance with the extensive amount of metadata stored for each job (parameters, versioning of data and tools, etc.) but the absence of full-fledged workflow capabilities limits the lineage between jobs. This design was done on purpose to limit the capacity of an attacker to easily infer a user's activity in the eventuality of a breach. The full-fledged provenance using the eAE is done through the use of Borderline (see Section 4.1.2). Nonetheless, it would be possible to implement it fully in the eAE in the eventuality where this would have to be handled by the eAE itself rather than an external tool as it was decided in the context of this research.

Privacy concerns, now enshrined in law, are strongly limiting the use and sharing of data, even when it has been shown to have a great potential for providing new insights (e.g. predict the spread of epidemic disease) and value (e.g. improve public services, inter-business processes). Because this awareness is only recent, all those platforms (apart from the eAE) did not include

any privacy capability by design even if they all include some security measures acting as a layer of protection. In order to unlock the potential of data while preserving privacy, we adopted a query-based paradigm for data release. Instead of publishing (de-identified) data, eAE stores the data in a protected environment and allows analysts to send queries about the data. Since analyses are computed using fine-grained data, eAE makes it possible to achieve better utility and stronger privacy compared to de-identification techniques. The eAE manages risks using a combination of server-side security (Differential privacy, secure computation, etc.), authentication, audit and network security. However, those protections are only the core layer upon which adopters can rely on and further extend it to meet the most stringent requirements. This work will be further described in Chapter 5.

Finally, we must point out that, whilst IBM Platform Conductor is a very strong candidate as a general-purpose analytical platform for Life Science, it is not an open source project, unlike the others. The cost of the license to operate the platform makes it less than desirable in a research environment where the associated cost might hinder independent reproducibility and increase the cost of doing public research.

2.5.3 Conclusion

The increasing integration of patient-specific data into clinical practice and research raises serious privacy concerns. Patient data, like 'Omics, clinical data, location data, iris scans, and other biometric data are very sensitive data as they can (re-)identify people uniquely. In most instances, those data cannot be fully anonymised since anonymizing to the point where an individual cannot be re-identified is equivalent to destroying the utility of the data altogether. It is therefore critical to provide an efficient and scalable platform to researchers to support privacy preserving life science analytics to process the humongous amount of medical data.

Chapter 3

eTRIKS Analytical Environment: Design Principles and Core Concepts

In the previous chapter, we highlighted the need for a shift in architecture paradigm due to large scale analysis in life science. In this chapter, we introduce the eTRIKS Analytical Environment framework for the efficient management and analysis of large scale medical data, in particular, the massive amounts of data produced by high-throughput technologies.

3.1 Introduction and users' needs

Translational research is the interdisciplinary branch of the biomedical field with the goal of combining disciplines, resources, expertise, and techniques to promote enhancements in prevention, diagnosis, and therapies [CMe15]. The eTRIKS Analytical Environment aims at catering for that very broad range of users ranging from biologists with limited notions of computing to computational biologists with very different needs in terms of computation requirements and habits. Medical doctors with no programming capabilities can only do analyses in an interactive fashion using a user interface such as tranSMART or Borderline. Running analytical pipelines in an interactive, intuitive and easy fashion against large cohorts could greatly facilitate the input of their expert knowledge into the research.

R is a fundamental requirement for statisticians and bioinformaticians who have an extensive range of solutions implemented for them to conduct their research. Bioconductor, Limma, WGCNA, and QDNAseq are some of the most popular R packages used in the field of 'omics research. Many life science visualizations packages have also been developed only in R. Thus, it is mandatory to support R and those packages to support the day to day needs of those researchers.

A substantial part of research being carried out in the field of crop and plant science is the systematic sequencing of genomes. The subsequent realignment needed after the sequencing requires a humongous amount of computing power. Indeed, it is not rare for those genomes to be many times larger than the human one (e.g. a rare Japanese flower named Paris Japonica is 50 times the size of a human genome [PFL10]) and there are an estimated 382,000 species of vascular plants currently known to science according to researchers at the Royal Botanic Gardens [RBG17]. The massively parallelisable nature of the alignment tasks makes Spark the perfect candidate to support those systematic needs.

The automated collection of biosignals (through IoT, connected devices, etc.) has become an important indicator for passive health monitoring and medical diagnosis. Clinical researchers use those time-series to discover meaningful features of the human functional state. Researches in this field are focusing more and more on deep learning techniques using GPU to automatically learn features from raw biosignals. Public health researchers are also extensive users of time-series to model and monitor disease outbreaks and propagation. The processing of those time-series requires flexible storage and efficient retrieval capabilities to enable monitoring in real time. The compatibility of the architecture with time-series databases (Timescale, InfluxDB, etc.) and streaming platforms (Apache Kafka, etc.) coupled with stream processing computation engines (Apache Spark or Storm) is essential in that regard.

Patient stratification is the division of the patient group into subgroups, also referred to as 'strata', by investigating distances between a variety of components of patient data. Each stratum represents a particular section of the patient population. Patient stratification is a crucial element in precision medicine and one of the most important logistical and statistical

challenges when carrying out clinical trials. Computational biologists use clustering algorithms and machine learning techniques based on their similarities in various features, including 'omic and clinical profiles, to group similar patients and identify subgroups of complex patients. R and Python have been widely used to carry out these analyses.

Consequently, Python was quickly added to the list of languages that need to be supported because of its extensive support for machine learning libraries (TensorFlow, scikit-learn, etc.), support by Spark, time-series and its higher utility value as a full-fledged programming language. The sheer size and variety of the data leveraged in translational research added another level of complexity. It mandated that the flow of the data between the user and the compute nodes would have to be transparent, seamless and highly efficient to meet the high level of users expectations.

Another aspect that became clear was the interest in multitenancy and collaboration between users. The goal of those capabilities was to enable researchers to easily share their results with each other in a standard and seamless fashion as well as enhance the reproducibility and checking of the results. Still nowadays, most workflows are built in a downstream fashion with researchers one after the other building on top of each other. The multitenancy opens the way to the concurrent collaboration between different types of scientists in a fashion that would otherwise be difficult. This is very much in line with the open science [MNB⁺17] and good practices philosophy of the eTRIKS project and promote the highest standards in science.

Finally, the eTRIKS Analytical Environment aimed not only at supporting current needs but also supporting emerging ones and even offer the possibility to support ones that do not exist yet. we will also present how we address those needs thanks to the flexibility, modularity and extensibility capabilities of the architecture.

3.2 Existing knowledge management platforms and their limitations

Platforms allowing the management and exploration of clinical and 'omics data have been developed to address issues of Big Data. A review [CRe15] of seven publicly available solutions (BRISK [TTD11], caTRIP [MDe08], cBioPortal [CGe12], G-DOC [MGe11], iCOD [SMe10], iDASH [OMBe12] and tranSMART [SKKP10]) yielded consistently the same finding: they usually support only a limited set of data types and analysis as they were designed to address extremely specific needs. On top of that, BRISK, caTRIP, and iCOD have been depreciated and GDOC has been replaced by a closed source version called G-DOC Plus. I2B2 [MMe06] is another translational research platform; however, it supports only a very limited set of 'omics data types and very limited technologies (Java 8 or older) and patterns (e.g. SQL star schema). A lot of work has been done towards the improvement of the storage scalability on these platforms. For example, a plugin has been developed for tranSMART using HBase for storing and managing large scale microarray data [WPe14]. The results showed that, in general, the key-value implementation using HBase outperformed the relational model on both MySQL Cluster and MongoDB. However, one aspect that this plugin did not cover is the flexibility and extensibility of a HBase solution compared to a MongoDB one. For example, the use of newer or richer microarray data formats would render the plugin unusable. Furthermore, these improvements close only a minority of identified gaps and many of them still rely on outdated paradigms and technologies.

A major limitation on all these platforms is their limited built-in capacities to run complex analytics on the selected data. Both tranSMART and cBioPortal provide builtin sets of analytics through R or Matlab. These analytics, which were once fit for purpose, are now showing their limitations both in terms of scalability and utility. The first version of tranSMART was implemented more than nine years ago. At that time, hardware resources were more limited than they are now, and technologies have evolved a lot since then. tranSMART's analytics were written in R and designed to process small to medium size data on a single node with a single

core. Some ameliorations have been implemented since then, but most of the original design has been kept due to the absence of any serious alternative or requests from the users. This is the primary motivation driving us to develop a new framework for analytics, supporting rich and scalable analytical workflows by using tranSMART as the data warehouse.

Originally developed by a subsidiary of Johnson & Johnson, tranSMART is an open-source data and knowledge management platform that enables researchers to develop and refine research hypotheses by investigating correlations between different types of data and assessing their analytical results in the context of published literature and other work. It accommodates phenotypic data, such as demographics, clinical observations, clinical trial outcomes, and adverse events, and high content biomarker data, such as gene expression, genotyping, metabolomics and proteomics data. The latest version of the tranSMART application (17.2) has been written using Grails 3.2.3 and follows the Model – View – Controller (MVC) pattern. A number of plugins have been implemented either to enrich the application with new functionalities or interface with other software or components such as HiDome. Originally tranSMART is provided with an “Advanced Workflows” module. The workflows (19 in the version 17.2 of tranSMART), which are all written in R, were once fit for purpose but have become very limited in recent years. Attempts at replacing the default R with RevolutionR and subsequent marginal improvements in performances were implemented, but those attempts were only temporary solutions and did not satisfy the needs required by the projects. The use of Bioconductor and various attempts to write libraries in different languages (CUDA, C, etc.) then add a wrapper for R have usually had limited impact because of poorly optimised code and/or designs and frequently doesn’t generalise at all. The size of data to be processed is increasing, and the analyses, which rely more and more on machine learning techniques rather than basic statistical methods, become more complex. In addition, the implementation of the “Advanced Workflows” module was far from optimal. It requires several reading and writing operations to disk for every single workflow run, all the visualizations were static images and there was no cache mechanism to store the computed results. Moreover, if the results were not saved or if the connection to the ongoing computation was lost (the session times out, browser refresh, internet issues, etc.) the whole computation had to be restarted while the former computation is still running on the

server for no reason consuming resources.

The development of the Galaxy [GNT⁺10] plugin for tranSMART was a first attempt at integrating an external framework for analytics to enhance the analytical capabilities already in tranSMART. Galaxy is an open-source software written in Python that can be downloaded, installed and customised to address specific needs. Galaxy is part of the GMOD project [ODC⁺08], and the whole Galaxy framework has been developed to address a broad scope of needs. The project has a well-defined set of APIs, a community wiki, many high-quality extensions, an active community of developers and a rich set of tutorials. Despite the high level of sophistication, the platform was designed to be accessible to everyone, especially to biologists with no computer science background. That concept made it very popular among researchers. All those characteristics made Galaxy a very attractive solution to address the need for complex workflows in a flexible fashion. The plugin in tranSMART enabled the seamless transfer of data from tranSMART to Galaxy using the powerful cohort selection of tranSMART. The user could then trigger the associated workflows to process the data and visualise the results in Galaxy. The first limitation of that approach was the impossibility to push back those results to tranSMART which forced the user to navigate between the two applications. The second limitation and most important one was that, even though Galaxy is a very capable and flexible tool, it was never designed to scale to terabyte level analytics. Some workarounds were developed [Gal17] to work in traditional HPC environments but only a few of those integrations were made public and for the others, the integration was tightly bound to the specialised hardware used which would bring no benefits to people who couldn't afford the pricey expenses.

The framework we propose, i.e. the eTRIKS Analytical Environment (eAE), on the other hand supports a wide scope of medical analysis tasks to be performed in high-performance and scalable fashion. These tasks range from high-throughput sequencing data analysis, Genome-Wide Association Study (GWAS), expression analysis of genomics, proteomics and metabolomics data, time series analysis to medical imaging analysis. This framework has the vocation to be scaling on numbers due to a design based on Big Data architecture with the capacity of terabyte (TB) level analysis [Dat14].

3.3 eTRIKS Analytical Environment

3.3.1 Introduction

Personalised medicine is quickly becoming a data-driven science. Improving patient care, developing personalised therapies and new drugs depend increasingly on an organizations ability to rapidly and intelligently leverage complex molecular and clinical data from a variety of internal, partner and public sources. As analyzing these large scale and complex datasets becomes computationally increasingly expensive, traditional analytical engines are struggling to provide a timely answer to the new sets of questions that medical scientists are asking.

Designing such a framework is developing for a moving target as the very nature of data science requires an environment capable of adapting and evolving at the same pace as medicine does. The resulting framework consequently must be a scalable, on-demand, flexible and efficient solution resilient to failure. In response to these trend of analyzing increasing amounts of medical data in a fast changing environment, we developed the eTRIKS Analytical Environment (eAE), a scalable and modular framework for the efficient management and analysis of large scale medical data, in particular the massive amounts of data produced by high throughput technologies.

We designed the eTRIKS Analytical Environment as a modular and efficient framework allowing us to add new components (public or private) or quickly replace ageing ones by more efficient or proprietary ones. The eAE relies on mature open source technologies with strong supporting communities, such as tranSMART, OpenStack, Jupyter [KRKP⁺16] and Apache Spark to provide efficiency and scalability.

The development of the eTRIKS Analytical Environment has the goals of enabling the scalable exploration of multi-modal medical data using a flexible and modular architecture. In the following, we discuss its architecture, the components we used and finally also how the eTRIKS Analytical Environment fits into the eTRIKS environment.

3.3.2 General Environment

We designed the eAE with four layers - *Endpoints Layer*, *Storage Layer*, *Management Layer*, and *Computation Layer*. Those layers aim to provide users with an analytics environment which (a) has a frontend/endpoints which are user-friendly, extensible as well as easily integrated into tools, (b) is modular and finally (c) is also scalable. We accomplished this by designing a loosely coupled multi-layer architecture of components to provide as much flexibility as needed. The modularity of this framework enables adding new components (public or private) or replacing ageing ones with better performing or proprietary ones.

The operating system used on both the physical and virtual machines as well as containers within this architecture is Ubuntu 16.04 LTS. This version provides the required stability throughout the life of the machines and necessary support for a large spectrum of libraries and drivers. Other Linux distributions such as Centos or Debian can also be used. With the release of version 18.04 LTS, and once the stability of this version is ensured, it is possible to upgrade to that version.

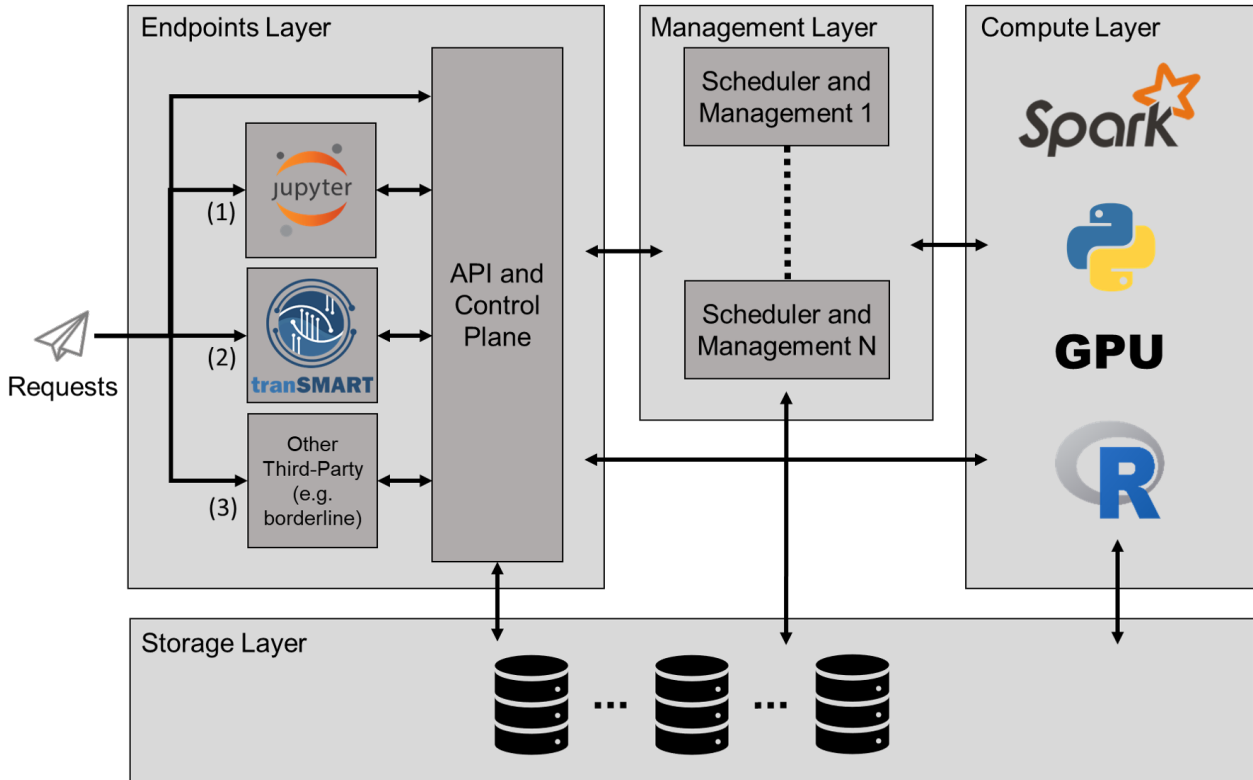


Figure 3.1: A schematic representation of the architecture of the eTRIKS Analytical Environment.

Figure 3.1 illustrates the architecture of the eTRIKS Analytical Environment. Each service is deployed in a Docker container and the services communicate with each other asynchronously through REST APIs. This architecture supports the possibility to deploy services multiple times across different host machines for scalability and resilience purposes. The platform is hosted behind a firewall and only the *Interface service* in the *Endpoints Layer* is exposed to the internet while all other services are interrelated only via an internal virtual private network.

At the top, interacting with users, is the *Endpoints Layer* which essentially hosts the containers which either provide the User Interface (UI) to users or the interface (REST API) to integrate it into third-party external tools or interact directly with the platform. The *Endpoints Layer* also contains the infrastructure to run smaller computations locally, user authentication, caching and auditing services. Interacting with the *Endpoints Layer* is the *Storage Layer* which stores analytics results and enables caches to be implemented for specific endpoints (see tranSMART example in Chapter 4) in order to avoid recomputation of frequent analysis, thereby making analyses more efficient. All the large scale data and meta-data associated with the platform (e.g. user details) are saved into the database via the *Storage Layer*. The *Storage Layer* supports all the other layers by providing replicated, distributed and scalable storage resources.

The *Management Layer* schedules the computation of the analyses on the compute nodes based on the availability of the nodes and type of analyses requested by the platform users. The *Computation Layer* is responsible for the execution of the scheduled computations on the scale-out infrastructure which can indifferently be a cluster, a cloud or any other specialised hardware. It also enforces the implemented privacy measures on each analysis. The final result is stored in the *Storage Layer* along with the analysis parameters and other details.

3.3.3 Endpoints Layer

The *Endpoints Layer* is the topmost layer and provides all the user entry points to the environment. The first aim of those endpoints is to cover the needs of a large spectrum of researchers ranging from biologists with limited computing and technological knowledge to the computational biologists who worry which parameter optimisation will yield the best results. In order

to achieve that goal, the eTRIKS Analytical Environment relies on three sets of tools: tranSMART, Borderline [OGA⁺18] and Jupyter. On the one hand, tranSMART focuses more on the hypothesis generation through a rich UI to explore the curated datasets stored, plot the associated statistics and a set of available workflows with their associated custom-made visualizations. Jupyter and Borderline, on the other hand, offer a larger set of possibilities as the user can write their own custom scripts and visualizations to harness the power of available libraries such as Matplotlib [Hun07] or Lightning [Lig16]. Furthermore, Jupyter and Borderline enable the researchers to manage the data provenance and lineage of their data sets and results thus facilitating the reproducibility of the experiments and data governance. The *Endpoints Layer* provides as well the only public interface to run analyses over user provided datasets directly on the platform. That interface enables the extensibility of the eAE to any third-party application and ensures that only verified and valid requests are processed, that all the requests are logged, and that the system remains responsive at all times. At the very minimum, the layer is composed of Interface, Authentication and Logging services.

The *Interface service* provides the client APIs for queries and user management. All the requests to the eAE are made over HTTPS and are first verified by the *Authentication service*. The authentication method has evolved between different versions and will be detailed in Chapter 4. Upon successful authentication, the *Interface service* validates the request and the job request is created in the database for the *Management Layer* to schedule.

The *Logging service* logs the queries made to the platform. The Logging records all queries both valid and invalid ones for audit purposes and traceability. The invalid queries must be easily accessible to administrators for periodic analyses to detect trends of any possible attack on the system. Thus, the valid queries are stored in an append-only text file, making it harder to tamper with without gaining physical access to the system, while the invalid queries are stored privately in the database.

3.3.4 Storage Layer

The *Storage Layer* manages the medical data and platform specific data. The first aim of the *Storage Layer* is to provide scalable, replicated and sharded storage capabilities to the platform. The scalability and sharding of the storage are fundamental to empower large scale analyses to run efficiently, while the replication guarantees the resilience of the platform against adverse events (e.g. a node failing, network disruptions, etc.). For the storage, we rely on three different tools: NoSQL database MongoDB 3.6.0 in the eAE backend, OpenStack Swift [Opec] for large scale object storage in the eAE backend and SQL database Postgresql [Pos14] in tranSMART and Timsecal [Tim]. The variety of database models supported gives flexibility to the architecture and a solid foundation to support new services.

MongoDB, with its absence of schema, is a perfect contender for adapting to any kind of data and acting as a cache. The platform-specific data consists of the user data (e.g. username, creation date, etc.), analyses available on the platform, query parameters, answers to previously requested queries, status of each micro-service including the available compute nodes, logged invalid requests, failed computations (output and error logs), and status of the ongoing computations. Unlike medical data, metadata does not have a fixed structure and a significant amount of meta-data is generated by each service periodically and for each query throughout its life cycle. Another substantial advantage is MongoDB's native support for high throughput read operations, scaling and resilience through sharding. That feature is essential for the correctness of our *Scheduling service* as it relies on the consistency of the fetched meta-data. MongoDB has a very powerful query language which enables selecting, filtering and projecting in a similar fashion as a SQL database would. However, for the storage and management of single very large files (terabytes and above), which do not require a database engine, MongoDB performances become a lot less attractive. Indeed, MongoDB stores objects in a binary format called BSON. The "binary JSON" or "BinData" datatype is used to represent arrays of bytes. However, MongoDB objects are generally limited to 16MB in size. In order to overcome that limitation, files are "chunked" into multiple objects that are less than 16MB each. This has the advantage of letting us efficiently retrieve a specific range of the given file, but it becomes

extremely slow when retrieving whole files of very large size.

OpenStack Swift is a high availability, distributed, eventually consistent object/blob store. Swift uses an eventual consistency model to replicate data across the different nodes, in contrast to the strongly consistent model that block storage uses for applications with real-time data requirements and databases. Eventually consistent object systems are intended to provide high availability and high scalability. They write data synchronously to multiple locations for durability, but when some nodes become unavailable due to a hardware failure, the replication is delayed. OpenStack Swift proxy servers ensure access to the most recent copy of the data, even if some parts of the cluster are inaccessible. Thus, Swift enables to store a very large amount of data efficiently, safely, and cheaply.

The open source version of tranSMART relies on PostgreSQL for historical reasons (it was migrated from Oracle). PostgreSQL suffers a number of limitations when it comes to high availability and scaling. Solutions exist to tackle these limitations. PostgreSQL’s default configuration is a very solid configuration aimed at everyone’s best guess as to how an “average” database on standard hardware should be set-up. By optimizing the configurations and distributing the queries to a replicated cluster of PostgreSQL instances using pgpool [PPS15] or pgclusterII for instance, we can improve the scalability and performance. However, most of those optimisations or modules are difficult to install, manage and are not supported natively by PostgreSQL. This is why the use of MongoDB in place of PostgreSQL as database for part of the data comes as a major speed-up for the analysis by exporting the required data much faster. The emerging NewSQL technologies (e.g. CockroachDB, TimeScale, MyRocks) could be an interesting addition to be a core part of the *Storage Layer* for high availability and scalability reasons as Chapter 5 will illustrate.

3.3.5 Management Layer

The *Management Layer* is the cornerstone of the eAE platform. It is responsible for monitoring jobs and compute nodes and scheduling the job requests for computation. It is, therefore, crucial for the *Management layer* to have downtime as close as possible to zero. *Management Layer*

also needs to be extendable out-of-the-box, i.e., it must be able to use the new compute nodes as soon as they are up and available. For the *Management Layer* to achieve as close as possible zero downtime, we avoid architectures prone to single points of failure. The *Scheduling and Management service* guarantees that the system performs efficiently by periodically purging the unresponsive compute nodes to ensure that the queries are scheduled only on working nodes. To avoid getting bottle-necked with unresponsive jobs or nodes, it decommissions unresponsive compute nodes and purges failed or stuck jobs. Decommissioning of nodes – locking the node’s status to *DEAD* – is done by checking that the last status update by the service exceeds the set threshold (1 hour by default). For purging unresponsive jobs, the jobs are fetched from the database and checked for their status and time of creation. They are purged (killed) if they have failed more than twice and are removed if they were created before a certain set time.

The *Management Layer* is composed of multiple *Scheduling and Management services* running independently in parallel, each service periodically fetches the jobs and compute nodes from the database for scheduling. This can be achieved through different means in terms of implementation and a proposed solution will be presented in Chapter 3. Nonetheless, a key novelty we introduce as part of the *Management Layer* is the introduction of a completely new scheduler designed to provide concurrent multi-master capabilities.

The core ideas behind the introduction of this novel scheduler are to support capabilities that would otherwise not be supported by third-parties schedulers. Increasing scale and the need for swift responses to always evolving requirements are challenging to meet with current monolithic cluster scheduler architectures. Monolithic architectures limit the rate at which new features can be deployed, increase maintenance downtime, decreases efficiency and utilisation, and will eventually limit cluster growth.

Utilisation and efficiency can be increased by running a mix of workloads on the same machines: CPU- and memory-intensive jobs, jobs requiring specialised hardware (TPU, GPU, etc.) available only on specific machines, small and large ones, and a mix of batch and low-latency jobs. This consolidation reduces the amount of hardware required for a workload, but it makes the scheduling more complex as a wider range of requirements and policies have to be taken into

account. The design of a multi-master scheduler with shared-states enables the concurrent scheduling of jobs across different machines. This shared-state mechanism has proven successful at delivering performances in the design of Google’s Omega scheduler [SKAEMW13] even in the eventuality where a single scheduling task may require up to several seconds to compute. We went further by taking a gang scheduling approach to ensure that the scheduling is never the scalability bottleneck. It allows for jobs to be resumed in the eventuality of a scheduler becoming unavailable and provides the guarantees necessary for downtime as close as possible to zero. The multi-master design allows for the deployment of improved versions of the scheduler side by side with the old ones without any downtime or hindering the currently running scheduling. One final feature that a multi-master design enables is the self-invalidation of a member, acting as a watchdog against itself, in the event that it fails repeatedly at scheduling or managing the jobs.

The introduction of cluster computing systems, such as Spark, as a supported analysis environment introduces a new set of challenges that we aim at addressing in our design. Those cluster computational capabilities themselves rely on clusters of machines to operate, thus creating a new layer of complexity for scheduling and resource management. In order to maximise the usage of the machines, we developed a reservation mechanism executed before any Spark computation, which allows Spark clusters to be used for other computation types (such as Python or R) while idle. The scheduler locks all the compute nodes of the cluster by setting their status to *RESERVED* and, once all the nodes are reserved, the job can be submitted for execution to the compute node hosting the Spark master. This is mandatory to avoid collisions between a Spark job running on workers and other jobs. However, in order to avoid any starvation of the Spark clusters by long-running R/Python jobs, the scheduling priority of Spark jobs is set higher on those nodes. Other types of jobs get scheduled on the Spark clusters in an overflow policy fashion when the number of waiting jobs and their pending duration threshold is reached.

Finally, the support for provenance is another important aspect we wanted to address in our design. In addition to supporting the sharing of the jobs states, the persistence of the jobs in the *Storage Layer* enables the traceability of the jobs with an extensive amount of metadata history being stored with every job. This metadata includes if the job failed to finish properly,

on which executors, the associated error logs (if any), the executor and language/packages versions used for that execution, the user who submitted the job, the start and end dates as well as the version of the data amongst others. The provenance can be seen as an audit trail which is of critical important in the context of security. Periodic analyses of the provenance data can be carried out to detect any potentially suspicious request or sequence of requests indicative of possible attacks on the system. The provenance data associated with every job is also vital at ensuring the efficient running of the clusters as they provide insights into the health of all the services across all layers. The continuous processing of that data allows the *Scheduling and Management services* to oversee the invalidation and notification to the system administrators of failing services following a given set of policies.

3.3.6 Computation Layer

The *Computation Layer* provides execution capabilities to the eTRIKS Analytical Environment. While scheduling jobs is handled by the *Management Layer*, running them, generating results, and ensuring the integrity of the results is the role of the *Computation Layer*. The first aim of the *Computation Layer* is to support efficiently a broad scope of analytical capabilities ranging from simple statistics to compute heavy deep learning models. That scope also includes supporting heterogeneous hardware (CPU, GPU or ASIC), which can be used to further enhance the computing speed. The layer's modularity gives the mean to support the addition of new compute resources on the fly, without downtime or loss of scalability, and the addition of new types of analytical capabilities. The *Computation Layer* has also a combination of security and privacy features which enables us to run privacy-preserving computations as described in Chapter 5. Nevertheless, the support of sandboxing is a key requirement when designing a platform for privacy-preserving analytics. Sandboxing ensures the integrity and isolation of the computations against unauthorised accesses from rogue analysts scanning the memory or disk from collocated containers on the host machines. This prerequisite is the fundamental reason that led us to write a new *Compute Service* as part of the *Computation Layer* as no suitable open-source third-party offered that possibility within the scope of analytical capabilities required.

In order to have the maximum compute efficiency (e.g. compute resources usage) on the platform, it is capital for the environment to support on-demand resource allocation (to support scaling out when more compute-heavy or multiple computations need to be executed). The first approach to solve that issue was a pure cloud computing one using Openstack [SAE12] Mitaka. Indeed, Openstack offers two particularly interesting components to address this on-demand requirement: the Heat project [Opea] and the Glance project [Opeb]. Both enable the creation of predefined templates or disk images to launch multiple composite cloud applications. Setting up the development environment, therefore, is seamless for the user and enforcing a version control of the software and libraries ensures the integrity of the environment. The eTRIKS Analytical Environment uses a private instance of OpenStack and servers in house, however, there is no technical limitation to deploy it in public clouds such as AWS or Azure.

However, to further improve the architecture, we have replaced Heat and Glance by Docker containers and Kubernetes which contain separate services to be assembled to tailor a custom environment for the user and vastly improves performance for all the services. For instance, one user might require a specific set of languages (e.g. Python and R), while another user might need Spark support or GPU resources. Besides, we could support different versions of the software at the same time but hosted in different containers. The Spark computation clusters, HDFS, Swift and MongoDB are all installed in bare metal environments with RAID-10 storage to obtain the best performances possible and fault-tolerance. For security purposes, all those instances are password protected and running in HTTPS mode to ensure secure communication over the network between the different instances while MongoDB connections are encrypted using TLS/SSL. The eTRIKS Analytical Environment currently uses Cloudera CDH 5.11.0 for the deployment of the Hadoop [DG08a] stack (including Spark [ZCF⁺10]). We also considered MapR and Horton Works deployment tools. Each one presents its own set of advantages and drawbacks. The reason for choosing Cloudera's over the others is because it is arguably the best in terms of management interface and the availability of supported software in their stack. If the user's requirements differ from those set here, however, e.g., if the user prefers to use Amazon's AWS or own in-house components, some components would have to be adapted to adjust the interfaces of the eTRIKS Analytical Environment.

3.3.7 Interaction between Layers

Figure 3.1 illustrates the architecture of the eTRIKS Analytical Environment.

(1) Each user owns a Virtual Machine (VM) or a docker container containing a version of the Jupyter server, a set of kernels (R, Python [RD10], Spark, etc.) and a minimal set of standard libraries (Numpy, Scipy, Scikit-learn [PVe11], Bioconductor [GCe04], etc.) supporting them. This instance is one of the points of access to the eTRIKS Analytical Environment. The users can upload their data sets to the server and write their own scripts for analysis. Jupyter, through the selected kernel, sends the requested computations to the local engines which in turn sends it back to Jupyter. If the user requires more compute power, they can remotely submit their script to be scheduled on a larger centralised cluster to the *Interface service*. When the required resources become available, the scheduler triggers the computation. The Spark clusters are Hadoop stack production clusters installed on bare metal servers for performance reasons. Each one runs CDH 5.11.0 with the full Hadoop stack. The GPU clusters rely on TensorFlow 1.0 for Deep Learning and Nvidia CUDA 9¹ otherwise. The R servers rely on Microsoft R [R C14] Open, formerly known as Revolution R Open (RRO), which is the enhanced distribution of R from Microsoft Corporation. The results are sent back to Jupyter or MongoDB (depending on the user's choice). The user can explore the results using advanced visualizations (lightning, etc.).

(2) The second native entry point to the eAE is through a tranSMART's plugin specifically developed for this integration. The plugin manages and interfaces with the MongoDB cache. The plugin can submit a job to the *Interface service* using data stored either in MongoDB or in tranSMART. The results are sent to the MongoDB cache. The user can explore the results in tranSMART and compare them with previously run computations held in their personal cache history.

(3) The third native entry point to the eAE is through Borderline. Borderline is a user-facing set of services responsible for locating data, querying it across multiple heterogeneous sources, tracking its provenance as it travels through the platform and allowing users to maintain com-

¹<https://developer.nvidia.com/cuda-zone>

plete control over the process. It is the glue that puts together the eTRIKS Analytical Engine (eAE), the eTRIKS Data Platform (eDP) (including tranSMART and eHS) and capitalises on efforts conducted over the past four years. Besides allowing seamless data flow and tracking between these components, it provides the user with an enriched interface capable of handling complex scenarios. It also provides a dynamic data query editor for selection of patient subgroups from the entire corpus of data accessible by a given user (checking access according to a permission engine), as well as a code editor offering analysis code templates and the possibility of better control and customization over what the computational platform will execute.

3.3.8 Security of the architecture

The platform has been designed with security and privacy in mind. The eAE manages risks using a combination of server-side security, authentication, audit and network security. However, those protections are only the core layer upon which adopters can build upon and further extend it to meet the most stringent requirements. The extensibility and modularity of the security in the architecture will be further demonstrated in Chapter 5.

Server-side security: Many attacks on privacy and services employ a relatively large number of queries to circumvent protections (e.g. DDOS attacks, data leakage, etc.). To thwart brute-force attacks on the client API, we developed a query rate limitation mechanism. This ensures that any analyst can only submit a limited amount of queries in a certain time period, defined by the curator (e.g. 100 queries in 7 days). As detailed in Chapter 5, the architecture supports the secure execution of algorithms in sandboxed environments. This execution isolation comes at a cost in terms of performances, but prevents rogue algorithms to snoop illegally on other computations which might be running at the same time on the platform. The sandboxing relies on AppArmor (“Application Armor”) which is a Linux kernel security module. The module supplements the traditional Unix discretionary access control (DAC) model by providing mandatory access control (MAC).

Authentication: Access is provided only for authenticated users with the right authorization levels. Three levels of users are supported: super admin, admin, and standard users. Admin

can create, delete and check users through the API as well as monitor the status of the services. Super admins have the additional right to create new admins. In addition to those core levels, the users are given additional rights levels for data and analysis control. For example, in the context of the population density algorithm (see Chapter 5.2.3), different users will be authorised different levels of regional access: one user might be authorised to access at commune level while another only at regional level. The granularity can be as well temporal in the context of the mobility where the access time frame can be larger for some users compared to others. Further restrictions can also be implemented such as maximum sampling size for a given analysis.

Audit: Auditing is an important part of the security of the platform as it enables system administrators, governance board members for ethical oversight and data owners to review all previous queries and detect tentative of attacks by logging illegal requests. The auditing helps preserving the health of the clusters as well by providing computation times of the queries and clusters loads to the administrators. Those indicators can help them identify nodes that might be throttling or clusters which are over/underutilised. The administrators could then act on them by commissioning/decommissioning nodes and thus provide the best experience to users.

Network security: To prevent attacks where people intercept HTTP packets, all communications with the API and between the different services is exclusively done in HTTPS. Any non-HTTPS requests is discarded and logged for auditing purposes. Furthermore, the connections of the services to MongoDB are encrypted using TLS/SSL. In order to shield the platform from external brute force attacks, the layers are deployed into two different VLANs. This siloing enables to expose only the *Interface service* of the *Endpoints Layer* to client's applications while the data and compute services would be safely hidden from the rest of the network.

Chapter 4

Implementation of the eTRIKS Analytical Environment

In this chapter, we will present the base implementation of the architecture. Then, we will present as well the work carried out to facilitate the training and development of deep learning models and how they were later integrated into the production deployment of the eTRIKS Analytical Environment.

4.1 Implementation

4.1.1 General Environment

All the microservices of the platform follow ECMAScript 8 and use ExpressJS and NodeJS [TV10] as runtime environment. Agile development methodology (e.g., automated testing and building of the Docker containers) was employed for the development of the platform. The use of Docker containers has enabled to create endpoints containers tailored to the needs of the users to use the hardware in an optimal fashion. One user might, for example, require a specific set of languages (e.g., Python and R), while another user may need Spark support or GPU resources. Besides, we can support different versions of the software at the same time but hosted in differ-

ent containers. That custom environment gives more flexibility to the user and vastly improves overall performances. To ensure that the platform up-time is close to 100%, Docker Compose (in auto-restart mode) is used for the deployment of the services. This allows for the deployment of the services multiple times, across different host machines, for scalability and resilience purposes. The only entry point to the platform, for submitting requests, is through the public APIs exposed by the *Interface service* in the *Endpoints Layer*. The platform is hosted behind a firewall and only the *Interface service* is exposed to the Internet. This limited exposure to the outside limits the surface of attacks by malicious entities. All the services communicate with each other asynchronously through the REST APIs via an internal virtual private network.

All services must self-register on startup with MongoDB which is acting as the service registry with the cooperation of the *Management service*. A service instance is in charge of registering itself with the service registry. On startup, the service instance registers itself (IP address and host) with the service registry and becomes available for discovery. The client must send a heartbeat periodically to renew its registration so that the registry knows it is still alive. If the last heartbeat exceeds a defined threshold, the service is unregistered. This design pattern allows for flexible deployment of the platform, easily add or remove services and can run different versions concurrently without any downtime (although that would not be a recommended best practice).

4.1.2 Endpoints layer

The eAE uses a REST API that enables to:

- Remotely submit Spark, Python, GPU as well as R jobs
- Integrate the analytics environment with the examples of tranSMART, Borderline and Jupyter
- Monitor the health of the clusters, audit and jobs history

Borderline, like eAE, is written in JavaScript with Node.js. The choice of a rather uniform Node.js-based stack enables a simplified maintenance of all lightweight microservices. Likewise, all communication between the components relies on a set of similarly designed HTTP API.

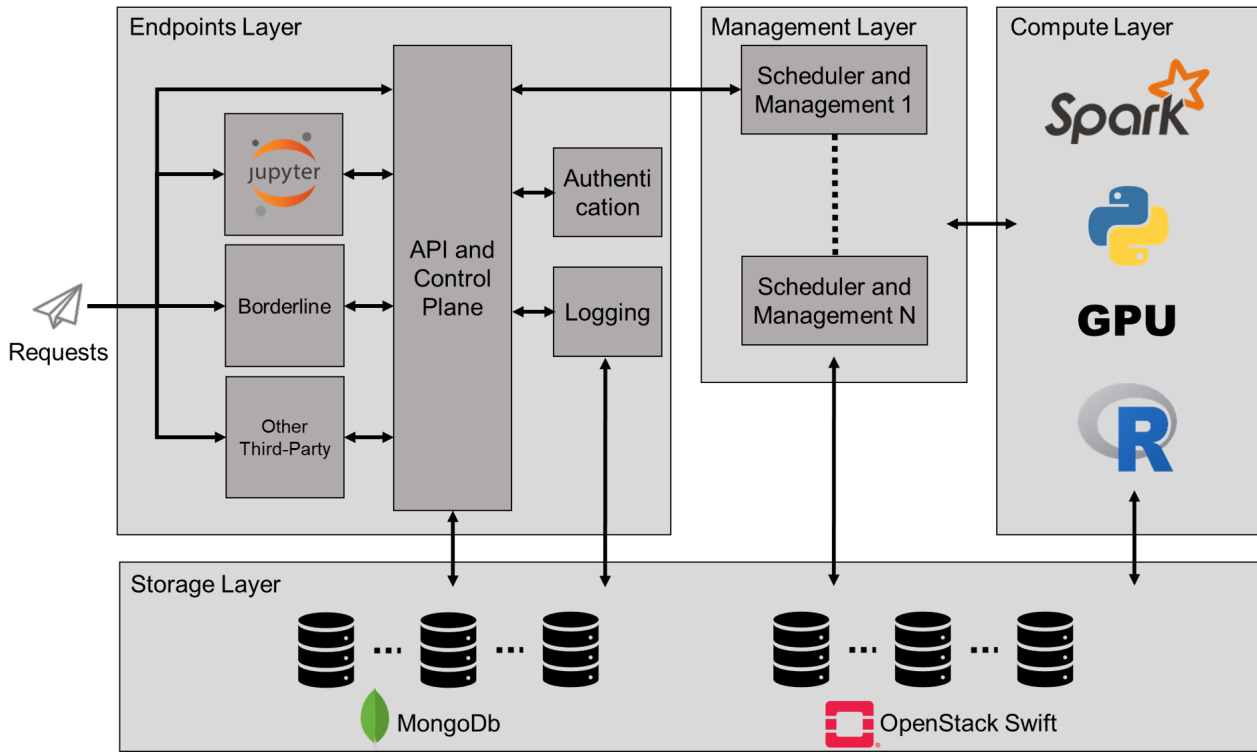
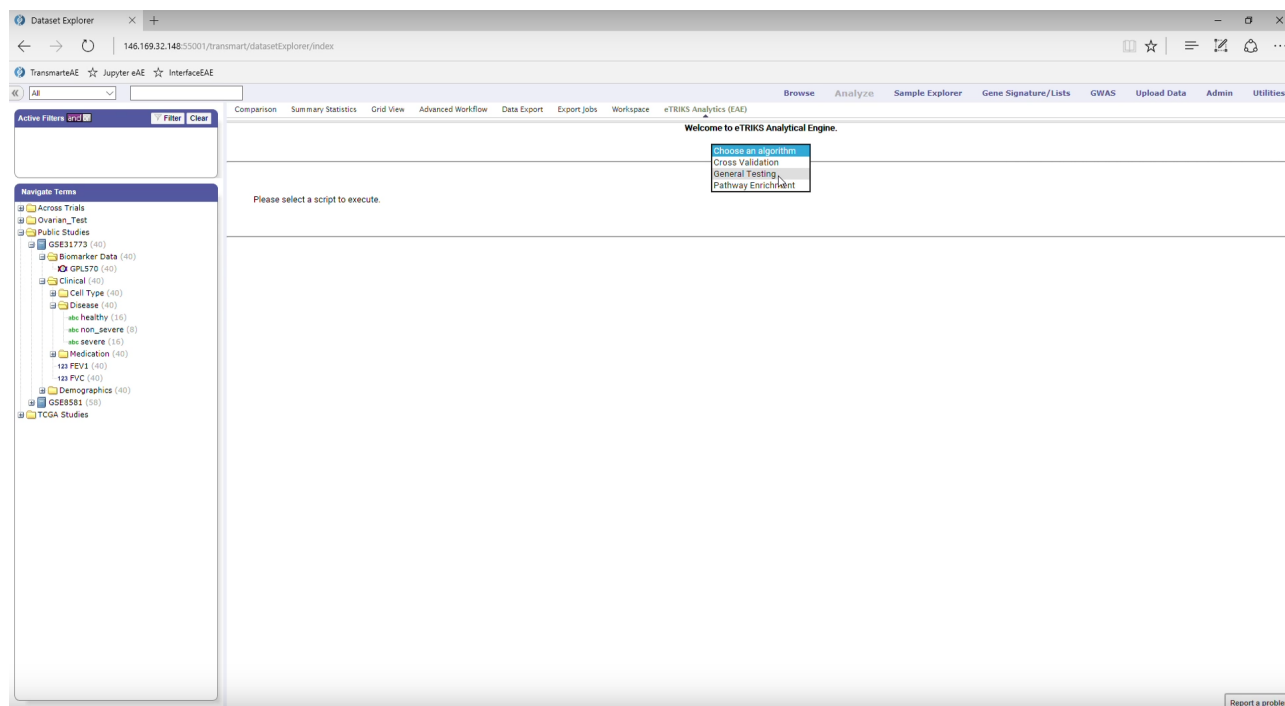


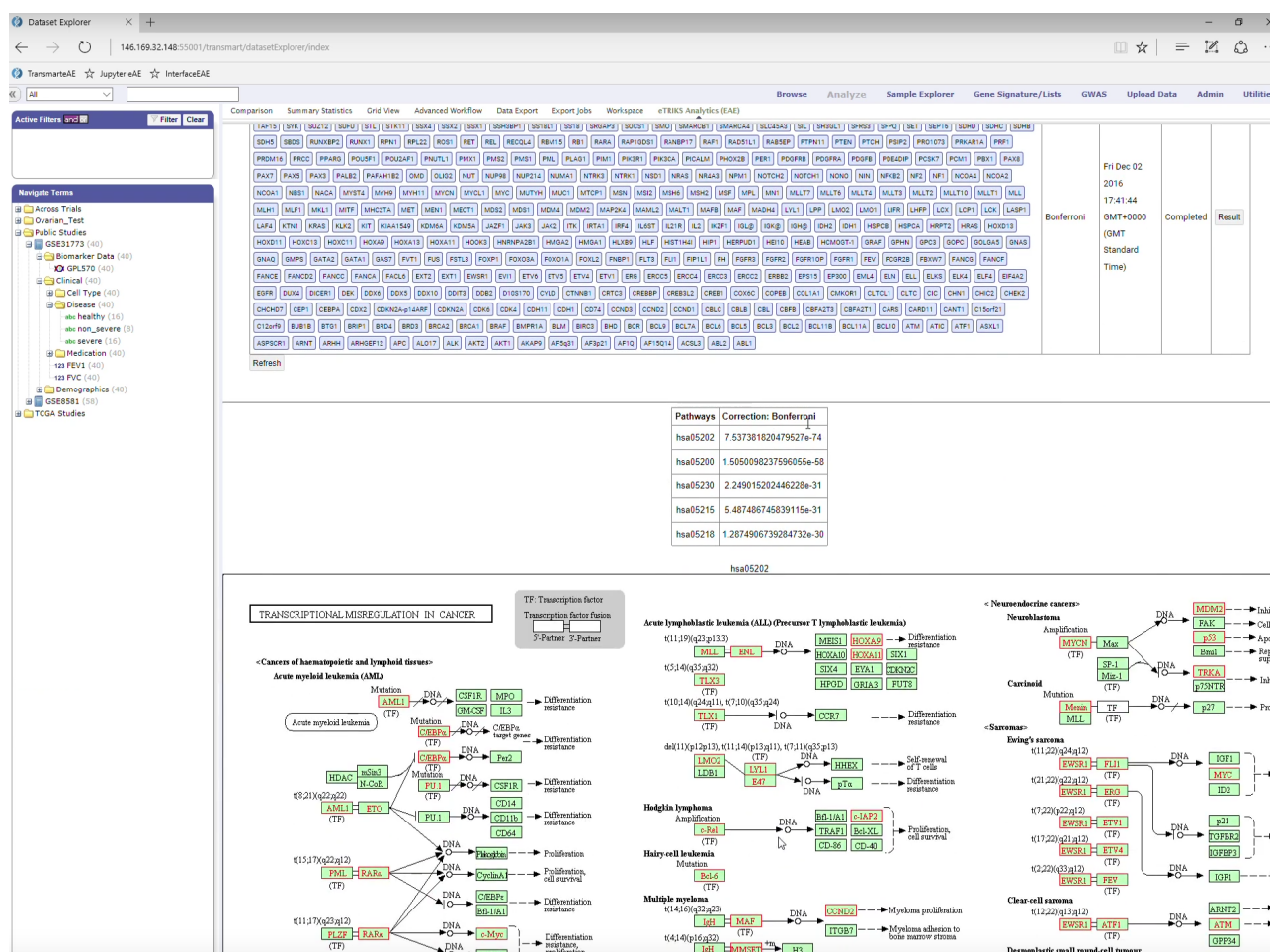
Figure 4.1: A schematic representation of the eTRIKS Analytical Environment implementation.

Plugin for tranSMART

As discussed before, the workflows originally developed in tranSMART were not intended for large scale computations. It is with the intention of closing this gap that a plugin for tranSMART 16.2 to interface with the eAE has been developed. The core features of this plugin are to transfer the data to the eAE, implement a cache mechanism for the users to track the status of the computation and access their results at any time. This mechanism has been implemented using the MongoDB layer. The cache itself enables individual users to browse through their results whenever they want to. Each cache history is visible only by the user. However, if one user wants to run the same analysis as another user, the cache will retrieve the results instead and add the computation results to the user's own cache history. This allows for better use of computing resources. The plugin is also responsible for the associated visualizations of the eAE workflows developed for tranSMART (workflows are detailed in Section 6.1). Those visualizations have been implemented using D3.js [BOH11]. In general, D3.js does not have any specific data format for visualization. Generic CSV tables can be used for all types of visualizations, and its core provides data loaders for this format fitting perfectly with MongoDB.



(a) Home page of the eAE plugin for tranSMART with the three algorithms developed.



(b) Example a pathway enrichment analysis run through the eAE. The table represents the top 5 most correlated pathways with the given list of genes and the image is the top pathway retrieved dynamically from KEGG.

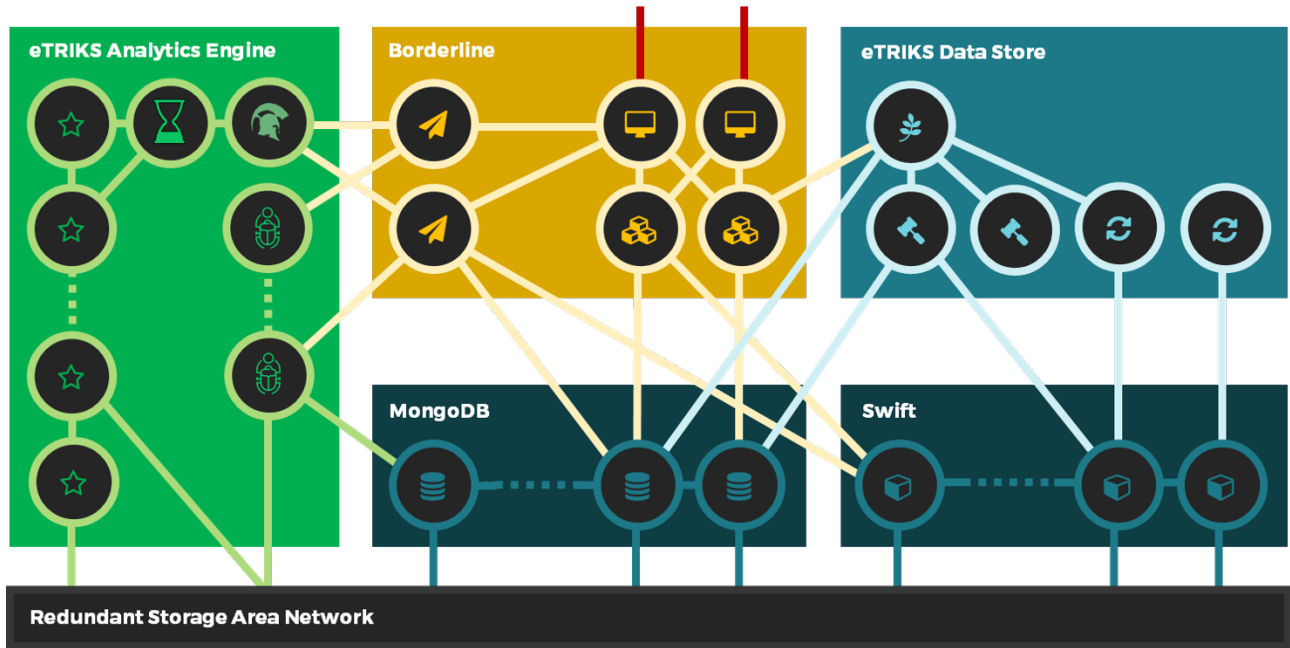


Figure 4.3: A schematic representation of the integration of Borderline with the eAE and the integration of tranSMART as part of the eDP. Users would access the platform through (Borderline web UI(s)) where they can select datasets via (Borderline data-source middleware(s)). Data is retrieved from a target such as (eDP External API) which in turns relies on (eDP File Parser(s)) and (eDP Query Executor(s)) to compile the selection. Once extracted, the data is pushed to (Swift Object Store cluster). In addition to selecting data, users might use (Borderline UI) to write custom analysis code and workflows. These are bundled with the data and made available to (Borderline Cloud Connector(s)). From there, it is sent to (eAE External API) via (eAE File Carrier(s)). It is then dispatched by (eAE Scheduler(s)) on the appropriate (eAE Compute Node(s)). Once the computation has finished, results come back to (Borderline Cloud Connector(s)) and are pushed to (Swift Object Store cluster) to be accessible to the users. Operational items such as service health, sessions and routes are stored in (MongoDB cluster).

Borderline

Borderline [OGA⁺18] is a user-facing set of services responsible for locating data, querying it across multiple heterogeneous sources, tracking its provenance as it travels through the platform and allowing users to maintain complete control over the process. It is the glue that puts together the eTRIKS Analytical Environment (eAE), the eTRIKS Data Platform (eDP) and capitalise on efforts conducted over the past five years.

The eTRIKS Data Platform (eDP) is a heterogeneous environment within which multiple data warehousing software can be deployed. In our lab we make use of the eTRIKS Harmonization

Service¹ (eHS), a home grown and CDISC [CDI04]/ISA [SRSe12] enabled software addressing the needs for data pipeline management between data collection and data analysis in the translational medicine; as well as the tranSMART Platform.

This alternative from connecting tranSMART directly to the eAE and instead rely on Borderline to interact with it presents two main benefits. The first benefit being that starting with version 17.1, tranSMART underwent a major overhaul resulting in a vastly improved API, improved external data query times and a complete decommission of the former UI. Thus, rewriting the plugin for that new version would have been moot as most users wouldn't have been able to use it from the UI. Secondly, Borderline offers the users a far richer experience, so we bring more value to the users by leveraging tranSMART through Borderline. Indeed, besides allowing seamless data flow and tracking between these components, it provides the user with an interface capable of handling complex scenarios and a data provenance feature with the workflow management. Borderline leverages the eAE's own provenance data to offer the possibility of better control and customization over what the computational platform will execute and facilitate reproducibility.

Because of the shared common technological stack and design principals, we have been able to develop a deeper integration between Borderline and eAE. That integration has enabled to gain a major data transfer speed up between applications and enhanced the interactivity of the user interface by bringing the compute capabilities closer to the user.

Jupyter

In order to achieve the best user experience, we discussed with other researchers to define how to best support their needs. We first explored the implementation of a Jupyter plugin to connect to the platform. However, this solution would not fully achieve the desired user experience for the power users (our primary users) and put more constraints than necessary on the architecture. For those reasons, we took a more programmatic approach and moved away from the UI.

¹<https://ehs.dsi.ic.ac.uk/>

The integration of Jupyter relies on a Python PIP package called eae². PIP is a package management system used to manage software packages written in Python. PIP manages full lists of packages and corresponding version numbers, this permits the efficient re-creation of an entire group of packages in a separate environment (e.g. another computer) or virtual environment. That embedded versioning allows different PIP versions to support different versions of the eAE API throughout its life cycles. Hence, it gives stability to the users while retaining possibilities for a safe evolution for the developers. In addition to facilitating the interaction between Jupyter and the eAE, that approach is agnostic to the environment considered. PIP packages can be installed in any Python environment, thus users can very well send jobs from their own machines or other hosting environments (e.g. Zeplin³) regardless where they are and are not constrained to Jupyter. That flexibly, while being less user-friendly, is far more powerful for the power users, lower maintenance and more future-proof towards supporting new endpoints.

The package enables the users to interact with the eAE back-end seamlessly by handling all the API calls and the data transfer in the background. As illustrated by Figure 4.4, that approach allows the users to easily define a large number of tasks to be submitted to the back end in a programmatic and stable fashion (e.g. avoid the tasks to be submitted twice by accident).

4.1.3 Storage Layer

The Storage Layer manages the medical data stored in tranSMART and platform specific data. We used MongoDB 3.4 to store the platform data, cache data, support TensorDB (see Section 4.3) and support new data types. However, the cluster was deployed in Docker containers in sharded mode where each shard is an independent three-member replica set. That new deployment further enhances the performances of the MongoDB cluster while retaining a high level of flexibility as it is integrated into the platform in the same fashion as the other services making the deployment and maintenance processes transparent to the administrator.

In addition to storing an extended amount of platform specific data, the *Storage Layer* under-

²<https://pypi.org/project/eae/>

³<https://zeplin.io/>

```

1  # We import the eAE package
2  from eAE import eAE
3
4  # We create the connection to the backend
5  eae = eAE.eAE("example", "password", "interface.eae.co.uk")
6
7  # We list the jobs with their associated parameters
8  parameters = [ "first_analysis_type 0 1",
9                  "first_analysis_type 1 2",
10                 "second_analysis_type 0.3 delta" ]
11
12 # We list the required files for the analysis to be sent to the back-end
13 data_files = ["job.py", "faust.txt"]
14
15 # We submit a job
16 answer = eae.submit_jobs("python2", "job.py", parameters, data_files)
17
18 # We check that the submission has been successful
19 print(answer[0])
20
21 """
22     answer = { "status": "OK",
23                "jobID": "5b080d28e9b47700118f0c99",
24                "jobPosition": 1,
25                "carriers": [
26                    "carrier:3000"]
27            }
28 """
29
30 # We download the results
31 result = eae.get_job_result('', answer[0]['jobID'])
32
33 # We have a look at the computed result
34 """
35 Hello World !
36 first_analysis_type
37 The Project Gutenberg EBook of Faust, by Johann Wolfgang Von Goethe
38
39 This eBook is for the use of anyone anywhere at no cost and with
40 almost no restrictions whatsoever. You may copy it, give it away or
41 re-use it under the terms of the Project Gutenberg License included
42 with this eBook or online at www.gutenberg.net
43 """

```

Figure 4.4: Illustration of a simple submission of three jobs using the python eae package.

went some conceptual changes with the addition of OpenStack Swift. Swift implements object storage with interfaces for object-, block- and file-level storage. Swift allows us to assume a highly available distributed and more efficient storage to serve the data between the different services and compute nodes. Objects and files are written to multiple disks spread across multiple servers in the data center, with the OpenStack software responsible for seeing to integrity and data replication across the cluster. Storage clusters scale horizontally straightforwardly by adding new servers. Should hard drive or a server fail, Swift replicates its content from other active nodes to new locations in the cluster. Thus, the *Storage Layer* has its own large scale storage capabilities to store any medical data before being processed by the *Compute Layer*.

4.1.4 Management layer

For the *Management Layer* to achieve close to zero downtime, we avoid architectures prone to single points of failure. Unlike master-slave architecture pursued by schedulers such as IBM's Load Sharing Facility (LSF) [IBMa] or OpenLava [JB16], eAE offers a completely new scheduler designed to provide a new design based on concurrent multi-master capabilities.

The *Management Layer* is composed of multiple *Scheduling and Management services* running independently in parallel, and each service periodically fetches the jobs and compute nodes from the database for scheduling. As the schedulers run independently, a soft lock mechanism has been developed to manage the concurrent access to the jobs' and the compute nodes' records in the database. This prevents inconsistency like the same job getting scheduled multiple times or a single node receiving multiple jobs at the same time. The soft locks are set in the database by the *Scheduler* fetching the records. The *Scheduler* periodically fetches the unlocked *IDLE* compute nodes and *QUEUED* jobs and, for each fetched job J , it checks if a compute node is available to execute it. On successful check, it tries to set the soft lock on the record and the selected compute node. If successful, a job request is sent to execute the job. The locks are removed once the request is acknowledged by the compute node. This internal concurrency management and the loosely coupled design are the enablers for the multi-master scheduling capabilities of the *Management Layer* and the subsequent benefits described in Section 3.3.5.

The *Scheduling and Management services* guarantee that the system performs efficiently. To avoid getting bottle-necked with unresponsive jobs or nodes, it decommissions unresponsive compute nodes and purges failed or stuck jobs. Decommissioning of nodes – locking the status of the node to *DEAD* – is done by checking that the last status update by the service exceeds the set tolerance (1 hour by default). For purging unresponsive jobs, the jobs are fetched from the database and checked for their status and time of creation. They are purged (killed) if they have failed more than twice and are removed if they were created before a certain time (2 days). The scheduling of jobs happens every one second, purging once a day, and decommissioning of nodes every 1 minute.

4.1.5 Computation Layer

The *Computation Layer* has seen the development of a dedicated *Compute service*. The *Compute* service fetches the required data and analysis algorithm from the *Storage Layer*, executes the functions and finally stores back the results into the *Storage Layer* to be collected at any point in time by the requesting user. The *Compute service* is also responsible for logging the execution time, exit code, standard and error outputs (if any) into the associated job record for audit and development purposes.

Each *Compute service* is associated with a list of type of jobs it can execute, e.g., Python2, Python3, R, Spark. In order to maximise usage of the services, the same physical containers run different types of computations (e.g. R and Python or Python and GPU) and can be allocated to more than one cluster. However, the idea has been pushed a bit further as Spark clusters can now be used to run other computations such as Python or R on individual nodes while they are not in use. In order to achieve that, we created a reservation mechanism to prepare the cluster for the computation. If a job is running on one of the nodes of the cluster, the scheduler waits for the end of the computation and then reserves it. Once all nodes have been reserved, the Spark cluster can be used and the job scheduled. In pursuance of avoiding Spark jobs to be waiting for too long, we took an overflow approach and those nodes have lower scheduling priority than the other nodes dedicated to that type of jobs.

The *Compute* service has been designed as well in a fashion that facilitates the addition of new types of jobs (e.g. Fortran or C/C++) or features (e.g. sandboxing as demonstrated in Section 5.2). The typical time needed to develop and test a new job type is less than a day, which should encourage developers to easily adopt the platform.

4.2 Benchmarking and Scalability

In this section, we demonstrate improved resource usage, the scalability, and performances of the architecture towards supporting the several bioinformatics analysis pipelines.

4.2.1 Resource usage

To evaluate the improved resource usage, we monitored the average CPU load across a set of twenty physical machines before and after deploying the eAE. In both cases, we recorded every fifteen minutes the load average over the last fifteen minutes provided by the host system in `/proc/loadavg` for each individual machine. The recording of the loads for those experiments spanned across three months in both instances. The set of machines was heterogeneous (different machines have different numbers of cores) so in order to give a representative idea, we provide the usage percentage as a ratio between load and total capacity. For instance, a machine with 24 hyper-threaded cores with a load recorded of 12 gives a usage percentage of 50%.

While the Figure 4.5 clearly shows improved resource usage on average, nevertheless it is interesting to note that the improvement varied substantially between the type of computations in the context of this experiment. In both cases, we had fourteen machines dedicated to R/Python and six to a single Spark cluster. R/Python machines have seen their average loads increase, while the Spark cluster has seen a marginal decrease. However, this could be partially explained by the fact that there was only one Spark user, unlike the others which had multiple.

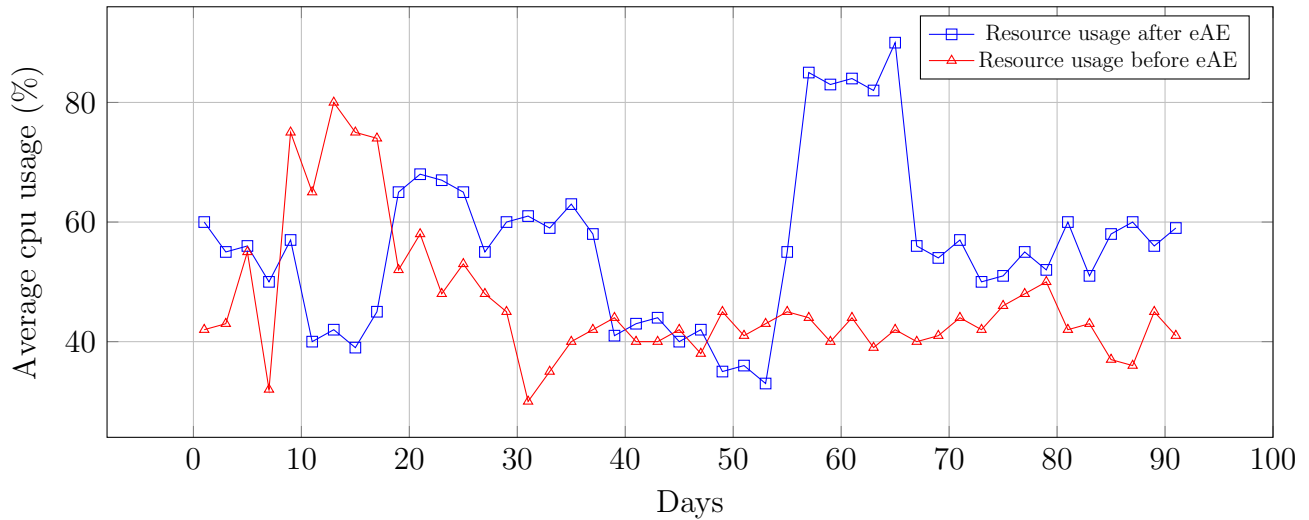


Figure 4.5: Evolution of the usage percentage aggregated per two days across all the machines during three months. We observe that, on average, the resource usage of the compute resources is significantly improved (21% on average).

4.2.2 Scheduler

We conducted several experiments to evaluate both the performance of the scheduler and the resilience of the *Management Layer* as we scale the system. For both experiments, we deployed a single instance of MongoDB with 1000 concurrent Compute services and submitted requests to schedule a sleep job of 1s. These jobs are scheduled, computed, and finally, the results are stored in MongoDB. The benchmarks were executed on a single machine with 256GB of disk (RAID 1 @ 7200RPM), 250GB of RAM (@ 1600 MHz), 48 cores (2x Intel Xeon, E5-2620 v2 @ 2.10GHz) and a network speed of 20Gb/s. The large number of Compute services ensures that the time measured truly evaluates the scheduler’s performance and not the bottleneck of the compute services.

To evaluate the scheduler’s maximal capacity to scale as the number of concurrent jobs increases, we submitted N batches ($N=1,10,50$ and 100 in this experiment) of 100 jobs to a single scheduler. A scenario of more than 10k is rather unlikely in a production environment as each job takes some time (in hours) to process. Figure 4.6 shows that submission and scheduling scales linearly with a large number of requests.

To evaluate the resilience of the *Management Layer* as we horizontally scale the system, we submitted a total of 10k jobs with a varying number of schedulers. This scenario evaluates, as well,

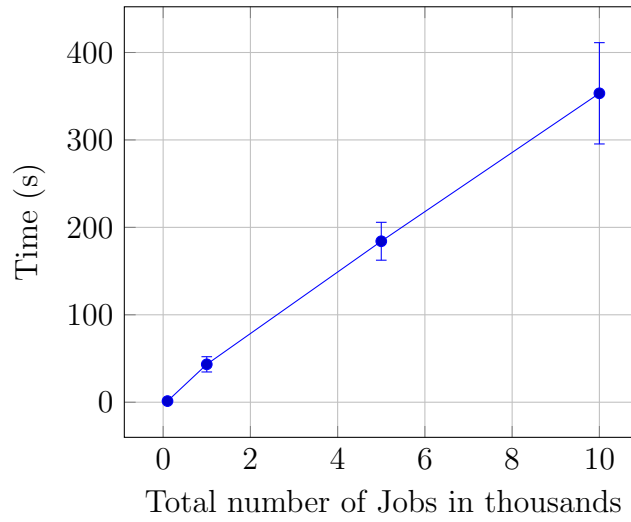


Figure 4.6: The performance of a single scheduler with respect to the submission size. Each point represents the average running time of 10 experiments along with the standard deviation.

the performances of the *Management Layer* in the eventuality one or more scheduler becomes unavailable. This simultaneous evaluation is possible thanks to its multi-master design where each scheduler operates independently from each other. Five independent clients (with batches of size 100) were used to insert jobs in MongoDB with M schedulers ($M=5, \dots, 1$) running in the background. Each scheduler is self-orchestrated in an asynchronous fashion, e.g. each scheduler is responsible for periodically (every 100ms in this experiment) retrieving a list of jobs to be scheduled and scheduling them on an available compute services. At any given time, the status of a job in MongoDB is the same for all schedulers which removes the need for synchronization between schedulers. Figure 4.7 shows that, as the number of schedulers decreases, the drop-in performance or the increase in computation time is not significant. This behavior re-asserts the resilience of our *Management Layer*, enabled by its multi-master design, by showing that the failure of multiple schedulers has a negligible effect on the overall performances of the system.

4.2.3 Compute Scalability

To evaluate the compute performance and scalability of the platform, we have performed benchmark scalability tests on data size, number of executors and number of users. The computation has been executed on a cluster of six machines: one driver and five executors. All nodes are identical and each one has 200GB of disk (Seagate, RAID 1 @ 7200RPM), 100GB of RAM

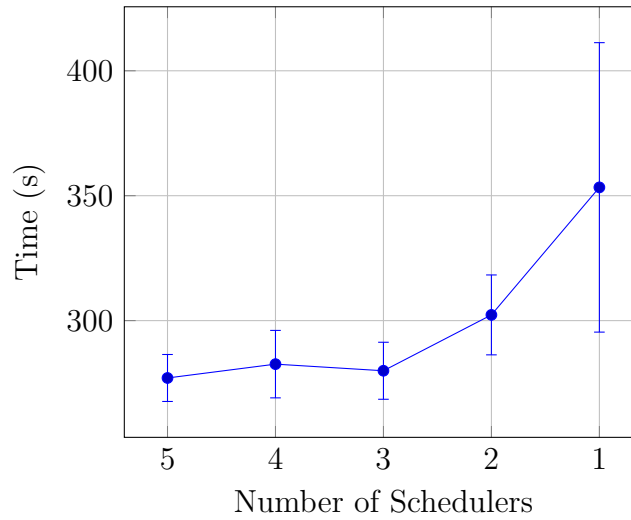


Figure 4.7: The performance of the *Management Layer* as the number of schedulers decreases. Each point is the average running time of 3 experiments along with the standard deviation.

(Micron, @ 1600 MHz), 24 cores (Intel Xeon, E5-2620 v2 @ 2.10GHz) and a network speed of 10Gb/s. The cluster itself relies on Cloudera CDH 5.11.0 which comes with Spark 1.6.0. The Spark configurations used in all experiments for the “spark-submit” are :

1. master = yarn-client
2. executor-cores = 16
3. driver-memory = 20GB
4. executor-memory = 20GB

The number of executors has been set to five for the first compute scalability experiment but has been set to different values in the other second compute experiment. The analysis executed in the executor scalability experiment is linear SVM which is a method available in Spark’s machine learning library.

The experiments are designed based on the iterative model generation and cross-validation pipeline mentioned in Section 6.1.1. In this experiment, we use mRNA data from a GEO dataset (GSE31773) [TW12] as a base to synthesise the required data. The synthetic data has been generated by first averaging the original values and then adding randomly generated noise for every new vector until the desired data size is reached. The label for each vector is randomly chosen. Six randomised datasets have been generated independently, with sizes of 1MB, 10 MB, 100MB, 1GB, 10GB, 100GB, respectively. The 1MB dataset has been used

to determine Spark’s initialisation time. This initialisation time has been subtracted from all other measurements in order to keep only the effective computation time of the model. The data has been placed in HDFS for Spark to use. From the results plotted in Figure 4.8, we can observe that the time required to process grows linearly with the size of the data, which constitutes the first indicator of scalability of the platform.

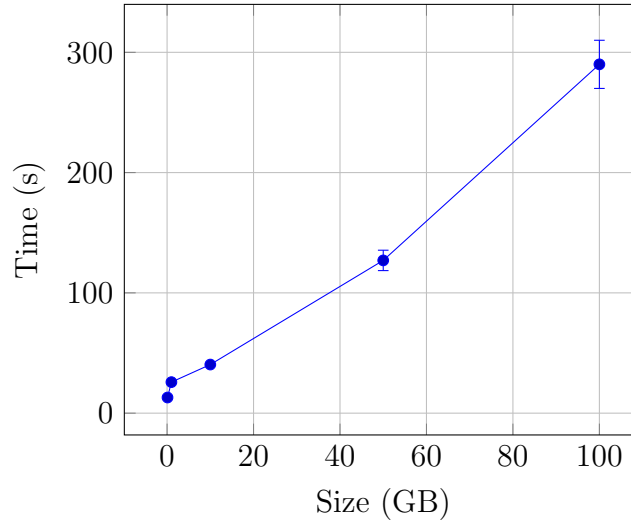


Figure 4.8: The scalability of the eTRIKS Analytical Environment with respect to the data size. Each point represents the average running time of 30 experiments, with the error bar representing the standard deviation.

In the second experiment, we reused the 100GB dataset generated beforehand but used a different number of executors every time to observe the impact on the computation time. We can see from Figure 4.9, a considerable decrease in the computation time when we increase the number of executors, which constitutes the second indicator of scalability of the platform. The inflexion occurring between the 3 and 5 executors indicates as well that adding too many nodes to the computation may prove counter-productive as the time gained for the computation is only small.

4.2.4 Storage Scalability

To evaluate the storage performance and scalability of the platform, we have performed benchmark scalability tests on data size. The Swift cluster has been deployed on three machines, all nodes are identical and each one has 3TB of disk from a disk array (Seagate, RAID 1

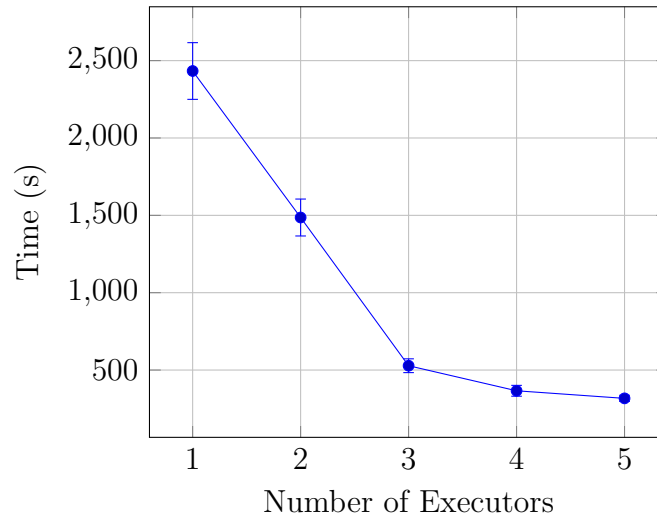


Figure 4.9: The compute scalability of the eTRIKS Analytical Environment with respect to the cluster size. Each point represents the average running time of 30 experiments, with the error bar representing the standard deviation.

@ 7200RPM), 100GB of RAM (Micron, @ 1600 MHz), 24 cores (Intel Xeon, E5-2620 v2 @ 2.10GHz) and a network speed of 10Gb/s. The client uploading the data to the cluster was a single machine with 1TB of SSD disk (Samsung 970 PRO) for storage, 16GB of RAM (Corsair, @ 2133MHz), 8 cores (Intel i7-6700, @ 3.40 GHz) and a network speed of 1Gb/s.

To evaluate the *Storage Layer* capacity to scale as the size of files grew, we uploaded/downloaded six files of varying sizes ranging from 0.1GB to 1TB. Figures 4.10 and 4.11 demonstrate that the upload/download capabilities of the *Storage Layer* scales linearly up to the terabyte level. It is to be noted, however, that those times could have been significantly better with a higher bandwidth on the client side.

We did not explore a varying number of replicas as only a three-replica cluster makes sense in a production environment. Indeed, a single replica would not provide any resilience to the platform and two replicas would not offer the possibility to continue operating the platform if the other replica were to fail at some point (the remaining replica would lock itself to avoid any data loss or corruption). Similarly, any deployment with more than three replicas would result in a huge amount of network IO between replicas and a lot of time and CPU power spent replicating the data without any overall performance improvement to the storage layer.

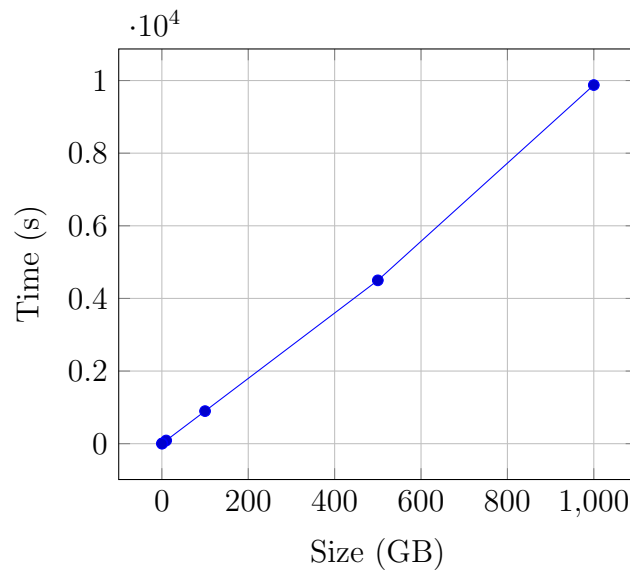


Figure 4.10: The storage upload scalability of the eTRIKS Analytical Environment with respect to the data size. Each point represents the average running time of 5 experiments, with the error bar representing the standard deviation.

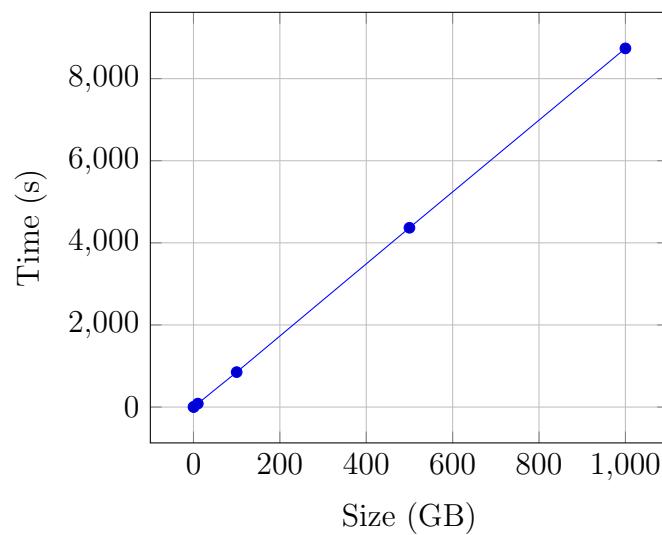


Figure 4.11: The storage download scalability of the eTRIKS Analytical Environment with respect to the data size. Each point represents the average running time of 5 experiments, with the error bar representing the standard deviation.

4.2.5 Summary

The resulting system provides an efficient and effective solution for data exploration and high-performance bioinformatics analytics while implementing strictly the architecture defined in Chapter 3. The technological stack uses only what is necessary, which facilitates new developments, new features and the adoption of the platform. The services are all strictly independent,

autonomous, even more resilient and streamlined with the best engineering practices and strict coding style put in place. The endpoints have the possibility to be fully standalone or to go for a deeper integration for optimal performances as the integration with another eTRIKS project called Borderline has shown.

The platform has a high density of logs, which enables better monitoring and easier maintenance of the platform. In addition, no downtime is needed to add new services or restart part of the platform as they can self-register directly with the other services while the management layer monitors and manages the health of the clusters and services at any given time.

The new compute layer has permitted to distribute more evenly the computation load across the different machines without any meaningful impact on the user. The load distribution is important as it allows to distribute the wear evenly across machines and thus avoid premature failing of the hardware.

In conclusion, this implementation is an elegant, efficient and scalable solution for data exploration and high-performance bioinformatics analytics. This version is production ready and is the base upon which extensions can be built to further extend the capabilities of the platform as we will demonstrate in Chapter 5.

4.3 TensorDB: Database Infrastructure for Continuous Machine Learning

This section introduces the TensorDB system, a framework that fuses database infrastructure and application software to streamline the development, training, evaluation and analysing machine learning models. That work was carried out with the goal to further facilitate the research and development of new machine learning models as well as the reproducibility and traceability of that research.

4.3.1 Introduction

In nature, learning is adaptive and progressive, that faculty enables animals to change their behavior as the environment changes [SYL13]. On the other hand, most machine learning algorithms, like deep learning, are static and never spontaneously evolve regardless of the situation. With the dominant batch learning approach, there is a clear separation between training and inference. Machine learning models are trained off-line with large amounts of training data and the parameters are fixed and optimised according to specific objective functions. For most applications, such methodology faces many limitations. Most businesses are dynamic, and the machine learning models must be updated continuously as the business operates. For e-commerce businesses, for example, the recommendation system must be updated as user preferences or market conditions change.

Continuous machine learning is the key towards machine intelligence and of increasing importance in research. Recent machine learning models such as reinforcement learning [SB16] and generative adversarial network [GPAM⁺14] are continuous in nature. We argue that the continuous learning, adaptive and progressive features of the biological system are indispensable for machine learning applications.

We introduce TensorDB, a framework to enable continuous training, evaluation, analysis, and deployment of machine learning models. TensorDB is not an algorithm for continuous machine learning, but an autonomous life-cycle management system of machine learning models. The core idea is based on the observation that continuous learning is achieved by continuously updating the training sets and continuously training and evaluating the models. Assuming unlimited storage and computing power, we can build millions of models with different architectures while the evaluation system will keep on mining the model repositories and always select the one with the best performance for deployment. When designing such a system, the key challenges lie in the data management of the models, training sets, parameters, and logs.

The data management system of TensorDB is based on a NoSQL database [MH13] and a map-reduce search engine [YDHP07]. It is implemented as a distributed system that fuses database

infrastructure and machine learning application software. All the components of machine learning development are connected by the database query mechanism, which sets up TensorDB as a flexible system that enables each component to be updated continuously. Even when continuous learning is not required, machine learning developers can benefit from TensorDB for streamlining the development process. The framework can be used as a data warehouse for managing the training sets, a logging system for recording training process, a version control system for machine learning models, an intelligent system for model selections, and a load balancing system for distributing machine learning jobs. TensorDB system provides all the above functions and allows for fully autonomous pipelines for sampling, training, evaluating, analyzing and deploying machine learning models.

TensorDB is made of three components:

1. Cohort recruitment for management of training set and models
2. A building/job system for training models and a logging system for recording the training process
3. Model mining system for evaluation and model selection

4.3.2 Related work

While many libraries for building and training deep learning models have been developed, little work has been done for the models' management and provenance. Recently ModelHub [MLDD16] has been proposed as a version control system for exploring and storing all the models. However, TensorDB is more ambitious and focuses on the working flow. From the technology perspective, a critical difference is that TensorDB is based on NoSQL database technology and uses map reduce as the searching method while ModelHub is SQL based and thus less flexible. The current implementation of TensorDB is based on MongoDB and thanks to the schemaless design, users can alter the database schema at any point in time. The key feature in the TensorDB design is the flexibility in the binding system. Flexible binding is

implemented as data search query thanks to the support of a database. This design is inspired by the Katana lighting system [HSL14] developed by Sony Image Works. The logging system is inspired by the EHR system for healthcare record management and it follows the NoSQL solution for Pharmacovigilance [AAoNLR⁺16].

4.3.3 Architecture

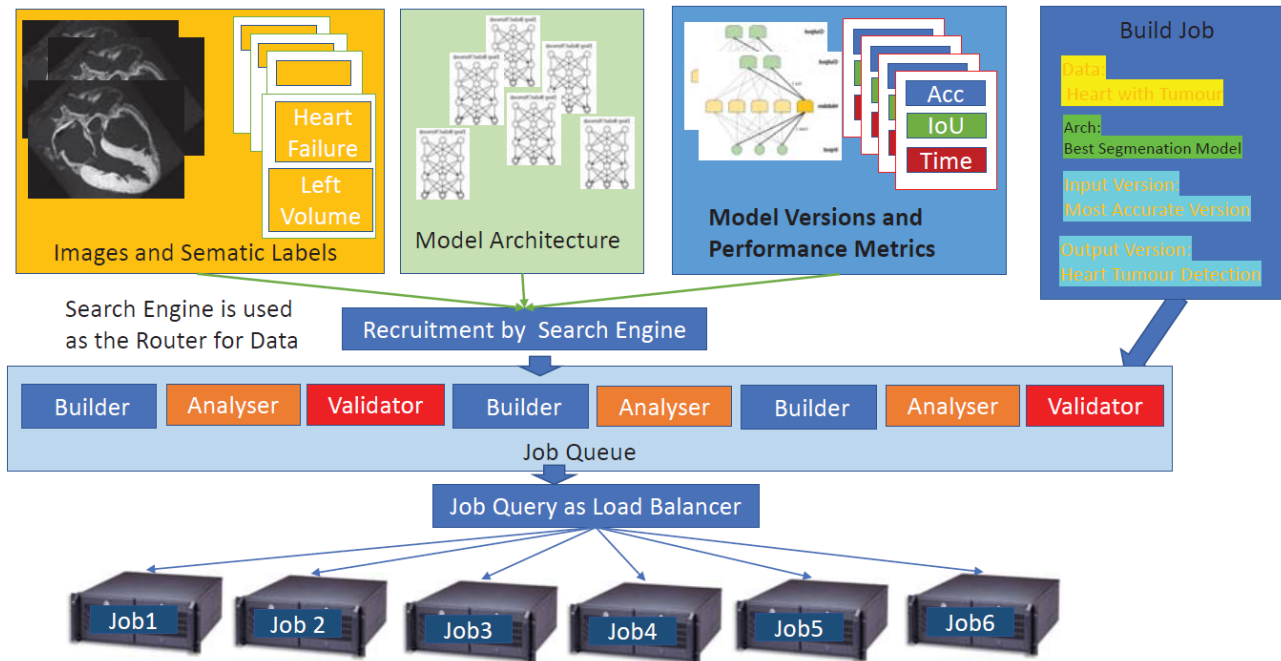


Figure 4.12: TensorDB workflow: the database query mechanism connects all the components. The work is distributed across multiple machine.

TensorDB is built around several core components as presented in Figure 4.12. The cohort recruitment system is a data warehouse for managing training sets and models. Our MongoDB based implementation stores the data as JSON documents, which is a dictionary of many key-value pairs (MongoDB replaces tables by collections which is a set of documents). Searching is performed by Map Reduce methods, which MongoDB supports natively thanks to built-in capabilities, and this is then mapped to the filtering, projection, and sorting methods. TensorDB maintains five collections.

1. The Model Collection which contains all the model architecture and parameters.
2. The Dataset Collection which contains all the training and validation set.

3. The Training Log which records the performance of each step during training
4. The evaluation Log which records all metrics for models on different validation sets.
5. The Job System which maintains all the building jobs and is also served as a queue for load balancing.

To be more precise, there are also two collections for models and datasets which are used as files system. There is also corresponding software operating on the data to run the TensorDB system :

1. Data Importer which imports training datasets into the database.
2. Recruiter is implemented as a query program based on MongoDB query language. Each cohort is not physically stored in the database, the searching program is executed during run time when the cohort is assembled.
3. Builder reads the job queue and executes the training and evaluation for the job. All the model parameters of each epoch are stored in the database. The logs for each training step are stored in the logging system as well.
4. Evaluation and Model Mining system also relies on the job queue and executes all the validating jobs. The evaluation results of each validation set and model metrics are stored in the database.
5. The analysis and visualization system explores the training and evaluation logs.

In the application, most of the code and database schema is fixed. The flexibility comes from the searching-based recruiter. Search-based recruiter never directly binds a model to data. For example, the training set could be specified as the last 1000 imported, which will keep on changing if data are generated continuously, and the evaluation to validate only the five models that have the least training errors.

The Cohort Recruitment System

The cohort management system is in principle a data warehouse that stores training datasets and models. There are three kinds of data 1) training sets, 2) model parameters, and 3) model architecture. The storage scheme separates metadata files and actual data. The query is performed on the metadata fields, while the actual data are stored as chunks in a database-based file system (GridFS). In our implementation, the raw data are stored as binary strings, which are the result of a serialization operation. Images and parameters are based on the Numpy serializer, models are based on the Tensorflow graph serializer. Software tools are required to extract the meta information, which is the role of Data Importer. The meta information is designed to be information rich, containing as much valuable information as possible. For medical images, besides the information in the dicom headers, it can also include semantic information such as the parts of the body scanned, volume of brain ventricle, and grade of tumour. This is greatly facilitated with a NoSQL database, which enables a very flexible data structure. For training data, we put both data and labels in one document. For models, the architecture and parameters are stored separately. The meta information of parameters includes references to training datasets, upload time, training errors, epoch number and step time of backpropagation. The recruitment system is implemented as a query-based search on the meta information fields. They are used for finding the case of interests, which is very valuable for the development of machine learning models, such as controlling the balance of positive and negative cases. A daemon process is invoked regularly to check if all machines are running their jobs properly and, in the event of a machine becoming unavailable or breaking down, redeploys the jobs on an available one.

The Building System

The Building System treats a training machine learning model as a software building. By combining datasets and algorithms, the building system just runs model training forever. In the implementation of TensorDB, we define a collection called jobs that serves as a queue and load balancer. The job document contains model architecture; the model initialises parameters,

training datasets and time stamp which indicates when the job is generated. De facto, the Building System continuously creates a hyperparameter space exploration from the initialise parameters and the constraints provided when creating the request. The Building System generates the hyperparameter space randomly at first and then follows the paths towards the best model accuracy after each iteration. There is also a status flag to indicate whether the job is ready, being processed or finished. The evaluation system shares the job queue with the building system, with different job types. The building software automatically accesses the database, finds an idle job from the job queue, and updates its status in an atomic manner. When a job is running, training results of each step are recorded in the logging system. Each record document contains the model architecture, epoch number, time, accuracy, performance metrics, study id, and information of hosting computer. After each epoch, the building system uploads the parameters into the cohort system, which updates the job fields with the latest update time, current epoch, and model accuracy. Evaluation and Model Mining as a system is straight forward to be implemented with building blocks from the cohort system, building system, and analysis system. In principle, the model mining system selects representative test data and evaluates all the possible models. After the system finds the best model, the system can also continue to improve it by recruiting new training sets and launching new training jobs.

The Analysis and Visualization System

The Analysis and Visualization system explores the system's log data for insights of model development. Its applications include visualization of the model learning speed, comparing convergence speed of different architectures or execution speed. The requirement is very flexible and opens many new opportunities such as meta-learning that needs to select the best architecture. Rather than providing tools, TensorDB facilitates such analysis. Finally, MongoDB being a popular database, many packages are available to support further data analysis and visualizations or export the data to other external software.

Evaluation and Model Mining

Such a system is straight forward to be implemented with building blocks from the cohort system, building system, and analysis system. In principle, the Model Mining system selects representative test data and evaluates all the models. After the system finds the best model, the system can also continue to improve it by recruiting new training sets and launching new training jobs.

4.3.4 Application Evaluation

We tested TensorDB with several challenging deep learning research projects. For our recent work on brain tumor segmentation, several modes were required to segment high/low-grade tumor from multiple imaging modalities [MGAS12]. In order to test our modes of interest, we performed a 5-fold cross-validation. The combination of modality, training sets and model architecture produced more than 100 models. Manual management such as moving the data, models and parameters across machines are error-prone and waste a lot of programming and computing time by forcing the machine into unnecessary idle time. It is also very hard to monitor all the training processes in one view. The introduction of TensorDB solves all problems by centralizing the data in a database and tools are easily built and can help the research with a global view of different model and architecture performance. The building and testing of the 100 models sharing the same application code with only different datasets specified by the query language. For distributed training, TensorDB was deployed as an independent remote server. The drawbacks of TensorDB is mainly from a performance perspective. Logging into the database is more time consuming than printing on the screen. However, for our image segmentation, the overhead is marginal. Each step consumes about 1.4 seconds and logging costs about 1.5 ms. The epoch consumes about 600 seconds while the upload of model parameters to the database takes about 3 seconds and the downloading of parameters about 2 seconds. The bottleneck of TensorDB solution is the loading of training data. Compared with loading the training data locally, loading the training dataset with the recruiting system is usually about 10 times slower. Loading about 6000 high-resolution images from the local disk costs about 30

seconds while from TensorDB, it costs about 300 seconds. If we constrain the loading time to be 5% for model update and model will finish in 20 epochs, each job will have a time resolution of about two hours. In terms of the storage burden, after running a hyperparameter exploration for a week, the system generated about 500 million log data points where each point was only about 20 bytes. Each log data contains the current performance of the step and a status code. Those logs can be used by the researcher for model development insights. The training data size was about 30GB which is fixed. The real storage pressure comes from saving all the model parameters of each epoch, which consumed about 300GB of storage. For analysis, the speed depends on the counts of query result. When the count number is less than 100,000, it usually takes about 4 seconds while 10,000 records consume about 0.5 seconds. The system is not real-time but can performance analysis interactively.

4.3.5 Conclusion

TensorDB is a flexible framework to support machine learning research and development. The database-based solution and the search-based binding make the learning development more flexible and easy to manage. It allows the system to continuously update the training sets, the learning mode and update the deployment in a fully autonomous manner. The current system has some performance issues, is not real-time and the loading is time-consuming. However, it is good enough for the machine learning application when models do not have to be updated more than 10 times a day.

Chapter 5

eTRIKS Analytical Environment with Privacy

In this chapter, we will present the use of location data in the context of public health research, the work done in the context of data privacy for location data and the extension of the eAE to support privacy preserving analytics. This work illustrates the modularity and extensibility capabilities of the eTRIKS Analytical Environment architecture.

5.1 Building Privacy capabilities

5.1.1 Location data as a support for public health

Diseases propagation using mobility data

Epidemic outbreaks are an important healthcare challenge, especially in low- and middle-income countries where they represent one of the major causes of mortality [MVL⁺12, BCJ⁺10]. To make the matter worse, it has been shown that re-occurring outbreaks due to preventable infectious diseases contribute to hinder the economic development even in developed countries [SKBBT09]. However, the prevention and containment of an infectious disease outbreak

can be greatly improved if health care response and outbreak control measures can be focused to areas predicted to be at the highest risk of experiencing new outbreaks [VBS⁺06].

Human mobility is indisputably one of the main spreading mechanisms of infectious diseases [CZG⁺17]. Therefore, understanding human movement and mobility is important for forecasting, characterizing and controlling the temporal and spatial spread of infectious diseases. The prediction of the temporal evolution of epidemics, once outbreaks have progressed beyond a small initial group of cases, has greatly improved in recent years. However, predicting spatial transmission routes of epidemics has proven to be remarkably difficult, due to the importance of rare, long-distance transmission events [Ril07], limited data on population mobility, unknown population immunity levels [GF08], low sensitivity and specificity of case reports [VBS⁺06] and limited access to accurate and spatio-temporally resolved case data.

In the last decade, the widespread adoption of mobile phones and other ubiquitous technologies are generating vast amounts of high-resolution location data. The penetration rate of mobile phones (percentage of mobile numbers in use per 100 citizens) in developing countries can vary from 70% in Colombia to 99.9% in Senegal while most developed countries are well above the 100% mark. Indeed, it is not uncommon for professionals in developed countries to have both a professional and personal number. Prepaid SIM cards also contribute to the multiple lines people may own. The analysis of individual Call Detail Records derived from mobile phone data has provided plentiful insights into the quantitative patterns that characterise human daily life [BDK15]. Notably, mobile phone data have demonstrated to be an excellent source to describe human trajectories at the finest scales, providing unprecedented details on individual mobility and highlighting some shared features, such as the high degree of predictability of individual patterns which coexists within strong heterogeneities of collective patterns [SKWB10, GHB08, PTB⁺17].

Based on this observation, several studies have shown that it is indeed possible to describe the countrywide-scale infectious disease spread even as individuals change location over time using mobile phone call data records. Wesolowski et al. [WET⁺12] have successfully demonstrated that human movements contribute to the transmission of malaria on spatial scales that exceed

the limits of mosquito dispersal. They used spatially explicit mobile phone data of nearly 15 million individuals over the course of a year and malaria prevalence information from Kenya to identify the dynamics of human carriers that drive parasite importation between regions. The maps (from their study) in Figure 5.1 highlights remarkably well how human mobility fosters the propagation of the parasite.

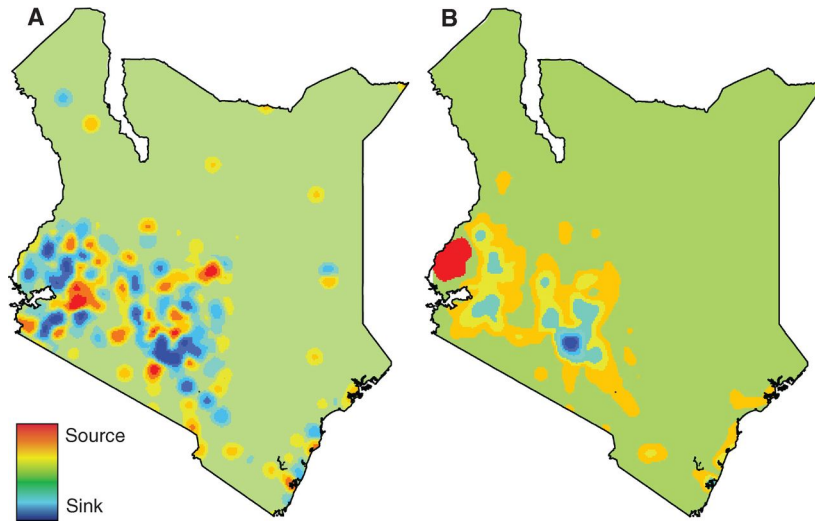


Figure 5.1: Sources and sinks of people and parasites from Wesolowski et al.’s [WET⁺12] study. Kernel density maps showing ranked sources (red) and sinks (blue) of human travel and total parasite movement in Kenya, where each settlement was designated as a relative source or sink based on yearly estimates. (A) Travel sources and sinks. (B) Parasite sources and sinks.

Their mobility data contained a high spatial resolution which allowed them to pinpoint particular settlements that are expected to receive or transmit an unexpectedly high volume of parasites compared with surrounding regions.

From that ascertainment, Bengtsson et al. [BGL⁺15] have shown that, not only was it possible to model the spread, but also to predict the spatial spread. Using mobile phone data, they have highlighted in their study that approaches that can rapidly target sub-populations for surveillance and control are critical for enhancing containment and mitigation processes during epidemics. Modern epidemic modelling recognises the central role of population structure and patterns of interactions and mobility, as components that can considerably influence the likelihood of disease propagation [RSM17]. The consideration of this complexity is the first step towards the integration of location data into traditional epidemic model.

Populations displacement during crises using density data

Most severe disasters, such as earthquakes and tsunamis, cause large population movements. In 2017, there were 30.6 million new displacements associated with conflict and disasters across 143 countries and territories [iDM18]. Among those 30.6 million displacements, 18.8 million were directly linked to disasters, which represents the equivalent of 51,500 people being displaced each day. These movements hinder the work of relief organizations to efficiently reach people in need. These movements can take place prior to events, due to early warning messages, or occur post-event as a result of damages to refugees, livelihoods and long-term reconstruction efforts. Those displaced populations are immensely vulnerable and often in urgent need of support (e.g. clean water, food, blankets, etc.). Timely and accurate data on the numbers and location of displaced populations are exceedingly difficult to collect across temporal and spatial scales, especially in the aftermath of disasters.

Nonetheless, similarly to the diseases' propagation, mobile phone Call Detail Records were shown by Lu et al. [LBH12] to be a reliable data source for estimates of population movements after the 2010 Haiti earthquake. They used 1.9 million mobile phone users ranging from 42 days before up to 341 days after the Haiti earthquake of the 12th of January 2010. They showed that the predictability of people's trajectories remained significant and even increased a little during the three-month period after the earthquake. In addition to that predictability, they have noticed a strong correlation between those destinations and their mobility patterns during normal times, and specifically with the locations in which people had significant social bonds.

The importance of CDRs analytics in the humanitarian space was confirmed by Wilson et al. [WZESAe16] in the context of the 2015 Nepal earthquake. Their analyses unveiled national level population mobility patterns and return rates which are extremely difficult if not impossible to acquire using other methods. They estimated that 390,000 people above normal left the Kathmandu Valley soon after the earthquake. Many of these moved to the highly-populated areas in the central southern area of Nepal and the surrounding areas. People who left their home area after the earthquake have gradually returned to the affected areas, with the return rate varying between regions.

These analyses are of tremendous relevance to humanitarian agencies, as density patterns can help identify where aid should be directed, and low return rates can identify areas where recovery and reconstruction work may not be progressing well. However, in both cases those presented two critical limitations which are the speed with which the analyses were delivered, and the absence of privacy guarantees regarding the data used by researchers. Indeed, the first analyses for responders to the Haiti earthquake were distributed several months after the earthquake which is far too late to support any early humanitarian response, while it took nine days to provide spatiotemporally detailed estimates of population displacements for the Nepal earthquake.

Those examples illustrate perfectly the need for an open-source, scalable, and privacy-preserving platform to support real-time key statistics computations from location data for a wide range of potential use cases.

5.1.2 Attempts at sharing location data

In order to exploit the potential of location data, attempts at sharing that data have been made in the past. Orange’s Data for Development (D4D-Senegal) challenge [dMST⁺14] was an open innovation data challenge on anonymous call patterns of Oranges mobile phone users in Senegal. The challenge took place in 2014 and consisted of three mobile phone datasets. The datasets were built upon Call Detail Records of text exchanges and phone calls between January 1, 2013 and December 31, 2013 of more than nine million of Oranges customers in Senegal. The datasets are: (1) antenna-to-antenna traffic for 1666 antennas on an hourly basis, (2) fine-grained mobility data on a rolling 2-week basis for a year with Bandicoot [dMRP16] behavioural indicators at individual level for about 300,000 randomly sampled users, (3) one year of coarse-grained mobility data at arrondissement level with Bandicoot behavioural indicators.

Despite the fact that those data sets were quite granular, many noteworthy achievements have been made and several papers have been published. Martinez-Cesena et. al. [MCMNS15] have demonstrated that even with coarse spatial and temporal resolution data, it was possible to offer unprecedented insights into the spatio-temporal distribution of people thus enabling

efficient electricity infrastructure planning in rural areas where information on human activity are typically limited. Those pieces of information are immensely valuable as a basis for planning the development of electric power infrastructure in a country where 70% of the rural population has no electricity while close to 99% of all Senegalese carry cell phones. Other contributions have seen the optimization of road network [WdACdRS18], poverty analysis [PDG15], public health [FGM⁺16] or assess the impact of natural disasters and disease outbreaks in real-time [WZESAe16, DLM⁺14, BGL⁺15]. After the success of the Senegalese challenge, Orange has started two other similar challenges in Ivory Coast [BEC⁺12] and Niger.

It can be mentioned that a similar initiative was carried out by Telefonica in the context of their Smart Steps project¹ where they aim at providing specific business and sale optimization through the analysis of travellers behaviour.

From such promising research paths, it is thus reasonable to assume that, with higher spatial and temporal resolutions for the data, a lot more could be achieved for developing new indicators, higher accuracy for the proposed models, improved public health, disease control and facilitate the economic growth of developing countries. However, mobility traces are extremely sensitive. In 2009, the Electronic Frontier Foundation listed examples of sensitive information that can be inferred about an individual from his location history. These include the attendance of a particular church, meeting an AIDS counselor or an individuals presence in a specific house or at an abortion clinic. These legitimate privacy concerns for the potential misuse of mobility data need to be addressed.

5.1.3 Sensitivity of location data

Even though mobility data, containing accurate user-level trajectories of visited places across long time periods, has been extremely valuable for researchers and organizations, and still holds significant potential for the public good as Orange's D4D-Senegal challenge highlighted, these advancements have been possible only thanks to the widespread adoption of mobile phones. That penetration rate (percentage of adults in possession of at least one phone) in developing

¹<https://www.wholesale.telefonica.com/en/services/digital/big-data/smart-steps/>

countries can vary from 70% in Colombia to 99.9% in Senegal while most developed countries are well above the 100% mark. While these types of engagements offered evidence of the promise and demand, these modalities limit the full realization of Big Data’s social potential as they fail to meet the standards that experimentations such as MIT’s OpenPDS [dMSWP14] initiatives have shown to be both possible: safe, stable, and scalable access to data for public good purposes.

Historically, data has been anonymised through de-identification, i.e. the process of transforming personal data to mask the identity of participants. Fully de-identified data is not considered personal data and can be shared or sold without limitations. However, a large body of research has shown that de-identification is not resistant to a wide range of re-identification attacks [Swe97, NS08, dMHVB13, CRT17, Ohm10]. This is especially true for geolocation data, as individual mobility traces are highly unique even among large populations, making them particularly vulnerable to re-identification. de Montjoye et al. [dMHVB13] showed that 4 spatio-temporal points are enough to uniquely identify 95% of people in a mobile phone database of 1.5M people, and that just a couple more points are needed if the data is heavily coarsened.

OPAL (for Open Algorithms) is a project with the goal to allow for private data to be used in privacy-conscious ways for good and to unlock the potential of mobility data. This is achieved by fundamentally changing the paradigm of data release: rather than publishing (de-identified) data, OPAL stores the data in a protected environment and allows analysts to send queries about the data. OPAL’s core consists of an extended version of the eAE with a new set of services and a set of open algorithms that can be run on the servers of partner companies behind their firewalls to extract key development indicators of relevance for a wide range of potential users. Since analyses are computed using fine-grained data, it is possible to achieve both better utility and stronger privacy compared to de-identification techniques.

A typical use case for the platform would be to compute the population density and mobility of a certain area for any given time interval, without releasing the full geolocation dataset to them. Ideally, a query-based system such as this one would enjoy some important properties:

Secure. The platform is secure against penetration attacks that aim at gaining unauthorised access to data.

Privacy-preserving. The outputs to queries should consist of aggregated data and should never disclose private information of users whose records are in the dataset. This guarantee should hold when analysts obtain and combine outputs for multiple queries.

Flexible. Data analysts should be able to submit different queries that serve a large array of statistical purposes by enabling developers to propose new algorithms that can be loaded on the platform.

Open. The code of the platform and of the algorithms should be open-source. This allows for better security, privacy and utility, as everybody can review and contribute to the code, or build new algorithms on top of the existing ones.

5.2 Privacy preserving eTRIKS Analytical Environment

This work has been supported by the OPAL project² and the platform is based on the implementation presented in Chapter 4. The platform is designed with the same four layers - *Endpoints Layer*, *Storage Layer*, *Management Layer*, and *Computation Layer* as shown in Figure 5.2. However, in addition to the privacy features, new services to support additional features required by the OPAL project and two algorithms –population density and population mobility– have been implemented. The algorithms and the privacy module for **density** have mostly been contributed by the project while the rest of the work is my own contribution.

5.2.1 New services and features

A new set of services and features have been developed to address the requirements of the projects. Some services are generic (with application specific configurations) while others are

²<https://www.opalproject.org/>

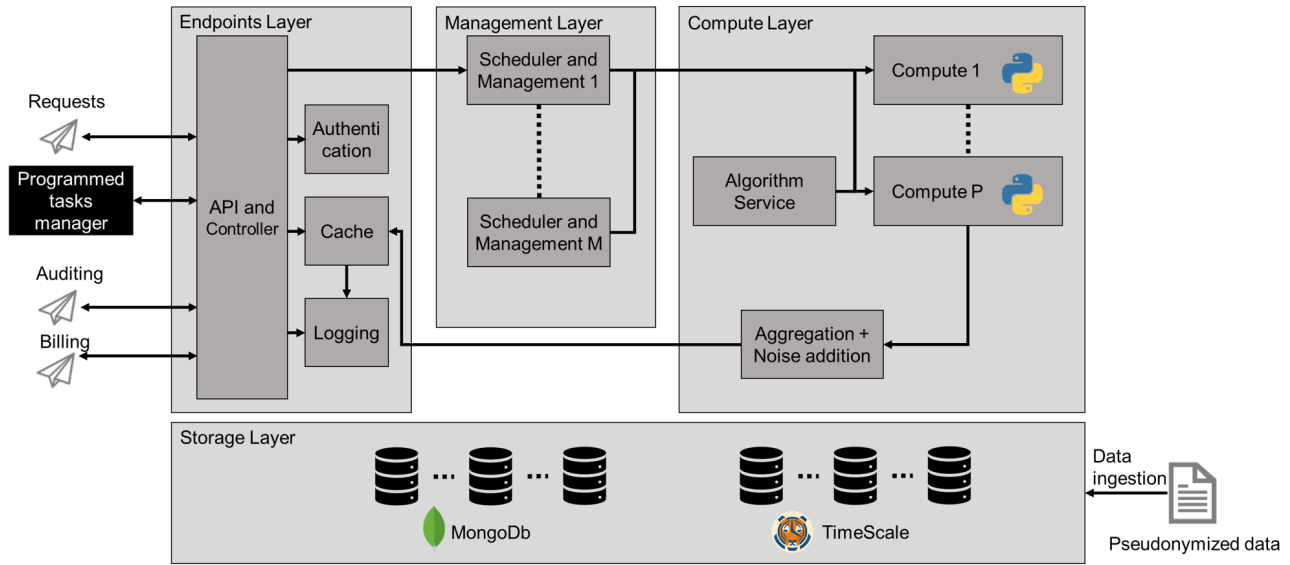


Figure 5.2: A schematic representation of the architecture of the OPAL platform.

data and application specific. The specialization of some services was necessary to meet performance requirements.

Generic components

Endpoints Layer

In the current configuration, the REST API in the *Endpoints Layer* provides the only public interface to run analyses over the data sets. A query is a request for running an analysis against the data available for the requested time interval and an answer is the output of the requested analysis. Each query must contain the name of the analysis (e.g. density, mobility, etc.), start and end date for the data on which it needs to run and other parameters required by the analysis. The platform technically supports a temporal resolution for queries down to the second, but the operational resolution has been set to 10 minutes for privacy reasons that will be detailed in Section 5.2.3. *Interface service* validates each query by checking all the required parameters are available and well-defined. If the request is not well-formed, the platform rejects the request, sends back to the user which field is not compliant and why, and finally logs the illegal request into the *Logging service*. We used Another JSON Schema Validator (Avj) [Evg18] which supports JSON Schema draft-07 to ensure both strong guarantees on the parameters validity and flexibility/extensibility in the future developments of the platform.

The *Cache service* works in the same way as in the vanilla version of the eAE, e.g. it stores the answers for all the computed queries with the query parameters and other metadata (date and duration of computation, etc.). However, a new feature has been added which enables answers to be signed with a private key to allow users to verify them a posteriori.

The *Logging service* has been also further extended to log more information for audit and billing purposes. The invalid queries are now logged to enable periodic analyses for detecting trends of any possible attack on the system (see Section 5.2.3). A new set of APIs have been added to enable users to retrieve the public logs of all valid queries that have been run on the platform. The valid queries are stored in an append-only text file, making it harder to tamper with without gaining physical access to the system, while the invalid queries are stored in the *Storage Layer*.

Management Layer

The *Management Layer* is exactly the same as described before as all necessary features were already available. For more details please see Chapter 4.

Application specific

Compute Layer

The *Compute Layer* has been extensively modified and the *Algorithm* and the *Aggregation and Privacy* services have been added to the layer. The *Compute Layer* has a combination of security and privacy features which enables us to run privacy-preserving computations.

The platform relies on the MapReduce[DG08b] paradigm for defining and executing analysis algorithms (see Section 5.2.3). Each analysis algorithm is defined as a *Map* function that runs on the data of an individual user and a *Reduce* method to aggregate the results of the map over all the users. For example, a *Map* function could receive the Call Detail Records of a user and return the antenna ID that appears most in the data. A *Reduce* method would then count the number of times each antenna occurs in the output and returns the count for each antenna

id as the computation output. All the algorithms are audited and then added to the platform (see Section 5.2.3).

The *Map* function runs in a sandbox environment. It ensures that the *Map* function interacts with only one user's data in one system process. This guarantees that the computations run independently over the users (see Section 5.2.3). Furthermore, it prevents the function from accessing any other files besides the user being processed and from accessing any other system resources such as the system network.

Privacy mechanisms mitigate the risk that an attacker can infer details about an individual from an output or combination of the outputs from various analyses on the platform. We provide a functionality to add algorithm-specific privacy-modules which helps maximizing utility and privacy for their use cases (see Section 5.2.3).

The *Algorithm* service manages and versions the analysis algorithms, the *Compute* service fetches data and executes the map functions, the *Aggregation and Privacy* service aggregates the outputs from the *Compute* and applies privacy mechanisms on the aggregated result. Each *Compute* service is associated with a list of type of jobs it can execute, e.g., Python2, Python3, R, Spark.

On receiving the execution request for job *J*, the *Compute* node updates its status to *BUSY*, the status for *J* to *SCHEDULED* and sends the acknowledgement to the *Management Layer*. It fetches the algorithm requested by *J* from the *Algorithm* service and the data for the requested interval from the *Storage Layer*. The request is sent to the *Aggregation* service signalling the start of the aggregation for *J* and the aggregation method to be used. The compute node starts executing the Map function, for each user, in a sandbox and results are sent to the aggregation service. At the end of the computation, the request is sent to the *Aggregation* service indicating the end of the aggregation for the job *J* and the compute node updates its status to *IDLE*. Upon receiving the aggregation end request, the *Aggregation* service runs the privacy algorithm on the aggregated result, saves the privacy-preserving output to the database and updates the status of the job to *DONE*.

Storage Layer

The *Storage Layer* has been modified to manage the mobile phone data stored directly in the platform. Mobile Phone Data consists of pseudonymised CDRs captured by the telecommunication service provider and the GPS coordinates of antennas. A CDR contains 9 fields (see Table 5.1) that forms the basis for the development of an algorithm on the platform. An antenna is defined by a unique antenna ID, location details (borough, commune, and region) and the presence interval (installation and removal times). Storing installation and removal times is essential for mobile antennas which can be moved from one location to another and the same antenna might have different locations at different times.

Field	Definition
Timestamp	Datetime at which record is captured by the telecom operator
User ID	Pseudonymised ID of the user whose record was captured
User Country	Country code of the user
Correspond ID	Pseudonymised ID of corresponding user
Correspond Country	Country code for corresponding user
Antenna ID	ID of the antenna the user was connected to during the initiation of interaction
Interact Type	Type of interaction - <i>call</i> or <i>text</i>
Interact Direction	<i>Out</i> if user initiated the interaction, else <i>In</i>
Duration	Duration of call in seconds, -1 for text

Table 5.1: Structure of a Call Detail Record.

The mobile phone data is pseudonymised and ingested periodically into the database by the system administrators. A small to medium sized country contains billions of records for a year of data [dMST⁺14]. Each computation fetches the data within the requested time interval and can typically range from hundreds to billions of records. The database, for mobile phone data, needs to scale to terabytes of data without significant decrease in performance and to

provide high speed data retrieval capability for each concurrent request. Ingestion speeds can, however, be slower without compromising the overall performance of the platform. Based on these requirements and detailed evaluation (see Section 5.2.2) we chose Timescale [Tim] for storing mobile phone data.

Data Flow

The data flow is designed to meet strict privacy requirements. Figure 5.3 describes the flow of the data and the subsequent transformations it undergoes from the raw data to the output of a query.

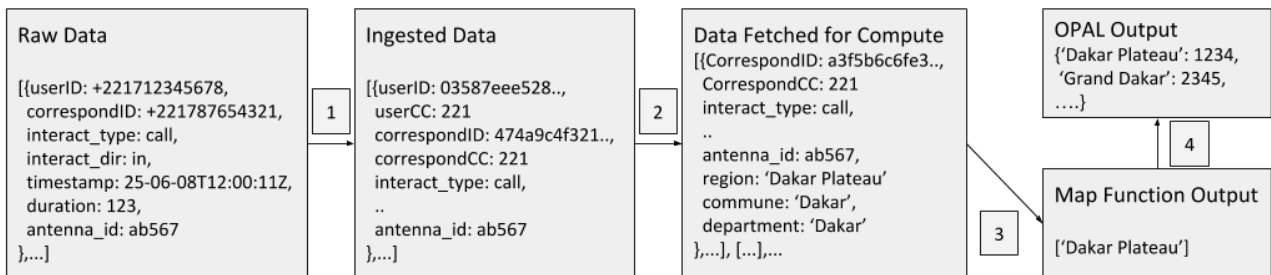


Figure 5.3: A schematic representation of the flow of the data from the raw data to the platform's output. 1. Pseudonymizing and ingesting data 2. Data Fetching for compute and creation of user specific CSVs 3. Executing Map function 4. Outputs aggregation and applying privacy mechanisms

The raw data, extracted from the data curator's database, consists of Call Detail Records (CDRs) and antenna details. Records for the users choosing to opt-out are removed and the remaining data is distributed across multiple files. Multiple parallel workers are created for data ingestion. Each worker fetches a file from the shared queue, extracts the country code from the phone number in each record, pseudonymises the phone numbers, and adds the modified record to a list. Records in the list are ingested in batches. The batch size and number of workers are tuned as per the system configuration. All pseudonymization steps are done using a salt and MD5 [Riv92] hashing function. MD5 has a sufficiently large domain space to avoid any collision while being small enough to minimise the impact on the storage.

The compute node creates a unique salt for each job execution and the fetched data is pseudonymised again using this salt. This ensures that distinct computations receive data for the same user with different user IDs, making attacks using multiple queries harder to accomplish. Each

record is joined with the antenna database to add antenna location to the records. Precaution is taken to ensure that the record timestamp is between the antenna installation and removal time. The compute node stores the fetched data for each unique user in a separate CSV file to sandbox the computations for each user.

5.2.2 Scalability of the platform

In this section, we will evaluate the scalability and performance of the new services of the platform.

Database

One key requirement of the platform is to be able to store and serve mobile phone data efficiently in range of billions of records (see Section 5.2.1). Different potential solutions already existed, after a thorough review, we narrowed the list to four candidates for further testing: MongoDB, Timescale, InfluxDB, and Druid. To evaluate the candidates, we conducted the following benchmarks:

1. Insert a month of mobile phone data (74GB, which is the typical size for a month of data in a small to medium size country, stored in a single CSV file) in a single process with batch size 10000 asynchronously (whenever possible). Evaluated insertion time is the average of two runs.
2. Run 5 different select queries fetching records in random 5 min intervals. Each retrieval fetched on average 60000 records. Average time over all the queries is reported.

Each solution was deployed in a container either provided by the project or manually created. The benchmarks were executed sequentially using two identical machines (24 cores, 100GB RAM, 7200RPM disks), one hosted the containers and the other executed the scripts.

<i>Databases</i>	Insertion time	Select Time
MongoDB	13h	34m
Timescale	46h	0.7s
InfluxDB	34h	7s
Druid	>48h	16m

Table 5.2: Comparison of core operations for the four potential solutions considered for the database.

From Table 5.2, we can see that Timescale provides a select time an order of magnitude lesser compared with the other solutions while having still acceptable performances for insertion time. The select is of key importance for the platform as it allows for faster computations of the requests in a production environment. It is for those reasons that we selected Timescale as the database. By choosing Timescale, we also get standard SQL engine and the extensibility capabilities of PostgreSQL.

Then, we evaluated the performance of Timescale in a single deployment instance as the scale of data increases. All the experiments were performed with Timescale-0.11, Postgres-10, Python 3.5 and asyncpg [Pyt] deployed as a Docker container on a 48 core machine with 189 GB memory and 8.7 TB of HDD (RAID 5, 10k RPM). We had 6 months of data, comprising of 8,778,751,539 unique records.

First, we evaluated the ingestion speeds to store an increasing number of records in the database. It is calculated as the total of data ingested divided by the total time taken. The raw data was stored in gzipped CSV files, each file containing an hour of data. 8 clients were created, each of them retrieved a raw file from a shared queue, uncompressed it and parsed one record at a time. Records were processed to pseudonymise the user and the correspondent IDs, and inserted in the database in batches with a batch size of 2 million. Ingestion speed is monitored in the background. The final results were reported as an average over 3 runs.

We observe in Figure 5.4 that the ingestion speed remains essentially stable at around 13MB/s as the amount of data ingested increases from 0 to 973 GB. This stability can be attributed to the transparent time-space partitioning provided by Timescale[Tim]. Overall, 6 months of

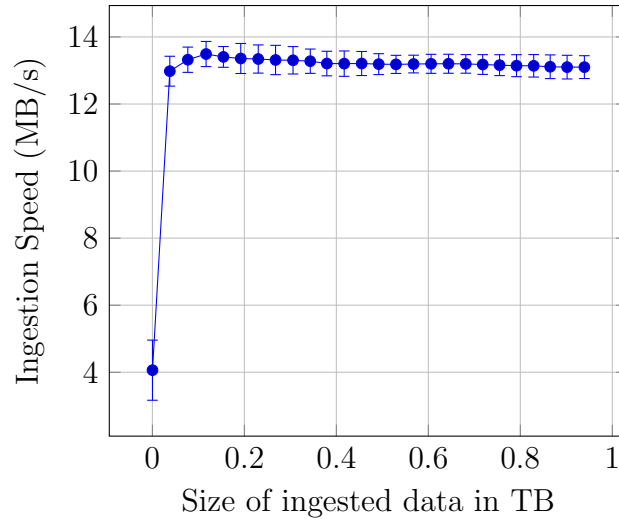


Figure 5.4: The insertion performance of Timescale with amount of data inserted. We observe that the speed remains essentially stable as the amount of data being ingested increases.

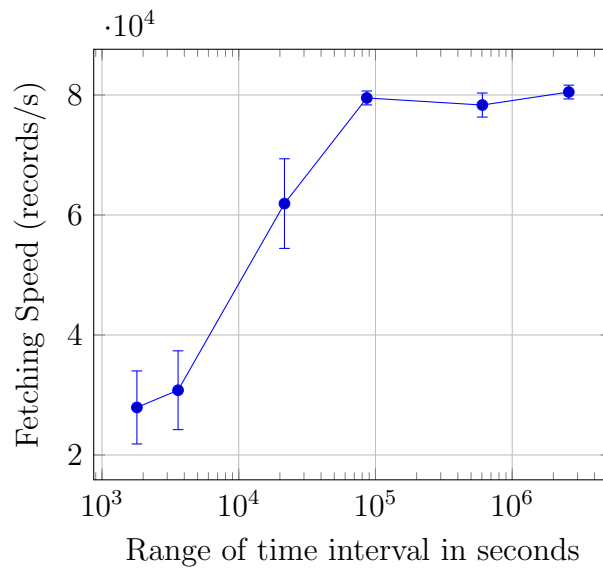


Figure 5.5: The data fetching performance of Timescale with time interval of the query. We observe fetching overheads to be significant for smaller queries which decreases as the query interval size increases.

data were inserted in less than a day.

Then, we evaluated the selection speed provided by the database as we fetch data in a range of time intervals. We fetched data for 20 queries each for 6 different interval lengths - 30 minutes, 1 hour, 6 hours, 1 day, 7 days, 30 days and measured the time taken from sending a query to completely parse the result and number of records fetched. Figure 5.5 shows the speeds to be largely similar as the intervals go from 1 day to 1 month. The lower speeds for smaller interval lengths can be attributed to the database overheads incurred during selection which becomes negligible as the interval length increases.

Compute Scalability

We evaluated the scalability of the *Compute* service to process large amounts of data while using Codejail and AppArmor sandbox. This experiment was conducted with two weeks of data containing over 44 million records for 320,000 users. We measured the time to Compute density for 3 different intervals (1 hour, 1 day and 1 week) over 3 different user sampling parameters (1%, 10% and 100%). Each compute fetches data of users in batches (of size 50,000) and processing happens in parallel. We used 6 workers for processing, 1 for fetching data and 1 master process. The experiment was done on a single machine (8 cores, 64 GB RAM, 7200 RPM disks), with the complete platform deployed and requests were sent to the *Interface* service for the query to be processed.

In Figure 5.6, we plot the time taken for compute and number of users in that interval. Figure 5.6 shows that the increase in compute time for intervals of 1 hour to 1 day to 1 week is directly proportional to the number of users in that interval. This behavior is attributed to the fact that sandboxing is the bottleneck in the computation. Sandboxing requires each user data to be saved as a distinct file, and a new process is spawned for processing data of each user, making it IO and CPU intensive. It takes less than an hour and a half to compute density for all the users for a week of data while the analysis on 10% of the users over the same period requires less than 9 minutes. Thus, sampling is tailored for scenarios when quick answers are required and utility can be slightly compromised.

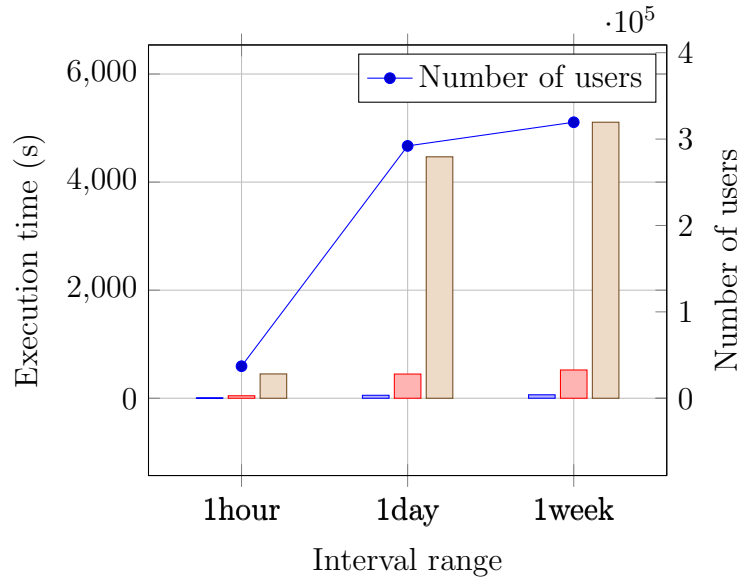


Figure 5.6: Measuring time taken for Compute and number of users for various interval range sizes with sampling parameters. Sampling Parameters used - Blue: 1%, Red: 10%, Brown: 100%.

5.2.3 Privacy of the platform

This platform belongs to the class of query-based systems, offering data analysts a remote interface to ask questions and receive answers aggregated from several, potentially many, records. Granting access to the data only through queries, without releasing the underlying raw data, mitigates the risk of typical re-identification attacks [Swe97, NS08, dMHVB13, dMRSP15, CRT17, Ohm10]. Yet, a malicious analyst can often submit a series of seemingly innocuous queries whose outputs, when combined, will allow them to infer private information about participants in the dataset [DSSU17, GHRdM18]. In section 5.2.6, we give an overview of the literature on privacy attacks and privacy-preserving mechanisms.

In this section, we describe how the platform protects privacy in query outputs, so that no individual-level data is leaked to the analyst. This is achieved by combining several protection layers that, together, significantly reduce the risk of personal data leakage.

User authentication When the platform receives a query, the request is first authenticated. A unique token is associated with each user account and must be supplied through the request headers. Each user is assigned an access level during their registration. The access level

defines the potential restrictions on the user for algorithms they can access, limit of spatial and temporal resolution requested in each query, and other settings defined by the data curator.

Limited number of queries Many attacks on privacy employ a relatively large number of queries to circumvent privacy protections, e.g. by averaging out noise [DSSU17]. To mitigate this risk, the platform includes a query rate limitation mechanism. This ensures that any analyst can submit a limited amount of queries in a certain time period, defined by the curator (e.g. 100 queries in 7 days by default).

MapReduce The platform relies on the MapReduce [DG08a] paradigm for computation. The Reduce methods currently supported are *count*, *sum* and *median*. The use of MapReduce helps privacy in two ways. First, it is easier to audit algorithms, since the Map function is applied independently to each user in the dataset, and hence makes it very hard for an attacker to hide conditions that try to re-identify specific records. For example, an IF statement that checks whether the user made a call with duration 5m23s on 10/07/2018 would look suspicious and raise a flag in the auditing phase. Second, the MapReduce paradigm ensures that every output is the result of a final aggregation step (i.e. the Reduce method). For example, if the *count* function is selected, this ensures that every user can contribute by at most 1 to the final output. While this is not enough to guarantee privacy – as we explain in the next section – it offers a first layer of protection and simplifies the design of additional privacy-preserving measures.

Algorithm auditing All algorithms are evaluated by a committee before being installed on the platform. Only system administrators can install an algorithm. Furthermore, if an algorithm needs to use its own privacy module (see below), a special token has to be passed in the request body that is verified against the token in the *Algorithm* service configuration file. This token is made available only to algorithm auditors and, hence, ensures that no algorithm with custom privacy module is added without being audited. The audit phase includes automatic and semi-automatic techniques, such as fuzz testing.

Query monitoring The *Logging service* in the *Endpoints Layer* provides APIs for accessing the valid and the invalid requests made on the system. These logs will serve as base for future query monitoring algorithms to identify any potentially suspicious request or sequence of requests.

Privacy module Every algorithm consists of two components:

1. The *analysis algorithm*, composed of the Map function and Reduce method (see Section 5.2).
2. A privacy module that provides privacy-mechanisms for the query. It is not essential for each algorithm to have a privacy module. If not provided, the platform uses a default noise addition and query suppression mechanism.

The privacy module ensures that the final output to each query does not disclose personal data. This is typically achieved via noise addition, query set size restriction, and other techniques. The privacy module can provide differential privacy [DMNS06] or any other privacy protection that the developer wants to implement. Although solutions for general-purpose privacy-preserving data analytics have been proposed [McS09, PGM14, RSe10, Mea12, Fra17, JNS17], they present limitations for utility, flexibility, or privacy [JNS17, McS18, GHRdM18]. Algorithm-specific techniques can give strong privacy guarantees and yield accurate results, but need to be designed and tuned for each new algorithm. The platform allows every algorithm to include a privacy module specific to that algorithm, allowing developers to achieve a better privacy/utility tradeoff in their algorithms. In the next sections, we present the two algorithms developed for the platform and a privacy module for the **density** algorithm that protects privacy and provides good utility. In particular, the algorithm enforces geo-indistinguishability [ABe13], a variant of differential privacy.

5.2.4 Algorithms on the platform

Two algorithms have been developed and deployed for the platform: the **mobility** algorithm and the **density** algorithm. Those two algorithms present a high utility potential and are compatible with the framework of the platform on both the privacy and computing capabilities (MapReduce) aspects. In the next Section, we will give formal verification of the privacy guarantees for our population **density** algorithm. However, research is still ongoing to give similar guarantees for the **mobility** algorithm as it hasn't been solved yet. This is why, in order to avoid any privacy breach, only most trusted users will be given access to this algorithm on the platform.

The population mobility algorithm

The **mobility** algorithm is used to release the number of users who moved from a certain area at time interval T_1 to another area at time interval T_2 . The algorithm accepts exactly seven parameters from the analyst: *resolution*, *keySelector*, *startDate₁*, *endDate₁*, *startDate₂*, *endDate₂*, *sample*.

The *keySelector* is a list of directed 2-tuple of areas (e.g. *area₁.area₂* which translates to mobility from *area₁* to *area₂*) for which the mobility needs to be computed. Similarly to before, this could be, for example, the ID of a specific borough or the name of a city. The *resolution* parameter selects the spatial resolution of the requested locations. There are three different resolution levels: *borough*, *commune* and *region*. The list of all available areas and corresponding resolution level is made available in the API documentation.

The *startDate*s and *endDate*s parameters specify the time intervals of interest. Both parameters can select any day and any time of the form *hh:m0:00*. Finally, the *sample* parameter is a value between 0 and 1 that specifies the (random) fraction of users sampled by the algorithm to compute the query on. There are four available values: 0.01, 0.1, 0.25, 1. A larger sample parameter yields better accuracy but requires more time to compute, as the platform needs to process more users and more records per user. In our mobility algorithm, sampling is not used to improve privacy guarantees.

The output of the mobility query is a list of $(key, value)$ pairs, where each key is one of the 2-tuple of areas specified in `keySelector` and value represents the number of users who moved from a certain area at time interval T_1 to another area at time interval T_2 . Note that a user might visit multiple locations in the same time interval, but our algorithm adopts a *winner-takes-all* approach, i.e. it assigns each user to at most one area (where the user spent most of the time in the requested interval). A user can thus contribute to the count of only one element in the `keySelector`.

The population density algorithm

The **density** algorithm is used to release the number of users who spent most of their time in a certain area in a given time interval. The algorithm accepts exactly five parameters from the analyst: *resolution*, *keySelector*, *startDate*, *endDate*, *sample*.

The `keySelector` is a list of areas for which the density needs to be computed. This could be, for example, the ID of a specific borough or the name of a city. The *resolution* parameter selects the spatial resolution of the requested locations. There are three different resolution levels: *borough*, *commune* and *region*. The list of all available areas and corresponding resolution level is made available in the API documentation.

The `startDate` and `endDate` parameters specify the time interval of interest. Both parameters can select any day and any time of the form `hh:m0:00`. Finally, the *sample* parameter is a value between 0 and 1 that specifies the (random) fraction of users sampled by the algorithm to compute the query on. There are four available values: 0.01, 0.1, 0.25, 1. A larger sample parameter yields better accuracy but requires more time to compute, as the platform needs to process more users and more records per user. In our density algorithm, sampling is not used to improve privacy guarantees.

The output of the density query is a list of $(key, value)$ pairs, where each key is one of the areas specified in `keySelector` and value represents the number of users that spent most of their time in that area during the specified time interval. Note that a user might visit multiple locations

in the same time interval, but our algorithm adopts a *winner-takes-all* approach, i.e. it assigns each user to at most one area (where the user spent most of the time in the requested interval). A user can thus contribute to the count of only one element in the keySelector.

We denote by $\text{density}(L, T_1, T_2, \rho)$ the output of the **density** algorithm for the location L in the interval $[T_1, T_2]$ with sampling parameter ρ . To simplify the exposition, in the rest of this section we present the details of **density** when a single location L is selected. L denotes a generic geographic area for an arbitrary resolution, and we denote by $\mathcal{L}$ the set of locations at the resolution of L . So, for example, $\mathcal{L}$ could be a set of cities or a set of regions. If keySelector contains a list of locations $(L_1, \dots, L_n)$ that belong to $\mathcal{L}$, the **density** algorithm (including its privacy module) is run independently on each L_i .

5.2.5 Privacy module for **density**

We now present our design of the privacy module for the **density** query. This serves as an example to demonstrate the flexibility of the platform and offers a starting point for the development of other algorithms.

The **density** algorithm is based upon three layers of protection: geo-indistinguishability, query set size restriction, and output noise addition. Each layer relies on a solid set of established literature to maximise the privacy guarantees while that specific combination, and their respective implementations, constitute a novel attempt at tackling the privacy issues for the **density** algorithm in a practical way.

Geo-indistinguishability Geo-indistinguishability (GI) is a formal notion of location privacy [ABe13] to obfuscate single user locations, and is a variant of differential privacy (DP) [DMNS06]. GI is not a property of the data, but rather of a randomised algorithm that takes location points as input (and generally outputs location points). Intuitively, given as input a location x , the output y is determined according to a probability distribution that ensures privacy at a level that decreases exponentially with the distance from x . More concretely, if x is located in Dakar,

y will likely point to a different location within Dakar, but it is extremely unlikely that y will point to a different city. In practice, this can be achieved by adding two-dimensional random noise to the input location, drawn from a specific distribution.

The formal definition of **GI** depends on a free parameter ε that essentially controls how quickly the privacy guarantees vanish for points that are far from the true location. The parameter ε is called *privacy loss*.

Definition 5.1 (ε -geo-indistinguishability) *Let $\mathcal{X}$ be a set of locations and let $\mathcal{A}: \mathcal{X} \rightarrow \mathcal{X}$ be a randomised algorithm. Denote by $d(\cdot, \cdot)$ the Euclidean distance. $\mathcal{A}$ satisfies ε -geo-indistinguishability if, for all $x, x' \in \mathcal{X}$ and $S \subseteq \mathcal{X}$,*

$$\Pr[\mathcal{A}(x) \in S] \leq e^{\varepsilon d(x, x')} \Pr[\mathcal{A}(x') \in S].$$

Definition 5.2 (Planar Laplace distribution) *Let $\varepsilon \in \mathbb{R}^+$ and $x \in \mathbb{R}^2$. The planar Laplace distribution centered at x is the probability distribution on $\mathbb{R}^2$ with pdf*

$$D_\varepsilon(x)(x') = \frac{\varepsilon^2}{2\pi} e^{-\varepsilon d(x, x')}.$$

Consider the mechanism that, on each input x , output x' is drawn from $D_\varepsilon(x)(x')$. It can be proven that this mechanism satisfies ε -**GI**. In practice, the obfuscated location x' is obtained by adding to the true location a noise value sampled from the planar Laplace distribution centered in zero (see [ABe13]).

In **density**, the location associated with every CDR is obfuscated using planar Laplace noise with parameter ε , producing a sanitised database (this method is discussed in [Bor14]). In our implementation, we sanitise the data on-the-fly, using pseudo-random generators, to avoid the need to store a sanitised copy of the database and give more flexibility to developers (see next paragraphs). Nevertheless, to simplify the exposition, sometimes we use the expression “sanitised database” to refer to the database that we would obtain if we saved every user location obfuscated while executing the algorithm.

Query set size restriction The algorithm counts how many users spent most of their time in the selected area. The count is computed on the dataset sanitised with GI. If the count for a certain area is below a fixed threshold B , then the count is suppressed and the value associated with that area is set to `ValueTooLow` in the output.

Output noise addition Finally, `density` adds random noise to every count that is not suppressed by the query set size restriction. The random noise value is drawn from a normal distribution $\mathcal{N}(0, \sigma^2)$, and is sampled using a pseudo-random number generator. The seed is set to:

$$seed = \text{hash}(L, T_1, T_2, \rho, \text{salt}),$$

where `salt` is a long string known only to the data curator but accessible from any algorithm, and the default hash function is SHA-512. This ensures that the noise value is the same for the selected area and time interval, and hence it cannot be averaged out by repeating the same query [FPEM17].

The full algorithm is presented in detail in Procedure `density`. In the algorithm, the *majority element* (or *location*) of a list is the element with most occurrences. If there is a tie, then the majority element is selected at random among the most frequent.

Choice of privacy parameters. The `density` algorithm depends on three parameters: ε , B and σ . As these parameters determine the privacy protections of the mechanism, they must be fixed by the data curator. The defaults are:

- $\varepsilon = 10 \text{ km}^{-1}$. This ensures that the obfuscated locations are statistically indistinguishable with confidence ε from all the other locations within distance d . For the exact meaning of this guarantee, we refer to [ABe13]. In particular, the expected distance between true and obfuscated location is $2/\varepsilon$, i.e. 200 meters for our default choice of ε .
- $B = 50$. We believe that such a high value does not affect utility significantly, as data analysts are not generally interested in precise density values for populations smaller than 50 individuals.

Procedure density($L, T_1, T_2, \rho; \varepsilon, B, \sigma$)

Input: Defined by analyst: location $L \in \mathcal{L}$, time start T_1 , time end T_2 , sampling parameter ρ .

 Defined by data curator: Gl parameter ε , minimum threshold B , noise stdev σ
Output: number of sampled users who spent most of their time in L during $[T_1, T_2]$

```

1  $\mathcal{D} \leftarrow$  fraction  $\rho$  of total users, selected at random
2  $density = 0$ 
3 for  $i \leftarrow 1$  to  $|\mathcal{D}|$  do
4    $\{x_1, \dots, x_n\} \leftarrow$  locations with timestamps of  $user_i$  between  $T_1$  and  $T_2$ 
5   for  $j \leftarrow 1$  to  $n$  do
6      $seed \leftarrow \text{hash}(user_i.time, x_j.loc, \text{salt})$ 
7      $x'_j \leftarrow$  random location drawn from planar Laplace pdf  $D_\varepsilon(x_j)(\cdot)$  seeded with  $seed$ 
8      $l_j \leftarrow$  nearest location from  $x'_j$  in  $\mathcal{L}$ 
9   end for
10  For each 10min interval in  $[T_1, T_2]$  select the majority location  $L_j$  for that interval
11   $\{L_1, \dots, L_m\} \leftarrow$  majority locations, one for each 10min interval
12   $L^* \leftarrow$  majority location of  $\{L_1, \dots, L_m\}$ 
13  if  $L^* = L$  then
14     $density \leftarrow density + 1$ 
15  end if
16 end for
17 if  $density < B$  then
18   return ValueTooLow
19 else
20    $seed \leftarrow \text{hash}(L, T_1, T_2, \rho, \text{salt})$ 
21    $noise \leftarrow$  draw random value from  $\mathcal{N}(0, \sigma^2)$  seeded with  $seed$ 
22   return  $density + noise$ 
23 end if

```

- $\sigma = 10$. Seen as an application of the Gaussian mechanism [DMNS06], this corresponds to the protection level guaranteed by (ξ, δ) -differential with privacy loss $\xi = 0.6$ and parameter $\delta = 2.5 \cdot 10^{-7}$ for a single query, which is generally believed to provide meaningful privacy [DS10]. Note that **density** does not really enforce DP in general, as we do not limit the privacy budget over multiple queries. We discuss this more in detail later on.

The utility of density. The **density** algorithm presents several advantages for utility, especially with respect to accuracy of the outputs and running time. We tested **density** on a real mobility dataset with millions of users and thousands of antennas, using antenna-level resolution and setting $\varepsilon = 10 \text{ km}^{-1}$. We ran **density** across all antennas over 100 different time intervals of length 1 hour, 6 hours, 1 day, 3 days and 10 days. When $\sigma = 0$ (i.e. there is no output noise addition), the average relative error between **density** outputs larger than $B = 50$ and the true density values was between 3-6% for all different lengths. Setting the default $\sigma = 10$, the average relative error goes up to 6-9%. This is because, even for longer time intervals, a significant fraction of the outputs are rather small, but the parameter σ for the Gaussian noise is not scaled to the output. This is needed to provide stronger privacy guarantees [DR13].

As for the computational efficiency, the average run-time overhead due to the **Gl** sanitization is only 3%.

The privacy guarantees of density. The **density** algorithm relies on **Gl** as a formal notion of privacy. The specific choice of **Gl** as an obfuscation method is due, among other things, to some of its mathematical properties. Most notably, it abstracts from the adversary's background knowledge and the total privacy loss grows naturally (linearly) with the number of observed user locations [ABe13]. Additionally, **Gl**'s guarantees are preserved under multiple queries and specifically they are compatible with sampling: running the same query with different sampling parameters does not affect the privacy protection provided by **Gl**.

Moreover, they hold even if we assume that several malicious analysts are colluding, because the output for each query is always the same, independently of the analyst that submits it. Hence

different analysts do not gain any advantage by combining the outputs they obtain (although for colluding attackers it might be easier to bypass the intrusion detection system).

Although inspired by DP, GI is fundamentally different. It is easy to check that simply releasing aggregate statistics computed on a dataset sanitised with GI does not enforce DP. Similarly, the **density** algorithm is not designed to enforce DP. Although the addition of Gaussian noise to each query output ensures (ξ, δ) -DP on that output, the total (theoretical) privacy loss for DP increases linearly with the number of queries. In the current implementation of **density**, we do not limit the privacy budget, hence the theoretical total privacy loss for DP is unbounded. Note, however, that the overall rate of queries per analyst is limited by the platform's *Interface service*.

Applying DP to mobility data provides very strong guarantees, but it is extremely challenging to preserve utility and flexibility. One example is the mechanism by Acs et al. [AC14] to release the population density in Paris. While the proposed solution enforces DP, it presents two important limitations that make it impractical for our use case. First, the data is available only for a period of one week. This allows to pre-sample a limited number of locations for each user, hence improving privacy while preserving good utility. In contrast, our algorithm can be used to query data that spans several months or even years. Second, the density is computed for slots of one hour. To obtain the density across larger time frames, one would then take the sum over the selected one-hour slots. However, this leads to biased estimates for larger time intervals, as the same user may be counted multiple times in different slots.

Attacks. We now present two attacks with different threat models to illustrate strengths and limits of GI and **density** in practice.

Suppose that a user (the victim) was the only user potentially connected in a specific region R for a relatively long time frame. Suppose that this is known to the attacker, but she does not know whether the victim was indeed in R . A completely accurate density query for that time frame would output 0 or 1, revealing to the attacker the presence (or absence) of the victim. Now consider **density** _{$\varepsilon=10, B=0, \sigma=0$} , i.e. the **density** algorithm that enforces only GI with $\varepsilon = 10 \text{ km}^{-1}$, without query set size restriction and output noise addition. This algorithm

already thwarts this attack: thanks to **GI**, the victim’s noisy location might fall in a different area, while the noisy locations of other users might be assigned to R , making the output of **density** _{$\varepsilon=10, B=0, \sigma=0$} useless to the adversary. The probability that the attack is thwarted can be controlled with a rigorous choice of ε that takes into account the geography of the surrounding regions.

As an edge case variation, assume that an attacker has no specific background information about the victim, but she has access to the full dataset sanitised with **GI**, minus the full trace of the victim. Additionally, suppose the attacker can submit an unlimited number of **density** _{$\varepsilon=10, B=0, \sigma=0$} queries to the platform, which runs them on the full dataset (that includes the victim’s record). By submitting many such queries for varying time intervals, the adversary can easily recover the entire victim’s *sanitised* trace. This is done by comparing the platform’s outputs with counts on the dataset without the victim (which differ by 0 or 1). Although the locations inferred by the attacker are obfuscated, obtaining the *full* obfuscated trace (or a significant portion of it) constitutes a risk for privacy [dMHVB13].

A similar scenario is very unlikely, but the design of **density** includes default output noise addition and query set size restriction to mitigate the risks. These two additional measures make the second attack very hard to perform too, although they do not completely rule out all attacks by even more powerful adversaries (i.e. assuming even more detailed background knowledge). **GI** alone does not provide close to perfect privacy like differential privacy, but we employ **GI** as a measure of risk in worst-case scenarios. Together with output noise addition, query-set size restriction and the platform’s defense-in-depth measures, we are convinced that **density** prevents inference attacks in realistic, non-pathological settings.

5.2.6 Related work

A large range of attacks on query-based systems have been developed since the late 70’s [Den78, Bec80]. Most of these attacks show how to circumvent privacy safeguards (e.g. query set size restriction and noise addition) in specific setups. In 2003, Dinur et al. [DN03] proposed the first example of an attack that works on a large class of query-based systems. Since then,

numerous other attacks have been proposed in the literature. These attacks address different limitations of previous ones, particularly the computational time required to perform them. A recent survey from Dwork et al. [DSSU17] gives a detailed overview of attacks on query-based systems.

Privacy research has been increasingly focused on providing provable privacy guarantees to defend query-based systems against such attacks. However, the development of a privacy-preserving platform for general-purpose analytics is still an open problem [JNS17].

General-purpose analytics usually refers to systems that allow analysts to send many queries of different type, using a rich and flexible query language. Some solutions based on differential privacy have been proposed, the main ones being PINQ [McS09], wPINQ [PGM14], Airavat [RSe10], and GUPT [Mea12]. All of these systems however present limitations [JNS17], e.g. in simplicity of use for the analyst, which must provide additional query parameters that interface with the differential privacy implementation. In particular, Airavat is based on MapReduce like this platform and enforces differential privacy by using a simple application of the Laplace mechanism [DMNS06]. Like other general differentially private mechanisms, a straightforward application of the Laplace mechanism often destroys the utility of the data for multiple queries [DR13]. Specifically, every aggregation method supported by the platform (count, sum, median) could be easily made differentially private with the standard Laplace or exponential mechanisms [DMNS06, MT07], but this solution would require to add a lot of noise to outputs in order to provide meaningful guarantees.

In 2017, Johnson et al. [JNS17] proposed a new framework for general-purpose analytics, called FLEX. FLEX enforces differential privacy for SQL queries without requiring any knowledge about differential privacy from the analyst. However, the actual utility achieved – level of noise added – by the current implementation of FLEX has been questioned [McS18]. Diffix, a patented commercial solution that acts as an SQL proxy between the analysts and a protected database [FPEM17, FPEO⁺18] has recently been proposed as an alternative to differential privacy. However, Diffix’s anonymization mechanism has been shown to be vulnerable to some re-identification attacks [GHRdM18].

5.3 Discussion and future work

The current implementation of the architecture presents some limitations.

System.

There is currently no live ingestion of new data, as the data gets loaded periodically in bulk. This limits the platform's real-time capabilities to, e.g., monitor a crisis and provide adequate information for search and rescue parties. However, this could be easily solved by bridging the platform with the CDR databases of the telecommunication companies as there is no technical limitation on the platform itself. The current implementation is targeted to scale up to the medium-sized countries (e.g. up to 50M people). In order to scale up to larger countries, further work would be required on the database side to ensure the efficient storage and retrieval of the data across the different workers and further improve the computation performances in the sandboxed environment. Furthermore, the platform management either requires the admin to go through the platform API or directly connect to the physical servers. A more administrator-friendly set of tools could be implemented to facilitate the platform monitoring (status of the services and privacy alerts). On the analytics side, the move from antenna locations to a geographic grid (e.g. Voronoi tessellation) would enable to add a geo-semantic layer (forest, rural/urban, etc) and allocate presence probability. Also, it would address concerns of some telecommunication companies which are not authorised to provide antenna locations.

Privacy.

Strict cache, such as the one currently in place, presents a higher surface of attack compared to a cache capable of query similarity. That query similarity engine could as well detect unusual query patterns and be used as another defense against potential attacks from malicious users. A more generic approach to privacy on the platform is another challenge. At the moment, generic noise addition, strict caching and query-set size restriction offer only a basic layer of protection. Privacy is mainly provided by the privacy module, which is algorithm-specific. General privacy-preserving mechanisms that apply automatically to the outputs of any query could simplify the development of new algorithms. However, preserving good utility for general-purpose analytics is still an open research problem (see section 5.2.6).

Chapter 6

Analytics Developed using the eTRIKS Analytical Environment

6.1 Analytics for tranSMART

To illustrate how the eTRIKS Analytical Environment can be used for managing and analysing large scale translational research data in tranSMART, we implemented three bioinformatic analysis pipelines, including a) an iterative model generation and cross-validation pipeline for biomarker identification, b) a general statistical analysis pipeline for hypotheses testing, and c) a pathway enrichment pipeline using KEGG to demonstrate the performance of the proposed architecture. Unlike the two others, the pathway enrichment pipeline forms a sub part of the iterative model generation pipeline or a pipeline in its own right.

Each one was implemented in the same fashion: the code was prototyped locally in a container (to ensure that the code operated as expected using a subset of the data or a smaller number of iterations) and the full computation was then submitted to the central clusters. All these workflows were designed to be highly parallelisable and, in order to enable their seamless scalability, Spark has been chosen for their implementation.

6.1.1 Iterative Model Generation and Cross-validation Pipeline

This pipeline is a generalization and a more robust approach than traditional ones used in translational research (for example in the context of identifying the gene signature for stage II colon cancer [KBe11]). Commonly, translational research approaches focus on a specific algorithm and published work to reduce the scope of genes in the analysis which inherently creates a bias. In addition, it is not always certain that a cross-validation has been carried out during the analysis to test the robustness of the model found. The iterative model generation and cross-validation pipeline, as described in Figure 6.1, aims at providing a robust and unbiased method for biomarker identification thanks to a massively parallel computational approach. Furthermore, the pipeline implementation impressively illustrates the seamless scalability of the eAE. Using the eAE, the pipeline scales at the same rate as the underlying hardware, a crucial aspect given the massive amounts of data involved.

In clinical trials, collecting further samples may be hazardous, costly or even impossible to do. In these cases, cross-validation is a powerful approach to prevent from testing hypotheses suggested by the data (called “Type III errors” [Mos48]). Cross-validation is a model validation technique for assessing how a statistical or computational model will generalise to an independent data set. It is mainly used in settings where prediction is the main objective and one aims to estimate how good a predictive model is in practice. In a prediction problem, a model is normally given a data set of known data on which the training is run (i.e., training set), and a data set of unknown data, or first seen data, against which the model is tested (i.e., testing set). To reduce variability, different partitions of the data can be defined and multiple rounds of cross-validation can be performed on them.

In practise, many statistical and computational approaches can be used for model generation. Possible candidates are (but not limited to): linear or nonlinear Support Vector Machine (SVM), Logistic Regression, Linear Regression, Alternating Least Squares, Lasso, etc. Non-parallelisable algorithms can be used as well in this context. Indeed, instead of distributing subsets of the data across different nodes (First type of parallelization) to parallelise the computation of the model, we can distribute different sets of parameters (Second type of paral-

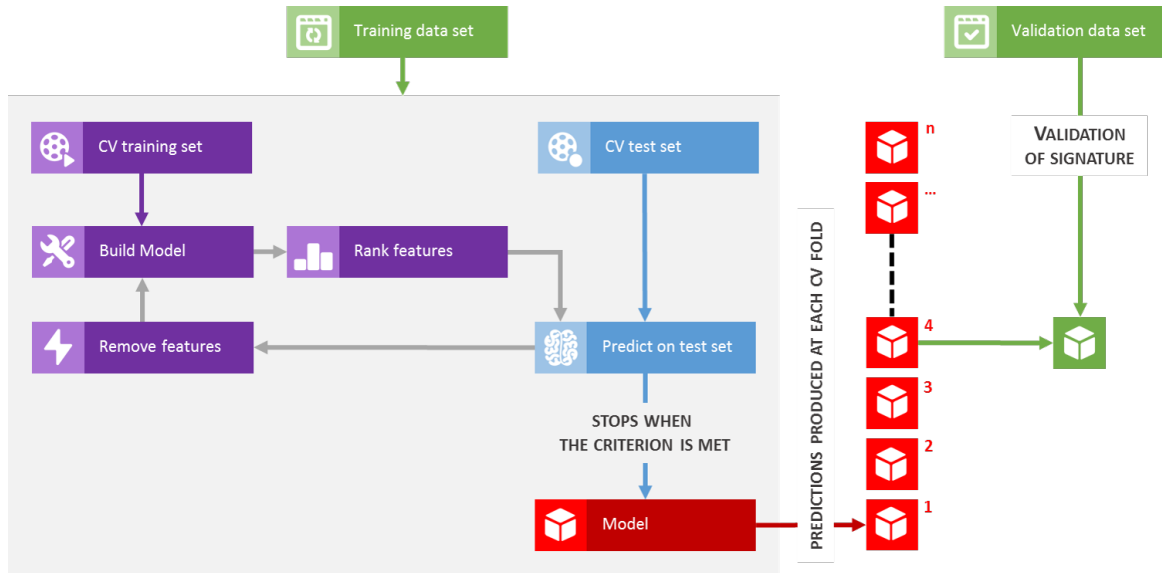


Figure 6.1: Iterative model generation and cross-validation pipeline

lization) on every worker to generate a different model every time.

The drawback with the second type of parallelization compared to the first one is that, when dealing with large datasets, the working nodes need to be equivalently large and the network might become a bottleneck performance wise.

This pipeline allows us further scaling up by distributing the computation to multiple clusters which work independently to generate models. Each cluster randomly samples the training set and starts generating models using the selected algorithm. Each model is then tested against the test set to evaluate its fitness according to the specified set of indicators. With the increasing number of iterations, it is expected that the model will converge to the optimal solution.

Biomedical data always consist of a large number of features, i.e., individual measurable properties that describe the observed phenomenon. To find the best-fitted model, with the least amount of bias, we generate a family of models using the same dataset with different selections of features. Once the model is built, a certain amount of features can be removed, and a new model is generated using the remaining features (Figure 6.1). An unbiased approach is to randomly remove a selected number of features and then check whether it improves the fitness of the model. If the metrics remain the same then we move to the next iteration, if not then we select another set of features to remove. A fixed small number of features removed at every

recursion will prevent any overlook, but it is computationally expensive. A relative step, e.g., a percentage of the total number of features, will greatly speed up the process.

A more efficient approach is to introduce a very small amount of bias by lowering the chances of a feature, which we know beforehand is a factor, to be removed. By introducing this, the generation of models will naturally tend towards the optimal solution much faster. Another option, as suggested by Vladimir Vapnik's group [GWBV02] in the context of gene selection, is to use the weight magnitude as ranking criterion. Compute the ranking criteria: $c_i = (w_i)^2$ for all i and find the feature with smallest ranking criterion $f = \operatorname{argmin}(c)$.

The scoring used to access the fitness of models can be done through a wide variety of measures, such as Area Under the Receiver Operating Characteristic (ROC) Curve - AUC, Sensitivity or True Positive Rate (TPR), Specificity or True Negative Rate (TNR), Negative predictive value (NPV), Positive predictive value (PPV) and F1-score. There is no such metric that can give the right answer for every situation. These metrics can only eliminate obvious "failures" due to performance, complexity, similarity to other models developed in similar ways and general stability. In the case of multiple thresholds with largely identical performance, the Hazard Ratio (HR) from Cox proportional hazards regression [CHe09] can be used as a tiebreaker to favour higher HR values.

Once that first selection is done by each individual cluster, we gather them into a NoSQL cache where other services take over. The role of this section is to further narrow down the candidate models by comparing the models against one another. Then, we enrich the results to give back as much information as possible to the scientist. It is thus important to implement different types of metrics to enable data scientists to select the right models for further extensive assessment or biological validation. Pathway enrichment is a method to identify classes of genes or proteins that are over-represented in a large set of genes or proteins and may have an association with disease phenotype. Applying a pathway enrichment using Kyoto Encyclopaedia of Genes and Genomes (KEGG) [KG00] and Gene Ontology (GO) [ABe00] with multiple test corrections (Bonferroni, Holm-Bonferroni and/or FDR) on a sub selection of high scoring models can provide another insight to the results and an additional quality check to every model generated.

This type of unbiased approach to model generation is not well supported on standard platforms. The reason is that model generation is a long-running computation and, the longer a computation runs, the more likely is a crash and loss of intermediate results. Indeed, the computations could be running for days at a time thus putting a lot of pressure on the hardware and software with no possibility to create milestones, a mechanism that would help to prevent full recomputation in the event of a crash. To address the issue of long-running computations crashing, the eAE leverages the versioning mechanisms of Spark and the temporary models are stored in MongoDB for later evaluation and provenance if the final model is not satisfactory. Each model is stored with the associated parameters and performances. The necessity to develop this mechanism gave the idea for what would later become TensorDB (see Chapter 4.3). The integration of Spark with the additional eAE layers on top enables users to run these large scale compute intensive experiments extremely easily and seamlessly through the end point of their choice. The stability, robustness, and fault-tolerance of the platform enables these computations in a high-performance fashion thanks to the fact that, even if a physical machine or a worker fails, the tasks get automatically rescheduled, thereby avoiding having to rerun the entire computation. The integration of Docker, Jupyter, Toree ¹ and eAE parts have enabled users to implement and prototype their algorithms efficiently without having to deal with how to set it up or link all the pieces together. Finally, once the algorithm is ready, no modification is needed and the submission to the centralised cluster is straightforward.

6.1.2 General statistics

The general statistical analysis pipeline aims at providing statistical insights about the datasets for further research, without any prior statistical knowledge, by performing multiple statistical tests on a given data set. Statistical methods test scientific theories when observations, processes or boundary conditions are subject to stochasticity. Performing multiple tests on the same data set at the same stage of analysis increases the chance of obtaining at least one invalid result. The benefit obtained from performing statistical methods across whole datasets,

¹<https://toree.apache.org/>

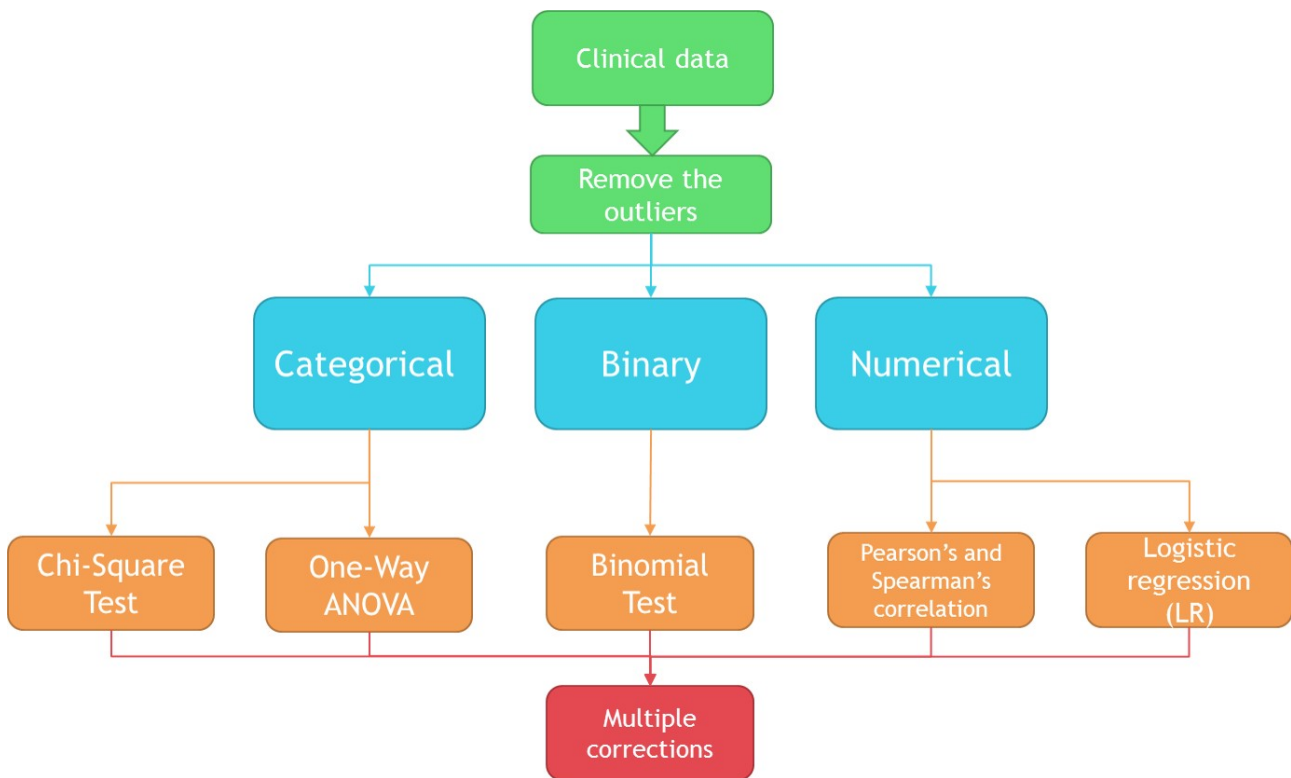


Figure 6.2: Modelling of the pipeline for an unbiased approach to statistical testing of whole datasets.

however, far offsets this drawback.

The first step of this pipeline is to divide the data into their basic data types: numerical, binary, categorical and unknown. The unknown category holds any column with three or less valid data points and any irrelevant data (e.g. phone numbers and free text). This data is not discarded as it might be used to extract insights at a later stage. The numerical data is then sub divided again by normality. Two methods are used to determine if a variable follows a normal distribution: Shapiro-Wilk's tests and Anderson-Darling's test. The variable is tagged as normally distributed only if both tests yield a positive answer. Applicable statistical methods are then applied to the data in each category.

For the categorical data, the χ^2 test is extensively used for accessing the associations between different clinical variables. The χ^2 test determines whether there is a significant difference between the observed frequencies and the expected frequencies in one or more categories. If one variable is categorical and one is numerical, the analysis of variance (ANOVA) test is used to provide a statistical test of whether or not the means of several groups are equal, which

comes as a complement to the χ^2 test. For the binary data, the binomial test has been chosen, which is an exact, two-sided test of the null hypothesis that the probability of success in a Bernoulli experiment is p (we chose $p = 0.5$).

We use two methods to analyse the relationships between numerical variables, i.e., logistic regression (LR) analysis and correlation analysis. LR has been successfully used to identify independent predictors of prostate cancer to improve diagnose accuracy [VGKS99]. LR models built on the regression fit on probabilistic odds between comparing conditions requires no specific distribution assumption (e.g., Gaussian distribution) but is often shown to be less sensitive than other approaches. For correlation analysis, we choose Spearman and Pearson correlation. The Spearman's correlation between two variables is equal to the Pearson's correlation between the rank values of those two variables; while Pearson's correlation assesses linear relationships, Spearman's correlation assesses monotonic relationships. If there are no repeated data values, a perfect Spearman correlation of $+1$ or -1 occurs when each of the variables is a perfect monotone function of the other. Bonferroni's correction is used for multiple corrections.

6.1.3 Pathway Enrichment

Gene expression data is usually interpreted using gene set enrichment analyses based on the functional annotation of the differentially expressed genes. This is effective for uncovering if the differentially expressed genes are associated with molecular function or a certain biological process. The Gene Ontology (GO) [OGS⁺99], containing standardised annotation of gene products, is often used for this purpose. This analysis is carried out by comparing the frequency of individual annotations in the gene list with a reference list (usually all genes on the microarray or in the genome).

To help the scientists to select the right model after the list of genes has been outputted by the Iterative Model Generation pipeline or simply do a pathway enrichment using a list of genes obtained from another analysis, a pathway enrichment using Kyoto Encyclopedia of Genes and Genomes (KEGG) [OGS⁺99] and Gene Ontology (GO) with multiple test corrections (Bonferroni, Holm-Bonferroni and/or FDR) on the sub selection of high scoring models has been

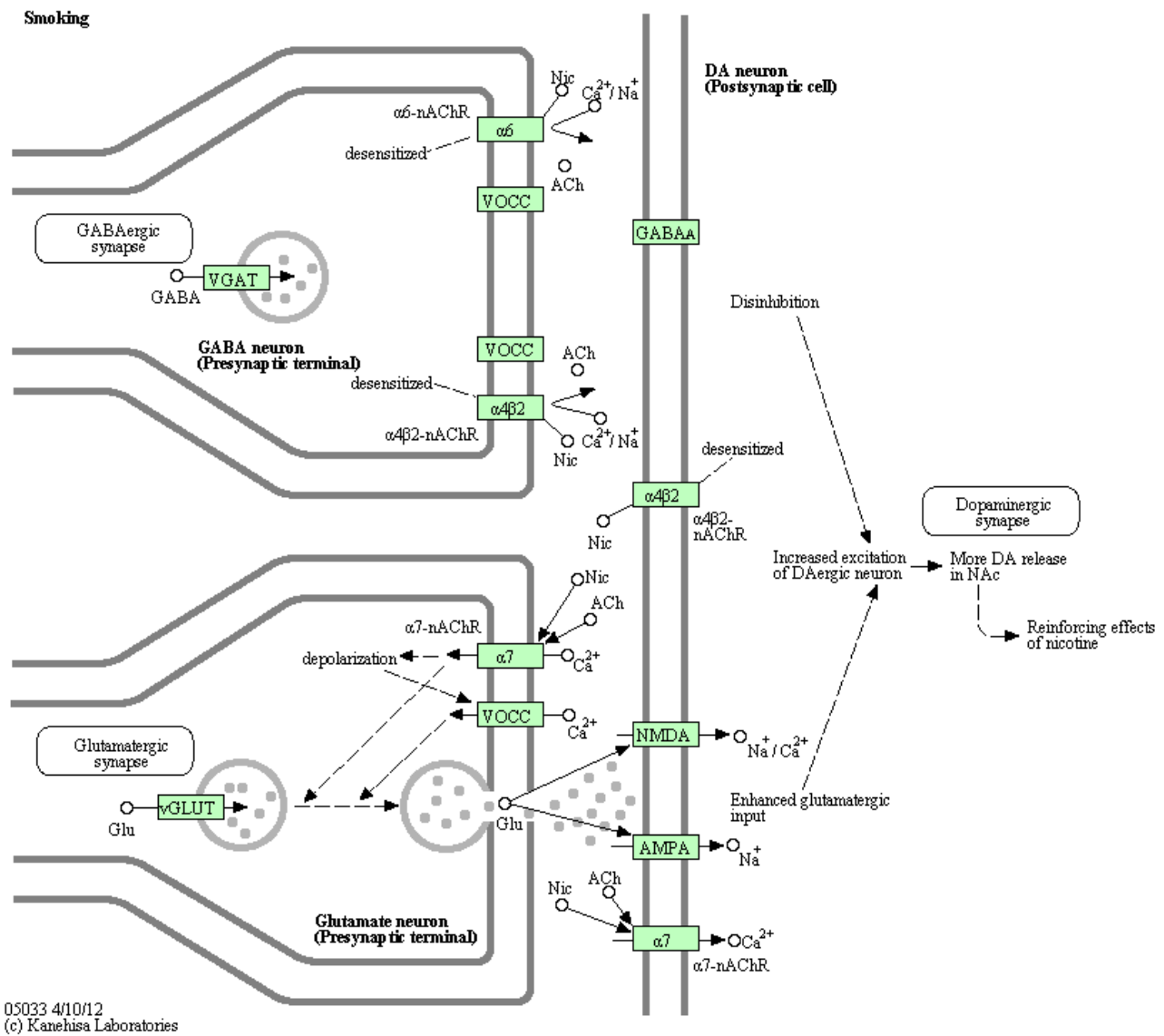


Figure 6.3: Illustration of a KEGG disease pathway with the differentially expressed genes associated with smoking.

implemented. The enrichment is done using a two-sided Fisher’s exact [Fis35] test after building the associated contingency table. Those enrichments add another insight to the results and an additional quality check to every model generated. The Spark re-implementation facilitates the large scale enrichments while building the models concurrently, unlike traditional platforms that wait for the final model to be outputted before performing the enrichment. Moreover, the number of pathways and their complexity gradually increases which requires more and more compute power in order to do a single analysis. The proposed implementation offers a way to overcome that issue and opens the possibility to researchers to integrate that procedure to any of their pipelines, thus improving the quality of their research seamlessly.

6.2 DeepSleepNet

This work was carried out in collaboration with researchers specializing in sleep disorders from the Data Science Institute. This example highlights the new possibilities and scalability benefits researchers in the field of distributed deep learning gain by leveraging the eTRIKS Analytical Environment and TensorLayer. It proposes a new deep learning model, named DeepSleepNet [SDWG17], for automatic sleep stage scoring based on raw single-channel EEG.

6.2.1 Introduction

Sleep plays an important role in human health. Being able to monitor how well people sleep has a significant impact on medical research and practice [WGWF10]. Typically, sleep experts determine the quality of sleep using electrical activity recorded from sensors attached to different parts of the body. A set of signals from these sensors is called a polysomnogram (PSG), consisting of an electroencephalogram (EEG), an electrooculogram (EOG), an electromyogram (EMG), and an electrocardiogram (ECG). This PSG is segmented into 30-s epochs, which are then classified into different sleep stages by the experts according to sleep manuals such as the Rechtschaffen and Kales (R&K) [AH69] and the American Academy of Sleep Medicine (AASM) [IACe07]. This process is called sleep stage scoring or sleep stage classification. This manual approach is, however, labor-intensive and time-consuming due to the need for PSG recordings from several sensors attached to subjects over several nights.

The new approach proposes a model for automatic sleep stage scoring based on raw single-channel EEG by utilizing the feature extraction capabilities of deep learning. The architecture of DeepSleepNet consists of two main parts as shown in Figure 6.4. The first part is representation learning, which was trained to learn filters to extract time-invariant features from each raw single-channel EEG epochs. The second part is sequence residual learning, which was trained to encode the temporal information such as stage transition rules from a sequence of EEG epochs in the extracted features.

Figure 6.4 details the specifications of the hidden size of forward and backward Long Short-

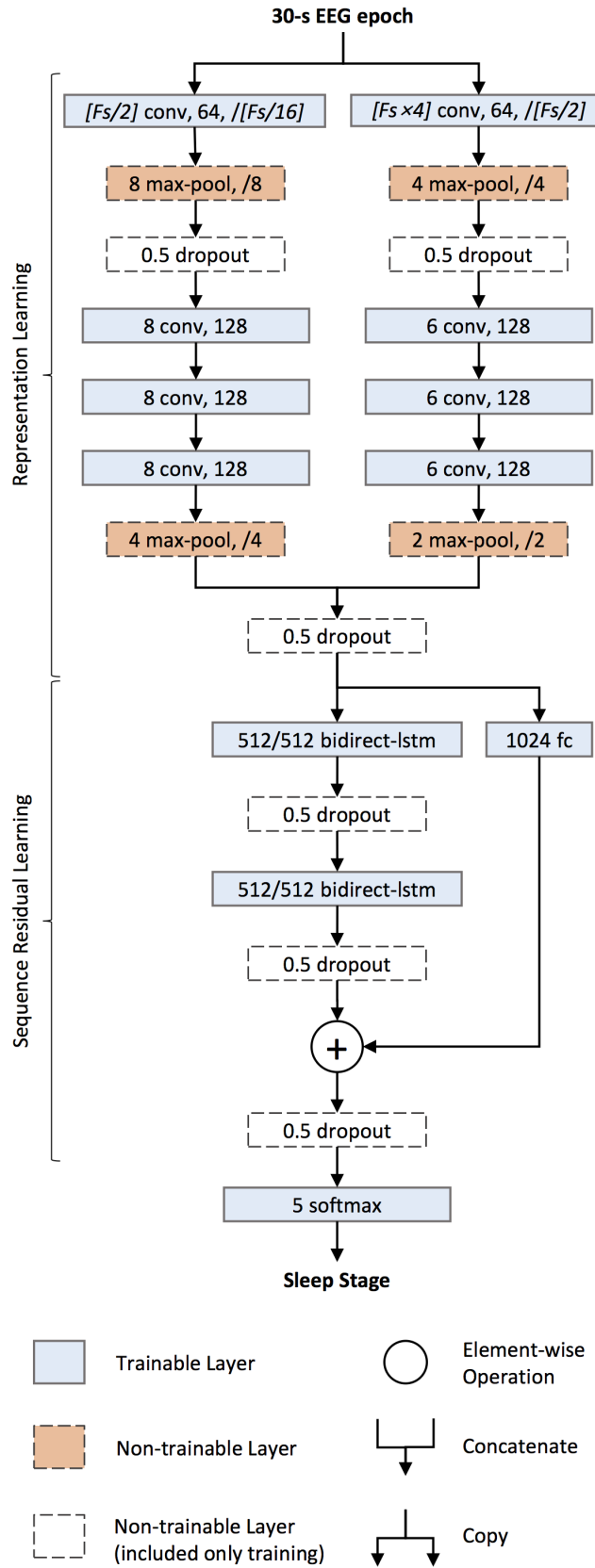


Figure 6.4: An overview architecture of DeepSleepNet from Supratak et al. [SDWG17] consisting of two main parts: representation learning and sequence residual learning. Each trainable layer is a layer containing parameters to be optimised during a training process. The specifications of the first convolutional layers of the two CNN depends on the sampling rate (F_s) of the EEG data.

Term Memories (LTSMs) along with the fully-connected layers. The *fc* block shows a hidden size, while each *bidirect-lstm* block shows hidden sizes of forward and backward LSTMs. For the representation learning part, we followed the guideline provided by Cohen et al. [Coh14] for capturing temporal and frequency information from the EEG.

6.2.2 Tackle class imbalances

Models built on large sleep datasets often suffer from class imbalances issues (i.e., learning to classify only the majority of sleep stages). In order to prevent this from happening, a two-step training algorithm (see Algorithm 1 in [SDWG17]) was developed as a solution to effectively train the model end-to-end via backpropagation and prevent the model from suffering class imbalance problem. The representation learning part of the model is first pre-trained by the algorithm with a class-balanced training set (to avoid any overfitting on the majority of sleep stages), then the algorithm fine-tunes the whole model using two different learning rates. The fine-tuning step encoded the stage transition rules and was trained on the sequence training set using a mini-batch Adam optimiser with the two different learning rates. The class-balanced training set was obtained by oversampling the minority sleep stages in the original training set until all sleep stages have the same number of samples. A cross-entropy loss was used to quantify the agreement between the predicted and the target sleep stages in these training steps. The last layer in the DeepSleepNet architecture (see Figure 6.4) is a combination of the softmax function and the cross-entropy loss which are used to train the model to output probabilities for mutually exclusive classes.

6.2.3 Results

Data and Performance Metrics

Evaluation of the model against other datasets is important in order to access the quality of the model. The evaluation of the model was done using different EEG channels from two public

datasets: Montreal Archive of Sleep Studies (MASS) [OGCN14] and Sleep-EDF [GAG⁺00, KZT⁺00].

The performances of the model were measured using per-class precision (PR), per-class recall (RE) and per-class F1-score (F1). The per-class metrics are computed by selecting a single class as a positive class and then combining all other classes as a single negative class.

Initial Experiments

Initially, experiments to determine the design of the architecture and the parameters for DeepSleepNet have been conducted with the first fold of the 31-fold cross-validation using the MASS dataset. For each model architecture, several configurations have been explored such as increasing/decreasing convolutional layers, changing the number of filters, the stride sizes, changing the number of hidden sizes in the bidirectional-LSTMs and the fully-connected layer. The architecture in Figure 6.4 gave the best performance.

Experimental Design and Implementation

The implementation of DeepSleepNet is an illustration of the user-friendliness for highly parallelisable computation and support for hardware accelerators (GPU in this instance) using the eAE. The eAE allows users to seamlessly and quickly configure and launch the complex training of multiple models concurrently.

Project implementation had two major constraints: the first one was that the project involved three different researchers working concurrently on the workflow, and the second one was that the GPU cluster was only available outside of working hours. To make things even more complicated, the GPU cluster had to be switched between Windows and Ubuntu daily (except on weekends). This switch happened forcefully in an automated fashion interrupting the computations in the morning. Thus, it was of paramount importance that the eAE restarted the compute services seamlessly and reschedule interrupted jobs whenever a resource becomes available.

In order to overcome those constraints and develop the new workflow, an eAE Jupyter container was deployed on a specific box with two GPU resources available to enable the researchers to prototype their workflow simultaneously, share their code with one another seamlessly and access their data. To overcome the limited availability of the GPU cluster, the eAE's computes were deployed in containers that were configured to restart whenever the machine boots in Ubuntu. If the reported health of a compute node was unavailable, upon restart of the container, the eAE would commandeer the host machine of the failing container and attempt to restart up to three times the container automatically to ensure the good health of the cluster.

The eAE has proven instrumental in tackling the class imbalance by providing an easy way to test different hypotheses quickly. The class imbalance has proven a challenging issue and different approaches were necessary to appropriately develop the two-step training algorithm. The data provenance has greatly helped the researchers in versioning the different steps of the algorithms and trace the associated data and parameters used at the time to evaluate the quality of the model and algorithm. Furthermore, the optimization of the two learning rates for the whole model has required extensive grid search. This step was facilitated by the eAE by providing easy access to a large pool of compute resources directly from the development environment of the researchers.

In order to build and assess the quality of the model, a k -fold cross-validation scheme was used on the eAE, where k was set to 31 for the MASS and Sleep-EDF datasets. In each fold, we used recordings from 60 subjects to train the model and use the two remaining subjects to test the model. This process is repeated 31 times so that all recordings are tested. Finally, we combine the predicted sleep stages from all folds and compute the performance metrics. We ran a large number (several hundred) of 31-fold cross-validation iterations for hyperparameters tuning of the model and various experiments. Each cross-validation task takes roughly 6-7h to execute and the total execution time consequently is 170.5 hours. The eAE enabled the researchers to schedule during the day two iterations to run every night without any intervention necessary as the tasks would be triggered as soon as the compute nodes become available thanks to the eAE's scheduler and management services. Then, we combined the predicted sleep stages from all folds and computed the performance metrics.

The only alternatives to this would be either to schedule the tasks on each machine individually or sequentially on one machine. The former is tedious and far from practical as one needs to give the user access to all machines, and the latter simply takes too long. The eAE provides a user-friendly web UI, which allows to train multiple models with different configurations concurrently across a cluster of high-performance machines. The scheduling of these tasks through the eAE takes approximately 2-3 minutes compared to an hour if done manually. Another benefit is the possibility to queue jobs to be run once machines become available. For this experiment, the GPU resources were only available at night as they were used for other projects during the day. The option to schedule two iterations at a time for 31-fold cross-validation tasks to be run during the night, without any external intervention, is a key feature for the timely delivery of DeepSleepNet. In the case of this workflow, the experiments span almost an entire year and were made possible only thanks to the eAE.

Sleep Stage Scoring Performance

Tables 6.1 and 6.2 show confusion matrices obtained from the cross-validation from the MASS and Sleep-EDF datasets respectively. Fpz-Cz channel yielded the best performance when compared with the Pz-Oz channel from the Sleep-EDF dataset, thus we did not include the confusion matrix obtained from the Pz-Oz channel. Each row and column represent the number of 30-s EEG epochs of each sleep stage classified by the sleep expert versus the model respectively. The numbers in bold indicate the number of epochs that were correctly classified by the model. The last three columns in each row indicate per-class performance metrics computed from the confusion matrix.

The poorest performance came from the stage N1, with the F1 less than 60, while the F1 for other stages were significantly better, with the range between 81.5 and 90.3. It is also important to notice that the confusion matrix is almost symmetric via the diagonal line (except for the pair of N2-N3), which indicates that the misclassifications were less likely to be due to the imbalance-class problem. Figure 6.5 presents of a manually scored hypnograms against an automatically scored one done by the DeepSleepNet model for Subject-1 from the MASS dataset.

	Predicted					Per-class Metrics		
	W	N1	N2	N3	REM	PR	RE	F1
W	5433	572	107	13	102	87.3	87.2	87.3
N1	452	2802	827	4	639	60.4	59.3	59.8
N2	185	906	26786	1158	499	89.9	90.7	90.3
N3	18	4	1552	6077	0	83.8	79.4	81.5
REM	132	356	533	1	9442	88.4	90.2	89.3

Table 6.1: Confusion matrix from Supratak et al. [SDWG17] obtained from the cross-validation on the F4-EOG channel from the MASS dataset

	Predicted					Per-class Metrics		
	W	N1	N2	N3	REM	PR	RE	F1
W	6614	745	181	81	306	86.0	83.4	84.7
N1	295	1406	631	30	442	43.5	50.1	46.6
N2	391	618	14542	1473	775	90.5	81.7	85.9
N3	29	9	291	5370	4	77.1	94.2	84.8
REM	360	457	419	7	6474	80.9	83.9	82.4

Table 6.2: Confusion matrix from Supratak et al. [SDWG17] obtained from the cross-validation on the Fpz-Cz channel from the Sleep-EDF dataset

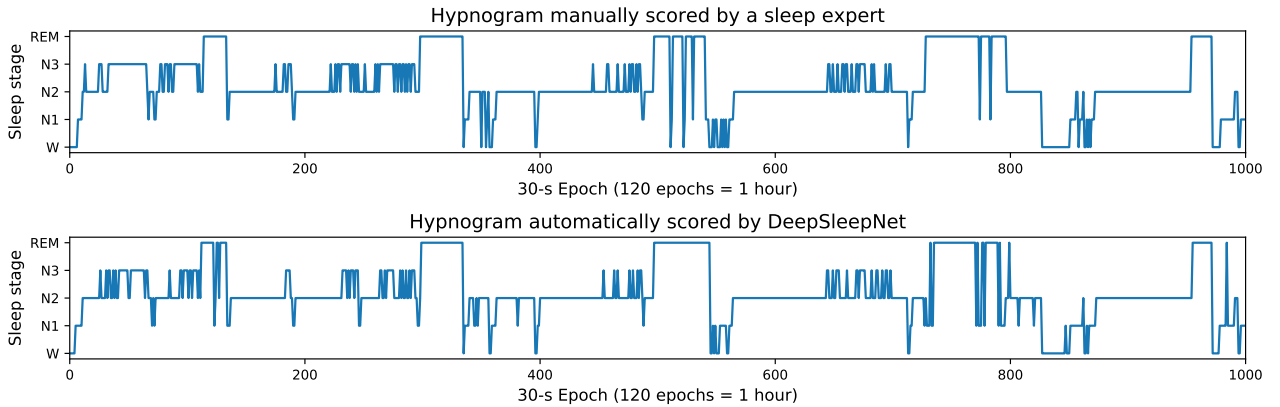


Figure 6.5: Examples from Supratak et al. [SDWG17] of the hypnogram manually scored by a sleep expert (top) and the hypnogram automatically scored by DeepSleepNet (bottom) for Subject-1 from the MASS dataset.

Conclusion

The results demonstrated that the model could flexibly be applied on different EEG channels (F4-EOG, Fpz-Cz and Pz-Oz) without any change both in the model architecture and the

training algorithm. Also, the model achieved similar overall accuracy and macro F1-score compared to the state-of-the-art hand-engineered methods on both the MASS and Sleep-EDF datasets despite having different properties such as sampling rate and scoring standards (AASM and R&K). It is interesting to note that temporal information learned from the sequence residual learning part helped improve the classification performance. We can conclude that the proposed model was capable of automatically learn features for sleep stage scoring from different raw single-channel EEGs. This work has moved us one step closer to the possibility of remote sleep monitoring from home environments which would be less costly and less stressful for the patients and at a larger scale than current hospital setups. Remote monitoring could potentially help elder people and people with stress or sleep disorders on a daily basis and doctors to easily follow up on their patients.

Conversely, the eAE has benefited as well from that close collaboration with the DeepSleepNet project. Firstly, the researchers have provided valuable feedback on the user experience side of the first implementation of the eAE. That feedback has been included in the design of the architecture bringing more value to the users. Secondly, this project acted as a test-bed for validating the architecture and identify shortcomings of the implementation which have been subsequently addressed. Finally, the integration and optimization of TensorDB and TensorLayer into the architecture was only made possible thanks to the collective efforts of the group. As this use case illustrates, all those innovations have opened the way for better science and continuous deep learning to build better applications.

6.3 Characterizing Political Deception On Twitter

In this section, we will present the work done in the context of the identification of deceptive news, the identified relevant features and an ensemble model built from those features to facilitate the automatic labelling of news tweets. This work is a continuation of preliminary work done in [AOMS17a] and has been submitted for publication to IEEE Access for publication at the time of writing.

6.3.1 Background

Political fake news have become a major challenge of our time, and flagging them successfully is a main source of concern for publishers, governments and social media. Our approach is focused on Twitter, and in this project we aimed at finding characteristic features that can help in the process of automating the identification of tweets containing fake news. In particular, we look into a dataset of four months-worth of tweets related with the 2016 US presidential election.

Even if the term *fake news* —understood as deliberate misleading pieces of news information— reached the mainstream in the 2016 electoral campaign in the United States, the phenomenon, and in particular the worries on how it affects people’s beliefs and perceptions, have been present every time a new technological breakthrough in communications comes along [AG17, PR17].

However, the identification of the nature of fake news is a challenge of its own. There is no official definition to define precisely what is a fake news either from a legal, philosophical or conceptual standpoint. Many governments, such as the ones from France and the UK, are currently drafting laws which would try to legally define and fight against fake news. The debate regarding those laws is at a stalemate as nobody agrees on a general definition as the perception of what is a fake news varies a lot between people and any restrictive law would be regarded as threat against freedom of speech.

This shift in perception is more acute when the time variable is taken into account: time may show that what was considered a genuine and honest real news turn out to be false. This transformation could be caused by an evolution of the public opinion, a new discovery (scientific discovery for example) or a new piece of information that was unavailable before. The impact of perception on the classification of a news as fake or not is especially strong as it is subject to emotions and not just rational thinking.

The introduction of cheap and improved presses and advertising business models allowed newspapers to increase their reach dramatically. Partisans and ideologues as well as ‘entrepreneurs’, were among the many beneficiaries that, by adopting such innovations and promoting sensational and fake news stories, managed to cheaply and effectively spread their message and

increase sales [Iof17, Tho17].

The 20th century has seen new technologies such as radio and television further enabling the distribution of content, raising fears that whomever controlled these platforms could influence public opinion. More recently, the emergence of the internet connectivity together with the proliferation of online social networking sites have dramatically reduced publishing and distribution costs, increased the diversity of viewpoints, but also pushed out significant editorial filtering and fact-checking [Set17].

While technology facilitating the distribution of fake news has evolved, strategies and motives behind their consumption, production and distribution appear to remain constant. For instance, in relation to consumption, Pennycook and Rand [PR17] explain that the dominant psychological explanation for why individuals fall for fake news is that previous exposure to fake news primes individuals to become familiar with them and repetition reinforces previous beliefs about such news.

With respect to production, the online information ecosystem is particularly fertile ground for sowing misinformation. Fake news outlets follow strategies such as mimicking names of legit news outlets, and mixing factual articles with partisan slang and fake content. Social media can be easily exploited to manipulate public opinion thanks to the low cost of producing fraudulent websites and high volumes of software-controlled profiles or pages, known as social bots [SCe17]. Also, fake news outlets tend to disappear more often but their content is shared like it is coming from trusted sources. And although their analysis is limited to a sample of online data, anecdotal evidence gathered by Thompson [Tho17] seems to support these findings for different periods of time. In relation to motives, Allcot and Gentzkow [AG17] argue *‘there appears to be two main motivations for providing fake news. The first one is pecuniary [...]. The second motivation is ideological’*. Regularities in strategies and motives together with the abundance of data available suggest the possibility of identifying features to automatically tag fake news outlets. As such, the aim of our work is to put forward a simple framework that facilitates identifying fake news content in online social media. Many attempts have been made at characterising fake news, most recently and famously was the

one published in Science [VRA18] where they leveraged external sources to automatically label tweets. However, this approach presents many limitations. First, this approach only works with news that contain specific URL that can be traced back and checked against those verification websites which drastically reduces the scope. The second issue is that there is no guarantee that those external websites can always be trusted as well. It would be easy for them to suddenly shift the classification of a news to better align with one of their interest. To solve this, complicated and very advanced NLP techniques would be required to parse the text automatically, extract the informations and then check the veracity of the facts against verified data.

Though theoretically sound, this approach has a serious caveat: current NLP techniques and automatic fact-checking are far from being fully developed. With this path still closed, researchers are forced to look for alternatives or simplifications.

As related by several authors [Pol12, PR17, FNR17], previous exposure to fake news makes individuals become more familiar with the piece of information they were exposed to. This familiarity makes it more likely for individuals to believe a previously seen fake news item the next time they encounter it. Moreover, as reported several times [AG05, AG17, BMA15, BFJ⁺12], fake news articles tend to include more polarizing content. Putting these two elements together, we hypothesise that efforts aiming to identify fake news should look for large levels of exposure and large levels of polarization. To validate our hypothesis, we use a dataset containing tweets collected during 4 months just after the 2016 US presidential election [ADLOMS17]. This dataset includes tweets that got re-tweeted more than 1000 times (regarded as *viral tweets*), and offers two sets of manual annotations for each tweet into fake news or not, attending to the categories established by [CCR15]. From the dataset, characteristics signalling exposure, polarization and diffusion have been analysed. Among other features, the date tweets were created, their number of retweets, favourites, hashtags, and text, the number of friends and followers of the creator of the tweet, and whether the account was verified have been examined with care.

Among other results, it has been found that tweets containing fake news were retweeted more than those containing other types of viral content, but are in turn less 'favourited'. The authors

of fake news have more friends and favourites but fewer followers and lists than the authors of tweets labelled as not containing fake news. Those characteristics are representative of more ‘common’ Twitter users (i.e. not celebrities or official sources). Finally, fake news tend to be less positive than tweets not containing them.

This research can be related to different strands of literature and to work in political science and psychology [Pol12, PR17, PCR17, FNR17, SBLE17], to economics [AG17, BM18, Wir18], and to computer science [AG05, BMA15, BFJ⁺12, CCR15, RCC15].

The key contributions of this work are as follows:

- Identification of relevant indicators of political fake news from Twitter metadata and derived metadata, using statistical significance of differences in distributions.
- Proposal of additional indicators using state-of-the-art text processing techniques to improve the identification of political fake news.
- Proposal and testing of several classification methods (including an ensemble method which outperforms other state-of-the-art methods) using the above raw and derived data.

6.3.2 Data and Methodology

The wealth of data available online and through social networking sites present certain difficulties when analysing them. Many approaches to identify fake news and its effects on behaviour have been suggested [CCR15]. However, the question of how do viral fake news effectively differ from other types of viral content remains unanswered. Leveraging from the literature presented in our first paper [AOMS17a] we sketched a model for singling out fake news based on two dimensions: exposure and polarization.

Exposure. Following findings from Polage [Pol12], Pennycook et al. [PR17], Pennycook et al. [PCR17] and Flynn et al. [FNR17], repetition and familiarization with previously seen information should increase the likelihood of believing certain pieces of information

are true. In this way, we would expect that an individual will be more likely to consume a fake news item if he has been previously exposed to it.

Polarization. Swire and colleagues [SBLE17] explained that in highly politicised environments, partisanship highly influences how information affects the way in which individuals process information. The authors claim that the specific mechanism through which this operates is confirmation bias. In view of this, it is reasonable to expect that it is more likely that fake news will contain highly polarizing content.

Following these two dimensions, we define the following topology that will allow us to identify viral fake news in online social media:

1. *Fake News.* High level of exposure and high level of polarization.
2. *Viral Tweet.* High level of exposure but low level of political polarization.
3. *Regular Tweet.* Low level of exposure and low level of political polarization.
4. *Partisan Tweet.* Low level of exposure and high level of political polarization.

We now turn to Twitter’s data to put forward evidence about the existence of these categories. We used Twitter’s public streaming API to collect publicly available tweets related to the 2016 presidential election in the United States. It is important to highlight that tweets collected are subject to Twitter’s terms and conditions which imply that users posting tweets consent to the collection, transfer, manipulation, storage and disclosure of the data generated by the tweet. Because of that, no ethical, legal or social implication derived from the usage of the tweets is expected. The sample was collected using the following search terms and user handles: #MyVote2016, #ElectionDay, #electionnight, @realDonaldTrump and @HillaryClinton. The dataset can be found in [ADLOMS17].

An important feature within Twitter is the ability to share someone’s tweet through ‘retweets’. This functionality enables users to pass forward to their followers an exact copy of someone else’s

		Second Label		
		Other Tweets	Fake News	Unknown
First Label	Other Tweets	6482	1444	330
	Fake News	213	133	7
	Unknown	250	98	44

Table 6.3: This table of contingency reports the differences and the similarities between the labelling performed by the two teams in the used dataset.

tweet. There are many reasons why users might decide to retweet; e.g. to spread information to new audiences, to show one’s role as a listener, or to agree with or validate someone else’s point of view [BGL10]. In the context of this research, we used retweets to define virality. We consider that a tweet went viral if it was retweeted more than a 1000 times. We chose 1000 to single out those tweets that get some traction, but this threshold can be chosen differently. This filtering reduced the number of tweets to be categorised from 57 million to 9000 tweets (a factor of 6375).

Having singled out every viral tweet, we proceeded to eliminate duplicates and manually inspected the text field of the tweets to categorise them. Two teams of individuals labelled tweets as *viral fake news* if its text could be considered within any of the categories mentioned below, and as *regular tweet* (i.e. viral tweet not containing fake news) otherwise.

In this way, the statistical analysis allows us to look for evidence supporting the dimensions proposed for our framework. Our focus moves on to find statistically significant differences between viral tweets and viral fake news within those two dimensions: exposure of the tweet to others, number of entities (i.e. URLs and images), life of the accounts/social connections, and the extent to which tweets differed in expressing polarised political views. We use the fullest extent of available fields within a tweet’s meta-data (such as `created_at`, `retweet_count`, `favourites_count`, etc.) and some derived fields (such as `user_name_caps`, `user_screen_name_digits`, etc.) to investigate differences along these dimensions.

The original dataset contained 57,379,672 tweets ranging from November 2016 until March 2017. That number includes original tweets and retweets. From them, we extracted only the viral tweets (defined as those that have more than 1000 retweets), resulting in a total of 9001

tweets. This dataset can be found at [ADLOMS17]. However, a portion of those tweets are no longer publicly available through Twitter API: some accounts have been closed for infringing Twitter’s policy resulting in tweets no longer being available, while others have been deleted by their authors. As can be seen from Table 6.3 there are some discrepancies between the two teams. For the rest of this study, we focus solely on the labelling of the second team (which contains 1675 fake news (18.6% of the total) and left aside their 381 unknowns (4.23%) to avoid any conflict or ambiguity in the results. The dataset enabled us to perform a statistical analysis looking for statistically significant differences in the features between viral tweets and viral fake news.

It should be noted that there is a growing resistance to use the term *fake news* in the scientific literature as it has somehow lost its literal meaning, and its acute characterization and definition is elusive [LBBe18]. In turn, researchers are favouring the term *false news* in an attempt at reducing the complexity and side-lining interpretation and philosophical debates. We will stick to it because our dataset was compiled with its labellers using the broad interpretation of fake news and not merely acting as fact-checkers.

In order to be consistent with our previous research [AOMS17a], we followed the same labelling principles, i.e. the three categories defined by Rubin et al. [RCC15] plus two more used previously:

1. **Serious fabrication**

These are news stories created entirely to deceive readers. During the 2016 US presidential election, there were plenty of examples of this (e.g. claiming a celebrity has endorsed Donald Trump when that was not the case).

2. **Large scale hoaxes**

Deceptions that are then reported in good faith by reputable news sources. A recent example would be the story that the founder of Corona beer made everyone in his home village a millionaire in his will.

3. **Jokes taken at face value**

Humour sites such as the Onion or Daily Mash present fake news stories in order to satirise the media. Issues can arise when readers see the story out of context and share it with others.

4. **Slanted reporting of real facts**

Selectively chosen but truthful elements of a story put together to serve an agenda. One of the most prevalent examples of this is the well-known problems of voting machine faults.

5. **Stories where the ‘truth’ is contentious**

On issues where ideologies or opinions clash for example, territorial conflicts— there is sometimes no established baseline for truth. Reporters may be unconsciously partisan or perceived as such.

For each tweet collected, the Twitter API provided the main features (number of retweets, favourites, media, URL...). Additionally, we derived extra features such as the number of capital letters in the user name, the time a tweet takes to get to a certain number of retweets, or the global sentiment of the tweet.

In an attempt to minimise labelling errors and variability in perception, tweets were manually labelled by two different teams of people with the same guidelines and categories defined above, who were tasked to determine if a tweet was a fake tweet or not. It resulted in huge discrepancies between the two labels as it can be seen in Table 6.3. Understanding these discrepancies and getting more people to label could be a great improvement track for the future. As the second label was more trustworthy, we will use this one for the following studies with 1675 fake news (18.6%) and 381 unknowns (4.23%)

The collection of the tweets over those few months was done using Twitter Stream API with the desired filters and the eAE coupled with Apache Kafka to store the tweets in MongoDB and produce part of the derived features using Spark. This setup was put in place as an initial step towards an automated platform for fake news identification using the eAE.

Figure 6.6 presents the proposed architecture for the FakeNews Platform using the eAE as back-end and the proposed model for identifying fake news.

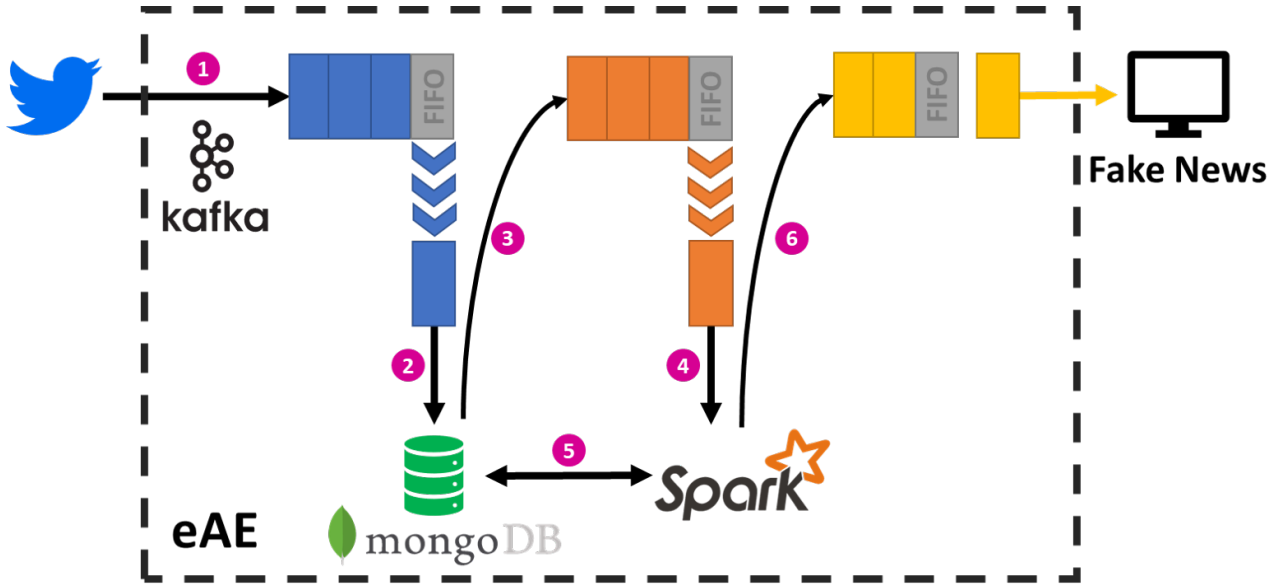


Figure 6.6: A schematic representation of the architecture of the proposed FakeNews Platform.

We use Twitter Stream API (1) and filter real-time tweets using the desired filters. Those tweets get then stored (2) into MongoDB which acts both as a sink and a source, reading from one topic and writing (3) to another one. This new topic gets in turn consumed (4) by Spark to run our models against the collected tweets. Once a tweet is consumed the resulting label is saved (5) back into MongoDB by updating the record. If the tweet is labelled as fake news, it gets added as well (6) to the `fake_news` topic to be consumed by any external service either for further analysis or browsing. The by-default storage of all the streaming data into a database removes the need of researchers to worry about the storage of the data, making it readily available in a high throughput fashion. We also considered developing a module for knowledge enrichment pulling from different websites or external services as a preprocessing step of the tweets before being stored in the database. However, this platform was never put in production as the external service for the subsequent exploration was never implemented due to a lack of resources and was out of the scope of this research.

Regarding the analysis itself, we used the eAE as the analytical platform for all the analyses that have been carried out in this project. The project illustrates the flexibility, comprehensiveness towards the end to end support of data science workflows and the multitenancy capabilities of

the eAE to support large scale concurrent loads. The submission of the jobs was done using the eAE's Python PIP package. This work was implemented using two different languages (R and Python) and various packages (word2vec, scikit-learn, etc.) to be able to use a broad variety of techniques for exploratory and analysis purposes. The purpose of using two different languages was to rely on the individual strength of each one. In addition, every analysis was done using a 5 folds cross-validation to ensure the robustness of the analyses. We used R 3.4 for the statistics and the NLP tasks. In order to speed up the computation, the NLP tasks were executed out on GPUs. Python 3 was used for the initial data transformations and remaining analyses including the data filtering, PCA and the different machine learning algorithms.

Finally, the deployment of the eAE at DSI has enabled the pooling of the hardware resources from different projects in a transparent manner to the users. This sharing gives individual projects access to larger compute capacities than they would otherwise. It has been critical in this instance as the exploration, hyperparameter tuning and cross-validations alone have consumed well over 200k hours of compute time during the span of the project. Consequently, the eAE has enabled faster iterations, access to resources that otherwise wouldn't have been available (GPU) and increased the average load across machines. All those benefits de facto decreased as well the financial cost of conducting that research.

6.3.3 Feature Selection

As previously indicated, several pieces of information about the tweets are contained in the dataset or can be derived from it. This section focuses on describing those features (obtained directly from Twitter API, derived from them, or extracted from the texts) and on checking whether or not their value distributions differ when referred to viral tweets containing fake news versus tweets not containing them. To do so, we look for statistically significant differences in the distributions of features through a Kolmogorov-Smirnov test [MJ51] between the set of tweets containing viral fake news and complementary set (viral tweets not containing fake news). The null hypothesis was *H0: The two samples come from the same distribution* against the alternative hypothesis *H1: The two samples come from different distributions*.

	p-value	t-stat	Mean Fake News	Mean Other Tweets
fav_count	0.00	0.184	2896	4022
user.followers_count	0.00	0.129	140609	235797
user.listed_count	0.00	0.121	907	1679
user.verified	0.00	0.209	49.219%	70.161%
num_hashtags	0.00	0.169	0.721	0.550
num_mentions	0.00	0.267	1.731	2.076
user.favourites_count	1.65e-12	0.103	1921	1161
user.friends_count	5.07e-11	0.094	992	696
num_media	2.07e-10	0.091	0.268	0.177
user.statuses_count	2.32e-08	0.081	4103	12355
retweet_count	5.35e-01	0.022	2284	2328
user.default_profile	4.96e-01	0.022	20.879%	18.646%
num_urls	1.00e+00	0.005	0.353	0.346
user.default_profile_image	1.00e+00	0.001	0.231%	0.319%
user.profile_use_bg_image	1.00e+00	0.009	76.750%	75.888%

Table 6.4: Analysis of features coming from Twitter API. The results (p-value and t-stat) come from the Kolmogorov-Smirnov test [MJ51] on the distributions between the viral fake news and the other viral tweets. Rows are ordered by p-value. Variables above the line are those whose differences are considered statistically significant (p-value smaller than 0.01).

We consider that a difference between sets is statistically significant if the p-value is lower than 1%. For the continuous variables with extreme values, we did the test on the (decimal) logarithm in order to have a more representative scale. For others (e.g. *num_hashtags*, *num_mentions*, *num_media*, *num_urls*), we compute the number of items per tweet. p-values smaller than $1e-16$ are reported in all tables as 0.00.

Those key features (i.e. with statistically significant differences in the distributions), if any, will be the ones we will ultimately feed to different classification algorithms (see next section) in order to attempt the automatic flagging of tweets containing fake news.

Features from Twitter API

The main data features were those directly given by Twitter API. Table 6.4 lists all the variables we considered (a detailed description of each one can be found at Twitter’s API website) together with the results from the Kolmogorov-Smirnov test. As seen, there are indeed several

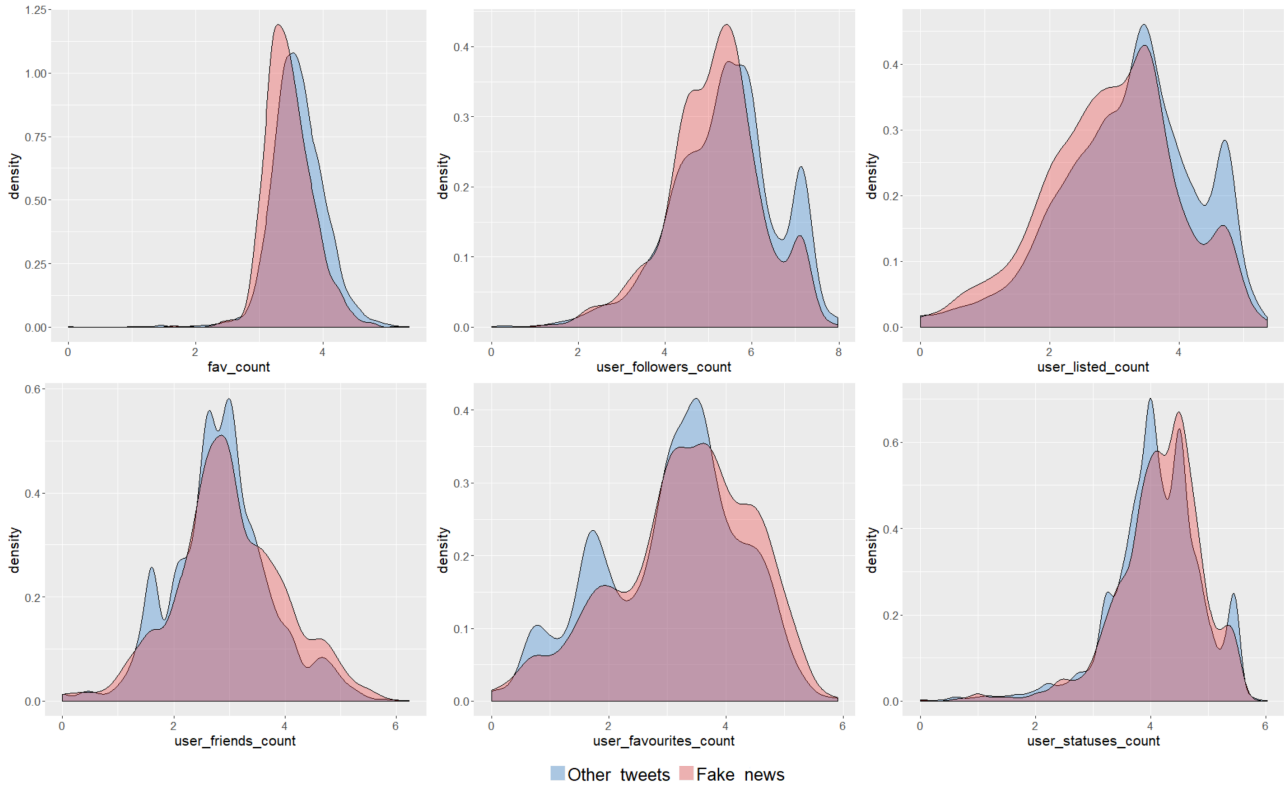


Figure 6.7: Density distribution of the decimal logarithm of the continuous variables from Table 6.4 that are statistically significant. From the image, we can see that viral tweets not containing fake news (in blue) tend to have *peakier* distributions.

variables in which the differences in distributions are different in terms of statistical significance (those above the horizontal line).

Note that a statistically significant difference does not mean that difference is large or ultimately meaningful. Therefore, in order to accurately select a group of promising features for the discrimination task, we need to consider both the size of the difference and its statistical significance. Visual inspection of the distribution, together with the actual value of the difference can help to identify those features. Figures 6.7 and 6.8 visually display the density distributions of the features that do have statistically significant differences in their distributions.

Apart from the features listed in Table 6.4, we looked at the distribution of tweet sources (iPhone, Android, web client, media studio, etc.) for both viral tweets containing fake news and those not containing them, and both distributions are very similar: 40% vs 42% for iPhone, 33% vs 32% for the web client and 10% vs 9.5% for Android. Those marginal differences confirm there is no significant difference for this feature and do not offer any other meaningful insight.

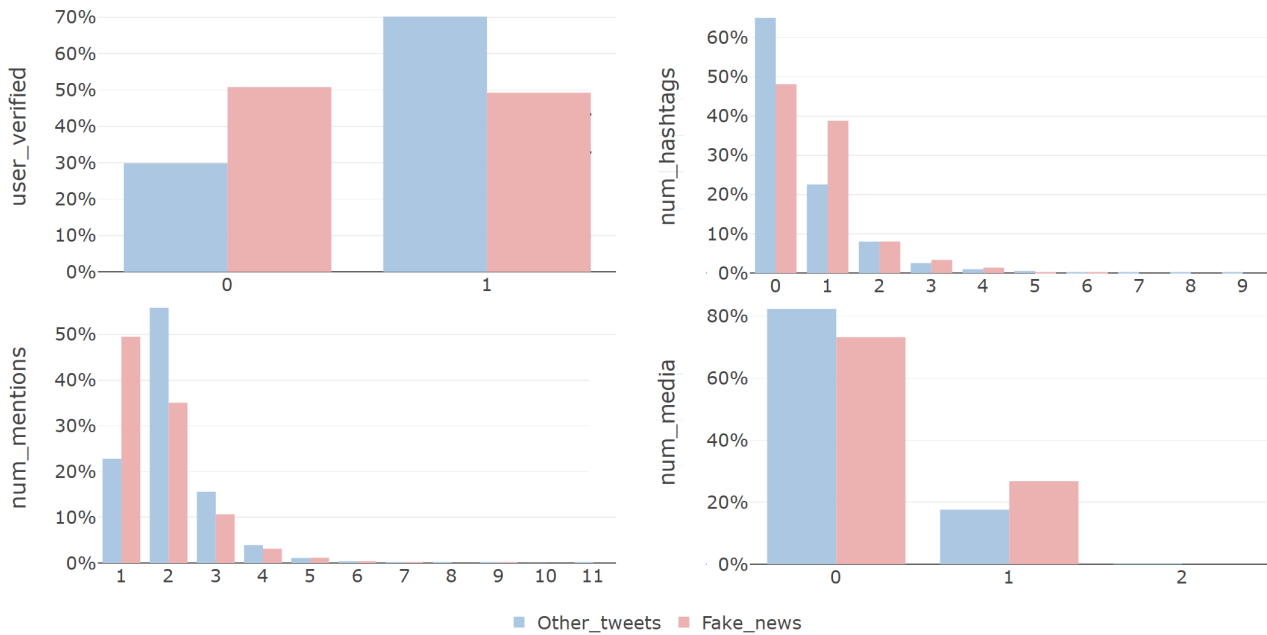


Figure 6.8: Distribution of the four significant discrete variables (user.verified, num.hashtags, num.mentions and num.media) from Table 6.4. The test for the proportion of verified account confirms an expected fact: the proportion of verified account is much weaker for viral tweets containing fake news than for other viral tweets, suggesting that fake news tend to be created by more ‘anonymous’ people. Besides, tweets with fake news have generally more hashtags and media but fewer mentions.

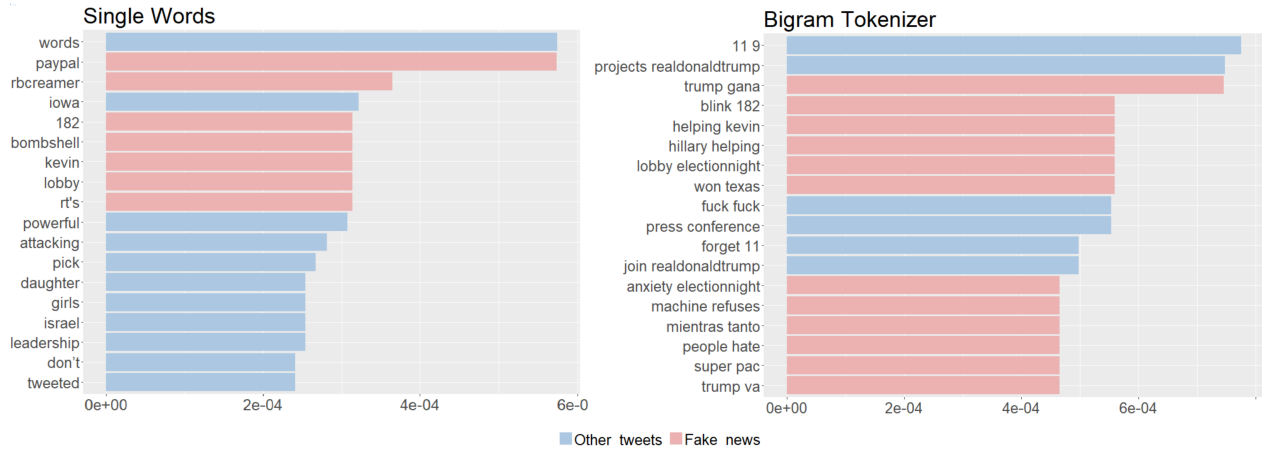


Figure 6.9: Most recurrent words in the tweets (single and bigram)

We also analysed words’ frequencies (as single words and bigrams) within the text of the tweets; the most recurrent ones are depicted in Figure 6.9. Even with huge discrepancies between the distributions, we believe this is not a very resilient feature to consider for flagging fake news as they are too specific to this dataset and to the type of news, making them unusable for different topics, contexts and languages.

Finally, we also analyzed the most used hashtags in both subsets of tweets. Of course, the most common were the ones used to collect the tweets. Leaving those aside, and as for the remaining ones, there is no statistical difference between hashtags used in tweets containing fake news and tweets not containing them. Figure 6.10 shows their frequency distribution. It is interesting to see that a couple of hashtags only appear in tweets labelled as fake news. While this might be a product of the dataset, it is an issue that probably deserves further research. However, once again, the frequency of hashtags is hardly generalizable or usable on another dataset.

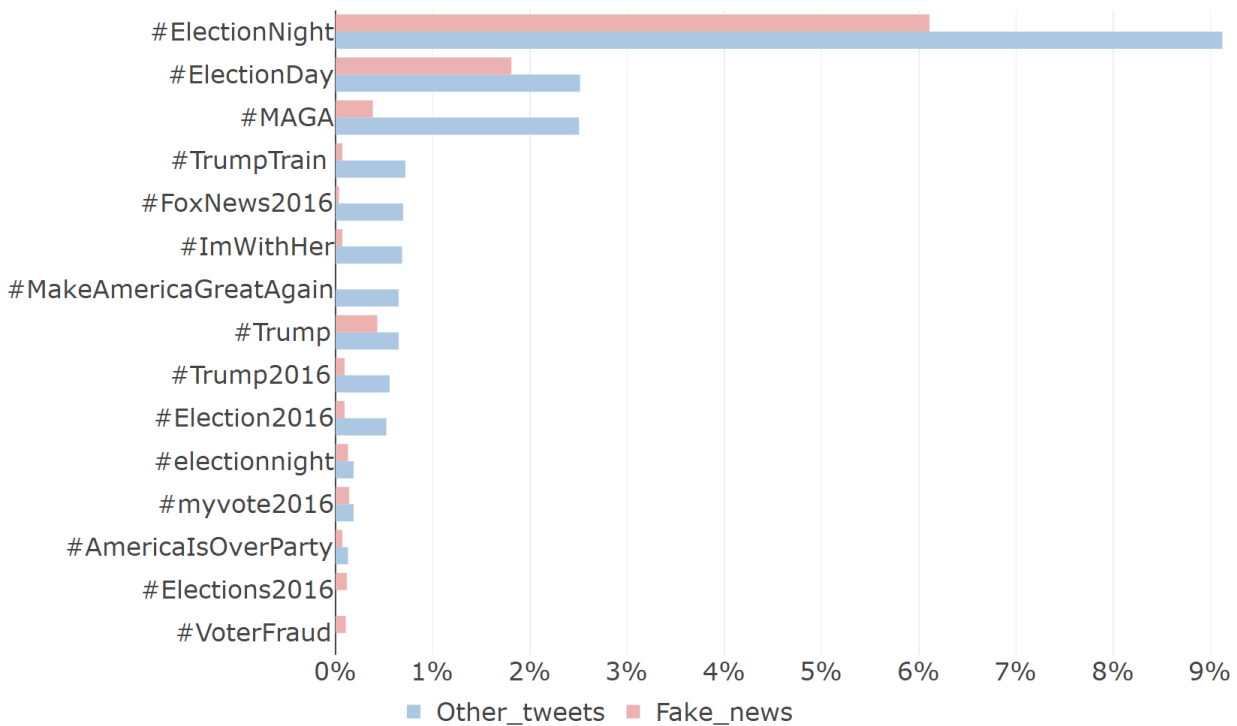


Figure 6.10: Frequency of appearance of most used hashtags in tweets containing fake news (red) and not containing them.

Features about diffusion of tweets

In order to obtain additional useful features that might have discriminating power, we enriched the original dataset by computing the day and the hours at which the tweet was published (in the EST time zone). As these variables are categorical, the average difference test does not make sense for them.

However, the number of days which separates the creation of a tweet and the creation of its

Twitter account is, on average, 1941 days for tweets containing fake news and 2100 days for tweets not containing them. This delta represents a computed p-value of $1.04e-08$ which is indeed highly significant. This result suggests that accounts spreading fake news were created more recently (also reported by [VRA18]). This is coherent with the observed fact that users who spread fake news have their accounts eventually deleted and are forced to create new ones.

We believe that the viral propagation pattern of the tweet is important. In fact, the evolution in the number of retweets (and favourites) greatly varies depending on the tweets. For example, it can be linear, exponential or polynomial over time and the patterns between fake news and the rest would allow us to distinguish them easily. We can observe from Figure 6.7 that the retweet count and the favourite count are correlated (coefficient of 0.665) as well as the number of followers of a user and the number of times he has been listed (coefficient of 0.911).

For each tweet that we have the relevant data, we computed the time (in hours) it takes to get to 10, 20, 50, 100, 250, 500 and 1000 retweets (and to equal number of favourites, respectively). Figure 6.11 shows that, for most of the tweets, these features are very correlated with the exception of 1000 retweets, which is not that correlated to any of the favourite features. It is also interesting to see that there is no negative correlations between these variables.

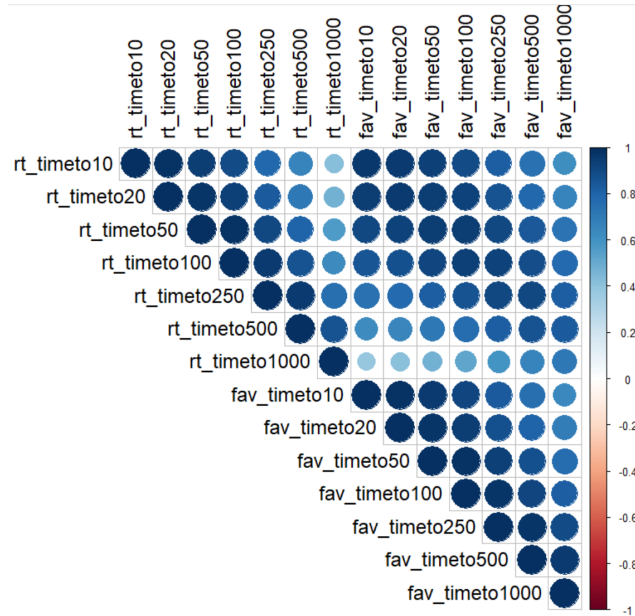


Figure 6.11: Correlation of the features related to the spreading of tweets. *rt* stands for retweet (e.g. *rt_timeto10*, time to get to 10 retweets), and *fav* for favourite (e.g. *fav_timeto10*, time to get to 10 favourites).

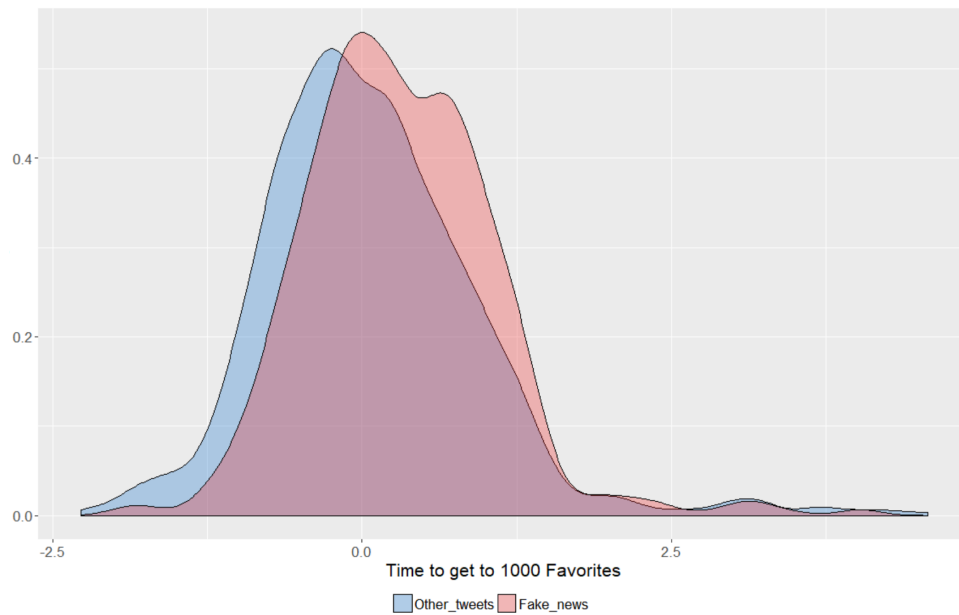


Figure 6.12: Distribution of the decimal logarithm of the time (in hour) to get to 1000 favourites (*fav_timeto1000*) for both tweets containing fake news and tweets not containing them. The associated p-value is $9.60e-11$ which proves the significance of the propagation pattern.

Figure 6.12 shows, in particular, the different distributions of the time to get to 1000 favourites (which is the most distinctive feature of this set) for tweets containing fake news and those not containing them. Tweets containing fake news are generally slower to get a thousand favourites.

Features extracted from text

Finally, and in order to obtain a more complete picture of what viral fake news look like we explored some textual features of the tweets; more precisely, we looked at:

- User's name
- User's screen name
- User's description
- Tweet's text

In particular, we extracted several text features (by means of regular expressions) about the previous variables; e.g. percentage of capital letters, of digits, of special characters Our aim is again to analyse if any of those have a different value distribution that can be useful to discriminate between tweets containing fake news and tweets not containing them.

	p-value	t-stat	Mean Fake News	Mean Other Tweets
text.digits	8.02e-09	0.084	2.676%	2.104%
text.caps	1.41e-08	0.082	14.016%	12.665%
user.description.caps	1.50e-04	0.059	11.970%	12.511%
user.name.caps	8.27e-04	0.053	23.592%	23.287%
user.name.weird_char	2.95e-03	0.049	7.533%	4.606%
user.screen_name.caps	3.64e-03	0.048	18.444%	17.664%
text.exclam	4.40e-03	0.047	0.222%	0.309%
user.screen_name.weird_char	2.18e-01	0.028	3.162%	2.502%
user.description.exclam	2.53e-01	0.027	0.241%	0.283%
user.screen_name.digits	4.44e-01	0.023	1.812%	1.462%
user.description.digits	6.52e-01	0.020	1.458%	1.552%
user.screen_name.underscores	8.02e-01	0.017	1.350%	1.040%
user.description.nonstandard	8.14e-01	0.017	2.879%	2.892%
user.name.digits	1.00e+00	0.006	0.510%	0.288%
user.name.underscores	1.00e+00	0.000	0.004%	0.006%
text.nonstandard	1.00e+00	0.008	0.458%	0.447%

Table 6.5: Features extracted for the text analysis. Again, rows are ordered by statistical significance; significant variables are above the line. It is interesting to see that those are mostly the ones associated with spelling used by bots (randomly generated to avoid collisions).

Results reported in Table 6.5 make us conclude that individuals with higher numbers of special characters and caps in their user name are more likely to tweet fake news. In addition to that, a tweet with a high proportion of capital letters, digits and exclamation points has more chance to be a fake news.

Sentiment Analysis features

Finally, we perform a sentiment analysis on the actual content of the tweets. Sentiment analysis aims at identifying, extracting, quantifying and studying affective states and subjective information. Often, Sentiment analysis is used to determine the attitude of a speaker [SOS⁺67]; in our case, the author of a tweet.

When human readers approach a text, they use their understanding of the emotional intent of words to infer whether a section of text is positive, negative or neutral, or perhaps characterised by some more nuanced emotion like surprise or disgust. Text mining tools are available to

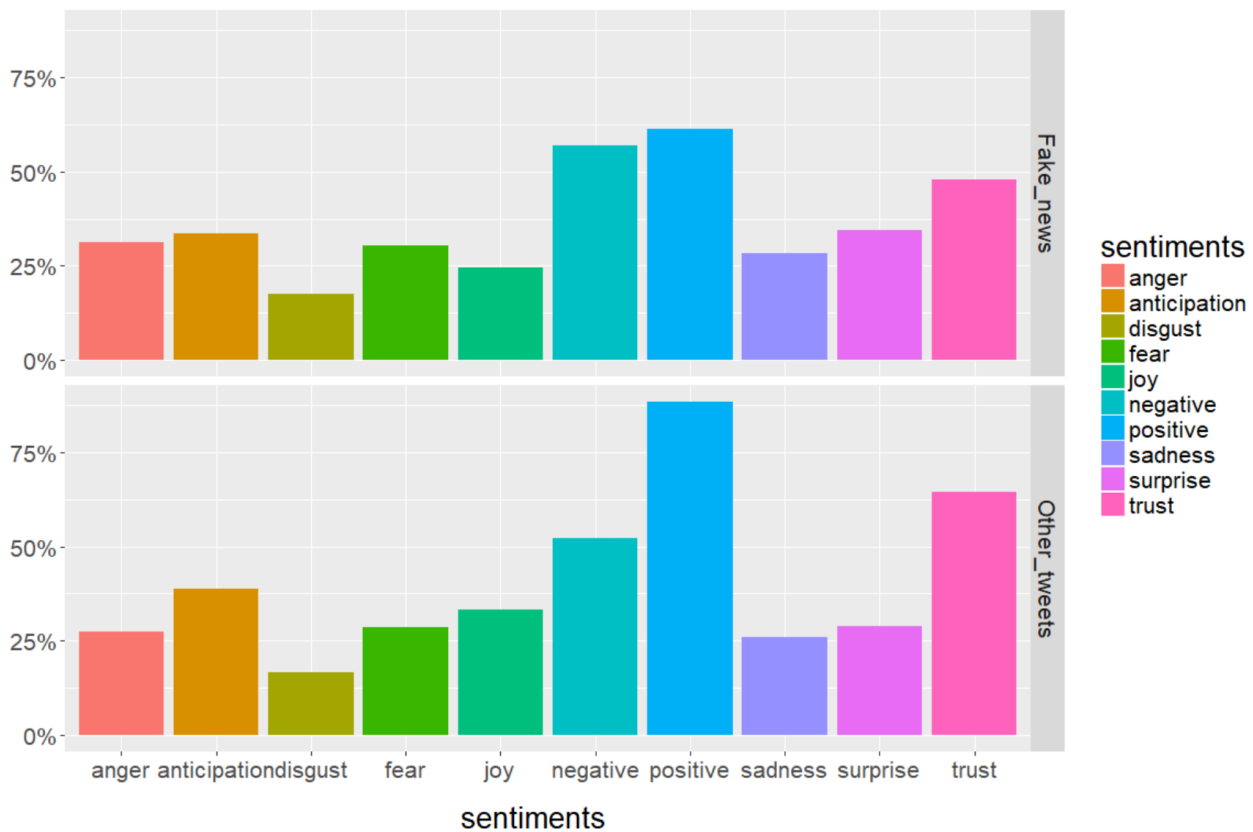


Figure 6.13: Comparison between the different core sentiments between tweets containing fake news and tweets not containing them.

programmatically extract the emotional content of text [SR17], and they usually boil down to two main approaches [KSTA15]: lexical approach and machine learning approach.

Our first step was to analyse the sentiments of the whole text field (which includes hashtags) of tweets in our dataset using National Research Council Canada (NRC) lexicons [MT13]. Figure 6.13 clearly shows that fake news have generally less joy, trust and positive emotions but more surprise. By looking at the temporal evolution of the emotions (see Figure 6.14) we noticed that the emotions present a higher level of randomness in tweets with fake news, while trust and positive emotions dominated in tweets not containing fake news).

A sentiment score can also be computed using Deep Learning techniques [ZWL18], and sometimes perform better than the lexicon approach as it can capture some subtleties that the lexicon approach misses. In our case, we used both *word2vec* from Google [MCCD13] (to compute a sentiment score with the vectors given by a pre-trained model on the *sentiment140* dataset [GBH17]) and *fasttext* from Facebook [JGBM16] (to identify informative text features

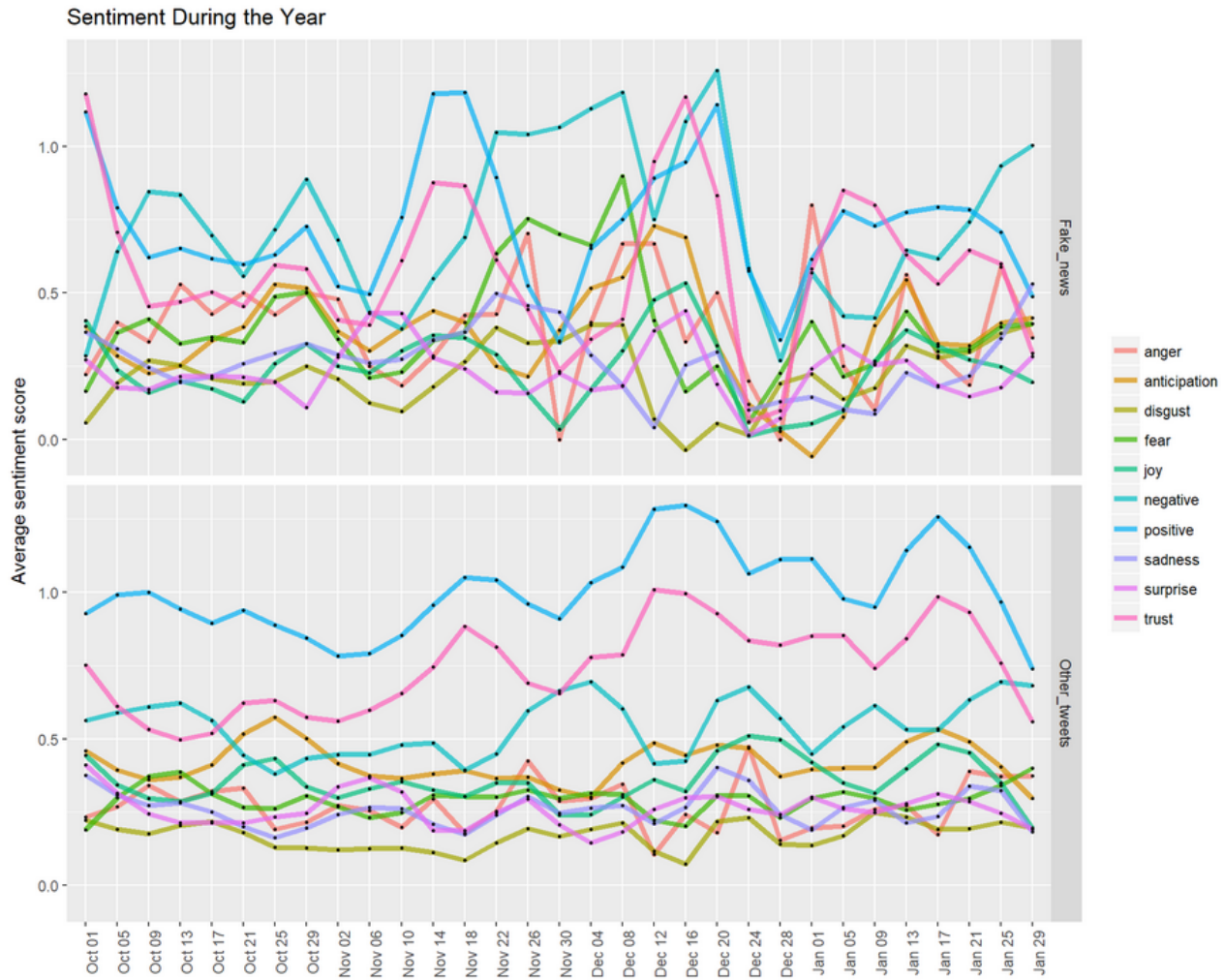


Figure 6.14: Evolution of the different core sentiments over the course of the four months, between tweets containing fake news and tweets not containing them.

that could be directly added to the final classification model).

Figure 6.15 shows the overall probability of a tweet being positive given the *word2vec* algorithm. The average probability over the 4 whole months is more positive than negative, regardless of it being fake news or not. However, the trend for fake news is lower than for the other tweets, which is coherent with the previous results stating that tweets with fake news tend to be more negative. The wider spread can be explained by the scarcity of the fake news at some periods and doesn't constitute a significant indicator.

As input model for *fasttext*, we used a model pre-trained on a Wikipedia database [BGJM16] with word vectors of dimension 300. For each tweet, we computed the associated vectors for each word in the main text. Then, we computed the mean and the standard deviation for



Figure 6.15: Difference of the evolution of the sentiment computed by *word2vec* between tweets containing fake news and other tweets. Each point represents a tweet in the timeline of our dataset and the probability of the tweet for being positive. The blue line represents the average probability per day.

each of these words, finally obtaining two vectors of length 300 for each tweet. In order to reduce dimensionality, we applied a Principal Component Analysis (PCA), choosing only four components for the average ($text_ft_mean_c_i$) and two for the standard deviation ($text_ft_sd_c_i$). Those correspond to 40% of the total information. This specific number of components have been selected using the elbow method.

In addition to those features from the main text of the tweets, we also analysed emojis contained in them. 7.13% of the tweets in the dataset contained at least one emoji with an average of 2.4 emojis per tweet. Some tweets had a surprisingly high number of emojis (the largest number is 55 emojis in a single tweet) but it is quite common that a tweet only contains one single emoji

repeated many times. We narrowed our study to tweets containing between 1 and 10 emojis (this cover 98.7% of the cases on which at least an emoji is present). Despite some interesting information yielded by the analysis (e.g. the American flag is by far the most used emoji, see Figure 6.16), this particular study did not lead to any meaningful insight that could contribute to the creation of an indicator to separate fake news from non-fake news. Figure 6.17 shows that the distribution of emojis and their sentiment score in both the fake news and the other tweets are very similar.

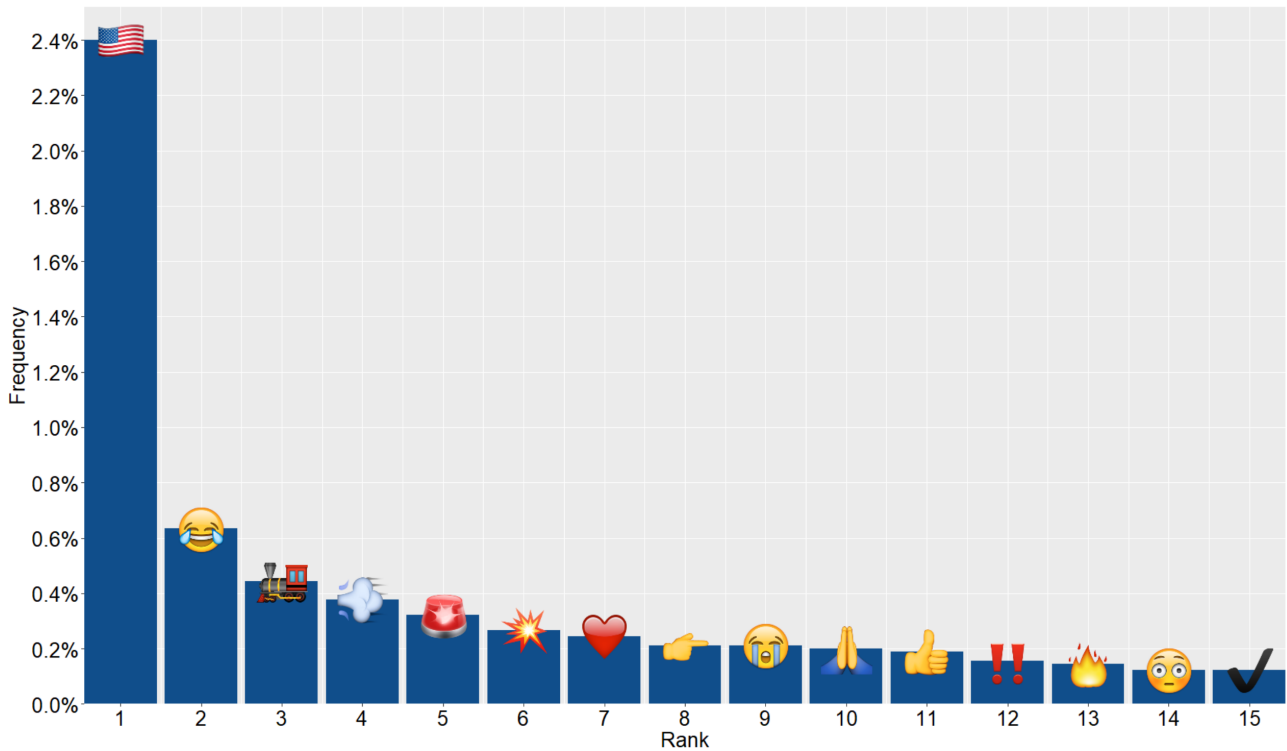


Figure 6.16: Most used emojis in the dataset

Our results confirm our previous findings [AOMS17b] on a smaller dataset (just the election day) about the relevance of distribution of followers, the number of URLs on tweets, and the verification of the users.

6.3.4 Fake news classification

Having described and selected a set of appropriate features about tweets, we focus in this section on the task of using those futures to classifying tweets into fake news or not. In other

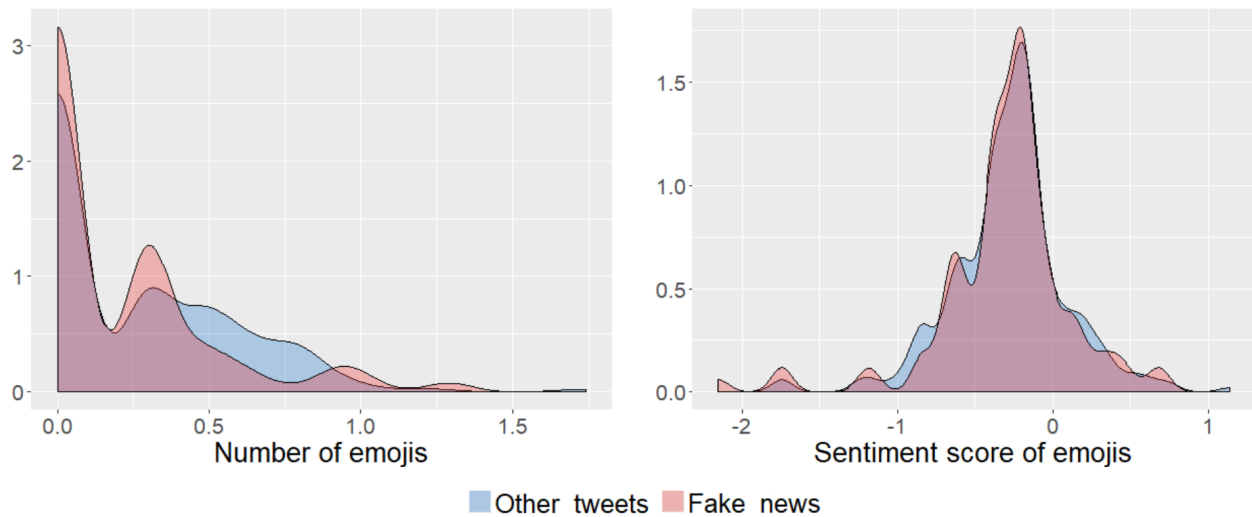


Figure 6.17: Distribution of the decimal logarithm of the number of emojis and the sentiment score of tweets with emojis for the fake news and the other tweets

words, our aim is to build a classifier capable of tagging unseen viral tweets into fake news or not.

In order to have a larger coverage of our experimentation, we considered six different subsets of features from the ones previously described:

- Only the features coming from Twitter API (main)
- Features from Twitter API + features extracted from text (main_with_regex)
- Features from Twitter API + features from the sentiment analysis - lexicon dictionary approach (main_with_sentiments_dict)
- Features from Twitter API + features from the sentiment analysis - deep learning approach (main_with_DL_NLP)
- Features from Twitter API + time-related features (main_with_time)
- All the features (all_features)

For the last two subsets, we only used 3717 tweets. The reason is that the collection of the tweets in the dataset dates back to November 8th 2016 but some of the tweets were first published

before that date (within the collection window only retweets to them were collected). Hence, some pieces of the original tweets (i.e. data only available when first published) is not included in the dataset and cannot be obtained now. Because of that, we limit ourselves to only use the time series for tweets that were published after the collection started.

In all cases, we first split the dataset into:

- A training set (80% of the data). For the algorithms that need a hyperparameters-tuning, we applied cross-validation with 5 folds within this sub dataset.
- A test set (20% of the data) to compute the final performance of the models.

In order to check the predicting capability of our selected features, we have tested different classification algorithms, including a polynomial kernel SVM, k-NN, Logistic Regression, Random Forests, and Boosted Trees (for this, we chose XGboost which is usually used to achieve state of the art results in this context [CG16]).

Due to our dataset being highly unbalanced (as shown in Table 6.3) accuracy is not a proper metric to measure performance of the models. Instead, we turn our attention to two other metrics:

Recall (for a threshold of 0.5): since a fake news has an impact, we want to optimise the ratio of fake news detected and be sure that the number of fake news not detected is really low, even if it means to over filter and having some false positives.

AUC: It gives a good overview of how true positive rate and false positive rate trade-off, regardless of any threshold.

The computed AUC for each of them on the different sets of features is summarised on Figure 6.18. Unsurprisingly, XGboost gave the best performance, which is coherent with what others have reported for this type of classification [OPSS11, Wai16, Gor17]. Because of this, we will solely focus on this model for the rest of this section.

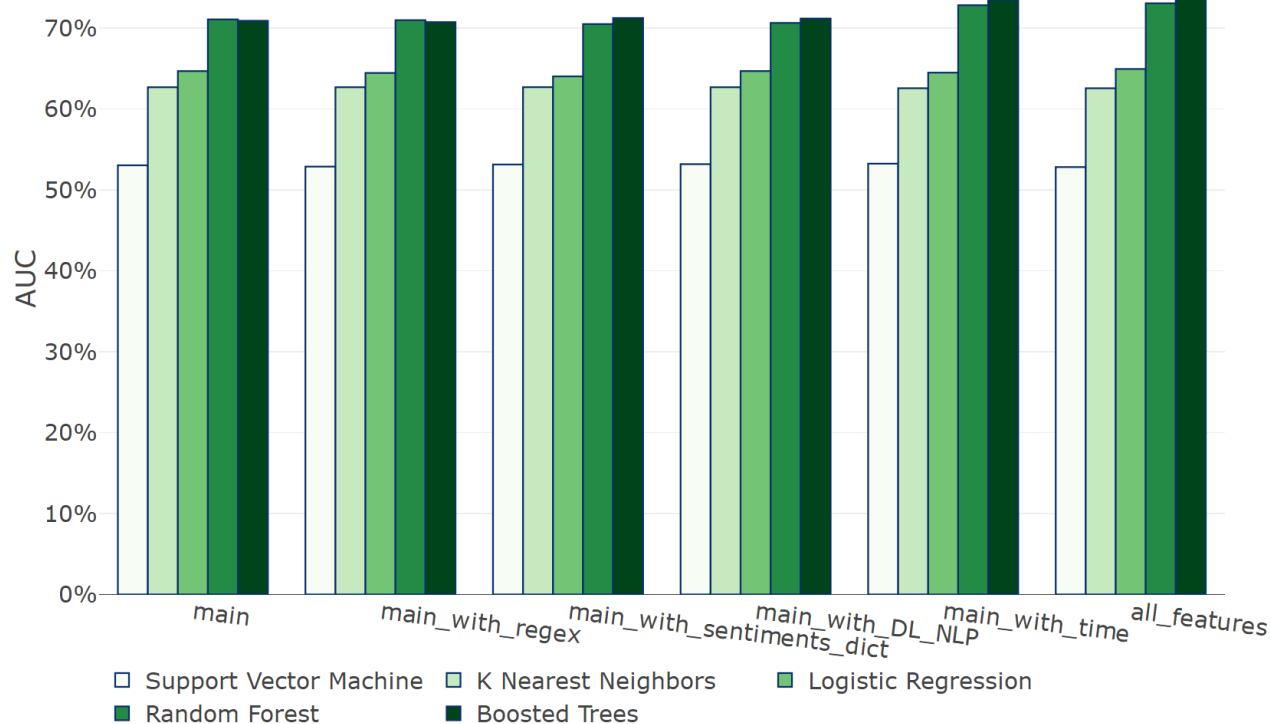


Figure 6.18: AUC computed on all subsets of features for the different machine learning algorithms evaluated.

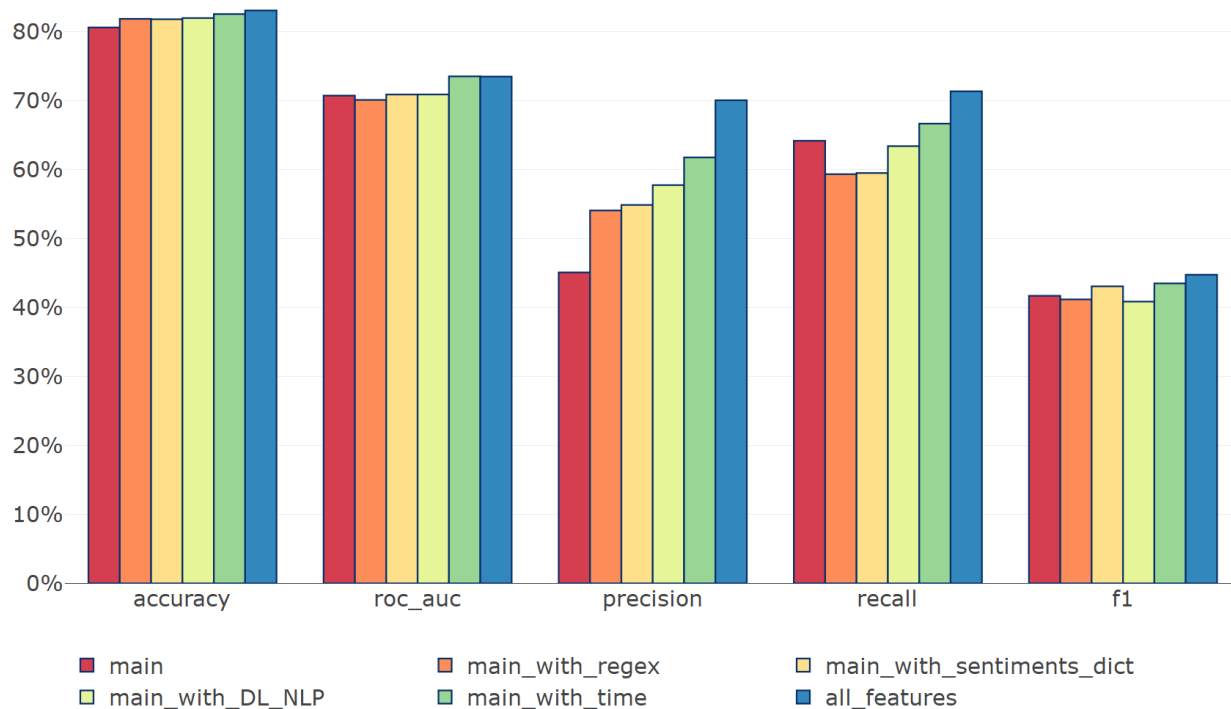


Figure 6.19: Best performances for each subset of features, and for each metric of performance.

Figure 6.19 shows the results obtained for those metrics for each set of variables. Accuracy, precision, recall, and f1-score have been computed for a threshold of 0.5 (meaning that each observation for which our model computed a probability over 0.5 has been labelled as fake news). It is clear from the Figure, that all these metrics improve as more features get added. In particular, AUC went from 70.73% with only Twitter features to 73.49% with all the features; likewise, recall went from 64.18% to 71.13%.

Impact of the hyperparameters

We also looked at the impact of hyperparameters on the performances of the XGboost model (which was the best performing amongst the ones tested). For this, we used a 5-fold cross-validation, and calculated optimised values for:

- The number of iterations
- The maximum depth of each tree at each step
- The learning rate
- The maximum number of features

The main problem we notice from Figure 6.20 is that precision and recall are inversely correlated (which explains why f1-score is so low). This effect is most obvious for maximum depth of the trees: while it grows, the precision increases but the recall decreases as shown in the figure).

Impact of selected variables

Finally, we looked at the contribution of the different variables to the final model (XGboost) in order to evaluate their importance in the context of identifying viral political fake news. In order to do so, we have plotted in Figure 6.21 the most important variables for each model where one metric is maximal. There are some interesting insights to report:

- It seems that the favourite count is always one of the three most important variables for every metric.

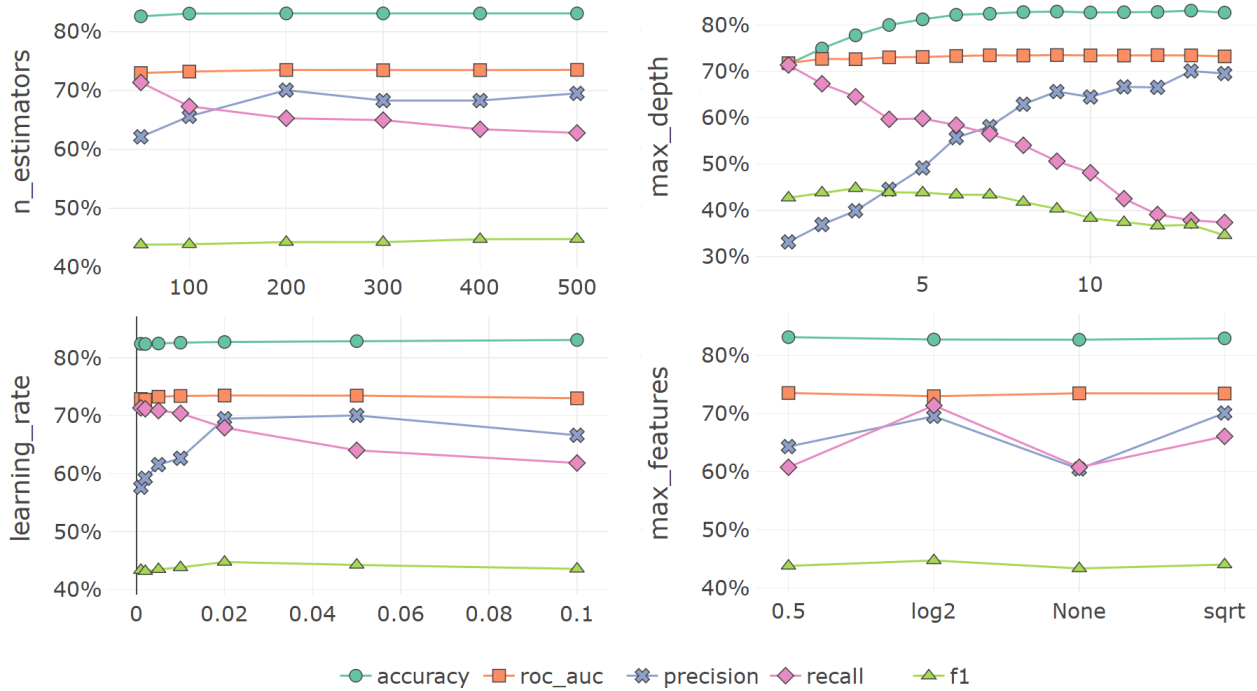


Figure 6.20: Evaluation of the impact of hyperparameters on XGboost performance

- The first component of the PCA for the standard deviation of the fasttext is also a very important variable for the models with respectively the best accuracy, AUC and precision.
- The contribution of features coming from Twitter outweighs the contribution of the other features (propagation, extracted from text, and sentiment analysis). Indeed, accuracy and AUC only goes up by 2,76% when all of them are added to the final model.

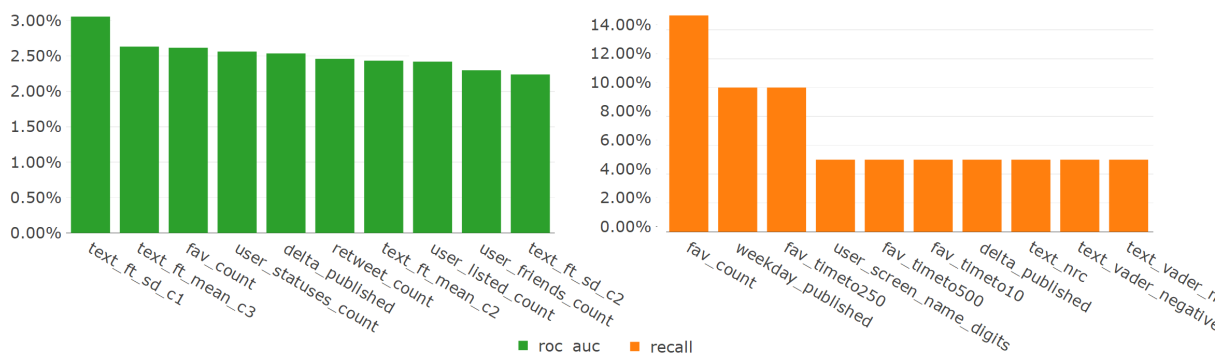


Figure 6.21: Most important variables for the best model for the AUC and the recall

6.3.5 Conclusion

In this section, we presented a study of features that could facilitate the automatic detection of political fake news, and validate our findings by building an ensemble method for fake news identification on Twitter. As reported, it is a promising line of action for efficient and effective early detection of fake news.

We used state-of-the-art machine learning techniques to identify significant features and used them to build an ensemble model of trees to identify fake news with a significant performance. We improved the model by introducing derived features indicating how fast the tweets spread and the contents of their texts. However, the results are still not ideal as we were forced to sacrifice some precision to get a better recall. Arguably, it is more relevant in this context to identify true news as fake news rather than missing on some fake news.

The first major limitation that we faced was the missing data from the public version of the Twitter API due to Twitter's limitation. We believe that our model would be more accurate if we could get access to more data from the Premium version of the Twitter API. The Premium version would allow us to:

- Have more metadata on each tweet (some of it is available for some tweets but is missing for others in the public open API or we failed to capture it properly).
- Build more accurate derived features (such as the evolution of the favourites and retweets counts).
- Study further interactions between tweets. Our hypothesis is that users react differently to fake news and real news (as recently reported by [VRA18]).

An added difficulty of the current dataset is the hardness of having an accurate and unbiased labelling (as Table 6.3 illustrates). It seems clear that different cultural backgrounds, knowledge of the American culture, and English language proficiency (among the most relevant) induced vastly different perceptions on whether a piece of news was fake or not. Additionally, the lack

of a proper fake news definition (thus, leaving it to individual interpretation) play a major role, that other researchers have addressed by focusing on *false news* instead.

As the dataset is only based on the US elections of 2016, it is likely that our model might be overfitting. Indeed, the model we built might work well in this specific context but might not be applicable in another scenario. Our next steps would be to add data on other elections to make the model more robust for this context. Shared features are likely to be the ground truth of any model trying to identify political fake news.

Regarding this generalization effort, one area of improvement for our model would be to identify the actual context and the subject of the tweet automatically with NLP algorithms. An extensive amount of work has been carried out in this domain and we would start by reusing as much as possible this work [SLC17]. Once the subject and context are extracted, we could automatically target a specific model better suited for the identification. To further improve the accuracy, we plan on applying deep learning's convolutional neural network to extract features from the profile pictures and the background pictures to create new derived features.

Finally, our results are mostly in agreement with the findings reported by Vosoughi et al. [VRA18]. One notable exception is regarding the spreading of fake news (generally slower than other tweets in our case; faster in the Science study). This can be due to the fact that our study is specific to the US elections whereas Vosoughi et al.'s collected tweets from 2006 to 2017 and this property may be specific to this sample of the data.

Chapter 7

eTRIKS Analytical Environment supporting Open Science

In this chapter, we will present measures that have been taken to ensure the sustainability, openness, and adoption of the platform. We present as well how the reconciliation between the research and engineering worlds has enabled us to bring the best of both worlds into this work. Indeed, a balance between good practices from engineering (TTD, microservices architecture, etc.) and the cutting-edge from research provides the optimal efficiency to researchers in the field of medical research.

7.1 Sustainability of the platform

7.1.1 Hosting of the project and supporting the users

We believe that quality research depends on the availability of tools and applications to the researchers. This belief is aligned with the Infrastructure School [FF14] proposed by Fecher and Friesike for Open Science. With that goal in mind, we have chosen to host the eAE on GitHub (see Figure 7.1) to make the code available to the community in an open source fashion. Github is a web-based hosting service for version control using Git and is most well known to be among

the top open source hosting platforms. The eAE has been open-sourced under the MIT license to be as permissive as possible. We leverage GitHub's built-in capabilities for bug reporting and tracking, documentation (through the wiki and README) and agile project management. Following that *modus operandi*, users and new developers can contribute to the projects in a straightforward way without scattering across different platforms. The issue tracker allows the users to report issues to the developers, discuss with them to identify the root cause and track the progress on the fix.

The use of Git as the version-control system for tracking changes warrants better management of the Master branch by the owners and encourages the potential contributors to contribute to the project in a safe fashion. Any contribution made through a 'pull request' can be reviewed by the other members and, if the request meets the quality standards and the build is successful, the request is securely merged into the code base.

In order to facilitate the interactions with the platform and to make it accessible to a wider audience, we have published on Postman¹ a set of example queries (status of the clusters, user creation/deletion, job creation, etc.) and their associated answers. Extensive documentation resources of each supported query for the users and administrators to use are also available. Figure 7.2 illustrates all the queries available to the users and administrators. Those queries can either be run directly in a user's local Postman by importing from the web those published examples or in Python by copying the code generated automatically by Postman for the given request. Postman can generate the requests for PHP, GO, jQuery, Ruby, cURL and node as well, giving substantial freedom to the developers and users. This initiative is comparable to the Public School [FF14] of thought for Open Science by addressing the challenge of ease of use for interested non-experts.

¹<https://www.getpostman.com/>

dsi-id / eae-scheduler

Watch 2 Star 1 Fork 2

Code Issues 1 Pull requests 10 Projects 1 Wiki Insights

eAE - Scheduler micro-service

236 commits 13 branches 0 releases 5 contributors MIT

Branch: master New pull request Find file Clone or download

Commit	Description	Time
fgruitton Merge pull request #47 from dsi-id/renovate/eslint-5.x	Latest commit b801c06 on 9 Jul	
DockerHub	Pin node Docker tag	6 months ago
Documentation	Update of the documentation	a year ago
config	adding disable archiving feature	9 months ago
src	Upgrading the necessary component for switch to MongoDB Node driver v3	6 months ago
test	Upgrading the necessary component for switch to MongoDB Node driver v3	6 months ago
.babelrc	first commit	a year ago
.dockerignore	first commit	a year ago
.editorconfig	small bug fix and adding swagger doc, editorconfig and bits of travis...	a year ago
.eslintrc	Fix eslint problems in files	a year ago
.gitignore	Merge branch 'master' into master (reverted from commit 2f7c76c)	9 months ago
.travis.yml	Test that jobs which have failed three times are set to dead	a year ago
LICENSE	Create LICENSE	a year ago
README.md	Merge branch 'master' into master (reverted from commit 2f7c76c)	9 months ago
doc-api-swagger.yml	Merge branch 'master' into master (reverted from commit 2f7c76c)	9 months ago
package.json	Update dependency eslint to v5.1.0	5 months ago
renovate.json	Add renovate.json	6 months ago

README.md

eAE-scheduler Travis

made with Phoenixes build passing dependencies up to date dev dependencies up to date

David Dependencies Manager

eAE - Scheduler micro-service

The eae-scheduler service provides scheduling capabilities to the eAE eco-system. While running jobs is handled by the eae-compute, managing and scheduling them is the role of eae-scheduler. To do so, the eae-scheduler runs continually in the background, checking periodically for jobs to be scheduled, queued or archived.

We provide the [API documentation](#) in swagger 2.0 format. You can paste the content in the [swagger editor](#) to render the API documentation.

Figure 7.1: Illustration of the eAE's *Scheduling and Management service* hosted on GitHub. The repository contains a README describing the main features of the service, the docker file to build the docker container, the YAML build file (.travis.yml) for Travis, the tests to automatically validate the build, and the code of the service. The issues tab contains all the issues reported by users and developers or future features related to the service. The pull requests tab contains all currently open pull requests to be merged into the development branch by the admins. The wiki tab contains all the necessary documentation of the service (design of the service, description of the API, comments, etc). The shields (build and dependencies) are dynamic which allows anyone to check the current status of the project.

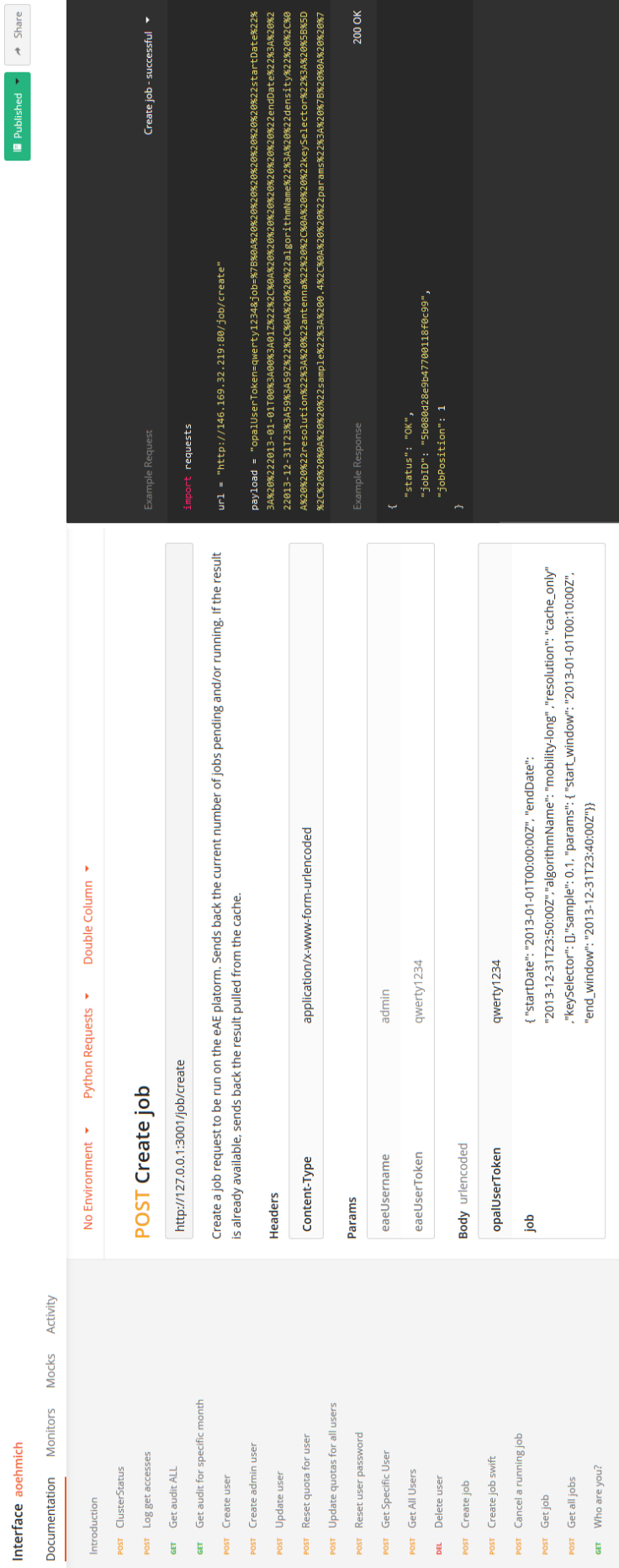


Figure 7.2: Illustration of the create job query in Postman with the type of request (POST), the URL, a description of the request, the parameters in the body of the request and a Python version example of the request. We can see as well an example response for the query with the associated code (200 in this instance).

7.1.2 Continuous integration and system deployment

Continuous Integration (CI) is a development practice that requires developers to integrate code into a shared repository at regular intervals. The main aim of CI is to limit integration problems. The check-ins are done against individual branches dedicated to a specific bug-fix or feature which are then merged into a general development branch. Each check-in is then verified by an automated build that compiles the code and runs the dedicated tests of the project, allowing teams to detect problems early. Those builds are carried out using Travis² which is GitHub's native build factory. Once the build is green (e.g. validated by the build factory), the repository is pushed to DockerHub³ to build the associated Docker container. Those containers become publicly available, once successfully built, for the projects to always deploy the latest version. To ensure the platform up-time is as close as possible to 100%, Docker-Compose (in autorestart mode) in a Docker Swarm environment is used for the deployment of the services on adopters' premises. This allows for the deployment of the services multiple times, across different clusterised host machines, for scalability and resilience purposes. The Docker files are all located in the projects under the DockerHub folder to ease the individual modifications of the microservices by the developers.

To ensure the robustness and quality of the platform, we have deployed a dependency management tool for Node.js projects called David⁴ and a dependency update automation called Renovate⁵. Those tools help ensure that the projects' dependencies are always up to date and that the updated dependencies do not break the projects.

7.1.3 Agile methods

Agile methodology has become very popular for software development in the last decade. Agile methodologies all started based on four core principles as outlined in the Agile Manifesto [BBB⁺01]. The first and foremost intent of agile methods is to enable complex systems

²<https://travis-ci.org/>

³<https://hub.docker.com/>

⁴<https://david-dm.org/>

⁵<https://github.com/renovatebot>

and product development with dynamic, non-deterministic and non-linear characteristics to be perennial and successful in their development.

We broke down development work into small increments that minimise the amount of up-front planning and design. Those increments were particularly well suited to the microservices architecture of the eAE. Each iteration represented a single microservice and involved a cross-functional analysis in all functions: planning, analysis, design, coding, unit testing, and acceptance testing. Our approach was very much alike to the test-driven development (TDD) and, as such, every new function added to the code base was accompanied by a set of tests. For the continuous integration part, we relied on Travis CI which helps automate the non-human part of the software development process with continuous integration and facilitating technical aspects of continuous delivery. Thus, at the end of each iteration, a new microservice was implemented, thoroughly tested, documentation was available and ready for production. That atomicity ensured that all necessary requirements were met. Once all iterations for a given Layer were completed, the microservices were integrated together and tested. Thanks to that incremental approach, the integration between services and the layers has been almost instantaneous and seamless and overall has saved a lot of development time.

The agile methods have enabled as well an easier communication with the two other persons who contributed to the platform. The daily stand-up meetings helped coordinate the development efforts, ensure the soundness of the designs and fluidify any impediments that could hinder their timely delivery.

Finally, making the code understandable quickly and easy to edit is key towards the sustainability of the platform as it would encourage adoption, lower entry costs and lower the cost of developing new features. The agile methods have helped us to keep the platform “evolvable” by decreasing the total cost of ownership of the application and helped us design the application in a way that facilitates the adoption of newer frameworks (if necessary), encourages extensive unit tests, detailed documentation with illustrations and reduces technical debt (duplicated/dead code or misleading comments).

7.2 Future of the platform

7.2.1 Adopters

The first adopters and strongest supporters have been the Data Science Institute and the eTRIKS project at Imperial College London. The adoption of the platform by some projects has already led to scientific contributions such as DeepSleepNet [SDWG17], FakeNews (publication pending) and OncoTrack [GYVdS⁺19]. We have also been able to support various analytical efforts in the context of the Innovative Medicines Initiative (IMI) and the European Union's Horizon 2020 projects such as BioVacSafe⁶, U-BIOPRED⁷, Pioneer⁸ and AiPBAND⁹. The ITMAT project ¹⁰, which is part of the National Institute for Health Research Imperial BRC, has also adopted the eAE as their analytical environment to propel their analyses.

Another major adopter of the eAE was International Business Machines Corporation (IBM). They added it to their portfolio of supported projects, and it is advertised as part of their large scale computing platforms and POWER architecture to their clients.

The most recent adopter has been the OPen Algorithm (OPAL) project as detailed in Chapter 5. They have made substantial contributions to the project through the financing of development work for the privacy related features and the use of the platform in the context of pilot projects with Orange-Sonatel in Senegal and Telefónica in Colombia. Those two projects are now running production deployments of the eAE, and around a dozen people (from the Senegalese government, United Nation, Agence Nationale de Statistique et de la Démographie, researchers from Orange and Telefónica among others) are actively using them in each country. There are, at the time of writing, an additional five other actors which are evaluating the adoption of the platform in their respective countries.

⁶<http://www.biovacsafe.eu/home>

⁷<https://www.europeanlung.org/en/projects-and-research/projects/u-biopred/home>

⁸<https://prostate-pioneer.eu/>

⁹<https://www.aipband-itn.eu>

¹⁰<https://www.imperial.ac.uk/itmat-data-science-group>

7.2.2 Community building

Community building is an important aspect of the life of an open source software as it ensures lasting support and development of the platform. We have started building a community around the eAE by presenting the work done at various conferences¹¹ and providing educational resources for parties interested in trying out the platform. Those educational resources are an important aspect of our Open Source efforts as they aim at lowering the entry cost for the adoption of the eAE and encourage peers to review the platform and give us feedback for improvements.

Those presentations have enabled new collaborations (with the United Nations for example) and created momentum around the project. Some branding and visual identity have been started to differentiate it from the other competitors and highlight its benefits. However, we believe that further work in that area would be beneficial to reflect a clear positive image to the collaborators. However, to ensure further extension of the platform, we believe we should engage further the medical and bioinformatics communities by inviting them to demonstrations of the platform and show how much they could benefit from it. Those communities are key for the future success of the platform to ensure the accumulation of credits and citations. The second community that will benefit greatly from the work done in the context of location data is telecommunication companies. They hold an enormous amount of location data through their Call Detail Records (CDRs) while being unable to use it efficiently for privacy reasons.

¹¹tranSMART conferences 2015 & 2016, BioTransR 2017, IEEE Big Data 2017, Early Warning Systems Workshop 2018, ICDCS 2018

Chapter 8

Conclusion

8.1 Summary of Thesis Achievements

The automated collection of biosignals (through IoT, connected devices, etc.), the recent developments of new technologies in both clinical and 'omics fields, the abundance of high-precision diagnostic devices (producing massive amounts of data) as well as the increasing inclination towards open science and public initiatives like UK Biobank lead to massive amounts of diverse data available to medical researchers. However, the availability of that data alone cannot enable a medical revolution and moving from a symptom- to evidence-based medical diagnosis (and treatment). In order to achieve that paradigm shift, new tools, infrastructure, and standards must be created.

In this thesis, we propose a new component-based, distributed framework for distributed data exploration and high-performance computing in order to leverage that humongous amount of data in an accessible and seamless fashion to any researcher. The concrete contributions of this thesis are as follows:

- **Platform:** The development of the eTRIKS Analytical Environment (eAE) as an answer to the needs of analysing and exploring massive amounts of medical data. The eAE is a modular framework which enables the analysis of medical data at scale. Its modular

architecture allows for the quick addition or replacement of analytics tools and modules with little overhead, thereby ensuring support of users as the data analytics needs and tools evolve. The eAE is flexible enough to support a variety of use cases across the biomedical domain from statistical analysis to machine learning and deep learning. Each component has been developed with resiliency and redundancy in mind to ensure that the platform delivers the highest uptime and performance possible. A multi-master scheduler has been specifically developed to handle internally all the scheduling and management of jobs and services. The languages currently supported are Python, R and C with the addition of Spark and TensorFlow.

- **Privacy:** The specialization of the eAE architecture into a privacy-preserving platform for location data and privacy preserving analytics in the context of public health. This work illustrated the modularity, resilience and scalability of the platform to support new classes of problems. We presented a formal verification as well for our population density algorithm, the privacy mechanisms that have been put in place on the platform and relevant use cases in the context of public health analytics using location data.
- **Analytics in life science:** To illustrate how the eTRIKS Analytical Environment can be used for managing and analysing large scale translational research data in tranSMART, we have implemented three bioinformatics analysis pipelines, including an iterative model generation and cross-validation pipeline for biomarker identification, a general statistical analysis pipeline for hypotheses testing, and a pathway enrichment pipeline using KEGG to demonstrate the performance of the proposed architecture. We have presented a proposed deep learning model as well, named DeepSleepNet, for automatic sleep stage scoring based on raw single-channel EEG and how the eAE has successfully and seamlessly supported the needs of the researchers for distributed deep learning computations at scale.
- **Extensibility of the eAE and identification of deceptive news:** Politically deceptive news have become a major challenge of our time, and its successful flagging a main source of concern for publishers, governments and social media. The approach we presented in the context of tweets collected during the 2016 US presidential election suc-

cessfully identifies characteristic features (including temporal diffusion and NLP) that can help in the process of automating the identification of tweets containing deceptive news. Subsequently, we make use of these features to propose a predictive ensemble model that allows to assess if a tweet contains a deceptive news or not.

8.2 Future Work

Even though we have developed an architecture for the efficient and scalable analysis of massive amounts of medical data, it is clear that many challenges remain in order for it to truly reach its full potential. Here, we discuss the remaining challenges and suggest potential future works to address them:

- The need for data privacy in the context of medical data is of paramount importance and the first efforts to address it have been started. Even though the issues are now well understood and legislation and guidelines have been implemented in recent years in a growing number of countries, the number of systems with a privacy by design are still extremely limited. In the future, we need to agree on a common platform to support those privacy requirements in the medical field while allowing the use of data distributed across countries to accelerate discoveries and facilitate reproducibility for a better science. In addition to the platform, a shift in paradigm on the design of analytics needs to take place to meet strict privacy rules similar to our implementation of the density algorithm presented in Chapter 5.
- The current implementation of the architecture do not integrate the concept of federation of the computation and storage layers. There is no technical limitation from an architectural standpoint and federation would enable researchers to leverage different data repositories for any analysis seamlessly and maximise the usage of the pooled resources. The compute service would have to be re-implemented from monolithic design to a master/slave design (already in use in grid computing) coupled with the integration of a grouping mechanisms for geolocality of the data. The development and integration of

that feature would probably display the full potential of the architecture and open new technical possibilities to researchers as demonstrated with the DeepSleepNet project in Section 6.2.

- As the dataset is only based on the US elections of 2016, it is likely that our proposed model for deceptive news detection might be overfitting. Indeed, the model we built might work well in this specific context but might not be applicable in another scenario. Our next steps would be to add data on other elections to make the model more robust for this context. Shared features are likely to be the ground truth to any model trying to identify political deceptive news. Regarding this generalisation effort, one area of improvement for our model would be to identify the actual context and the subject of the tweet automatically with NLP algorithms. Extensive amount of work has been carried out in this domain and we would start by reusing as much as possible this work. Once the subject and context are extracted, we could automatically target a specific model better suited for the identification. To further improve the accuracy, we plan on applying deep learning's convolutional neural networks to extract features from the profile pictures and the background pictures to create new derived features.

Bibliography

- [20117] GDPR et droit français : une ordonnance tardive et limitée en préparation, 2017.
- [AA16] J. Archenaa and E. A. Mary Anita. Interactive Big Data Management in Healthcare Using Spark. In *Proceedings of the 3rd International Symposium on Big Data and Cloud Computing Challenges*, pages 265–272. Springer, Cham, 2016.
- [AAoNLR⁺16] Yong American Academy of Neurology., Yang Li, Joel Raffel, Matt Craner, Cheryl Hemingway, Gavin Giovannoni, James Overell, Robert Hyde, Johan Van Beek, Fiona Thomas, Yike Guo, and Paul Matthews. *Neurology.*, volume 86. Advanstar Communications, 4 2016.
- [ABe00] Michael Ashburner, Catherine A. Ball, and et. al. Gene Ontology: tool for the unification of biology. *Nature Genetics*, 2000.
- [ABe13] Miguel E Andrés, Nicols E Bordenabe, and et al. Geo-Indistinguishability: Differential Privacy for Location-Based Systems. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*. ACM, 2013.
- [AC14] Gergely Acs and Claude Castelluccia. A Case Study: Privacy Preserving Release of Spatio-Temporal Density in Paris. In *KDD '14 Proc. of the 20th ACM SIGKDD int. conf.*, 8 2014.

- [ADLOMS17] Julio Amador Diaz Lopez, Axel Oehmichen, and Miguel Molina-Solana. Fake news on 2016 US elections viral tweets (November 2016 - March 2017), 11 2017.
- [AG05] Lada A Adamic and Natalie Glance. The political blogosphere and the 2004 U.S. election. In *Proceedings of the 3rd international workshop on Link discovery - LinkKDD '05*, New York, New York, USA, 2005. ACM Press.
- [AG17] Hunt Allcott and Matthew Gentzkow. Social Media and Fake News in the 2016 election. *Journal of Economic Perspectives*, 2017.
- [AH69] J. Allan Hobson. A manual of standardized terminology, techniques and scoring system for sleep stages of human subjects. *Electroencephalography and Clinical Neurophysiology*, 26(6):644, 6 1969.
- [And04] David P. Anderson. BOINC: A system for public-resource computing and storage. In *Proceedings - IEEE/ACM International Workshop on Grid Computing*, 2004.
- [ANe13] Fariba Aghajafari, Tharsiya Nagulesapillai, and et. al. Association between maternal serum 25-hydroxyvitamin D level and pregnancy and neonatal outcomes: systematic review and meta-analysis of observational studies. *BMJ*, 2013.
- [AOMS17a] Julio Amador, Axel Oehmichen, and Miguel Molina-Solana. Characterizing Political Fake News in Twitter by its Meta-Data. *arXiv*, 2017.
- [AOMS17b] Julio Amador, Axel Oehmichen, and Miguel Molina-Solana. Characterizing Political Fake News in Twitter by its Meta-Data. *arXiv*, 2017.
- [Ash15] Euan A. Ashley. The precision medicine initiative: A new national effort, 2015.
- [BBB⁺01] K. Beck, M. Beedle, A. Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries, J. Kern, B. Marick, R. C.

- Martin, S. Mellor, K. Schwaber, J. Sutherland, and D. Thomas. Manifesto for Agile Software Development, 2001.
- [BBM17] BBMRI-ERIC. The EU General Data Protection Regulation Answers to Frequently Asked Questions Updated Version 2.0. Technical report, BBMRI-ERIC, 2017.
- [BCJ⁺10] Robert E Black, Simon Cousens, Hope L Johnson, Joy E Lawn, Igor Rudan, Diego G Bassani, Prabhat Jha, Harry Campbell, Christa Fischer Walker, Richard Cibulskis, Thomas Eisele, Li Liu, Colin Mathers, and Child Health Epidemiology Reference Group of WHO and UNICEF. Global, regional, and national causes of child mortality in 2008: a systematic analysis. *The Lancet*, 375(9730):1969–1987, 6 2010.
- [BDK15] Vincent D Blondel, Adeline Decuyper, and Gautier Krings. A survey of results on mobile phone datasets analysis. *EPJ Data Science*, 4(1):10, 12 2015.
- [Bec80] Leland Beck. A Security Mechanism for Statistical Database. *ACM Trans. Database Syst.*, 1980.
- [BEC⁺12] Vincent D. Blondel, Markus Esch, Connie Chan, Fabrice Clerot, Pierre Deville, Etienne Huens, Frdric Morlot, Zbigniew Smoreda, and Cezary Ziemlicki. Data for Development: the D4D Challenge on Mobile Phone Data. *arXiv*, 9 2012.
- [Ben09] Y. Bengio. Learning Deep Architectures for AI. *Foundations and Trends® in Machine Learning*, 2009.
- [BFJ⁺12] Robert M Bond, Christopher J Fariss, Jason J Jones, Adam D I Kramer, Cameron Marlow, Jaime E Settle, and James H Fowler. A 61-million-person experiment in social influence and political mobilization. *Nature*, 2012.
- [BGJM16] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching Word Vectors with Subword Information. *arxiv*, 2016.

- [BGL10] D Boyd, S Golder, and G Lotan. Tweet, Tweet, Retweet: Conversational Aspects of Retweeting on Twitter. *Proc. 43rd Hawaii Int. Conf. on System Sciences*, 2010.
- [BGL⁺15] Linus Bengtsson, Jean Gaudart, Xin Lu, Sandra Moore, Erik Wetter, Kankoe Sallah, Stanislas Rebaudet, and Renaud Piarroux. Using Mobile Phone Data to Predict the Spatial Spread of Cholera. *Scientific Reports*, 2015.
- [BGO⁺16] Brendan Burns, Brian Grant, David Oppenheimer, Eric Brewer, and John Wilkes. Borg, Omega, and Kubernetes. *Queue*, 14(1):70–93, 1 2016.
- [BH00] I. A. Basheer and M. Hajmeer. Artificial neural networks: Fundamentals, computing, design, and application. *Journal of Microbiological Methods*, 2000.
- [Bis06] C. Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag New York Inc., 2006.
- [BJGS12] Ziv Bar-Joseph, Anthony Gitter, and Itamar Simon. Studying and modelling dynamic biological processes using time-series gene expression data. *Nature Reviews Genetics*, 2012.
- [BM18] Vian Bakir and Andrew McStay. Fake News and The Economy of Emotions: Problems, causes, solutions. *Digital Journalism*, 2018.
- [BMA15] Eytan Bakshy, Solomon Messing, and Lada A Adamic. Exposure to ideologically diverse news and opinion on Facebook. *Science*, 348(6239):1130–1132, 2015.
- [BOH11] Michael Bostock, Vadim Ogievetsky, and Jeffrey Heer. D3 data-driven documents. *IEEE Transactions on Visualization and Computer Graphics*, 2011.
- [Bon00] Andr B. Bondi. Characteristics of scalability and their impact on performance. In *Proceedings of the second international workshop on Software and performance - WOSP '00*, 2000.

- [Bor14] Nicols E Bordenabe. *Measuring Privacy with Distinguishability Metrics: Definitions, Mechanisms and Application to Location Privacy*. PhD thesis, École Polytechnique, 2014.
- [Car15] Carter Moore. How long does it take to pass a bill in the US? - Quora, 2015.
- [CCF⁺16] Yu-Ting Chen, Jason Cong, Zhenman Fang, Jie Lei, and Peng Wei. When Spark Meets FPGAs: A Case Study for Next-Generation {DNA} Sequencing Acceleration, 2016.
- [CCR15] Niall J. Conroy, Yimin Chen, and Victoria L. Rubin. Automatic Deception Detection: Methods for Finding Fake News. *Proceedings of the 78th ASIS&T Annual Meeting: Information Science with Impact: Research in and for the Community*, 2015.
- [CDI04] CDISC. Clinical Data Interchange Standards Consortium, 2004.
- [CEW⁺18] Hongming Chen, Ola Engkvist, Yinhai Wang, Marcus Olivecrona, and Thomas Blaschke. The rise of deep learning in drug discovery. *Drug Discovery Today*, 23(6):1241–1250, 6 2018.
- [CG16] Tianqi Chen and Carlos Guestrin. XGBoost: A Scalable Tree Boosting System. *KDD*, 2016.
- [CGe12] Ethan Cerami, Jianjiong Gao, and et. al. The cBio Cancer Genomics Portal: An open platform for exploring multidimensional cancer genomics data. *Cancer Discovery*, 2(5):401–404, 2012.
- [CHe09] Dan Cao, Mei Hou, and et. al. Expression of HIF-1alpha and VEGF in colorectal cancer: association with clinical outcomes and prognostic implications. *BMC Cancer*, 2009.
- [CMe15] Randall J. Cohrs, Tyler Martin, and et. al. Translational Medicine definition by the European Society for Translational Medicine, 2015.

- [Coh14] Mike X Cohen. *Analyzing Neural Time Series Data: Theory and Practice*. MIT Press, 2014.
- [Cos14] Fabricio F. Costa. Big data in biomedicine, 2014.
- [Cox07] D.R. Cox. *Principles of statistical inference*. arXiv, 2007.
- [CRe15] Vincent Canuel, Bastien Rance, and et. al. Translational research platforms integrating clinical and omics data: A review of publicly available solutions. *Briefings in Bioinformatics*, 16(2):280–290, 2015.
- [CRT17] Chris Culnane, Benjamin I P Rubinstein, and Vanessa Teague. Health Data in an Open World. *ArXiv e-prints*, 12 2017.
- [Csá01] B. Csáji. Approximation with artificial neural networks. *MSc. thesis*, 2001.
- [CSB14] Davide Chicco, Peter Sadowski, and Pierre Baldi. Deep autoencoder neural networks for gene ontology annotation predictions. In *Proceedings of the 5th ACM Conference on Bioinformatics, Computational Biology, and Health Informatics - BCB '14*, pages 533–540, New York, New York, USA, 2014. ACM Press.
- [Cur] Inc. Curoverse. Arvados — Open Source Big Data Processing and Bioinformatics.
- [Cyb89] Cybenko. Approximations by superpositions of sigmoidal functions. *Mathematics of Control, Signals, and Systems*, 1989.
- [CZG⁺17] Vivek Charu, Scott Zeger, Julia Gog, Ottar N. Bjørnstad, Stephen Kissler, Lone Simonsen, Bryan T. Grenfell, and Ccile Viboud. Human mobility and the spatial transmission of influenza in the United States. *PLOS Computational Biology*, 13(2):e1005382, 2 2017.
- [Dat14] Databricks. Apache Spark the fastest open source engine for sorting a petabyte, 2014.

- [Dea16] Jeffrey Dean. Large-Scale Deep Learning For Building Intelligent Computer Systems. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining - WSDM '16*, 2016.
- [Dee18] DeepMind. AlphaFold: Using AI for scientific discovery, 12 2018.
- [Den78] Dorothy E Denning. Are statistical data bases secure. In *Proc. AFIPS*, 1978.
- [DG08a] Jeffrey Dean and Sanjay Ghemawat. MapReduce. *Communications of the ACM*, 2008.
- [DG08b] Jeffrey Dean and Sanjay Ghemawat. MapReduce: simplified data processing on large clusters. *Communications of the ACM*, 2008.
- [DJS14] George E. Dahl, Navdeep Jaitly, and Ruslan Salakhutdinov. Multi-task Neural Networks for QSAR Predictions. *ArXiv*, 6 2014.
- [DLM⁺14] Pierre Deville, Catherine Linard, Samuel Martin, Marius Gilbert, Forrest R Stevens, Andrea E Gaughan, Vincent D Blondel, and Andrew J Tatem. Dynamic population mapping using mobile phone data. *Proceedings of the National Academy of Sciences of the United States of America*, 2014.
- [dMHVB13] Yves-Alexandre de Montjoye, Csar A. Hidalgo, Michel Verleysen, and Vincent D. Blondel. Unique in the Crowd: The privacy bounds of human mobility. *Scientific Reports*, 2013.
- [DMNS06] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating Noise to Sensitivity in Private Data Analysis. In *Theory of Cryptography*. Springer, 2006.
- [dMRP16] Yves-Alexandre de Montjoye, Luc Rocher, and Alex Sandy Pentland. bandicoot: a Python Toolbox for Mobile Phone Metadata. *Journal of Machine Learning Research*, 2016.
- [dMRSP15] Yves-Alexandre de Montjoye, Laura Radaelli, Vivek Kumar Singh, and Alex Sandy Pentland. Identity and privacy. Unique in the shopping mall:

- on the reidentifiability of credit card metadata. *Science (New York, N.Y.)*, 2015.
- [dMST⁺14] Yves-Alexandre de Montjoye, Zbigniew Smoreda, Romain Trinquart, Cezary Ziemlicki, and Vincent D. Blondel. D4D-Senegal: The Second Mobile Phone Data for Development Challenge. *arXiv*, 2014.
- [dMSWP14] Yves-Alexandre de Montjoye, Erez Shmueli, Samuel S. Wang, and Alex Sandy Pentland. openPDS: Protecting the Privacy of Metadata through SafeAnswers. *PLoS ONE*, 2014.
- [DN03] Irit Dinur and Kobbi Nissim. Revealing information while preserving privacy. In *ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. ACM, 2003.
- [DQR12] Jens Dittrich and Jorge-Arnulfo Quiané-Ruiz. Efficient big data processing in Hadoop MapReduce. *Proceedings of the VLDB Endowment*, 5(12):2014–2015, 8 2012.
- [DR13] Cynthia Dwork and Aaron Roth. The Algorithmic Foundations of Differential Privacy. *Foundations and Trends in Theoretical Computer Science*, 2013.
- [DRW07] Leticia Duboc, David Rosenblum, and Tony Wicks. A framework for characterization and analysis of software system scalability. In *Proceedings of the the 6th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering - ESEC-FSE '07*, 2007.
- [DS10] Cynthia Dwork and Adam Smith. Differential Privacy for Statistics: What We Know and What We Want to Learn. *Journal of Privacy and Confidentiality*, 2010.
- [DSM⁺17] H. Dong, A. Supratak, L. Mai, F. Liu, A. Oehmichen, S. Yu, and Y. Guo. TensorLayer: A versatile library for efficient deep learning development. In *MM 2017 - Proceedings of the 2017 ACM Multimedia Conference*, 2017.

- [DSSU17] Cynthia Dwork, Adam Smith, Thomas Steinke, and Jonathan Ullman. Exposed! A Survey of Attacks on Private Data. *Annual Review of Statistics and Its Application*, 2017.
- [DY14] Li Deng and Dong Yu. Deep Learning: Methods and Applications, 5 2014.
- [EKe17] Andre Esteva, Brett Kuprel, and et. al. Dermatologist-level classification of skin cancer with deep neural networks. *Nature*, 2017.
- [ERAEB05] Hesham El-Rewini and Mostafa Abd-El-Barr. *Advanced Computer Architecture and Parallel Processing*. arXiv, 2005.
- [Evg18] Evgeny Poberezkin. Ajv - Another JSON Schema Validator, 2018.
- [Fei15] Dror G Feitelson. *Workload Modeling for Computer Systems Performance Evaluation*. Cambridge University Press, New York, NY, USA, 1st edition, 2015.
- [FF14] Benedikt Fecher and Sascha Friesike. Open Science: One Term, Five Schools of Thought. In *Opening Science*, pages 17–47. Springer International Publishing, Cham, 2014.
- [FGM⁺16] Flavio Finger, Tina Genolet, Lorenzo Mari, Guillaume Constantin de Magny, Nol Magloire Manga, Andrea Rinaldo, and Enrico Bertuzzo. Mobile phone data highlights the role of mass gatherings in the spreading of cholera outbreaks. *Proceedings of the National Academy of Sciences of the United States of America*, 113(23):6421–6, 6 2016.
- [Fis35] R. A. Fisher. The Logic of Inductive Inference. *Journal of the Royal Statistical Society*, 1935.
- [FNR17] D J Flynn, Brendan Nyhan, and Jason Reifler. The Nature and Origins of Misperceptions: Understanding False and Unsupported Beliefs about Politics. *Advances in Political Psychology*, 2017.

- [FPEM17] Paul Francis, Sebastian Probst Eide, and Reinhard Munz. Diffix: High-Utility Database Anonymization. In *Privacy Technologies and Policy*. Springer International Publishing, 2017.
- [FPEO⁺18] P Francis, S Probst-Eide, P Obrok, C Berneanu, S Juric, and R Munz. Extended Diffix. *ArXiv e-prints*, 6 2018.
- [Fra17] Paul Francis. MyData 2017 Workshop Abstract: Technical Issues and Approaches in Personal Data Management. <https://aircloak.com/mydata-2017-workshop-abstract-technical-issues-and-approaches-in-personal-data-management>, 2017.
- [GAG⁺00] Ary L. Goldberger, Luis A. N. Amaral, Leon Glass, Jeffrey M. Hausdorff, Plamen Ch. Ivanov, Roger G. Mark, Joseph E. Mietus, George B. Moody, Chung-Kang Peng, and H. Eugene Stanley. PhysioBank, PhysioToolkit, and PhysioNet: Components of a New Research Resource for Complex Physiologic Signals. *Circulation*, pages 591–596, 2000.
- [Gal17] Galaxy Project. GitHub - galaxyproject/pulsar: Distributed job execution application built for Galaxy, 2017.
- [GBH17] Alec Go, Richa Bhayani, and Lei Huang. Twitter Sentiment Classification using Distant Supervision. *arXiv*, 2017.
- [GCe04] Robert Gentleman, Vincent Carey, and et. al. Bioconductor: open software development for computational biology and bioinformatics. *Genome Biology*, 2004.
- [GF08] Nicholas C. Grassly and Christophe Fraser. Mathematical models of infectious disease transmission. *Nature Reviews Microbiology*, 6(6):477–487, 6 2008.
- [GHB08] Marta C. González, Csar A. Hidalgo, and Albert-Lszl Barabási. Understanding individual human mobility patterns. *Nature*, 453(7196):779–782, 6 2008.

- [GHRdM18] Andrea Gadotti, Florimond Houssiau, Luc Rocher, and Yves-Alexandre de Montjoye. When the Signal Is in the Noise: The Limits of Diffix’s Sticky Noise. *arXiv:1804.06752 [cs]*, 4 2018.
- [GHS16] Erik Gawehn, Jan A. Hiss, and Gisbert Schneider. Deep Learning in Drug Discovery. *Molecular Informatics*, 35(1):3–14, 1 2016.
- [Glo] Glosser.ca. File:Colored neural network.svg - Wikimedia Commons.
- [Glu05] C. Gluud. Evidence based diagnostics. *BMJ*, 2005.
- [GNT⁺10] Jeremy Goecks, Anton Nekrutenko, James Taylor, Enis Afgan, Guruprasad Ananda, Dannon Baker, Dan Blankenberg, Ramkrishna Chakrabarty, Nate Coraor, Jeremy Goecks, Greg Von Kuster, Ross Lazarus, Kanwei Li, James Taylor, and Kelly Vincent. Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences. *Genome Biology*, 2010.
- [Gooa] Google. Balancing Strong and Eventual Consistency with Google Cloud Datastore — Cloud Datastore Documentation — Google Cloud.
- [Goob] Google. Quantifying the performance of the TPU, our first machine learning chip — Google Cloud Blog.
- [Gor17] Ben Gorman. Gradient Boosting Explained — GormAnalysis, 2017.
- [GPAM⁺14] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Nets, 2014.
- [GPe16] Varun Gulshan, Lily Peng, and et. al. Development and Validation of a Deep Learning Algorithm for Detection of Diabetic Retinopathy in Retinal Fundus Photographs. *JAMA: Journal of the American Medical Association*, 2016.

- [GWBV02] Isabelle Guyon, Jason Weston, Stephen Barnhill, and Vladimir Vapnik. Gene Selection for Cancer Classification using Support Vector Machines. *Machine Learning*, 2002.
- [GYVdS⁺19] Wei Gu, Reha Yildirimman, Emmanuel Van der Stuyft, Denny Verbeeck, and et al. Data and knowledge management in translational research: implementation of the eTRIKS platform for the IMI OncoTrack consortium. *BMC Bioinformatics*, 20(1):164, 12 2019.
- [HA17] Thomas Heinis and Anastasia Ailamaki. Data Infrastructure for Medical Research. *Foundations and Trends in Databases*, 2017.
- [Has95] Mohamad H. Hassoun. *Fundamentals of artificial neural networks*. MIT Press, 1995.
- [Hen95] Robert L. Henderson. Job scheduling under the Portable Batch System. In *Job scheduling under the Portable Batch System*, pages 279–294. Springer, Berlin, Heidelberg, 1995.
- [HIL65] A B HILL. The environment and disease: association or causation? *Proceedings of the Royal Society of Medicine*, 1965.
- [Hin09] Geoffrey Hinton. Deep belief networks. *Scholarpedia*, 4(5):5947, 2009.
- [HKP⁺11] Benjamin Hindman, Andy Konwinski, A Platform, Fine-Grained Resource, and Matei Zaharia. Mesos: A platform for fine-grained resource sharing in the data center. *Proceedings of the ...*, 2011.
- [Hor91] Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 1991.
- [HSK⁺12] Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *ArXiv*, 7 2012.

- [HSL14] Brian Hall, Jeremy Selan, and Steve LaVietes. Katana's Geolib. In *ACM SIGGRAPH 2014 Talks on - SIGGRAPH '14*, pages 1–1, New York, New York, USA, 2014. ACM Press.
- [Hun07] John D. Hunter. Matplotlib: A 2D graphics environment. *Computing in Science and Engineering*, 2007.
- [HZRS16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2016.
- [IACe07] Iber C, Ancoli-Israel S, Chesson AL Jr., and et. al. The AASM Manual for the Scoring of Sleep and Associated Events: Rules, Terminology and Technical Specifications. In *AASM Manual for Scoring Sleep*. AASM, 2007.
- [IBMa] IBM. IBM Platform LSF V9.1.3 documentation.
- [IBMb] IBM. IBM Spectrum Conductor.
- [IBMc] IBM Big Data Hub. The Four V's of Big Data.
- [iDM18] iDMC. Global Report on Internal Displacement. Technical report, iDMC, 2018.
- [ILC13] Thomas R. Insel, Story C. Landis, and Francis S. Collins. The NIH BRAIN Initiative, 2013.
- [Int] International Standardization Organization. ISO 25237:2017 - Health informatics – Pseudonymization.
- [Iof17] Julia Ioffe. The History of Russian Involvement in America's Race Wars, 2017.
- [IWCCtT04] France) IFIP World Computer Congress (18th : 2004 : Toulouse. *History of computing in education : IFIP 18th World Congress, TC3/TC9 1st Conference on the History of Computing in Education, 22-27 August 2004, Toulouse, France*. Kluwer Academic Publishers, 2004.

- [JB16] Pranav Joshi and Muda Rajesh Babu. Openlava: An open source scheduler for high performance computing. In *International Conference on Research Advances in Integrated Navigation Systems, RAINS 2016*, 2016.
- [JBBe17] Norman P. Jouppi, Al Borchers, Rick Boyle, and et al. In-Datacenter Performance Analysis of a Tensor Processing Unit. *ACM SIGARCH Computer Architecture News*, 2017.
- [JC16] Ian T. Jolliffe and Jorge Cadima. Principal component analysis: a review and recent developments. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 2016.
- [JGBM16] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of Tricks for Efficient Text Classification. *arxiv*, 2016.
- [JNS17] Noah Johnson, Joseph P Near, and Dawn Song. Towards Practical Differential Privacy for SQL Queries. *ArXiv e-prints*, 2017.
- [Kar18] Karl Rupp. 42 Years of Microprocessor Trend Data — Karl Rupp, 2018.
- [KBCG03] Yuval Kluger, Ronen Basri, Joseph T. Chang, and Mark Gerstein. Spectral biclustering of microarray data: Coclustering genes and conditions, 2003.
- [KBe11] Richard D. Kennedy, Max Bylesjo, and et. al. Development and independent validation of a prognostic assay for stage ii colon cancer using formalin-fixed paraffin-embedded tissue. *Journal of Clinical Oncology*, 2011.
- [KG00] M Kanehisa and S Goto. KEGG: kyoto encyclopedia of genes and genomes. *Nucleic acids research*, 2000.
- [KH12] Alex Krizhevsky and Geoffrey E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. *Neural Information Processing Systems*, 2012.
- [KRKP⁺16] Thomas Kluyver, Benjamin Ragan-Kelley, Fernando Pérez, Brian Granger, and et al. Jupyter Notebooks - a publishing format for reproducible computa-

- tional workflows. In *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, 2016.
- [KSTA15] Olga Kolchyna, Tharsis T. P. Souza, Philip Treleaven, and Tomaso Aste. Twitter Sentiment Analysis: Lexicon Method, Machine Learning Method and Their Combination. *arxiv*, 2015.
- [KZT⁺00] Bastiaan Kemp, Aeilko H. Zwinderman, Bert Tuk, Hilbert A C Kamphuisen, and Josefien J L Oberyé. Analysis of a sleep-dependent neuronal feedback loop: The slow-wave microcontinuity of the EEG. *IEEE Trans. Biomed. Eng.*, 47(9):1185–1194, 2000.
- [Lan01] Doug Laney. 3D Data Management: Controlling Data Volume, Velocity, and Variety, 2001.
- [LBBe18] David M. J. Lazer, Matthew A. Baum, Yochai Benkler, and et. al. The science of fake news. *Science*, 2018.
- [LBH12] Xin Lu, Linus Bengtsson, and Petter Holme. Predictability of population displacement after the 2010 Haiti earthquake. *Proceedings of the National Academy of Sciences of the United States of America*, 109(29):11576–81, 7 2012.
- [LBH15] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 5 2015.
- [Lig16] Lightning Viz team. Lightning Viz, 2016.
- [LL73] C. L. Liu and James W. Layland. Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment. *Journal of the ACM*, 20(1):46–61, 1 1973.
- [LLM88] M.J. Litzkow, M. Livny, and M.W. Mutka. Condor-a hunter of idle workstations. In *Proceedings. The 8th International Conference on Distributed*, pages 104–111. IEEE Comput. Soc. Press, 1988.

- [Lug09] George F Luger. *Artificial Intelligence: Structures and Strategies for Complex Problem Solving*. Addison Wesley, 2009.
- [MA63] M.G. Kendall and A. Stuart. *The Advanced Theory of Statistics, Volume 1 Distribution Theory*. Journal of the Staple Inn Actuarial Society, 1963.
- [Mar13] Vivien Marx. Drilling into big cancer-genome data. *Nature Methods*, 2013.
- [MCCD13] T Mikolov, K Chen, G Corrado, and J Dean. Efficient estimation of word representations in vector space. *NIPS*, 2013.
- [MCMNS15] Eduardo Alejandro Martinez-Cesena, Pierluigi Mancarella, Mamadou Ndiaye, and Markus Schlöpfer. Using Mobile Phone Data for Electricity Infrastructure Planning. *arXiv*, 4 2015.
- [McS09] Frank D McSherry. Privacy Integrated Queries: An Extensible Platform for Privacy-preserving Data Analysis. In *Proceedings of the 2009 ACM SIGMOD*, 2009.
- [McS18] Frank McSherry. Uber’s differential privacy .. probably isn’t. <https://github.com/frankmcsherry/blog/blob/master/posts/2018-02-25.md>, 2018.
- [MDe08] Patrick McConnell, Rajesh C Dash, and et. al. The cancer translational research informatics platform. *BMC medical informatics and decision making*, 8:60, 2008.
- [Mea12] Prashanth Mohan and Thakurta et al. GUPT: Privacy Preserving Data Analysis Made Easy. In *Proceedings of the 2012 ACM SIGMOD*, 2012.
- [Met10] Michael L. Metzker. Sequencing technologies the next generation, 2010.
- [MGAS12] Bjoern H. Menze, Ezequiel Geremia, Nicholas Ayache, and Gbor Székely. Segmenting Glioma in Multi-Modal Images using a Generative-Discriminative Model for Brain Lesion Segmentation, 2012.

- [MGe11] Subha Madhavan, Yuriy Gusev, and et. al. G-DOC: a systems medicine platform for personalized oncology. *Neoplasia (New York, N.Y.)*, 13(9):771–83, 2011.
- [MH13] A B M Moniruzzaman and Syed Akhter Hossain. NoSQL Database: New Era of Databases for Big data Analytics - Classification, Characteristics and Comparison. *arXiv*, 6 2013.
- [MHBe10] Aaron McKenna, Matthew Hanna, Eric Banks, and et al. The Genome Analysis Toolkit: a MapReduce framework for analyzing next-generation DNA sequencing data. *Genome research*, 20(9):1297–303, 9 2010.
- [Mit97] Tom M. (Tom Michael) Mitchell. *Machine Learning*. McGraw Hill, 1997.
- [MJ51] Frank J. Massey and Jr. The Kolmogorov-Smirnov Test for Goodness of Fit. *Journal of the American Statistical Association*, 46(253):68, 3 1951.
- [MLDD16] Hui Miao, Ang Li, Larry S. Davis, and Amol Deshpande. ModelHub: Towards Unified Data and Lifecycle Management for Deep Learning. *arXiv*, 11 2016.
- [MMe06] S Murphy, M Mendis, and et. al. Integration of clinical and genetic data in the i2b2 architecture. *AMIA Annu Symp Proc*, page 1040, 2006.
- [MMIGRGE03] Carnegie Mellon Mike Mesnier, Intel, Carnegie Mellon Gregory R. Ganger, and Erik Riedel. Object-Based Storage. Technical report, Seagate Research, 2003.
- [MML⁺11] Henry Markram, Karlheinz Meier, Thomas Lippert, Sten Grillner, Richard Frackowiak, and et al. Introducing the Human Brain Project. In *Procedia Computer Science*, 2011.
- [MMSW07] Maged Michael, Jos E. Moreira, Doron Shiloach, and Robert W. Wisniewski. Scale-up x scale-out: A case study using nutch/Lucene. In *Proceedings - 21st International Parallel and Distributed Processing Symposium, IPDPS 2007; Abstracts and CD-ROM*, 2007.

- [MNB⁺17] Marcus R. Munafò, Brian A. Nosek, Dorothy V. M. Bishop, Katherine S. Button, Christopher D. Chambers, Nathalie Percie du Sert, Uri Simonsohn, Eric-Jan Wagenmakers, Jennifer J. Ware, and John P. A. Ioannidis. A manifesto for reproducible science. *Nature Human Behaviour*, 1(1):0021, 1 2017.
- [Mos48] Frederick Mosteller. A k-Sample Slippage Test for an Extreme Population. *The Annals of Mathematical Statistics*, 1948.
- [Mov] Movidius. Movidius powersworld’s most intelligent drone.
- [MT07] F McSherry and K Talwar. Mechanism Design via Differential Privacy. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS’07)*, 2007.
- [MT13] Saif M. Mohammad and Peter D. Turney. Crowdsourcing a Word-Emotion Association Lexicon. *arxiv*, 8 2013.
- [Mur12] Kevin P. Murphy. *Machine Learning: A Probablistic Perspective*. MIT Press, 2012.
- [MVL⁺12] Christopher J L Murray, Theo Vos, Rafael Lozano, Mohsen Naghavi, Abraham D Flaxman, Catherine Michaud, Majid Ezzati, Kenji Shibuya, Joshua A Salomon, Safa Abdalla, Victor Aboyans, and et al. Disability-adjusted life years (DALYs) for 291 diseases and injuries in 21 regions, 19902010: a systematic analysis for the Global Burden of Disease Study 2010. *The Lancet*, 380(9859):2197–2223, 12 2012.
- [Nat] National Institutes of Health. Big Data to Knowledge — NIH Common Fund.
- [NP06] T. Newhouse and J. Pasquale. ALPS: An Application-Level Proportional-Share Scheduler. In *15th IEEE International Conference on High Performance Distributed Computing*, pages 279–290. IEEE, 2006.
- [NS08] A Narayanan and V Shmatikov. Robust De-anonymization of Large Sparse Datasets. In *IEEE Symposium on Security and Privacy*, 2008.

- [NVI] NVIDIA. Autonomous Car Development Platform from NVIDIA DRIVE PX2.
- [ODC⁺08] Brian D. O'Connor, Allen Day, Scott Cain, Olivier Arnaiz, Linda Sperling, and Lincoln D. Stein. GMODWeb: A web framework for the generic model organism database. *Genome Biology*, 2008.
- [OGA⁺18] Axel Oehmichen, Florian Guitton, Paul Agapow, Ibrahim Emam, and Yike Guo. A multi tenant computational platform for translational medicine. In *Proceedings - International Conference on Distributed Computing Systems*, 2018.
- [OGCN14] Christian O'Reilly, Nadia Gosselin, Julie Carrier, and Tore Nielsen. Montreal Archive of Sleep Studies: an open-access resource for instrument benchmarking and exploratory research. *J. of Sleep Research*, 23(6):628–635, 2014.
- [OGS⁺99] Hiroyuki Ogata, Susumu Goto, Kazushige Sato, Wataru Fujibuchi, Hide-masa Bono, and Minoru Kanehisa. KEGG: Kyoto encyclopedia of genes and genomes, 1999.
- [Ohm10] P Ohm. Broken promises of privacy: Responding to the surprising failure of anonymization. *UCLA Law Rev.*, 2010.
- [OKHC14] Ali Oghabian, Sami Kilpinen, Sampsa Hautaniemi, and Elena Czeizler. Bi-clustering methods: Biological relevance and application in gene expression analysis. *PLoS ONE*, 2014.
- [OMBe12] Lucila Ohno-Machado, Vineet Bafna, and et. al. iDASH: integrating data for analysis, anonymization, and sharing. *Journal of the American Medical Informatics Association*, 19(2):196–201, 2012.
- [Opea] OpenStack Foundation. Heat - OpenStack.
- [Opeb] OpenStack Foundation. OpenStack Docs: Welcome to Glances documentation!
- [Opec] OpenStack Foundation. OpenStack Swift.

- [OPSS11] Joseph O Ogutu, Hans-Peter Piepho, and Torben Schulz-Streeck. A comparison of random forests, boosting and support vector machines for genomic selection. *BMC proceedings*, 2011.
- [OTPG17] Simon Oya, Carmela Troncoso, and Fernando Pérez-González. Is Geo-Indistinguishability What You Are Looking For? *arXiv:1709.06318 [cs]*, 9 2017.
- [PCR17] Gordon Pennycook, Tyrone D Cannon, and David G Rand. Prior Exposure Increases Perceived Accuracy of Fake News, 8 2017.
- [PDG15] Neeti Pokhriyal, Wen Dong, and Venu Govindaraju. Virtual Networks and Poverty Analysis in Senegal. *ArXiv*, 6 2015.
- [Pea01] Karl Pearson. On lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, 11 1901.
- [PFL10] JAUME PELLICER, MICHAEL F. FAY, and ILIA J. LEITCH. The largest eukaryotic genome of them all? *Botanical Journal of the Linnean Society*, 164(1):10–15, 9 2010.
- [PGM14] Davide Proserpio, Sharon Goldberg, and Frank McSherry. Calibrating Data to Sensitivity in Private Data Analysis: A Platform for Differentially-private Analysis of Weighted Datasets. *Proceedings VLDB Endowment*, 2014.
- [PMG98] D. L. Poole, Alan Mackworth, and R. G. Goebel. Computational Intelligence and Knowledge. In *Computational Intelligence: A Logical Approach*. Oxford University Press, 1998.
- [Pol12] Danielle C Polage. Making up History: False Memories of Fake News Stories. *Europe’s Journal of Psychology*, 2012.
- [Pos14] PostgreSQL. PostgreSQL: The world’s most advanced open source database, 2014.

- [PPS15] T. Pohanka, V. Pechanec, and M. Solanska. Synchronization and replication of geodata in the ESRI platform. In *SGEM*, 2015.
- [PR17] Gordon Pennycook and David G Rand. Who Falls for Fake News? The Roles of Analytic Thinking, Motivated Reasoning, Political Ideology, and Bullshit Receptivity, 9 2017.
- [PTB⁺17] Cecilia Panigutti, Michele Tizzoni, Paolo Bajardi, Zbigniew Smoreda, and Vittoria Colizza. Assessing the use of mobile phone data to describe recurrent mobility patterns in spatial epidemic models. *Royal Society open science*, 4(5):160950, 5 2017.
- [PVe11] Fabian Pedregosa, Gal Varoquaux, and et. al. Scikit-learn: Machine Learning in Python. *The Journal of Machine Learning Research*, 2011.
- [Pyt] Python Software Foundation. A fast PostgreSQL Database Client Library for Python/asyncio.
- [QEG⁺10] Judy Qiu, Jaliya Ekanayake, Thilina Gunarathne, Jong Choi, Seung-Hee Bae, Hui Li, Bingjing Zhang, Tak-Lon Wu, Yang Ruan, Saliya Ekanayake, Adam Hughes, and Geoffrey Fox. Hybrid cloud and cluster computing paradigms for life science applications. *BMC Bioinformatics*, 11(Suppl 12):S3, 12 2010.
- [Qua] Qualcomm. Qualcomm Research brings server-class machine learning to everyday devicesmaking them smarter — Qualcomm.
- [R C14] R Core Team. R: A Language and Environment for Statistical Computing, 2014.
- [RBA⁺18] Albert Reuther, Chansup Byun, William Arcand, David Bestor, Bill Bergeron, Matthew Hubbell, Michael Jones, Peter Michaleas, Andrew Prout, Antonio Rosa, and Jeremy Kepner. Scalable system scheduling for HPC and big data. *Journal of Parallel and Distributed Computing*, 111:76–92, 1 2018.

- [RBG17] Kew Royal Botanic Gardens. State of the World's Plants. Technical report, Royal Botanic Gardens, London, 2017.
- [RCC15] Victoria L. Rubin, Yimin Chen, and Niall J. Conroy. Deception detection for news: three types of fakes. *Proceedings of the 78th ASIS&T Annual Meeting: Information Science with Impact: Research in and for the Community*, 2015.
- [RD10] Guido Van Rossum and Fred L Drake. Python Tutorial. *History*, 2010.
- [Ril07] S. Riley. Large-Scale Spatial-Transmission Models of Infectious Disease. *Science*, 316(5829):1298–1301, 6 2007.
- [Riv92] R. Rivest. The MD5 Message-Digest Algorithm RFC 1321. *arXiv*, 1992.
- [RN12] Stuart Russel and Peter Norvig. *Artificial intelligencea modern approach 3rd Edition*. arXiv, 2012.
- [Rot13] Mark A Rothstein. HIPAA Privacy Rule 2.0. *Journal of Law, Medicine & Ethics*, 2013.
- [RSe10] Indrajit Roy, Srinath T V Setty, and et al. Airavat: Security and Privacy for MapReduce. In *NSDI*, 2010.
- [RSM17] Stefania Rubrichi, Zbigniew Smoreda, and Mirco Musolesi. A Comparison of Spatial-based Targeted Disease Containment Strategies using Mobile Phone Data. 6 2017.
- [SAE12] Omar Sefraoui, Mohammed Aissaoui, and Mohsine Eleuldj. OpenStack: Toward an Open-Source Solution for Cloud Computing. *International Journal of Computer Applications*, 2012.
- [SB16] R S Sutton and A G Barto. *Reinforcement Learning : An Introduction*. MIT Press, 2016.
- [SBLE17] Briony Swire, Adam J Berinsky, Stephan Lewandowsky, and Ullrich K H Ecker. Processing political misinformation: comprehending the Trump phenomenon. *Royal Society Open Science*, 4, 2017.

- [SCe17] Chengcheng Shao, Giovanni Luca Ciampaglia, and et. al. The spread of misinformation by social bots. *ArXiv*, 2017.
- [Sch15] Jrgen Schmidhuber. Deep Learning in neural networks: An overview, 2015.
- [SDWG17] Akara Supratak, Hao Dong, Chao Wu, and Yike Guo. DeepSleepNet: a Model for Automatic Sleep Stage Scoring based on Raw Single-Channel EEG. *arXiv*, 3 2017.
- [Set17] Ricky J. Sethi. Spotting Fake News: A Social Argumentation Framework for Scrutinizing Alternative Facts. In *2017 IEEE International Conference on Web Services (ICWS)*. IEEE, 2017.
- [SGG05] Abraham Silberschatz, Peter Baer Galvin, and Greg Gagne. *Operating Systeme Concepts*. arXiv, 2005.
- [SHM90] A F Subar, L C Harlan, and M E Mattson. Food and nutrient intake differences between smokers and non-smokers in the US. *American journal of public health*, 1990.
- [SHMe16] David Silver, Aja Huang, Chris J. Maddison, and et al. Mastering the game of Go with deep neural networks and tree search. *Nature*, 2016.
- [SJFL⁺16] Aarti Sathyanarayana, Shafiq Joty, Luis Fernandez-Luque, Ferda Ofli, Jaideep Srivastava, Ahmed Elmagarmid, Teresa Arora, and Shahrad Taheri. Sleep Quality Prediction From Wearable Data Using Deep Learning. *JMIR mHealth and uHealth*, 4(4):e125, 11 2016.
- [SKAEMW13] Malte Schwarzkopf, Andy Konwinski, Michael Abd-El-Malek, and John Wilkes. Omega. In *Proceedings of the 8th ACM European Conference on Computer Systems - EuroSys '13*, page 351, New York, New York, USA, 2013. ACM Press.

- [SKBBT09] R. D Smith, M. R Keogh-Brown, T. Barnett, and J. Tait. The economy-wide impact of pandemic influenza on the UK: a computable general equilibrium modelling experiment. *BMJ*, 339(nov19 1):b4571–b4571, 11 2009.
- [SKKP10] Sndor Szalma, Venkata Koka, Tatiana Khasanova, and Eric D Perakslis. Effective knowledge management in translational medicine. *Journal of translational medicine*, 8:68, 1 2010.
- [SKWB10] Chaoming Song, Tal Koren, Pu Wang, and Albert-Lszl Barabási. Modelling the scaling properties of human mobility. *Nature Physics*, 6(10):818–823, 10 2010.
- [SLC17] Shiliang Sun, Chen Luo, and Junyu Chen. A review of natural language processing techniques for opinion mining systems. *Information Fusion*, 2017.
- [SLJ⁺15] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2015.
- [SMe10] Kazuro Shimokawa, Kaoru Mogushi, and et. al. iCOD : an integrated clinical omics database based on the systems-pathology view of disease. *BMC Genomics*, 11(Suppl 4):S19, 2010.
- [SNK⁺17] Eitam Sheetrit, Nir Nissim, Denis Klimov, Lior Fuchs, Yuval Elovici, and Yuval Shahr. Temporal Pattern Discovery for Accurate Sepsis Diagnosis in ICU Patients. *ArXiv*, 9 2017.
- [SOS⁺67] Marshall S. Smith, Daniel M. Ogilvia, Philip J. Stone, Dexter C. Dunphy, and John J. Hartman. The General Inquirer: A Computer Approach to Content Analysis, 1967.
- [SR17] Julia Silge and David Robinson. *Text mining with R : a tidy approach*. O’Reilly, 2017.

- [SRSe12] Susanna-Assunta Sansone, Philippe Rocca-Serra, and et. al. Toward interoperable bioscience data. *Nature Genetics*, 2012.
- [ST12] Nigam H. Shah and Jessica D. Tenenbaum. The coming age of data-driven medicine: Translational bioinformatics’ next frontier. *Journal of the American Medical Informatics Association*, 2012.
- [Sta] Stanford University CS231. Stanford University CS231: Convolutional Neural Networks for Visual Recognition.
- [Swe97] L Sweeney. Weaving technology and policy together to maintain confidentiality. *J. Law Med. Ethics*, 1997.
- [SYL13] Daniel L Silver, Qiang Yang, and Lianghao Li. Lifelong Machine Learning Systems: Beyond Learning Algorithms. Technical report, aaai, 2013.
- [Tam18] Olivia Tambou. THE FRENCH ADAPTATION OF THE GDPR. Technical report, Université Paris-Dauphine, 2018.
- [Tay10] Ronald C Taylor. An overview of the Hadoop/MapReduce/HBase framework and its current applications in bioinformatics. *BMC Bioinformatics*, 11(Suppl 12):S1, 12 2010.
- [Tho17] Derek Thompson. What Facebook and Google Can Learn From the First Major News Hoax, 2017.
- [Tim] Inc. Timescale. Timescale: Open-source time-series database powered by PostgreSQL. www.timescale.com/.
- [TT16] The European Parliament and The European Council. General Data Protection Regulation. *Official Journal of the European Union*, 2016.
- [TTD11] Alan Tan, Ben Tripp, and Denise Daley. BRISK-research-oriented storage kit for biology-related data. *Bioinformatics*, 27(17):2422–2425, 2011.

- [TV10] Stefan Tilkov and Steve Vinoski. Node.js: Using JavaScript to Build High-Performance Network Programs. *IEEE Internet Computing*, 2010.
- [TWe12] Eleni Tsitsiou, Andrew E. Williams, and et. al. Transcriptome analysis shows activation of circulating CD8+ T cells in patients with severe asthma. *Journal of Allergy and Clinical Immunology*, 2012.
- [UC] UC Berkeley. UC Berkeley Committee for Protection of Human Subjects.
- [VBS⁺06] C. Viboud, O. N. Bjornstad, D. L. Smith, L. Simonsen, M. A. Miller, and B. T. Grenfell. Synchrony, Waves, and Spatial Hierarchies in the Spread of Influenza. *Science*, 312(5772):447–451, 4 2006.
- [VGKS99] Arja Virtanen, Mehran Gomari, Ries Kranse, and Ulf Stenman. Estimation of prostate cancer probability by logistic regression: Free and total prostate-specific antigen, digital rectal examination, and heredity are significant variables. *Clinical Chemistry*, 45(7):987–994, 1999.
- [Vol18] Nicholas Vollmer. Article 4 EU General Data Protection Regulation (EU-GDPR). *EU official journal*, 9 2018.
- [VRA18] Soroush Vosoughi, Deb Roy, and Sinan Aral. The spread of true and false news online. *Science*, 2018.
- [Wai16] Jacques Wainer. Comparison of 14 different families of classification algorithms on 115 binary datasets. *arXiv*, 2016.
- [WC54] William L. Watson and Alexander J. Conte. Smoking and lung cancer. *Cancer*, 1954.
- [WCZQ18] Jingxue Wang, Huali Cao, John Z. H. Zhang, and Yifei Qi. Computational Protein Design with Deep Learning Neural Networks. *Scientific Reports*, 8(1):6349, 12 2018.
- [WdACdRS18] Yihong Wang, Gonalo de Almeida Correia, Erik de Romph, and Bruno Santos. Road Network Design in a Developing Country Using Mobile Phone Data: An

- Application to Senegal. *IEEE Intelligent Transportation Systems Magazine*, 7 2018.
- [WET⁺12] Amy Wesolowski, Nathan Eagle, Andrew J Tatem, David L Smith, Abdisalan M Noor, Robert W Snow, and Caroline O Buckee. Quantifying the impact of human mobility on malaria. *Science (New York, N.Y.)*, 338(6104):267–70, 10 2012.
- [WGWF10] Katharina Wulff, Silvia Gatti, Joseph G. Wettstein, and Russell G. Foster. Sleep and circadian rhythm disruption in psychiatric and neurodegenerative disease. *Nature Reviews Neuroscience*, 11(8):589–599, 8 2010.
- [Wir18] Wired UK. Fake news used to promote 'A Cure for Wellness' film — WIRED UK, 2018.
- [WPe14] Shicai Wang, Ioannis Pandis, and et. al. High dimensional biological data retrieval optimization with NoSQL technology. *BMC genomics*, 2014.
- [WZESAe16] Robin Wilson, Elisabeth Zu Erbach-Schoenberg, Maximilian Albert, and et al. Rapid and Near Real-Time Assessments of Population Displacement Using Mobile Phone Data Following Disasters: The 2015 Nepal Earthquake. *PLoS currents*, 8, 2 2016.
- [XNR14] Miguel Gomes Xavier, Marcelo Veiga Neves, and Cesar Augusto FonticIELha De Rose. A Performance Comparison of Container-Based Virtualization Systems for MapReduce Clusters. In *2014 22nd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*. IEEE, 2014.
- [XYH⁺15] Eric P. Xing, Yaoliang Yu, Qirong Ho, Wei Dai, Jin-Kyu Kim, Jinliang Wei, Seunghak Lee, Xun Zheng, Pengtao Xie, and Abhimanu Kumar. Petuum. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '15*, pages 1335–1344, New York, New York, USA, 2015. ACM Press.

- [YDHP07] Hung-chih Yang, Ali Dasdan, Ruey-Lung Hsiao, and D. Stott Parker. Map-reduce-merge. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data - SIGMOD '07*, page 1029, New York, New York, USA, 2007. ACM Press.
- [YHG⁺16] Ibrar Yaqoob, Ibrahim Abaker Targio Hashem, Abdullah Gani, Salimah Mokhtar, Ejaz Ahmed, Nor Badrul Anuar, and Athanasios V. Vasilakos. Big data: From beginning to future. *International Journal of Information Management*, 36(6):1231–1247, 12 2016.
- [YJG03] Andy B. Yoo, Morris A. Jette, and Mark Grondona. SLURM: Simple Linux Utility for Resource Management. In *Workshop on Job Scheduling Strategies for Parallel Processing*, pages 44–60. Springer, Berlin, Heidelberg, 2003.
- [ZCF⁺10] Matei Zaharia, Mosharaf Chowdhury, Michael J. Franklin, Scott Shenker, and Ion Stoica. Spark: cluster computing with working sets, 2010.
- [ZEEO16] Mark Ziemann, Yotam Eren, and Assam El-Osta. Gene name errors are widespread in the scientific literature. *Genome Biology*, 17(1):177, 12 2016.
- [ZTHZ17] Lu Zhang, Jianjun Tan, Dan Han, and Hao Zhu. From machine learning to deep learning: progress in machine intelligence for rational drug discovery. *Drug Discovery Today*, 22(11):1680–1685, 11 2017.
- [ZWL18] Lei Zhang, Shuai Wang, and Bing Liu. Deep Learning for Sentiment Analysis : A Survey. *arxiv*, 2018.
- [ZZWD93] Songnian Zhou, Xiaohu Zheng, Jingwen Wang, and Pierre Delisle. Utopia: A load sharing facility for large, heterogeneous distributed computer systems. *Software: Practice and Experience*, 23(12):1305–1336, 12 1993.