The 14th Conference of PhD Students in Computer Science

Volume of short papers

CS^2

Organized by the Institute of Informatics of the University of Szeged

July 3 – 5, 2024
Szeged, Hungary

Scientific Committee:

Organizing Committee:

Judit Jász, Balázs Bánhelyi, Tamás Gergely, Zoltán Kincses

Address of the Organizing Committee

c/o. Judit Jász
University of Szeged, Institute of Informatics
H-6701 Szeged, P.O. Box 652, Hungary
Phone: +36 62 546 728, Fax: +36 62 546 397
E-mail: `cscs@inf.u-szeged.hu`
URL: `http://www.inf.u-szeged.hu/~cscs/`

Sponsors

University of Szeged,
Institute of Informatics

SZTE TTIK
INFORMATIKAI INTÉZET

Preface

This conference is the 14th in a series. The organizers aimed to bring together PhD students working on any field of computer science and its applications to help them publishing one of their first papers, and provide an opportunity to hold a scientific talk. As far as we know, this is one of the few such conferences. The aims of the scientific meeting were determined on the council meeting of the Hungarian PhD Schools in Informatics: it should

- provide a forum for PhD students in computer science to discuss their ideas and research results;
- give a possibility to have constructive criticism before they present the results at professional conferences;
- promote the publication of their results in the form of fully refereed journal articles; and finally,
- promote hopefully fruitful research collaboration among the participants.

The papers emerging from the presented talks will be invited to be considered for full paper publication the Acta Cybernetica journal.

Szeged, July 2024

Judit Jász
Balázs Bánhelyi
Tamás Gergely
Zoltán Kincses

Contents

Program

Wednesday, July 3

09:00 – 09:40	Registration
09:40 – 09:50	Opening
09:50 – 10:50	Talks – Image Processing (2x30 min.)
10:50 – 11:00	Break
11:00 – 12:00	Plenary Talk
12:00 – 13:30	Lunch Break
13:30 – 15:00	Talks – Security (3x30 min.)
15:00 – 15:30	Break
15:30 – 17:00	Talks – Blockchain (3x30 min.)
17:00 – 19:00	Free program
19:00 –	Welcome Party

Thursday, July 4

09:00 – 10:30	Talks – Testing (3x30 min.)
10:30 – 11:00	Break
11:00 – 12:00	Plenary Talk
12:00 – 13:30	Lunch Break
13:30 – 15:30	Talks – Computation (4x30 min.)
15:30 – 16:20	Free program
16:20 – 18:00	Social event
18:00 – 19:00	Free program
19:00 –	Wine & Cheese

Friday, July 5

09:00 – 10:30	Talks – Development 1 (3x30 min.)
10:30 – 11:00	Break
11:00 – 12:00	Talks – Development 2 (2x30 min.)
12:00 – 12:15	Closing
12:15 –	Lunch

Detailed program

Wednesday, July 3

09:00	**Registration**
09:40	**Opening**
Session 1	**Image Processing** - Session chair: Kálmán Palágyi
09:50	A. H. M. Sajedul Hoque, Gergő Bognár and Fridli Sándor: *Quantitative Radiomics Analysis of Lung CT Images Using Radial Harmonic Fourier Moments*
10:20	Tarlan Ahadli and Hajder Levente: *Drone Localization using Stereo Vision and YOLOv7*
10:50	Break
11:00	**Plenary Talk** - Gábor Péter Nagy: *Graphs, Groups, and Geometry*
12:00	**Lunch Break**
Session 2	**Security** - Session chair: Ákos Kiss
13:30	Martin Farkas, Imre Kocsis and Bertalan Zoltán Péter: *Design Space Exploration of Verifiable Credential Schemas using Partial Graph Modeling*
14:00	Smiljana Knezev, Melinda Tóth and István Bozó: *Identifying security issues in Elixir web applications*
14:30	Attila Szász and Balázs Bánhelyi: *New interval-based training technique to parameter robustness*
15:00	Break

Session 3	**Blockchain** - Session chair: Tamás Pflanzner
15:30	Zsófia Ádám, Bertalan Zoltán Péter, Zoltán Micskei and Imre Kocsis: *Smart Contract in the Loop: Fault Impact Assessment for Distributed Ledger Technologies*
16:00	Damaris Kangogo and Imre Kocsis: *Design of Hyperledger Fabric Private Data Collections with Formal Concept Analysis*
16:30	Wilson Valdez: *Convergence of Fog Computing, Blockchain, and Federated Learning for Advancing New Generation Networks*
17:00	Free program
19:00	Welcome Party

Thursday, July 4

Session 4 09:00	**Testing** - Session chair: Attila Szatmári Patrik P. Süli, Judit Knoll and Zoltán Porkoláb: *Multithreading Atomicity Static Analysis checkers in Java*
09:30	Zsófia Erdei, Melinda Tóth and István Bozó: *Selecting Execution Path for Replaying Errors*
10:00	Norbert Vándor: *Evaluating GPT-4 on a real Python bug dataset*
10:30	Break
11:00	**Plenary Talk** Márk Jelasity: *Adversarial Robustness of Deep Neural Networks*
12:00	**Lunch Break**
Session 5 14:00	**Computation** - Session chair: Richárd Farkas Emília Heinc and Balázs Bánhelyi: *Effective heuristics for accelerated branch and bound solver of process network synthesis problems*
14:20	Mátyás Sebők: *Multi Model Recursion for Hungarian electricity load forecasting*
14:40	Imre Gera and András London: *Clustering and Community Detection in Nested Graphs*
15:00	Ronglin Zuo and Bálint Molnár: *Knowledge Graph Powered LSTM in Stock Investment Decision Making*
15:30	Free program
16:20	Social event
18:00	Free program
19:00	Wine & Cheese

Friday, July 5

Session 6 09:00	**Development 1** - Session chair: József Dániel Dombi Md. Easin Arafat and Tamás Orosz: *Enhancing SAP Ecosystem: Harmonizing Open-Source Technologies for Integration and Innovation*
09:30	Zoltán Ságodi, István Siket: *State-of-the-Art Business Intelligence Applications: A Journey Through Time and Technology*
10:00	Daniel Ferenczi and Melinda Tóth: *Towards correct dependency orders in Erlang upgrades*
10:30	Break
Session 7 11:00	**Develpment 2** - Session chair: András Erik Csallner Imre Munkácsi, Márta Alexy Angyalné and Tamás Gábor Orosz: *Optimizing SAP S/4HANA On-Premise with Cloud-Ready Extensions: a Clean-Core system*
11:30	Georgina Asuah, Md Easin Arafat and Orosz Tamas: *Optimizing SAP Machine Learning-based Solutions through Custom API Integration*
12:00	**Closing**
12:15	Lunch

Graphs, Groups, and Geometry

Gábor Péter Nagy
University of Szeged, Szeged, Hungary

A graph is said to be k-regular if every vertex is incident to exactly k edges, implying that each vertex has k neighbors. A strongly regular graph, on the other hand, is a k-regular graph with an additional property: for any two vertices, the number of common neighbors can only assume two distinct values, denoted by λ and μ, depending on whether the two vertices are adjacent or not.

Constructing strongly regular graphs with specified parameters (k, λ, μ) is not a trivial task. Often, these constructions rely on various combinatorial structures or geometric objects defined over finite fields, such as Latin squares, ellipsoids, or paraboloids in higher-dimensional spaces.

Graphs that exhibit numerous symmetries among strongly regular graphs are of particular interest to us. A prime example of such a graph is the Petersen graph on 10 vertices. In my presentation, I will discuss several general construction methods that highlight the captivating interplay between finite groups and finite geometries.

Adversarial Robustness of Deep Neural Networks

Márk Jelasity
University of Szeged, Szeged, Hungary

Deep neural networks (DNNs) are the foundations of AI systems, yet they are known to have several vulnerabilities that are unsolved to this day. A particularly interesting vulnerability is that it is possible to construct inputs to these networks that are extremely close to natural inputs (eg. invisible perturbations to images) yet result in completely unexpected behavior. In the talk I will give several examples of this problem, and I will also present some of our own results in the area. I will briefly touch on the formal verification of neural networks from the point of view of adversarial robustness, and I will also discuss some applications like attacking semantic image segmentation networks and attacking entire ensembles of models.

Drone Localization using Stereo Vision and YOLOv7

Tarlan Ahadli, Hajder Levente

Abstract: This study combines stereo vision and the YOLOv7 object detection model to detect Unmanned Aerial Vehicles (UAVs), aiming to enhance drone monitoring and address associated security and privacy concerns. Employing a Gaussian Mixture Model (GMM) for preprocessing, our approach improves detection and tracking efficiency for real-time applications. We introduce two datasets: one for general drone detection and another specifically for drone localization with stereo cameras, supporting the training and validation of UAV detection models. Conducted experiments utilizing robust Python libraries demonstrate the system's effectiveness in real-time surveillance scenarios, marking a meaningful contribution to drone management technology.

Keywords: UAV Detection, Stereo Vision, YOLOv7, Real-Time Tracking, Surveillance Systems

1 Introduction

Drones are increasingly utilized in fields like entertainment and delivery, offering efficiency but also posing privacy and security risks, calling for improved monitoring solutions and updated regulations [1, 2, 3]. Existing detection technologies, such as radar and acoustic systems, struggle with object differentiation and operational conditions, whereas RF detection can miss drones with low-signal emissions [4, 5, 6, 7, 8, 9].

Our research proposes a system combining stereo vision with the YOLOv7 model, refining drone detection and simplifying the tracking process, sidestepping YOLOv7's intensive computations [10]. Drawing from VOT advancements, such as transformer applications in MS-AOT [11, 12, 13], our method employs feature-based bounding box tracking and stereo vision for precise drone localization.

Further, we present two tailored datasets for training machine learning algorithms and enhancing drone surveillance technology, compiled from public sources for comprehensive research utility [1] [2].

2 Methodology

Our approach to UAV monitoring integrates Detection, Tracking, and Localization, using the YOLOv7 algorithm for its superior accuracy and processing speed in real-time applications [10]. This performance is attributed to an optimized CNN backbone and focal loss [14], addressing class imbalance. Before detection, input frames are preprocessed with a GMM-based Background Subtraction (GMM-BS) method [15], generating a binary mask $M(x)$ to distinguish between foreground and background:

$$M(x) = \begin{cases} 1 & \text{if } p(x|\text{GMM}) > \theta, \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

Post-processing refines this mask, and the YOLOv7 algorithm then processes the segmented moving objects. For tracking, a feature-based algorithm tracks the drone's trajectory using spatial movement, aspect ratio, and color similarity metrics. The Euclidean distance $d_{euclid} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$ is calculated between consecutive positions, aspect ratio consistency

[1] Drone Detection Dataset repository: https://github.com/tarlanahad/drone-detection-dataset

[2] Stereo Vision Drone Localization Dataset: https://github.com/tarlanahad/stereo-vision-drone-localization-dataset

$S_{aspect} = AR_{current} - AR_{previous}$ by the change in the drone's bounding box, and color similarity $S_{color} = \sum \sqrt{H_{current} \times H_{previous}}$ using the Bhattacharyya coefficient for color histograms.

Localization employs a calibrated stereo camera system for 3D positioning using stereo-vision triangulation techniques. The 3D world coordinates X correlate to image plane coordinates U_1 and U_2 for the first and second cameras, represented as $U_1 \sim P_1X$ and $U_2 \sim P_2X$, where $P_1 = K_1[I \mid 0]$ and $P_2 = K_2[R_2 \mid t_2]$ are the projection matrices. The drone's 3D position X is estimated using SVD or optimization techniques [16].

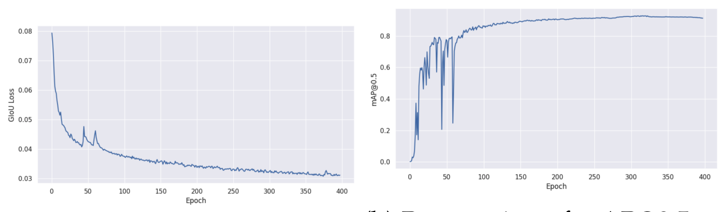
3 Implementation

System Architecture and Libraries. To construct an effective drone detection system, we leveraged Python's scientific computing libraries, integrating OpenCV for advanced image processing, NumPy for numerical computations and matrix operations, TensorFlow for neural network model development, and Scikit-learn for additional machine learning capabilities. The system's performance and computational efficiency were optimized through a high-performance Intel Core i7 workstation with 16GB RAM, facilitating the processing of large datasets and complex algorithms.

Dataset Preparation and Model Training. We prepared a dataset of approximately 4300 images, labelled in YOLO format, divided into training, validation, and testing sets with a 60/20/20 split.

Training involved random data augmentation to enhance model generalization and was conducted over 72 hours, adjusting the learning rate dynamically for efficiency. Checkpoints were set every 5 epochs to monitor and optimize performance.

The GIoU loss, as shown in Figure 1a, demonstrates a decreasing trend, indicative of the model's improving capability in accurately identifying the position and scale of drones over the training period.



(a) GIoU Loss trend during the training process.

(b) Progression of mAP@0.5 throughout the training epochs.

Figure 1: Performance metrics over the training epochs.

Subsequently, Figure 1b showcases the progression of the mAP@0.5 metric, reaching a high score of **0.94** on the validation dataset. This underscores the model's effectiveness in drone detection, highlighting its precision and reliability.

Object Tracking and Camera Setup. Utilizing NumPy and OpenCV, we crafted an algorithm for precise tracking, incorporating the GMM-BS algorithm and morphological operations for heightened detection fidelity. The deployment of a stereo camera system, featuring MV-CA020-20GM/GC cameras and SV-0614H lenses synchronized for time, was critical in acquiring high-definition images for precise localization and calibration.

The selection of the DJI Tello drone for data gathering, owing to its diminutive stature, facilitated the emulation of authentic long-distance drone detection contexts. This apparatus proved vital in curating a robust and varied dataset, encompassing 368 frames recorded at a rate of 5fps, featuring drones and pedestrians, thus amplifying the detection challenge's complexity and authenticity.

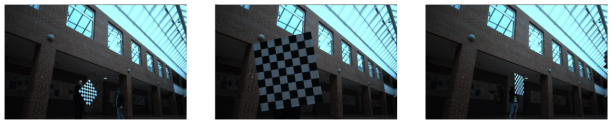


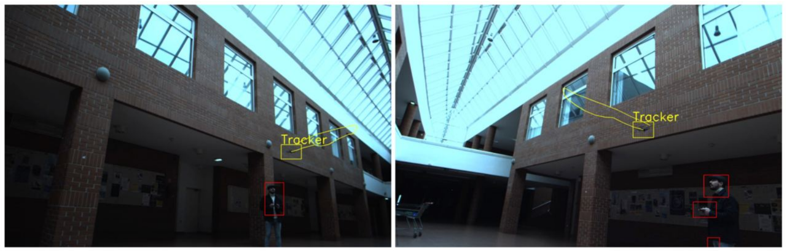
Figure 2: Stereo camera calibration with a checkerboard pattern.

Camera Calibration and Depth Perception. Once the checkerboard photos were taken, as shown in Figure 2, a meticulous calibration of the stereo camera system was conducted using MATLAB's Stereo Camera Calibration module. This step is crucial for precise depth perception and object localization in stereo images.

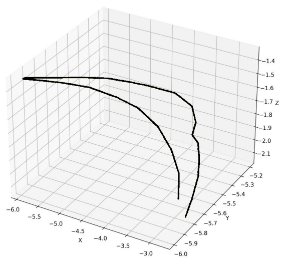
The calibration results were stored in a '.mat' file and integrated into our Python framework, pivotal for the precision of our 3D localization methods and the efficacy of our drone monitoring system.

4 Results

Our evaluation focused on the algorithm's effectiveness with a stereo camera drone dataset, demonstrating reliable drone tracking across frames and system robustness in real scenarios.



(a) Drone tracking using stereo cameras.



(b) Drone trajectory estimation.

Figure 3: Drone tracking and trajectory with stereo vision.

Initial GMM-BS adjustments were crucial for moving object identification, leading to successful drone tracking post-calibration. Despite a simple Visual Object Tracking (VOT) approach, the system managed bounding box tracking in all frames, leveraging GMM-BS for initial pixel distribution analysis.

Localization estimated the drone's coordinates via triangulation, facing challenges in quantitative accuracy assessment due to dataset limitations. The algorithm's trajectory visualization in Figure 3 offers qualitative precision insights. Future work should incorporate high-accuracy GPS for direct position comparison and enhanced localization accuracy.

The system delivered a detection rate of 3 FPS and a tracking rate of 7 FPS on a CPU setup, indicating a viable balance for real-time UAV monitoring.

5 Conclusion

This study presents a stereo-vision-based system for drone detection, using machine learning to enhance drone localization, addressing security and privacy concerns. It suggests using Kalman filtering for predicting drone movements, potentially increasing efficiency and accuracy. Key contributions include a new methodology for drone localization and two datasets: one for training machine learning models using various public sources, and another for stereo vision research in drone detection. Highlighting the growing importance of UAV surveillance technology, this research lays the groundwork for future studies aimed at improving UAV monitoring and exploring comprehensive UAV management for applications like urban surveillance and environmental monitoring.

References

[1] Z. Dukowitz. Thieves use drone to steal almost $150,000 from atm. 2022.

[2] Drone carrying drugs, phones crashes outside ohio prison. 2015.

[3] H. Brandes. Drone carrying drugs, hacksaw blades crashes at oklahoma prison. 2015.

[4] J. Gong, J. Yan, D. Kong, and D. Li. Introduction to drone detection radar with emphasis on automatic target recognition (atr) technology, 2023.

[5] C. Dumitrescu, M. Minea, I. M. Costea, C. I. Chiva, and A. Semenescu. Development of an acoustic system for uav detection. 2020.

[6] Y. Sun, J. Li, L. Wang, et al. Deep learning-based drone acoustic event detection system for microphone arrays. *Multimedia Tools and Applications*, 2023.

[7] M. M. Alaboudi, M. Abu Talib, and Q. Nasir. Radio frequency-based techniques of drone detection and classification using machine learning. 2021.

[8] I. Nemer, T. Sheltami, I. Ahmad, A. U. Yasar, and M. A. R. Abdeen. Rf-based uav detection and identification using hierarchical learning approach. 2021.

[9] S. Al-Emadi and F. Al-Senaid. Drone detection approach based on radio-frequency using convolutional neural network. 2020.

[10] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao. Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors, 2022.

[11] Z. Yang, X. Wang, J. Miao, Y. Wei, W. Wang, and Y. Yang. Scalable video object segmentation with identification mechanism, 2023.

[12] Z. Yang, Y. Wei, and Y. Yang. Associating objects with transformers for video object segmentation. 2021.

[13] Z. Yang, J. Zhang, W.-H. Wang, W. Han, Y. Yu, Y. Li, J. Wang, Y. Wei, Y. Sun, and Y. Yang. Towards multi-object association from foreground-background integration. 2021.

[14] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal loss for dense object detection, 2018.

[15] C. Stauffer and W. Grimson. Adaptive background mixture models for real-time tracking. 1999.

[16] R. I. Hartley and A. Zisserman. Multiple View Geometry in Computer Vision. Cambridge University Press, ISBN: 0521540518, second edition, 2004.

Enhancing SAP Ecosystem: Harmonizing Open-Source Technologies for Integration and Innovation

Arafat Md Easin, Orosz Tamás

Abstract: In today's rapidly evolving digital landscape, organizations are increasingly turning to open-source technologies to enhance their SAP systems. The concept of end-to-end integration of SAP Analytics Cloud (SAC) with Python-based technologies called Remote Function Modules (RFC) and Business Application Programming Interface (BAPIs), aims to provide insights into the benefits, challenges, and best practices. It explores a variety of open-source frameworks and platforms that effectively integrate with SAC across domains like data science, cloud computing, and DevOps. Through a comprehensive review of existing literature, case studies, and real-world benchmarks, we highlight the advantages of incorporating new open-source technologies within the SAP ecosystem, including improved flexibility, scalability, and cost-effectiveness. We discuss key strategies for successful integration, focusing on data interoperability, security, and performance optimization. By examining emerging trends and future directions, this paper offers insights for organizations aiming to maximize the value of integrating these technologies with SAC. Overall, this paper serves as a primer for researchers, practitioners, and decision-makers interested in understanding and maximizing the value of end-to-end integration and automation between SAP products and open-source technologies.

Keywords: SAP Analytics Cloud, Open-Source Technologies, End-to-end Integration

1 Introduction

The utilization of import data connections facilitates the process of integrating data from multiple sources. Organizations rely on resilient and scalable data and analytics solutions in the evolving landscape of data-driven decision-making. SAP's continuous adaptation and innovation necessitate an analysis of the dynamic SAP data and analytics architecture [1]. SAP Analytics Cloud (SAC) is another potent cloud-based solution for business intelligence and analytics, transforming raw data into actionable insights. With its robust data management capabilities, SAC empowers organizations to harmoniously monitor, analyze, and visualize data in real-time, offering a comprehensive platform for informed decision-making. The service also offers analytics, data modeling, version control, Smart Discovery, time series forecasting, R visualization, and Smart insights functionalities [2, 3].

In today's analytics landscape, real-time data is crucial for success. This integration benefits organizations by enabling proactive decision-making and enhancing responsiveness to opportunities and challenges. Achieving thorough and accurate data integration poses a common challenge for businesses, especially in large-scale projects involving diverse data silos containing critical information [4]. Utilizing real-time and historical customer data enables businesses to provide timely support, enhancing customer experience and revenues, and facilitating more accurate customer demand forecasting. It also enables organizations to simplify operations, leading to improved processes, reduced costs, and increased production across departments. Additionally, accessing historical and real-time data facilitates more accurate and timely customer demand forecasting.

In this continuum, SAC offers a range of integration options, allowing different integration with existing landscapes or as a standalone platform [5]. Moreover, open-source technology could emerge as a compelling alternative to proprietary software, offering a clear financial benefit [6, 7]. In recent years, integrating such technology has become a more viable option for enterprises seeking to enhance the security, quality, flexibility, and innovation potential of SAC. Integrating open-source technology broadens SAC users' access to diverse tools, enabling refinement of analytics workflows and targeted resolution of specific business needs. This introduction prepares us to find out how SAP Analytics Cloud and open-source technologies interact, highlighting the benefits, challenges, and possibilities that arise from integrating them.

In this paper, our focus is to present the powerful potential of real-time live connections through open-source technologies, as these data integration offers several advantages.:

- Firstly, the integration of Python-based open-source technologies and automation data extraction from SAP by executing Analytics Cloud functions from Python scripts eliminates the need for data replication, thereby avoiding data transfers from the source system.

- Secondly, this integration ensures automatic updates with the latest data through OData or REST API, facilitating the provision of real-time insights.

- Thirdly, SAC empowers users to create and analyse intricate models within the source systems, enhancing analytical capabilities.

- Lastly, when leveraging SAP HANA Python Clients with the SAC Platform, confidential data remains within the organization's internal network, protected by robust firewalls, while performing database operations and analytics directly on SAP HANA from Python.

2 Motivation

SAP Analytics Cloud (SAC) provides many ways to combine data, helping organizations make the most of their information and enabling users to combine diverse datasets for comprehensive insights. SAC facilitates integration with all data sources through both live connections and import connections. Live connections, also referred to as remote or online connections, provide direct access to tunnel connection data sources. This system allows us to import various data connections through on-premise and cloud-based systems, including SAP HANA (High-performance ANalytic Appliance), SAP Datasphere, SAP BW (Business Warehouse), SAP Universe, SAP SuccessFactors, SAP Integrated Business Planning, SAP Cloud for Customer, SAP ERP (Enterprise Resource Planning), Google BigQuery, Google Drive, SharePoint Online, Salesforce, SQL Database (including Azure Synapse), SAP Integration Suite Open Connectors, and more.

Likewise, in the context of exporting models to files, various methods such as exporting data to SAP Business Planning and Consolidation (BPC), OData Services (BW & BW/4HANA), SAP S/4HANA, SAP Integrated Business Planning (IBP), and Data Export API are commonly employed. Additionally, for establishing Live Data Connections, contemporary practices include utilizing SAP Datasphere, SAP HANA, SAP HANA Cloud, SAP S/4HANA, SAP BW and BW/4HANA, SAP BPC Embedded, SAP Universes, Web Intelligence Documents, and APOS Live Data Gateway [4, 8]. Through Live Data Connections, organizations gain direct access to real-time data sources, empowering decision-makers with the latest information [9]. Moreover, Import Data Connections facilitate the integration of data from diverse sources, allowing SAC to unify datasets from both connection types. Lastly, Export Data Connections enable businesses to export and store data on external platforms seamlessly. Needless to say, SAC not only transforms how organizations analyze data but also fuels innovation and drives strategic decision-making in today's dynamic landscape.

3 Proposed Approach

Setting up data connections correctly in SAP Analytics Cloud (SAC) is crucial, but it can be tricky. If not done properly, it can lead to problems like data errors, system downtime, and security risks. This can slow down operations, delay decision-making, and result in higher costs for resolution.

This article primarily focuses on how to affect the open-sources integration procedures of the SAP-based platform (shown in Figure 1). Therefore, some research questions are discussed to identify the objectives of this paper. The study aims to find solutions by investigating the advantages of implementing open-source technologies alongside the SAC. The listed questions are as follows:

1. Which optimal Python-based open-source technologies best integrate with the SAC platform, and why?

2. For current SAC infrastructure to be compatible and interoperable, how can enterprises ensure efficient integration of innovative technologies with their current SAC infrastructure to achieve compatibility and interoperability?

3. What opportunities and difficulties come with using such open-source technologies within the SAP ecosystem to drive innovation?

4. Which best practices and techniques should be used in SAC contexts to maximize the benefits of these technologies and speed up the integration process?

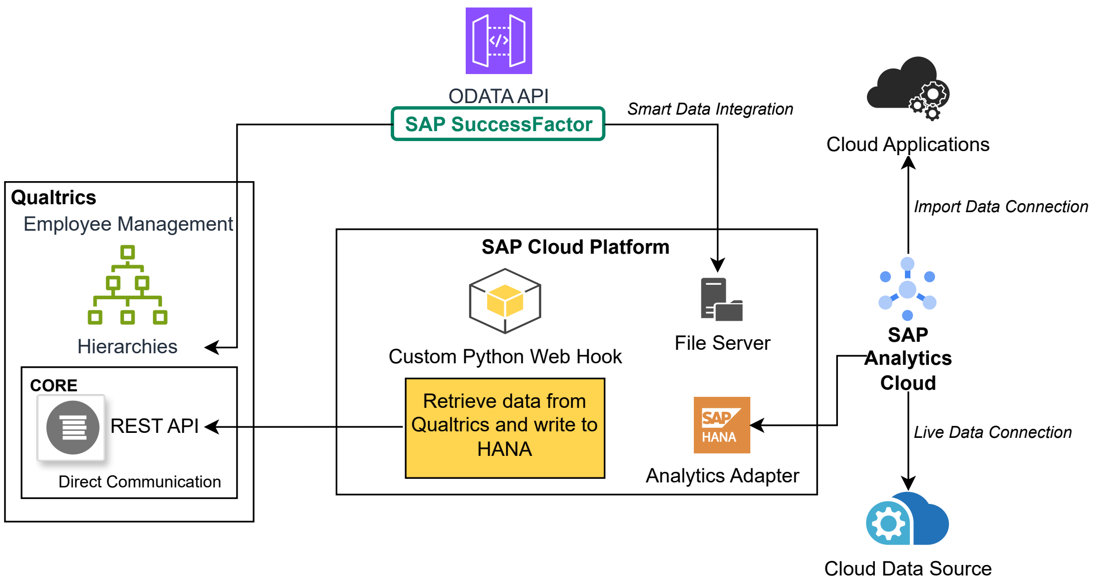


Figure 1: Integration of Open-Source Technologies with SAC Platform

The combination of such technologies with the SAC platform creates a technological foundation for advanced data science applications. Thus, using Python libraries like pandas, scikit-learn, TensorFlow, and PyTorch in the SAC environment allows for advanced data processing, statistical analysis, and machine learning applications [10]. Additionally, machine learning can be utilized to model existing visualization types and their associated user stories. This model is then employed to predict recommended visualization types for new user stories, enhancing data-driven decision-making.

Nowadays, Python has a vast modeling and predictive analytics toolbox, and this integration allows data scientists to easily access and leverage SAC's analytics and data visualization capabilities even more along with Matplotlib and Plotly. Organizations might accelerate the development and implementation of data science solutions, enabling the investigation of complicated datasets, the development of predictive models, and the production of useful insights, employing this technological combination. To further investigate these issues and concepts, researchers could essentially examine how open-source technologies are integrated into the SAP ecosystem. This investigation aims to reveal information that can facilitate innovation, enhance company procedures, and support decision-making.

4 Conclusion

The business environment is changing noticeably right now, and customers are expecting more. No one can argue against the importance of data analysis in meeting consumer needs and maximizing corporate success. Businesses rely heavily on predictive planning to help them make well-informed decisions that provide more accurate and more effective results. Likewise, organizations have the potential to benefit greatly from the integration of open-source solutions based on analytics cloud with SAP systems. The smooth integration of SAP Analytics Cloud (SAC) with RFC and BAPIs which are Python-based technologies has been talked about in this article, along with its advantages, difficulties, and recommended practices. This study not only provides valuable guidance for optimizing value from integration but also emphasizes the importance of predictive planning in helping organizations avoid costly mistakes and seize opportunities. As a result, SAC meets contemporary business operations needs to gain a competitive advantage in the data-driven era.

References

[1] Codorniz, R. SAP Analytics Cloud implementation-Step by step deployment. (2023)

[2] Gole, V., Shiralkar, S., Gole, V. & Shiralkar, S. Leverage SAC to Create "All-in-One" Analytics Platform. *Empower Decision Makers With SAP Analytics Cloud: Modernize BI With SAP's Single Platform For Analytics*. pp. 79-122 (2020)

[3] Nazarov, D., Morozova, A. & Kokovikhin, A. SAP analytic cloud: A tool for the formation of professional competencies of business analyst. *CEUR Workshop Proceedings*. **2570** pp. 1-4 (2020)

[4] Tran, D. Using SAP Analytics Cloud (SAC) for visualizing data and detecting problems. (2023)

[5] Sidiq, A. SAP Analytics Cloud. 2nd Edition. *Boston: Rheinwerk Publishing, Inc*. (2022)

[6] Olson, D., Johansson, B. & De Carvalho, R. Open source ERP business model framework. *Robotics And Computer-Integrated Manufacturing*. **50** pp. 30-36 (2018)

[7] Mladenova, T. Open-source ERP systems: an overview. *2020 International Conference Automatics And Informatics (ICAI)*. pp. 1-6 (2020)

[8] Wu. Unlock data integrations with SAP Analytics Cloud (SAC). Retrieved from https://community.sap.com/t5/technology-blogs-by-sap/unlock-data-integrations-with-sap-analytics-cloud-sac/ba-p/13578290 (Accessed on 23 March 2024).

[9] Arafat, ME., Georgina, A., Saha, S., & Orosz, T. Empowering Real-Time Insights through LLM, LangChain, and SAP HANA Integration. *Proceedings of 6th International Conference on Recent Innovations in Computing, Springer Nature Singapore*. Vol. 2 (2023)

[10] Xu, L. User Story based Information Visualization Type Recommendation System.. *International Journal Of Information Engineering & Electronic Business*. **11** (2019)

Optimizing SAP Machine Learning-based Solutions through Custom API Integration

Georgina Asuah, Arafat Md Easin, Orosz Tamás

Abstract: The business landscape is characterized by rapid changes, dynamic customer preferences, and evolving market trends. SAP HANA, with its in-memory processing architecture and robust foundation for real-time data analytics and processing, has emerged as a powerful platform to meet this demand. In this paper, we present a comprehensive implementation and deployment of an anomaly detection solution within the SAP HANA Fiori web application using a custom Application Programming Interface (API).

Keywords: SAP Fiori, Machine Learning, API Integration

1 Introduction

In today's world, where digital advancements and big data are everywhere, organizations are always trying to get useful insights from their large amounts of data [1]. The business environment is marked by swift transformations, ever-changing market trends, and dynamic consumer preferences [2]. Real-time decision support has become not only a competitive advantage but a necessity for organizations across industries [3]. SAP HANA with its in-memory computing architecture and robust foundation for real-time data analytics and processing, has emerged as a powerful platform to meet this demand [4]. However, a smooth integration of machine learning (ML) and artificial intelligence (AI) capabilities is necessary to fully utilize SAP HANA's potential [5]. With this connection, businesses can gain actionable insights from their data and make informed decisions instantly.

Nowadays, ML algorithms utilize data analysis techniques to identify patterns and correlations in historical data, enabling the extraction of valuable information and the creation of algorithms [6]. Integrating custom APIs with standard SAP ML-based solutions can significantly enhance their performance and capabilities. These APIs allow organizations to adapt ML models to their specific needs and requirements, enabling them to address unique business challenges effectively [7]. Moreover, the Predictive Analysis Library (PAL) in SAP HANA emerged as a robust framework for creating unsupervised anomaly detection algorithms [8]. Nevertheless, this method has certain constraints as it does not provide the user with the ability to personalize the algorithm's parameters or incorporate domain knowledge into the anomaly detection process. This necessitates the use of custom APIs to enhance the precision and efficiency of anomaly detection.

2 Literature Review

Following the architecture of ML in SAP HANA, we can explore the relevant examples and their results. [9] proposed a distributed and unified API service for machine learning models that helps the ensemble of multiple models, resulting in better predictions and other benefits such as wider availability, greater usability, and lesser resource constraints. The challenge of creating ML APIs that are easy to learn and use, especially for novices was addressed by [10]. Their work focused on analyzing the use of scikit-learn, a widely used ML API, by the Kaggle community. [11] in their paper presented a case study on integrating an SAP ERP system with an external web service using API access, demonstrating the implementation of algorithms and transactions in SAP ERP.

In a recent study by [8], the K-Means clustering algorithm, a widely used unsupervised anomaly detection technique, was employed to detect fraudulent transactions in a dataset of retail transactions. The K-Means technique divides the data into a predetermined number of clusters, with each cluster representing a set of data points that exhibit similar features. By analyzing the distribution of data points within these clusters, outliers or anomalies can be identified as those points that lie far from the cluster centers. They demonstrated that the K-Means algorithm effectively identified fraudulent transactions in the retail transaction dataset, highlighting its potential as a valuable tool for fraud detection in the retail industry.

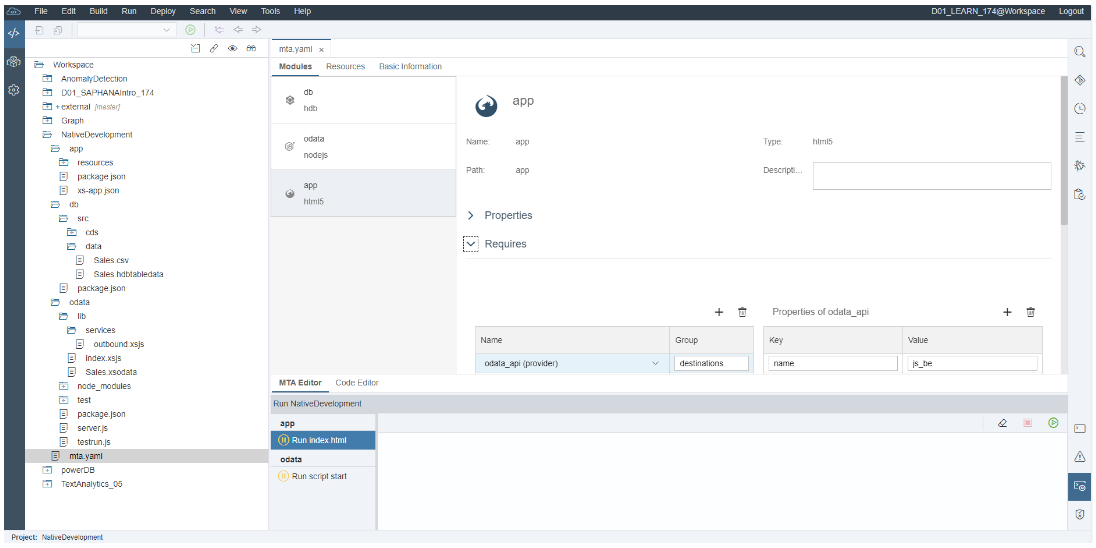
3 Research Methodology

The proposed solution is designed to leverage machine learning models from the renowned scikit-learn (sklearn) library. Whereas, the K-Means clustering algorithm for unsupervised anomaly detection is employed. K-Means partitions data into clusters, with anomalies identified as points distant from cluster centers. This method effectively detects anomalies in retail transactions, showcasing its potential for fraud detection. This approach ensures robust and well-established algorithms for data analysis and predictive modeling.

Afterward, the ML models are trained and validated, and scalable and accessible APIs are created for predictive capabilities. To achieve this, the FastAPI framework is employed, providing a high-performance and easy-to-use platform for building RESTful APIs in Python. The deployment of the API is orchestrated on the Azure cloud service provider, chosen for its robust infrastructure and seamless integration with FastAPI. The finalized API serves as a bridge between the ML models and the SAP HANA Fiori web application. This integration enhances the functionality of the web application by incorporating intelligent decision-making capabilities based on the ML models' predictions. The SAP HANA Fiori web application acts as the user interface, facilitating a smooth interaction between end-users and the predictive models.

4 Results and Discussions

This work involves the creation of a REST API using Python to enhance SAP ML-based solutions. The experimental results are shown and discussed below.



```
Workspace
  AnomalyDetection
  D01_SAPHANAIntro_174
  +external [master]
  Graph
  NativeDevelopment
    app
      resources
      package.json
      xs-app.json
    db
      src
        cds
        data
          Sales.csv
          Sales.hdbtabledata
      package.json
    odata
      lib
        services
          outbound.xsjs
        index.xsjs
        Sales.xsodata
      node_modules
      test
      package.json
      server.js
      testrun.js
    mta.yaml
  powerDB
  TextAnalytics_05
```

db hdb
odata nodejs
app html5

app

Name: app
Type: html5
Path: app
Descripti...

Properties
Requires

Name	Group
odata_api (provider)	destinations

Properties of odata_api

Key	Value
name	js_be

app
Run index.html
odata
Run script start

Project: NativeDevelopment

Figure 1: Overview of SAP HANA Fiori project configuration.

This application consumes OData API/Service and loads data from the system. The application interface (shown in Figure 1) is configured to present the sample data and make the external API call.

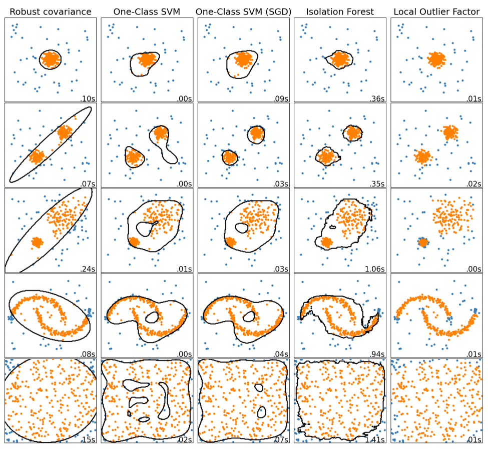


Figure 2: Comparison of different estimators for anomaly detection.

Sklearn estimators are examined and compared. Figure 2 compares Robust covariance, One-Class SVM, One-Class SVM (SGD), Isolation Forest, and Local Outlier Factor (LOF) estimators. The LOF was chosen for anomaly detection. It excels at identifying outliers in complex datasets because it captures local density deviation. Its non-parametric nature lets it adapt to different data distributions, making anomaly detection robust. LOF excels in cases where anomalies deviate from the norm to varying degrees [12].

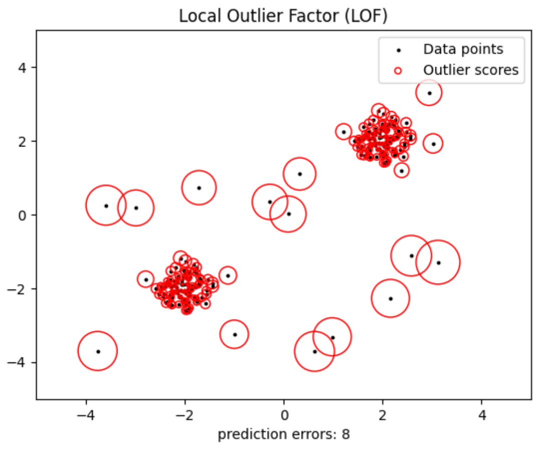


Figure 3: Plotting the local outlier factor using randomly generated data with outliers

Figure 3 illustrates the Local Outlier Factor of the trained model on the randomly generated data with outliers. LOF measures the local density deviation of a data point with respect to its neighbors, aiding in identifying anomalies. A prediction error of 8 indicates the model's accuracy in detecting outliers.

The proposed environment offers significant advantages by incorporating FastAPI for API development and scikit-learn's ML models to improve anomaly detection in the SAP HANA Fiori web application. This integrated system utilizes powerful algorithms, offering immediate insights for well-informed decision-making. Moreover, the adaptability of customized APIs effectively overcomes the constraints of SAP HANA, guaranteeing the streamlined identification of irregularities. Despite its advantages, it may encounter difficulties such as the intricate integration of APIs, the additional costs and effort required for maintenance, and the potential problems with scaling as the amount of data increases.

5 Conclusion

In summary, this paper introduces a comprehensive journey in implementing and deploying an anomaly detection solution within the SAP HANA Fiori web application. The scikit-learn's ML models, particularly the LOF, addressed the limitations experienced with SAP HANA's PAL, offering a more versatile and robust anomaly detection methodology. Therefore, the integration of FastAPI for API development facilitated the creation of a high-performance and efficient interface, enhancing the usability of the trained LOF model. The uninterrupted integration of the anomaly detection API into the SAP HANA Fiori web application represents a pivotal advancement. The establishment of connections between disparate systems, from scikit-learn models to FastAPI, Azure, and SAP HANA Fiori, culminates in a cohesive and powerful predictive analytics system. Consequently, this comprehensive solution not only addresses the technical challenges inherent in anomaly detection but also underscores a strategic approach to leveraging a diverse set of technologies.

References

[1] Shi, Z. & Wang, G. Integration of big-data ERP and business analytics (BA). *The Journal Of High Technology Management Research*. **29**, 141-150 (2018)

[2] Yang, K. Quality in the Era of Industry 4.0: Integrating Tradition and Innovation in the Age of Data and AI. (John Wiley & Sons,2024)

[3] Kulkarni, S. Implementing SAP S/4HANA. (Springer, 2019)

[4] Arafat, ME., Georgina, A., Saha, S., & Orosz, T. Empowering Real-Time Insights through LLM, LangChain, and SAP HANA Integration. *Proceedings of 6th International Conference on Recent Innovations in Computing, Springer Nature Singapore*. Vol. 2 (2023)

[5] Kohli, M. Using machine learning algorithms on data residing in SAP ERP application to predict equipment failures. *International Journal Of Engineering & Technology*. **7**, 312-319 (2017)

[6] SC, S. & Raja, B. A Review of Machine Learning and It's Method. (IJETIE,2019)

[7] Buitinck, L., Louppe, G., Blondel, M., Pedregosa, F., Mueller, A., Grisel, O., Niculae, V., Prettenhofer, P., Gramfort, A., Grobler, J. & Others API design for machine learning software: experiences from the scikit-learn project. *ArXiv Preprint ArXiv:1309.0238*. (2013)

[8] Oliveira, J. & Sousa, R. Unsupervised anomaly detection of retail stores using predictive analysis library on sap hana XS advanced. *Procedia Computer Science*. **181** pp. 882-889 (2021)

[9] Nandigramwar, H., Mittal, A., Bhatnagar, A. & Rashid, M. A distributed and unified API service for machine learning models. *2021 2nd International Conference On Intelligent Engineering And Management (ICIEM)*. pp. 480-485 (2021)

[10] Reimann, L. & Kniesel-Wünsche, G. Improving the learnability of machine learning APIs by semi-automated API wrapping. *Proceedings Of The ACM/IEEE 44th International Conference On Software Engineering: New Ideas And Emerging Results*. pp. 46-50 (2022)

[11] Peksa, J. Autonomous Data-Driven Integration into ERP Systems. *Design, Simulation, Manufacturing: The Innovation Exchange*. pp. 223-232 (2021)

[12] Angiulli, F. CFOF: a concentration free measure for anomaly detection. *ACM Transactions On Knowledge Discovery From Data (TKDD)*. **14**, 1-53 (2020)

Smart Contract in the Loop: Fault Impact Assessment for Distributed Ledger Technologies

Zsófia Ádám, Bertalan Zoltán Péter, Zoltán Micskei, Imre Kocsis

Abstract: Due to their decentralized and trustless nature, blockchain and distributed ledger technologies are increasingly used in several domains, including critical applications. The behavior of such blockchain-integrated systems is typically driven by smart contracts. However, smart contracts are application-specific software and may, therefore, contain faults with severe system-level impacts. This is especially true in the case of the extensively used Hyperledger Fabric (HLF) platform, where smart contracts are written in general-purpose languages (Java, among others), and applications can go far beyond handling virtual-currency-like assets. In this work, we present a novel formal-verification-based approach to smart contract verification and a high-level empirical model of a HLF platform. Our *Smart Contract in the Loop (SCIL)* method uses a model checker (Java Pathfinder) to check whether specific error properties hold for a given smart contract, while a predefined combination of platform-level fault modes is active. We facilitate the checking of HLF smart contracts without modification and enable the propagation or non-propagation of platform faults through the smart contracts to the system failure level.

Keywords: distributed ledger technology, blockchain, formal verification, model checking, Java Pathfinder, Hyperledger Fabric

1 Introduction

Distributed ledger technologies (DLTs) – especially blockchains – provide high-integrity distributed databases without requiring a trusted party. Initially developed with financial applications in mind and powering cryptocurrencies, blockchain technology now has a variety of use cases, including supply chain management, healthcare, and telecommunication. Blockchains have powerful properties, such as immutability, distribution, decentralization, and high security that make them fit for cross-organizational and possibly critical applications. Typically, such use cases are backed by permissioned platforms, such as Hyperledger Fabric.

Hyperledger Fabric (HLF) [1] is a widely used, mature, enterprise-grade permissioned blockchain platform maintained by the Hyperledger Foundation. It offers pluggable consensus mechanisms, identity management, flexible "subnetting" features, and privacy mechanisms. Fabric powers several projects in both development and production in various domains.

Smart contracts, which have been introduced with Ethereum [2], are akin to stored procedures and describe computations that are executed on the blockchain with effects that are persisted on-chain. They extend the original accounting "ledger" functionality of permissionless blockchains with rich, self-executing business logic. In HLF, the network *must* have smart contracts (called "chaincode") for any meaningful transactions to be able to occur.

However, smart contracts are software and thus susceptible to faults with potentially devastating consequences. Because of these risks, the verification of smart contracts has been a central research topic in the past years, bringing about a number of approaches for fault removal and prevention. These are mostly design-time methods, such as static analysis, and primarily target Ethereum and the Solidity programming language. There is significantly less support for the verification of enterprise smart contracts.

Formal verification approaches can be employed to reliably verify smart contracts in both public and consortial applications [3, 4]. However, to our knowledge, there is no tooling that enables the impact assessment of platform-level faults given an unmodified smart contract implementation, especially for enterprise platforms such as HLF.

In this work, we propose the application of model checking to show whether a smart contract may develop errors in the presence of certain platform-level faults. To this end, we present a simplified model of the HLF blockchain platform with its primary components and configurable fault modes. We demonstrate the viability of our approach with a Java-based prototype capable of simulating network faults in a hypothetical safety-critical application context using Java Pathfinder [5]. Our method provides the means for one to *plug in* their Java HLF smart contract to the framework and determine whether a predefined property holds while select platform-level faults are active. We dub this approach *Smart Contract in the Loop (SCIL)*.

The rest of this paper is organized as follows. In the next section, we provide the motivation and background for our research. Then, in Section 4, we present our prototype model of HLF and demonstrate the applicability of our approach. Finally, we conclude in Section 5.

2 Motivation

When high integrity is key, blockchains are already widely applied, even in critical applications, e.g., in the nuclear [6] or the railway [7] industry. However, where other extra-functional properties, such as timeliness, age-of-information, dependability, or availability are also matters of concern, the system-level analysis of critical applications is still largely an open challenge.

This is the primary motivation of our work; as soon as blockchains and DLTs are sought to be integrated into a system design process with extra-functional requirements to meet, there will be a need for a method to show that the system is compliant with the requirements. Especially in the case of enterprise blockchain platforms (such as HLF), there is no known support and tooling to argue about the safety of such applications.

Smart contract faults may result in the loss of financial assets in permissionless systems and the cryptocurrency world, but the potential effects are arguably far more devastating in the context of permissioned and especially critical applications. While smart contracts can be enhanced with various defenses (including runtime verification mechanisms or techniques such as n-version programming), faults of the platform itself may still induce unintended behavior. For example, in HLF, if a malicious ordering service intentionally reorders and selectively accepts (i.e., occasionally drops) transactions, ledger updates may not always reflect the expected world state. Platform-level faults include:

- malicious orderer behavior (transaction dropping, reordering),
- network issues (e.g., traffic congestion),
- hosting issues (e.g., a peer becomes unavailable),
- incorrect configuration (e.g., unsuitable endorsement policies),
- other malicious or unintentional behavior (e.g., client issues).

In a correctly configured network, Fabric protects against some of the potential faults. For example, endorsement policies can be designed to tolerate the downtime of some peers. However, due to the potential complexity of a HLF application, it is challenging to argue about the system-level effects of these platform faults on the applications. The model-checking-based approach presented in this paper can serve as a solution to this.

3 Verification Approach with a Smart Contract in the Loop

As discussed in Section 2, we focus mainly on system-level and smart contract faults. Even with state-of-the-art verifiers, out-of-the-box model checking of a large project, such as the implementation of HLF, is still not feasible due to numerous factors, such as scalability issues, libraries, and the distributed nature of the project.

Instead, we focus on a high-level model of a HLF network, into which we can add the

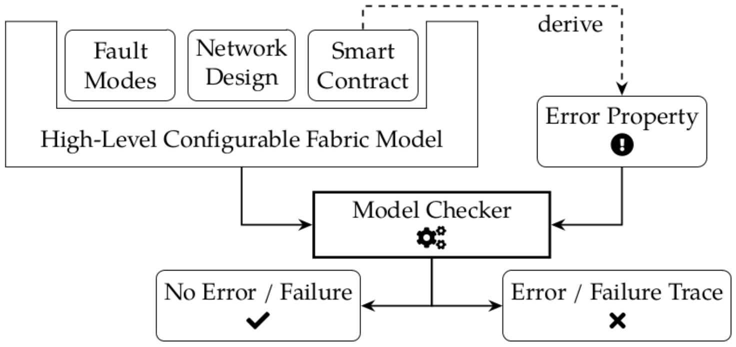


Figure 1: Verification Process

chaincode *as is* (hence the name SCIL) and the configuration of the network under verification (e.g., number of network participants such as orderers and nodes). Figure 1 shows an overview of our process.
Fault Modes and Error Property The faults discussed in Section 2 might enable errors in the network, e.g., valid transactions might not make it into blocks due to networking issues or a malicious orderer. Thus, we can derive *error properties* based on the smart contract – these are the issues the chaincode developer will be interested in. We can then separately enable different faults in the network by configuring fault modes and then check with a model checker whether the given combination of fault modes and the error property holds.

High-Level Configurable Fabric Model The difficulty of this approach lies in the empirical nature of modeling the network – verification outcomes are hard to trust on an abstract model based on informal documentation and some code. We identified abstraction as a key point regarding the quality of the model; i.e., finding the right abstraction to catch all relevant aspects to the faults and the error property while keeping the model and its limitations clear.

4 High-Level Model of Fabric

We summarized our Fabric model on the currently used level of abstraction in Figure 2. This high-level architecture serves as the basis of our model – it captures the main participants and flow of information, but it considers them to be *black box*, as we do not want to reimplement the underlying algorithms (e.g., Raft [8]). Thus, behavior is kept straightforward and declarative.

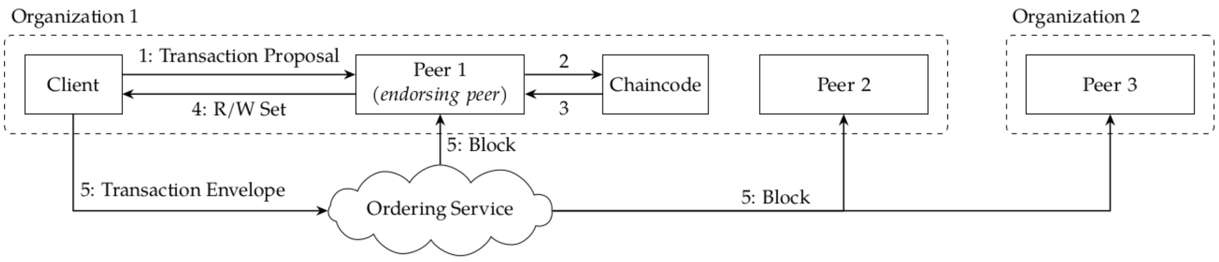


Figure 2: High-level overview of the Fabric architecture

The main components, as shown in Figure 2, are:

Organizations represent the participants of the network; i.e., members of the consortium using the network. Application clients, smart contracts, and peers are mapped to organizations.

Ordering Service The ordering service is an abstraction formed by all ordering nodes in the network and the consensus protocol among them. We did not model the individual orderers or the consensus protocol itself. Instead, the ordering service can have a selected fault mode, such as occasionally dropping transactions.

Peers maintain the distributed ledgers for the channels they are in. Furthermore, peers may receive and simulate client transaction requests and later validate published blocks.

Channels group a number of peers to form a "subnet" in the HLF network. Newly created blocks are broadcast to the peers in the channel. If endorsement policies are also modeled, they affect the behavior of the channels. Inappropriately chosen policies can have significant system-level effects.

Application Clients submit transactions via the peers.

Smart Contracts can be installed on the peers in the network and define application-specific business logic. The smart contract (chaincode) can be provided as is, without the need for any modifications.

Transaction Flow. HLF's transaction flow is part of the model: clients first submit transaction requests to endorsing peers, who respond with their endorsements and corresponding read/write sets. Then, the client sends the endorsed requests to the ordering service. During message exchange, it is possible to consider network issues (e.g., traffic congestion).

Prototype Implementation

Our prototype implementation of the above model in Java, along with further user documentation, is open-source and available on GitHub[1]. The currently implemented, most interesting fault modes are the realistic faulty behaviors of the ordering service (i.e., if it can reorder and lose transactions). Network configurations can be defined via builder classes, and error properties can be added as assertions.

In our current example application, the Fabric network controls the state of a train crossing, where an autonomous car decides whether to cross a railway based on the Fabric network. In this illustrative use case, an invalid ledger state read by the car can lead to a collision between the car and the train. Even on this simplified model, JPF's counterexample traces can show how an ordering service reordering or dropping messages can result in an undesired state (i.e., a collision).

The implementation already demonstrates the viability of our approach, but there are some key improvements currently left for future work, such as eliminating the need for implementing contract-specific mock classes when plugging in new smart contracts and making the transaction sequence more configurable.

5 Conclusion

We proposed a SCIL verification approach for enterprise smart contracts using model checking, which can show how platform-level behavior can impact service-level behavior through the smart contracts. Then, we analyzed HLF and proposed a high-level model of Fabric networks, which capture the main participants and transaction flow on an abstraction level sufficient for verification. Lastly, we implemented a prototype of this model, including a simple smart contract, configurable network, and fault modes to show how a model checker can prove the presence or absence of a given error property.

In the future, we would like to refine our model further and extend our prototype so that it may become a tool that can handle arbitrary chaincode – possibly enhanced with runtime defenses such as n-version programming – *out-of-the-box* and find its potential faults as well as how it is affected by platform-level faulty behavior.

[1] `github.com/AdamZsofi/hyperledger-java-model`

Acknowledgments

The project supported by the Doctoral Excellence Fellowship Programme (DCEP) is funded by the National Research Development and Innovation Fund of the Ministry of Culture and Innovation and the Budapest University of Technology and Economics under a grant agreement with the National Research, Development and Innovation Office.

The work of Zsófia Ádám was partially supported by the **ÚNKP-23-3-I-BME-9** New National Excellence Program of the Ministry for Culture and Innovation from the source of the National Research, Development and Innovation Fund.

The work of Bertalan Zoltán Péter was partially created under, and financed through, the Cooperation Agreement between the Hungarian National Bank (MNB) and the Budapest University of Technology and Economics (BME) in the Digitisation, artificial intelligence and data age workgroup.

References

[1] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich, S. Muralidharan, C. Murthy, B. Nguyen, M. Sethi, G. Singh, K. Smith, A. Sorniotti, C. Stathakopoulou, M. Vukolić, S. W. Cocco, and J. Yellick. Hyperledger Fabric: a distributed operating system for permissioned blockchains. In *Proceedings of the Thirteenth EuroSys Conference*, EuroSys '18, New York, NY, USA, 2018. Association for Computing Machinery.

[2] V. Buterin. Ethereum: a next-generation smart contract and decentralized application platform. 2014.

[3] I. Garfatta, K. Klai, W. Gaaloul, and M. Graiet. A survey on formal verification for solidity smart contracts. In *Proceedings of the 2021 Australasian Computer Science Week Multiconference*, ACSW '21, New York, NY, USA, 2021. Association for Computing Machinery.

[4] B. Beckert, M. Herda, M. Kirsten, and J. Schiffl. Formal specification and verification of hyperledger fabric chaincode. In *3rd Symposium on Distributed Ledger Technology (SDLT-2018) co-located with ICFEM 2018: the 20th International Conference on Formal Engineering Methods, Gold Coast, Australia, November 12, 2018*, pages 44–48. Institute for Integrated and Intelligent Systems, 2018.

[5] National Aeronautics and Space Administration (NASA). Java pathfinder, 2005.

[6] M. Díaz, E. Soler, L. Llopis, and J. Trillo. Integrating blockchain in safety-critical systems: An application to the nuclear industry. *IEEE Access*, 8:190605–190619, 2020.

[7] M. Kuperberg, D. Kindler, and S. Jeschke. Are smart contracts and blockchains suitable for decentralized railway control? *Ledger*, 5, 2020.

[8] D. Ongaro and J. Ousterhout. In search of an understandable consensus algorithm. In *Proceedings of the 2014 USENIX Conference on USENIX Annual Technical Conference*, USENIX ATC'14, pages 305–320, USA, 2014. USENIX Association.

Selecting Execution Path for Replaying Errors

Zsófia Erdei, Melinda Tóth and István Bozó

Abstract: The identification of the sources of a runtime error is a common task for Erlang developers. Dynamic and static tools can assist in this task. Our work aims to help Erlang developers in debugging processes to reproduce a runtime error. We would like to use and extend the static analyser framework of RefactorErl with new algorithms to support this fault localisation process. In our previous paper, we presented a symbolic execution-based analysis method to find the source of runtime errors. This paper extends that work with path selection heuristics to improve the efficiency of the algorithm in the RefactorErl framework.

Keywords: static analysis, Erlang, symbolic execution, fault localization, path selection

1 Introduction

Fault localization is a process of identifying the locations of faults in a program. Static source code analysis techniques may help programmers in various tasks: code comprehension, testing, debugging, etc. Although bugs in the software are usually discovered due to faulty behaviour (e.g. a runtime error occurs), finding the origin of the fault is a non-trivial task. Program analysis techniques with symbolic execution can help to solve this task.

In a concrete execution, a program is evaluated on a specific input, and a single control-flow path is explored. Symbolic execution [1] uses unknown symbolic variables in evaluation, allowing to simultaneously explore multiple paths that a program could take under different inputs. The use of symbolic execution can help us in fault localization.

We have previously implemented our prototype algorithm using backtracking and demonstrated how it finds an execution path to a given expression containing an error [2]. The algorithm uses a combination of the control flow graph and the RefactorErl [3] frameworks graph representation of the analysed code to determine an appropriate execution path that may lead to a given runtime error in Erlang software.

Because of the path-explosion problem, it is infeasible for symbolic execution tools to explore all execution paths of any nontrivial programs. Therefore, search heuristics are required elements of symbolic execution. Using a good search heuristic can maximize code coverage and improve the effectiveness of the analysis in practice.

In this paper, we examine several path selection heuristics that can be used to improve the efficiency of our algorithm to make our method feasible for error detection in larger software bases.

2 Background and related work

Symbolic execution [4, 1] is a technique used by many program analysis and transformation techniques, such as partial evaluation, test-case generation or model checking. It can be used for fault detection by exploring different execution paths of a program with symbolic values instead of concrete values. Symbolic values represent a range of possible values that can satisfy certain constraints. Tools based on such techniques can find errors that are hard to detect with conventional testing methods, such as buffer overflows, division by zero errors, etc.

Symbolic execution works by maintaining a symbolic state and a path condition for each execution path. The symbolic state contains the symbolic values of variables. The path condition contains the constraints on the symbolic values that are derived from branch conditions along the path. Symbolic execution uses a constraint solver to check the feasibility of each path and to generate concrete inputs that can trigger faults.

KLEE [5] uses two main search strategies: Random Path Selection and State-Based Search. Random Path Selection maintains a binary tree recording the program path followed for all active states, where the internal nodes are the ones where the execution has forked and the leaves represent the current states. The states are selected by traversing this tree from the root and randomly selecting the path to follow at branch points. During the symbolic execution when an internal node is reached, all child nodes of the given node have an equal probability to be selected by the algorithm regardless of the size of the subtrees. The biggest advantage of this strategy is that it avoids starvation occurring in loops containing symbolic conditions and resulting in quick new state creation.

While symbolic execution is not a new topic in the Erlang ecosystem, previously published papers mostly focus on formal [6, 7] and informal [8] definitions with the aim of program verification. In a previous paper [2], we present a symbolic execution technique for Erlang that can support debugging processes of Erlang developers through the RefactorErl framework. Our goal was not to verify Erlang programs but to support their debugging processes through the RefactorErl framework. Other related works on symbolic execution in Erlang focused on verification [6, 7] or used informal definitions [8].

Our proposed method was built on the RefactorErl [3] framework, which is a static code analysis tool that can be used to analyze and refactor existing Erlang code bases. The algorithm uses the control-flow graph of RefactorErl and applies backward symbolic execution to gather the constraints of the execution. Backwards symbolic execution is a technique that starts from the error location and works backwards to the input parameters of the function. This technique generates a set of constraints that must be satisfied by the input parameters to reach the error location.

The prototype algorithm uses the Z3 SMT solver to decide the reachability of a path and calculate possible input values for real execution. The Z3 SMT solver is a tool for solving logical formulas. The proposed method can be used to identify execution paths that may lead to a runtime error. We have implemented the presented algorithm for a reasonable subset of the Erlang language.

3 Path selection algorithms

The algorithm uses a kind of symbolic backward execution called call-chain backward symbolic execution [9]. This is a type of symbolic execution that mixes forward and backward symbolic execution. Inside each function, it explores the execution paths forward but it follows the call chain backwards from the target point to the program's entry point. Starting at the target expression, we search for a path from the entry point of the function containing the target expression itself. This intraprocedural part of the algorithm uses the control-flow graph of the function to look for possible paths to the target node.

Once a valid intraprocedural path is found, the next step is to determine the callers of the function. Using RefactorErl we can collect all expressions that contain such a function call. Now the expression containing the function call will be our target, and the new starting point will be the new function containing that expression.

We can see that our algorithm has two points when path selection is needed, once in the intraprocedural part and once in the interprocedural part. Using different strategies would make sense in each of these cases.

3.1 Intraprocedural strategy

For the intraprocedural part of our algorithm, we search for a path from the entry point of the function containing the target expression itself. Starting at the root of the control flow graph of the selected function, we explore as far as possible along each branch before backtracking.

If a path to the target node is found, we check the conditions along the path with the help of a constraint solver for feasibility. Depending on the result we either return the path or reject it and continue the backtracking to find another one.

Given the branching structure of the control flow graph, checking every possible path would not be feasible. To make our algorithm more efficient, we can use estimations in the intraprocedural part to reduce the problem space. Using the RefactorErl, we can query the depth of the target node in its intermediate source code representation, in the Semantic Program Graph. We can use this metric to reduce the size of the tree by removing sections of the tree that are deeper than our target.

Consider the simple example in Figure 1. This code snippet contains divisions, and if the denominator C is zero, a division by zero error occurs. Suppose that the error occurred in line 12. We can use the algorithm to find a realizable path to the target expression from the entry point of the program, and also determine a set of input values that may trigger the error. We need to traverse the control flow graph to find the target expression, but to enumerate all paths might be very expensive in larger functions. To reduce our search space we can cut branches that are deeper in the tree then our target expression. The tree next to the code snippet shows the path the algorithm traverses on the simple example function.

```
1  -module(example1).
2  -export([foo/2]).
3  foo(A, B) ->
4    C = A - B,
5    if
6      C == A -> 0;
7      C < A ->
8        if
9          B > C -> 1;
10         true -> 2
11       end;
12     C > A -> A / C
13   end.
```

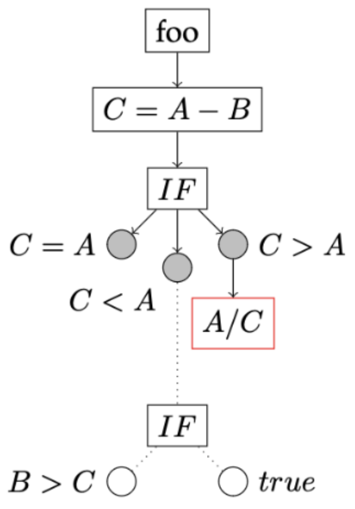


Figure 1: Example module and corresponding path selection

3.2 Interprocedural strategy

Random path selection maintains a binary tree that records the program paths that could be followed by all active processes. In this tree, the leaves represent the current processes, and the internal nodes correspond to places where execution forked. We traverse this tree from the root and randomly choose the path to follow at branch points. As a result, when we reach a branch point, the set of processes in each subtree has an equal probability of being selected, regardless of their size.

This strategy has two important properties. First, it favours processes that are high in the branch tree and therefore relatively unconstrained. Selecting these processes more frequently is valuable because they have greater freedom to reach uncovered code. Second, and most importantly, this strategy prevents starvation when some part of the program rapidly creates new states (a phenomenon known as fork bombing). In our case, this could happen for example at recursive function calls.

4 Conclusion

Our proposed method builds upon the RefactorErl framework, a static code analysis tool designed for analyzing and refactoring existing Erlang codebases. Our prototype algorithm utilizes call-chain backward symbolic execution, a combination of forward and backward symbolic exploration. Within each function, it analyzes execution paths forward, while tracing the call chain backwards from the target point to the program's entry point. Starting at the target expression, the algorithm seeks a path from the entry point of the function containing that expression. The intraprocedural phase uses the function's control-flow graph to identify potential paths to the target node.

Given the branching structure of the program graph, checking every possible path would not be feasible. To make our prototype algorithm more efficient we use various path selection strategies. We use backtracking within the functions, supplemented with improvements that take advantage of the information that can be extracted from the graph of RefactorErl, reducing the size of the graph to be traversed. In the case of the interprocedural part, we use random path selection to prevent starvation when some part of the program rapidly creates new states. Combining these strategies we can effectively identify execution paths that might lead to runtime errors.

References

[1] Baldoni, R., Coppa, E., D'elia, D. C., Demetrescu, C., and Finocchi, I. A survey of symbolic execution techniques. *ACM Comput. Surv. 51*, 3 (May 2018).

[2] Erdei, Z., Tóth, M., and Bozó, I. Supporting the debugging of Erlang programs by symbolic execution. *Accepted to Acta Univ. Sapientiae, Informatica,* (2024).

[3] Tóth, M., and Bozó, I. Static analysis of complex software systems implemented in erlang. Central European Functional Programming Summer School – Fourth Summer School, CEFP 2011, Revisited Selected Lectures, Lecture Notes in Computer Science (LNCS), Vol. 7241, pp. 451-514, Springer-Verlag, ISSN: 0302-9743, 2012.

[4] King, J. C. Symbolic execution and program testing. *Commun. ACM 19*, 7 (July 1976), 385–394.

[5] Cadar, C., Dunbar, D., and Engler, D. Klee: Unassisted and automatic generation of high-coverage tests for complex systems programs. In *Proceedings of the 8th USENIX Conference on Operating Systems Design and Implementation* (USA, 2008), OSDI'08, USENIX Association, pp. 209–224.

[6] Vidal, G. Towards symbolic execution in erlang. In *Perspectives of System Informatics* (Berlin, Heidelberg, 2015), A. Voronkov and I. Virbitskaite, Eds., Springer Berlin Heidelberg, pp. 351–360.

[7] Vidal, G. Towards erlang verification by term rewriting. In *Logic-Based Program Synthesis and Transformation* (Cham, 2014), G. Gupta and R. Peña, Eds., Springer International Publishing, pp. 109–126.

[8] Earle, C. B. Symbolic program execution using the erlang verification tool. In *9th International Workshop on Functional and Logic Programming, WFLP'2000, Benicassim, Spain, September 28-30, 2000* (2000), M. Alpuente, Ed., pp. 42–55.

[9] Ma, K.-K., Yit Phang, K., Foster, J. S., and Hicks, M. Directed symbolic execution. In *Static Analysis* (Berlin, Heidelberg, 2011), E. Yahav, Ed., Springer Berlin Heidelberg, pp. 95–111.

Design Space Exploration of Verifiable Credential Schemas using Partial Graph Modeling

Martin Farkas, Bertalan Zoltán Péter, Imre Kocsis

Abstract: With the advent of the use of standardized verifiable credentials, information verifiers begin to face a design space exploration problem: what kinds and potential sets of verifiable claims to request from persons and organizations to meet their verification needs, especially so that on the one hand, they have to maximize the trustworthiness of claim verification, but on the other hand, also observe the principle of data minimization and the correlation risk minimization needs of credential holders. We propose approaching this design problem through partial graph modeling and generation. We demonstrate that partial graph modeling can be amenable to express the potentially diverse information graph requirements of a verification scenario and generate admissible templates for so-called verifiable presentation grapht. This facilitates the application of established design space exploration approaches, including the enumeration of options integrated with multi-aspect evaluation and rank ordering.

Keywords: verifiable credentials, information graph, partial graph modeling

1 Introduction

In recent years, verifiable credentials (VCs) [1, 2] have stirred the interest of not just researchers but also legislators and corporations, as the European Union decided to adopt the ESSIF framework [3]. They are a crucial part of a novel and proliferating identity model called self-sovereign identity, which aims to offer its users more interoperability, privacy, and control over their identity on the Internet. VCs represent claims, and as more and more interdependent ecosystems and policies utilize them and the complexity of the claims grows, their design rapidly becomes non-trivial.

The following question arises: How do we maximize the expressive power of a related set of verifiable credentials while maintaining their holder's control over their claims by minimizing the amount of data shared and the risk of correlation?

To provide an answer to this question, we propose a partial-graph-modeling-based design space exploration workflow using the principles of model-driven architectures.

We aim to design a framework for systematically designing verifiable credential schemas that satisfy complex requirements. To this end, we use partial graph modeling and graph generation [4] to generate diverse VC architectures, as the verifiable credential data model can be expressed as **subject-property-value** relationships, and thus verifiable credentials can be modeled as information graphs. To help ensure interoperability, the generated designs follow the model-driven architecture (MDA) approach [5].

After a brief introduction of verifiable credentials, we present our approach, its architecture, and its benefits. Finally, we discuss our future aims regarding our method.

2 Preliminaries

Verifiable credentials (VCs), as defined by the W3C [1], are a digital form of credentials with cryptographically verifiable authorship, making them more trustworthy and tamper-evident than their physical counterparts. They represent claims about a subject backed by their authors. These claims can be expressed as **subject-property-value** relationships, e.g. `"Tom"-"has"-"blue eyes"`.

It is important to note that verifying a VC does not directly imply the truth of the claims encoded in it, only their authenticity and currency. Rather, after authenticity and currency are established, a verifier may evaluate the claims based on their own policy and trust in the author.

VCs can be presented as *verifiable presentations (VPs)*, where multiple VCs are cryptographically linked together by the holder, meaning that not only the authenticity of the credentials is ensured, but so is the authenticity of the entity presenting them. VPs can also contain additional arbitrary data, for which one use-case links the aligned entities between related VCs.

In this model, the W3C identifies five roles:

- The **holder**, who possesses and presents VCs;
- The **issuer**, who asserts claims about one or more subjects, and creates VCs;
- The **subject**, about which claims are made;
- The **verifier**, who receives and processes VCs presented to them;
- And the **verifiable data register (VDR)**, which mediates the creation and verification of publicly used data, such as identifiers, *verifiable credential schemas*, revocation registries, and so on.

Verifiable credential schemas are data schemas that enforce a specific structure over data collection. *Data verification schemas* ensure that the structure and contents of a VC conform to a published schema. *Data encoding schemas* are used to map the contents of a VC to an alternative representation format.

VC schemas make establishing the structure of common claim sets easier; their design influences the privacy characteristics of a verifiable credential ecosystem. How the claims are divided up between credentials plays a crucial role in how expressive a set of related VCs is and how precisely a holder can control what they disclose when they present a VC or VP.

An important concept in this case is *Entity Alignment*, which is the process of identifying nodes that represent the same entity across multiple ontologies. In our case, if two VCs contain claims about the same subject, there needs to be a process that enables us to identify them and thus enable us to reason over the joint information graph, as shown in Figure 1.

Reasoning over information graphs constructed from verifiable credentials allows us to make inferences and evaluate policies over subjects in a privacy-preserving way. Although many small examples have been demonstrated in existing research, a systematic design approach has yet to be proposed for VC shema sets, over which a policy may reason.

3 Approach

To provide a systematic method for designing credential schemas which contain claims from a predefined ontology, i.e. which credential should contain which claim from an information graph model, we propose a *design space exploration (DSE)* [6] process using *partial graph modeling* [4], which produces templates for platform-specific verifiable credential schemas, over

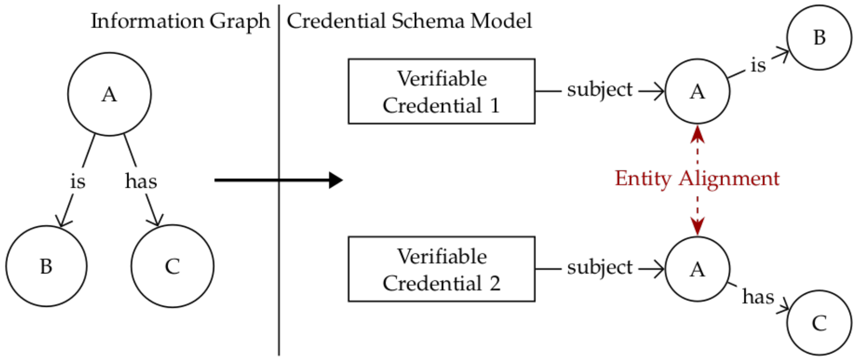


Figure 1: Decomposition of an information graph into verifiable credential schemas

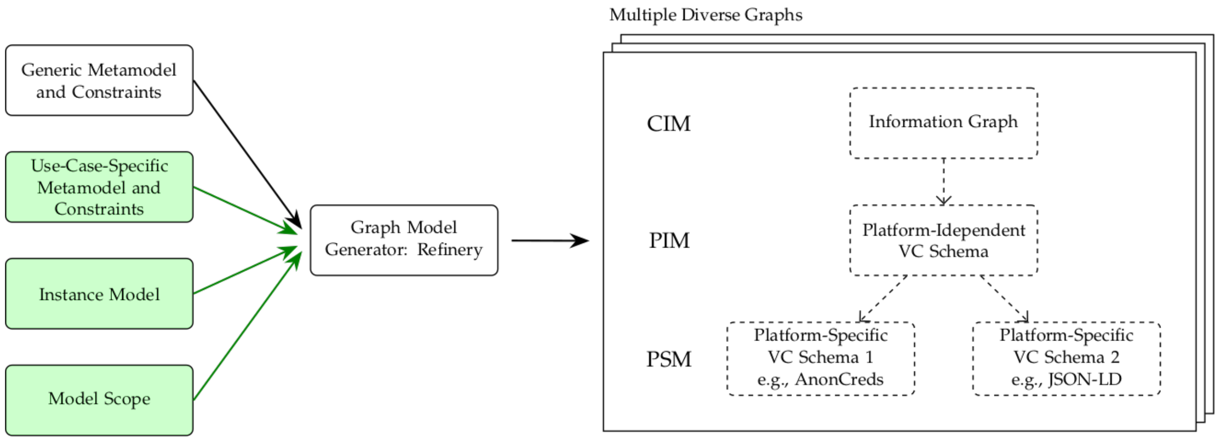

Multiple Diverse Graphs
Generic Metamodel and Constraints
Use-Case-Specific Metamodel and Constraints
Instance Model
Model Scope
Graph Model Generator: Refinery
CIM
PIM
PSM
Information Graph
Platform-Idependent VC Schema
Platform-Specific VC Schema 1 e.g., AnonCreds
Platform-Specific VC Schema 2 e.g., JSON-LD

Figure 2: The architecture of model generation using Refinery [8]

which VCs can be issued, from which the required information graph can be constructed and reasoned over.

These designs follow the MDA [5] approach, as we recognize that interoperability is desirable between ontologies and diverse sets of VC schemas, as well as between platform-independent and platform-specific VC schema models, on platforms such as Hyperledger Anoncreds [7] and JSON-LD [2].

To generate our models, we use Refinery [8][1], a partial graph refinement and graph generation tool, which can generate diverse graphs based on a ruleset written in its own domain specific language. It requires the following input:

- A **metamodel,** specifying nodes and edge types and how they can be connected;
- **Predicates,** which declaratively denote patterns in a given metamodel;
- **Error predicates,** which denote unwanted patterns, suitable for effectively shrinking the design space;
- **Instance models,** which are specific model parts that are required to be present in the generated model;
- and the **model scope,** which denotes the desired size range of the generated graph, with the option of including some specific model elements.

We define a generic metamodel and predicate set[2], which generates models that reflect the core logic of MDA. The generated model can be divided into three subgraphs analogous to the steps of MDA. Figure 2 provides an overview of the generated graphs.

The first is an **information graph**. This is the graph over which we want to later reason.

The relationships are then "partitioned" into claims within VCs in a traceable way, meaning there is a one-to-one correspondence between the relationships in the information graph and the claims they turned into within the credentials. This forms a **set of related VCs.**

Then, the abstract set of VCs is refined into **credential-format-specific representations**, from which they can be directly implemented. This ensures interoperability with multiple VC frameworks, making the final system design more versatile.

The model can be refined with use-case-specific metamodel elements, predicates, and an instance model. This means we can extend the model to satisfy complex requirements that can be stated as patterns on an information graph. Examples of such requirements include: the information graph should be a specific graph instance; two relationships must go into separate credentials; a credential must contain at most two claims; etc.

[1]Online version of the tool is available at https://refinery.services/

[2]Current assets are available on GitHub at https://github.com/BlackLight54/dse-vc-refinery

Through this approach, a designer can generate diverse verification requirement setups for their specific use case, which satisfy complex requirements, and model an interoperable, potentially provably privacy-preserving credential ecosystem.

4 Conclusion

In this paper, we proposed a novel method for systematically designing verifiable credential ecosystems from information graphs. We presented our approach for generating diverse data presentation requirement sets in the verifiable credential context, utilizing partial graph modeling. Our approach follows MDA principles, facilitating not only systematic design but also technical interoperability.

Based on our initial proof-of-concept models, we are working on elaborating the full graph generation facilities required for handling representative, real-life scenarios. We also aim to create the methodology and tooling for a complete design workflow.

Acknowledgments

This work was partially created under, and financed through, the Cooperation Agreement between the Hungarian National Bank (MNB) and the Budapest University of Technology and Economics (BME) in the Digitisation, artificial intelligence and data age workgroup.

The work of Bertalan Zoltán Péter, partially supported by the Doctoral Excellence Fellowship Programme (DCEP) is funded by the National Research Development and Innovation Fund of the Ministry of Culture and Innovation and the Budapest University of Technology and Economics under a grant agreement with the National Research, Development and Innovation Office.

References

[1] D. C. Manu Sporny, Dave Longley. Verifiable credentials data model v2.0. `https://www.w3.org/TR/vc-data-model-2.0/`.

[2] G. Cohen. Verifiable credentials json schema specification. `https://www.w3.org/TR/vc-json-schema/`.

[3] Essif lab. `https://essif-lab.eu/`.

[4] M. Famelis, R. Salay, and M. Chechik. Partial models: Towards modeling and reasoning with uncertainty. In *2012 34th International Conference on Software Engineering (ICSE)*, pages 573–583, 2012.

[5] S. J. Mellor, K. Scott, A. Uhl, and D. Weise. Model-driven architecture. In *International conference on object-oriented information systems*, pages 290–297. Springer, 2002.

[6] E. Kang, E. Jackson, and W. Schulte. An approach for effective design space exploration. In *Proceedings of the 16th Monterey Conference on Foundations of Computer Software: Modeling, Development, and Verification of Adaptive Systems*, FOCS'10, pages 33–54, Berlin, Heidelberg, 2010. Springer-Verlag.

[7] S. Curran, H. Yildiz, and S. Curren. Anoncreds specification. `https://hyperledger.github.io/anoncreds-spec/`, 2022.

[8] K. Marussy, A. Ficsor, O. Semeráth, and D. Varró. Refinery: Graph solver as a service. *ICSE 2024 Demonstrations*, 2024.

Towards Correct Dependency Orders in Erlang Upgrades

Daniel Ferenczi, Melinda Tóth

Abstract: Erlang tooling offers rich options to control the exact tasks to perform during an upgrade. This control aims to allow for zero-downtime upgrades. Upgrades affecting multiple dependant modules have to reflect the dependency order in the upgrade's configuration, as an erroneous configuration results in unintended behaviour, possibly even downtime. This paper presents two static analysis-based checkers for verifying module-related aspects of upgrades. In our first analysis, we compare the actual dependency order derived from the source code with that expressed in the upgrade's configuration. We also analyze the code to find cyclic dependencies among its modules. These pose a problem during upgrades and are generally good practices to avoid.

Keywords: Erlang; upgrades; static analysis; upgrade safety; dependency order; RefactorErl

1 Introduction

Ensuring continuous operation of IT services is considered the norm in today's software environment. While decades ago this was typically a feature of safety-critical systems, today we can encounter it in many customer-facing applications: retail, banking, and entertainment. The need for this is reasonable considering the global nature of these services. These applications not only operate around the clock but are changed in the meantime without any noticeable impact on the end user. The state-of-the-art tooling based on containers, serverless services and other features offered by hyperscalers standardizes some of the associated tasks. These solutions however require additional effort from operational specialists, for example, load balancers and draining periods have to be configured and the application state has to be preserved. These update schemes also work on a high granularity: even small code changes require the replacement of a unit typically composed of the whole application binary. As larger applications contain multiple such units, upgrades with a broader scope will require care when determining which unit to change at each upgrade step.

In our work, we researched this challenge in Erlang-based [1] software stacks. Erlang was developed with built-in features for concurrency, fault tolerance and continuous operation. This thins the software stack Erlang applications require. Consequently, developers maintaining them can create zero-downtime upgrades by solely using the features of the language and its runtime. With regards to upgrades, Erlang allows for live replacement of application modules and has upgrade-related tooling built-in into the language as well. Although modules are smaller units than the upgradeable units we discussed previously, the problem of identifying how they depend on each other inside a given application is still applicable. Module dependency structure is also a good candidate for analysis: cyclic dependencies are best to avoid in general, the Erlang release handling guidelines even advise against using them, as they might make safe upgrades impossible. We also support the detection of these with our checker, which will aid developers when structuring their code.

As the application modules' code meant to be upgraded and the upgrade specification are both expressed in Erlang, we based our work on existing static analysis tools to inspect the dependencies in the code. This paper is structured as follows: in the next Section we briefly present the Erlang language and the RefactorErl [2] static analyzer we base our work open. In Section we present the problems our checker focuses on, and its Erlang-specific aspects in more detail. In Section we present related research.

2 Erlang and RefactorErl

Erlang is a dynamically typed, functional programming language. It was developed at Ericsson for use in the telecommunications domain. Given its origins, it has been designed as a language apt for implementing highly scalable, fault-tolerant, distributed systems. These features are provided by the runtime and standard libraries that are included with Erlang distributions. Bundled tooling includes software for defining and managing upgrades on a fine-grained level to ensure disruption-free upgrades. These tools are used for defining applications, releases (entities composed of multiple Erlang applications) and respective upgrade files, `appup` and `relup`. These upgrade files are interpreted by the Erlang runtime's Release Handler [3].

RefactorErl [2] is a static analyzer for Erlang that also supports code comprehension and refactoring. It analyzes loaded code, generates its abstract syntax tree and enhances it with output from different analyzers: function, data-flow, etc. The resulting Semantic Program Graph (SPG) [4] allows for easy analysis of the loaded program through a query languageand the implementation of new checkers. Given its features, we chose it as our tool to implement our code checker.

3 Problem Statement

Discrepancy detection in upgrade definitions Erlang source files (modules) may contain references to functions exported in other modules. The hierarchy upon which modules depend on each other is important during a release's upgrade cycle. As complex upgrades involve changes in multiple modules, if these depend on each other, their dependency has to be reflected in the upgrade steps as well. For example, if module A depends on module B, we have to load B's new code and resume its process before we resume processes running module A. The steps that implement an upgrade for an application are typically declared in a `appup` file by the developer specific to the application being updated. As a release may consist of multiple applications, `appup` files are later combined into a `relup` file. Although `relup` files are typically generated from the `appup` files with the help of the release-related tooling offered by Erlang. Naturally, the `relup` file may also be written manually, so to account for both manual and automated workflows we compare the dependencies derived from the source files with those implicitly expressed in the `relup` files. An example of an `appup` and its derived `relup` file can be seen in Figure 1.
In the example's `appup` file we declared an upgrade from version 1.0 to 1.1 and a downgrade from version 1.1 to 1.0 respectively at lines 3 and 7. Specifically, we tell the Release Handler to load the newer version of `depmod` and update module `servermod`, which depends on `depmod`. The dependency relation is declared in the lists in lines 5 and 9. If we look at the list of instructions, both `load_module` and `update` atoms declare code changes. The difference lies in that `update` takes care of temporarily suspending processes running the target module, and transforming the internal state of the running process if the new version requires it. These additional operations allow for zero-downtime upgrades. `servermod` being a server implementation requires `update` for its code upgrade. The generated `relup` file is on a similar structure: it contains first the upgrade and then the downgrade instruction but contains the lower-level operations executed by Erlang's Release Handler. Without going into detail, we can observe how the dependency relation results in `depmod` being changed before the dependant `servermod` module in lines 19 and 20 for the upgrade, and 26 and 27 for the downgrade. In these files, we are looking for `suspend`, `load` and `resume` instructions to ensure that these dependencies are updated before dependant modules. Our research aims to verify whether the dependency order expressed in `relup` files is consistent with the actual dependency of the modules. The actual dependencies of an application are present in the source files. For their analysis, we rely on RefactorErl's feature for graphing and analyzing module dependencies.

```
%% appup file for application release_tst
{"1.1",
  [{"1.0", [
    {load_module, depmod},
    {update, servermod, [depmod]}
  ]}],
  [{"1.0", [
    {load_module, depmod},
    {update, servermod, [depmod]}
  ]}]
}.

%% relup file for release consisting of app release_tst
{"1.1",
 [{"1.0",[],
   [{load_object_code,{release_tst,"1.1",[servermod,depmod]}},
    point_of_no_return,
    {suspend,[servermod]},
    {load,{depmod,brutal_purge,brutal_purge}},
    {load,{servermod,brutal_purge,brutal_purge}},
    {resume,[servermod]}]}],
 [{"1.0",[],
   [{load_object_code,{release_tst,"1.0",[servermod,depmod]}},
    point_of_no_return,
    {suspend,[servermod]},
    {load,{servermod,brutal_purge,brutal_purge}},
    {load,{depmod,brutal_purge,brutal_purge}},
    {resume,[servermod]}]}]}.
```

Figure 1: Example of appup and relup files

Cyclical dependency detection Cyclical dependencies amongst modules may also put upgrades at risk. The difficulties in determining the correct order for defining an upgrade are noted in the Release Handling Section of the Erlang manual. In most cases, it is best to avoid them altogether for code meant to be upgraded. As their presence is safe for modules that are not updated in a given relup, we aim to only detect cyclic dependencies amongst modules configured for an upgrade. For this checker, we also rely on the graph analysis feature of RefactorErl. For the analysis, the dependency structure will be modelled as a graph, where the modules will be the nodes and dependency relations the edges. Our task will be to determine if such a graph, excluding modules not being subject to an upgrade has a topological ordering.

4 Related work

Previous research [5] has analyzed the issue of cyclical dependencies in Erlang. The main difference lies in the problem domain: while the author's work focuses on clean code and refactoring, our present work concentrates on supporting safe, disruption-free upgrades. Upgrade safety has been researched emphasizing runtime facets typically unrelated to an application's implementation language, like connection migration [6] between application versions or tools to support schema changes in backing databases [7, 8]. Erlang is singular in this regard, as its runtime provides facilities for handling state. Upgrade safety of Erlang [9] has been researched with RefactorErl with a focus on detecting references in the source code that might become invalid as the application is upgraded. Researching different unsafe patterns and a generic approach to support upgrade safety are further areas worth exploring.

5 Conclusions

Erlang offers the tools necessary to create application releases with fine-grained instructions to ensure disruption-free upgrades of modules. To achieve this, code has to be structured in an upgradeable manner, and the release's descriptor file should also reflect this structure correctly. Our research demonstrates our initial steps to implement checkers based on the static analysis tool RefactorErl, to aid developers in creating safe upgrades.

References

[1] Cesarini, F. & Thompson, S. Erlang Programming: A Concurrent Approach to Software Development. (O'Reilly Media,2009)

[2] Bozó, I., Horpácsi, D., Horváth, Z., Kitlei, R., Köszegi, J., M., T. & Tóth, M. RefactorErl - Source Code Analysis and Refactoring in Erlang. *Proceedings Of The 12th Symposium On Programming Languages And Software Tools, ISBN 978-9949-23-178-2*. pp. 138-148 (2011,10)

[3] Logan, M., Merritt, E. & Carlsson, R. Erlang and OTP in Action. (Manning Publications Co.,2010)

[4] Horváth, Z., Lövei, L., Kozsik, T., Kitlei, R., Víg, A., Nagy, T., Tóth, M. & Király, R. Modeling semantic knowledge in Erlang for refactoring. *Knowledge Engineering: Principles And Techniques, Proceedings Of The International Conference On Knowledge Engineering, Principles And Techniques, KEPT 2009*. **54(2009) Sp. Issue** pp. 7-16 (2009,7)

[5] Li, H. & Thompson, S. Refactoring Support for Modularity Maintenance in Erlang. *2010 10th IEEE Working Conference On Source Code Analysis And Manipulation*. pp. 157-166 (2010)

[6] Naseer, U., Niccolini, L., Pant, U., Frindell, A., Dasineni, R. & Benson, T. Zero Downtime Release: Disruption-Free Load Balancing of a Multi-Billion User Website. *Proceedings Of The Annual Conference Of The ACM Special Interest Group On Data Communication On The Applications, Technologies, Architectures, And Protocols For Computer Communication*. pp. 529-541 (2020)

[7] Maule, A., Emmerich, W. & Rosenblum, D. Impact Analysis of Database Schema Changes. *Proceedings Of The 30th International Conference On Software Engineering*. pp. 451-460 (2008)

[8] Meurice, L., Nagy, C. & Cleve, A. Detecting and Preventing Program Inconsistencies under Database Schema Evolution. *2016 IEEE International Conference On Software Quality, Reliability And Security (QRS)*. pp. 262-273 (2016)

[9] Ferenczi, D. & Tóth, M. Static analysis for safe software upgrade. *Annales Mathematicae Et Informaticae*. (2023)

Clustering and Community Detection in Nested Graphs

Imre Gera and András London

Abstract: Nestedness is a strict structural property that was first observed in mutualistic ecological networks. Since then, numerous metrics have emerged that quantify the global nestedness of a network, although methods to detect the detailed local nestedness structure of networks were missing for a long time. Our research goal is to compare multiple approaches to detect local nestedness in networks, and to introduce hierarchical clustering for this purpose.

In this paper, we compare both hierarchical clustering and overlapping community detection algorithms that detect nested structures in both bipartite and non-bipartite networks. We show that hierarchical clustering can be adapted to unveil nested structures and find a fully nested clustering of the network, while overlapping community detection can be used to find all nested communities of a network.

Keywords: nestedness, network science, clustering, community detection

1 Introduction

Detecting structural patterns and groups of nodes with common properties, often referred to as communities, in graphs has always been a crucial part of network science. Various methods and approaches have emerged to find these patterns, including clustering (partitioning) and overlapping community detection. Most of these methods are general clustering and community detection algorithms, relying on the definition of a community being a group of vertices densely connected inside and having less connections outside the group. Community detection and clustering, though, can also be used to identify groups of nodes with special properties.

In this paper we are going to look at an unweighted, undirected graph $G = (V, E)$, with V being the set of vertices ($n = |V|$), and $E = \{(i, j) \mid i, j \in V\}$ being the set of edges. Since we work with undirected graphs, $(i, j) \in E \Leftrightarrow (j, i) \in E$.

1.1 Clustering or community detection

While clustering and community detection are often used interchangeably, here we are going to refer to different tasks under these two names. In this paper, we are going to refer to *clustering* as partitioning, where every node is assigned a single label: the cluster it belongs to. On the other hand, when we refer to the task as *(overlapping) community detection*, a node can be assigned multiple labels (or communities it belongs to). In between the two approaches lies hierarchical clustering, where the result is a list of n clustering results, the i-th level containing i clusters. Hierarchical clustering can be agglomerative (bottom-up), or divisive (top-down).

1.2 Nestedness

We call a graph *fully nested* if, for all vertices (or in the case of bipartite graphs, all vertices of the same class) an ordering $v_1, v_2, \ldots, v_n$ can be given such that $N_{v_1} \subseteq N_{v_2} \subseteq \cdots \subseteq N_{v_n}$, where N_i is the neighborhood of vertex i.

Nestedness is a structural property that has been observed primarily in ecological bipartite networks [1], but largely remained one that was only quantified on graphs and treated as a global property, instead of a local one, at the node level. To quantify nestedness on a graph level, several nestedness metrics were introduced, including NODF [2], the binmatnest temperature [3] and discrepancy [4]. These, however, don't give us a node-level view of nestedness, only a single number that represents how nested the entire graph is.

For nestedness to work as a local property, we need to define a metric to quantify it between a pair of nodes. To do this, we will use

$$\mathrm{nest}(i,j) = \frac{|N_i \cap N_j|}{\min\{N_i, N_j\}}, \tag{1}$$

where N_i is the neighborhood of vertex i. In the case of non-bipartite graphs, we ignore the edge $(i,j) \in E$ for the calculation, if it exists, i.e., using $N_i \setminus \{j\}$ and $N_j \setminus \{i\}$ instead of N_i and N_j. As such, the bipartite and non-bipartite full graphs are considered fully nested.

2 Algorithms for detecting nestedness

2.1 Hierarchical clustering

First, we are going to look at using hierarchical clustering to detect nestedness, using both bottom-up (agglomerative) and top-down (divisive) methods.

Bottom-up clustering can be implemented by having a pairwise distance matrix of the nodes and merging the two clusters with the smallest distance. The distance of two clusters is determined by the *linkage method* used – in this paper, we use *single-linkage* (the minimum of the pairwise distances among clusters), *complete-linkage* (the maximum of the pairwise distances) and *average-linkage* (the average of the pairwise distances).

We implement top-down clustering by modifying the Girvan-Newman algorithm [5]. In the original algorithm, edges with the highest edge betweenness are deleted in every step until the number of components increases. We have made two modifications to the algorithm:

1. the algorithm works on the auxiliary *nestedness graph* instead of the original one as its input: a graph with edge weights $w_{ij} = \mathrm{nest}(i,j)$, excluding edges with $w_{ij} = 0$.
2. the edge to be deleted is the one with the maximal $\mathrm{bc}(i,j)/\mathrm{nest}(i,j)$ value, where bc is the edge betweenness centrality used by the original algorithm

We have also created an alternative "nested" version, where we skip recalculating the edge betweenness values and instead only use the $\mathrm{nest}(i,j)$ value.

2.2 Overlapping community detection

In order to find overlapping nested communities, we can use the following algorithm introduced in [6]. In summary, the steps are the following:

1. create an auxiliary *nestedness graph* G_{nest} where $(i,j) \in E(G_{\mathrm{nest}})$ if $N_i \subseteq N_j$
2. merge nodes with equal neighborhoods (this removes bidirectional edges from G_{nest}, making it acyclic)
3. perform transitive reduction, i.e., delete edge $(i,k) \in E$ if $(i,j),(j,k) \in E$
4. find maximal length (non-expandable) directed paths
5. replace compacted nodes with the original ones

The algorithm does not remove information related to nestedness and does not work with heuristics, and as such it finds all nested communities.

3 Results

In this section, we are going to highlight our key findings regarding nested clustering and community detection, obtained using the algorithms described in sections and . We are going to perform comparisons, devise new global nestedness metrics from them and compare them to existing solutions. We show that both hierarchical clustering and overlapping community detection are capable of providing an overview of the network's nestedness, and we also detail what situations they are useful in. We also show that our methods are not restricted to bipartite graphs only (like the previous nestedness metrics).

3.1 Data

We first test both algorithm families using synthetic networks. In the case of clustering, we generate fully nested bipartite graphs with multiple components, then introduce permutation inside each component with probability p_i and among the clusters using a different probability p_a. The perturbation is performed by replacing an original edge (part of a fully nested network) with a random edge, either inside the component or among two components. We can then measure how close the clustering algorithm's best clustering was to the original labels. In the case of community detection, we generate a random directed acyclic graph (a community graph) and then generate a bipartite graph with a matching nestedness structure.

Then, we use real bipartite graphs from the Web of Life dataset [7] to measure their nestedness, and finally we evaluate our algorithms on non-bipartite networks commonly used for traditional clustering and community detection [8].

3.2 Experiments

Synthetic networks

We have evaluated five hierarchical clustering methods: three linkage methods for the bottom-up algorithm (single-linkage, average-linkage, complete-linkage) and two versions of the top-down algorithm (one called "full", using betweenness centrality, and one called "nested", using $\mathrm{nest}(i, j)$ only).

We have found that the average-linkage and complete-linkage versions found the largest fully nested clustering on the synthetic networks, followed by the two top-down algorithms and finally the single-linkage bottom-up method. Despite this, the top-down algorithm's "full" version had the highest ARI index for smaller permutation values, while the single-linkage bottom-up method was the best at high perturbation levels.

On the synthetic bipartite networks for community detection, we have found that the algorithm was able to fully reconstruct the original communities.

Real networks

Evaluating the algorithms on bipartite graphs from the Web of Life dataset, we find that since average cluster sizes are high (the means being around 50%) and there is a large amount of overlapping communities, the networks show some local nestedness, but are far from being fully nested. This is confirmed by the other global nestedness measures, with all being in the 10-50% region. The Web of Life dataset does contain some fully nested graphs, for example, the `A_HP_015` host-parasite network.

The non-bipartite graphs showed much less nestedness overall. Zachary's karate club network showed lots of smaller nested clusters, while the Florentine families network had tiny nested communities, sometimes with very little overlap. In some networks, like the aforementioned families network, nestedness is best avoided – e.g., as a way to avoid influence from others in the network.

We have also derived metrics to quantify nestedness: in the case of clustering algorithms, the first step where all clusters are fully nested can be used to measure nestedness – the more steps are needed, the less nested a graph is. In the case of overlapping nested communities, we

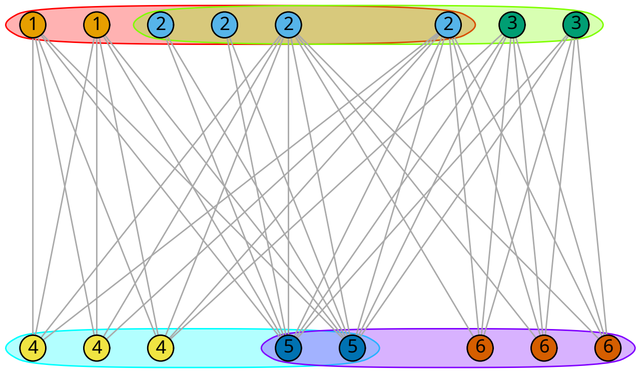


Figure 1: The `M_PL_070` pollination network with a fully nested clustering indicated by vertex colors and labels, and the overlapping nested communities indicated by colored groups.

can take the average of the fraction of communities each vertex is part of. If every vertex is part of only one community and there is a single community in the graph, the graph if fully nested.

4 Conclusion

We have compared algorithms using multiple approaches to detect local nestedness in networks. Hierarchical clustering allows us to select a desired clustering based on different criteria (e.g., the nestedness of the clusters), while overlapping community detection lets us see the overall nested structure of the graph, albeit at the cost of the difficulty to parse it. The real world networks we examined showed varying levels of nestedness: the ecological networks had higher, while the non-bipartite ones had low nestedness values.

References

[1] M. S. Mariani, Z.-M. Ren, J. Bascompte, and C. J. Tessone. Nestedness in complex networks: observation, emergence, and implications. *Physics Reports*, 813:1–90, 2019.

[2] M. Almeida-Neto and W. Ulrich. A straightforward computational approach for measuring nestedness using quantitative matrices. *Environmental Modelling & Software*, 26(2):173–178, 2011.

[3] M. Ángel Rodríguez-Gironés and L. Santamaría. How foraging behaviour and resource partitioning can drive the evolution of flowers and the structure of pollination networks. *The Open Ecology Journal*, 3:1–11, 2010.

[4] R. A. Brualdi and J. G. Sanderson. Nested species subsets, gaps, and discrepancy. *Oecologia*, 119:256–264, 1999.

[5] M. E. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical review E*, 69(2):026113, 2004.

[6] I. Gera and A. London. Detecting and generating overlapping nested communities. *Applied Network Science*, 8(51), 8 2023.

[7] Bascompte Lab. Web of life: ecological networks database. `https://www.web-of-life.es`.

[8] J. Leskovec and A. Krevl. SNAP Datasets: Stanford large network dataset collection. `http://snap.stanford.edu/data`, 6 2014.

Effective Heuristics for Accelerated Branch and Bound Solver of Process Network Synthesis Problems

Emília Heinc and Balázs Bánhelyi

Abstract: The P-graph methodology can be used to find the optimal solution for large process systems. This methodology solves the combinatorial part of the problem more efficiently than the traditional branch and bound method due to the relationships inherent in the structure. However, reducing the number of possibilities developed in the constraint functions also plays a major role in this algorithm. In this publication, we present a new constraint function that also takes into account the minimum cost structure and compares it with earlier versions.

Keywords: P-graph, Accelerated Branch and Bound, Heuristics

1 Introduction

The task of process synthesis is to determine the optimal structure of a process system and the optimal configurations and operating sizes of the functional units that make up the system and perform various operations [1]. Process synthesis plays a critical role in reducing material and energy consumption and negative environmental impacts, thereby increasing profitability. Several examples in the literature demonstrate that efficient process synthesis can reduce energy consumption by up to 50% and costs by 35% [2]. Ideally, the structure of a process and the operational configurations that make up the process could be designed and synthesized simultaneously because their performance interacts. In practice, however, it is extremely difficult due to the simultaneous continuous and discrete nature of the task. The discrete nature is caused by the structure of the process, which leads to the combinatorial complexity of the problem that makes it complex to find an optimal solution to the problem. The process network synthesis problems formulate a MIP problem with many binary variables. Finding the optimal subnetwork is an NP-hard problem. Combinatorial analysis can be applied to this type of problem. The method is used to reduce the number of possible solutions by exploiting the unique properties of the so-called PNS (Process Network Synthesis) problems is the accelerated branch and bound method [3]. It bases on the branch and bound method, i.e., the method uses a lower bound submethod to exclude the solutions that cannot provide a better solution than the current best solution. It is critical for the computation time of solving the problem with the B&B method to find a tighter lower bound submethod. The currently available implementations and the previous studies do not exploit all the information, considering only the continuous part of the problem by calculating the LP relaxation of the MIP problem. In this article, we introduce a better selection strategies for the ABB algorithm. Different heuristics for both material selection and decision mapping were presented. Furthermore, the ABB algorithm was reorganized so that we can also review our previous decisions.

2 P-graph

The P-graph (Process Graph) methodology was developed in the early 1990s for the complex chemical production system to model and optimize. Its name derives from a directed straight graph obtained by P-graph, which provides the ability to use combinatorial possible solution structures to determine the optimum for large tasks [4]. The P-graph methodology based on graph theory and combinatorial techniques provides a solution to facilitate finding the optimal PNS subproblem. In the methodology, we use a directed bipartite graph to represent the structure of a process system. We distinguish two kinds of nodes, the material (set of M) and operating units (set of O) in the graph. The directed edges represent the connection between the operating units and materials. The edges from the materials to the operating units mark the relation of the operating units that consume the materials. The edges from the operating units to the materials represent the relation of producing the materials. In the PNS problems, costs can be assigned to the operating units and raw materials. We distinguish three types of materials-raw materials must be consumed by the operating units and not be produced, the products must be produced by the operating units not be consumed, and the intermediate materials can be produced or consumed by the operating units. The goal of the model is to produce all of the products from the raw materials at minimum cost. To solve this problem, we first find the maximum structure which contains all combinatorically feasible process structures. This method is called the MSG method (Maximal Structure Generation) [5].

3 SSG algorithm

Further investigation is aided by the SSG (Solution Structure Generation) algorithm, which generates each combinatorially possible structure exactly once. The algorithm is based on decision mappings. Decision mappings involve deciding which material to use for one or more operational units, i.e., which operational units are involved in a given solution structure [6]. Consequently, during decision mapping, we also decide which operational units will be excluded from the given structure. We must be consistent in our decisions, because even if it has already been decided that an operating unit for one material should not be included in the structure, we cannot choose again when deciding for another material. All output materials for an operation unit, if included in the structure, must be specified; an inconsistent decision would result in certain substances being produced and certain substances not. The SSG implementation of the decision mapping based algorithm calls itself recursively [7, 8].

4 ABB method

Since this is a mixed-integer programming problem, they can also use general branch and bound-type methods to solve this model. Although the optimal solution to the problem can also be determined using these methods, their efficiency can be further improved since the special properties of the synthesis tasks are not taken into account in the search for a solution. Accordingly, the P-graph method for determining the optimal solution is a special algorithm of the Constraint and Separation type, ABB (Accelerated Branch and Bound) use.

This algorithm uses the previously described decision mappings of the SSG algorithm for binary variables in the B&B tree. Previously, the B&B method used continuous relaxation of the mathematical model in addition to the structural constraints of the constraint SSG. In this relaxed model, the binary variables (y_i) were not considered, and the model was limited to determining the optimal values of the continuous variables (x_i). This optimization task provided a lower bound on the operating costs.

5 Modified ABB method with heuristics

The original ABB algorithm is a traditional B&B algorithm, which decides for each decision a material to be manufactured with which machines to produce and which not. In our previous results, we showed that the choice of material can already greatly decrease the running time of the algorithm. In this result, we only made local decisions regarding the choice of materials and made simple FIFO and LIFO decisions regarding the materials.

In our present result, instead of this traditional B&B procedure, we created a list for decision mappings, in which the previous decisions are also stored. With this list, it becomes possible to choose the most encouraging branch for further development. But on top of that, it is also possible that if a branch is less successful, then we can continue with an earlier branch instead. Several heuristics were developed for this decision, such as one based on the results of the LP or simply based on the costs of the selected operating units. In addition, we are able to coordinate material and decision-mapping decisions.

6 Conclusion

A new heuristics were created for the ABB algorithm that give a better decisions in ABB algortihm. In this prezentation, we show that the overall running time will be reduced.

Acknowledgements

The research presented in this paper was funded by the National Laboratories 2020 Program - Artificial Intelligence Subprogram - Establishment of the ”National Artificial Intelligence Laboratory (MILAB) at Széchenyi István University (NKFIH-870-21/2020)” project.

References

[1] Naonori Nishida, George Stephanopoulos, and A. W. Westerberg. A review of process synthesis. *AIChE Journal*, 27(3):321–351, 1981.

[2] Jeffrey J. Siirola. Industrial applications of chemical process synthesis. 23:1–62, 1996.

[3] F. Friedler, J. B. Varga, E. Fehér, and L. T. Fan. *Combinatorially Accelerated Branch-and-Bound Method for Solving the MIP Model of Process Network Synthesis*, pages 609–626. Springer US, Boston, MA, 1996.

[4] F. Friedler, K. Tarjan, Y.W. Huang, and L.T. Fan. Graph-theoretic approach to process synthesis: axioms and theorems. *Chemical Engineering Science*, 47(8):1973–1988, 1992.

[5] F. Friedler, K. Tarjan, Y.W. Huang, and L.T. Fan. Graph-theoretic approach to process synthesis: Polynomial algorithm for maximal structure generation. *Computers and Chemical Engineering*, 17(9):929–942, 1993.

[6] F. Friedler, J.B. Varga, and L.T. Fan. Decision-mapping: A tool for consistent and complete decisions in process synthesis. *Chemical Engineering Science*, 50(11):1755–1768, 1995.

[7] F. Friedler, Á. Orosz, and J.P. Losada. *P-graphs for Process Systems Engineering: Mathematical Models and Algorithms*. Springer International Publishing, 2022.

[8] F. Friedler, K. Tarjan, Y.W. Huang, and L.T. Fan. Combinatorial algorithms for process synthesis. *Computers and Chemical Engineering*, 16:S313–S320, 1992.

Quantitative Radiomics Analysis of Lung CT Images Using Radial Harmonic Fourier Moments

A. H. M. Sajedul Hoque, Gergő Bognár, Sándor Fridli

Abstract: Radiomics is an emerging field of CT image processing, which utilizes quantitative image features that noninvasively quantify the tumour phenotypes. Radiomics analysis is promising for treatment planning, together with personalized medicine, as well as for predicting clinical factors. However, it usually involves the data mining of a large pool of features, where the optimal feature selection is not properly managed in the literature. In this paper, we investigate radiomic feature extraction using radial harmonic Fourier moments as higher-level decompositions. We perform radiomics analysis on lung CT images of non-small cell lung cancer patients from multiple annotated datasets. There, we evaluate the stability, reliability, and prognostic value of the proposed features following the literature guidelines, and compare the performance with state-of-the-art wavelet features.

Keywords: radiomics, lung CT, quantitative imaging, radial harmonics, orthogonal moments

1 Introduction

Medical imaging, especially X-ray computed tomography (CT), is a primary diagnostic tool of clinical oncology. CT, as an imaging modality, noninvasively quantifies the internal tissue density, that might help the localization and characterization of the tumour. CT imaging is routinely used in many areas of clinical oncology not only for diagnosis, but also for therapy planning and monitoring. In therapy planning, CT provides precise visualization of the geometric shape of the tumour and the normal tissue, which helps to determine the optimum radiation dose distribution in the tumour [1].

In this paper, we research quantitative imaging for lung CT motivated by personalized medicine. Personalized medicine is an emerging field that promises better patient care by taking the genetic differences of the tumour into account. In this personalized medicine, predictive and prognostic data factors coming from multimodal information including clinical, imaging, and molecular data are merged to forecast treatment outcomes [2]. However, the molecular characterization of cancer is challenging, and usually requires invasive approaches (biopsies and surgeries), which themselves may be limited if the tumour is heterogeneous. CT imaging is a promising supplementary tool to quantify tumour phenotypes [3]. As a quantitative imaging approach, radiomics [4] aims the robust extraction of image features through mathematical algorithms that describe the intensity, shape, and textural properties of tumour. It is already shown that radiomics correlate to tumour phenotypes [5], and can be utilized to predict distant metastasis [6].

Radiomics analysis usually involves a large number of features, including higher level decompositions of the CT image using scale-space transformations (e.g. wavelets and Laplace pyramids) [7]. In this paper, we investigate the radiomic features derived from Radial Harmonic Fourier Moments (RHFM) [8]. Image moments are widely used transformation-invariant feature descriptors [9], popular for pattern recognition, object representation, and feature extraction. In particular, orthogonal moments provide efficient and stable time-frequency decompositions, with the support for adaptivity. Here, we generate radiomic features from RHFM reconstructions, and we investigate their reliability, stability and prognostic value using annotated lung CT datasets of non-small cell lung cancer (NSCLC) patients.

2 Materials and Methods

Radiomics Features: Radiomics features are constructed employing advanced hard-coded algorithms, that provide a large set of quantitative imaging features. Radiomics features are usually grouped as follows:

- Shape and size related features illustrates the three-dimensional size and shape of tumour region, usually involving features like volume, surface area, surface to volume ratio, sphericity, spherical disproportion, maximum diameter, and compactness.
- First order statistical (FOS) features describes the gray distribution in the tumour area using descriptors like energy, entropy, kurtosis, maximum, minimum, mean, mean absolute deviation, median, range, root mean square, skewness, standard deviation, uniformity, and variance.
- Second order statistical features illustrate the statistical correlation between a voxel and its neighboring voxels addressing texture information. That is, second order features describe the heterogeneity of the tumour. In order to extract the texture features, matrices like Gray-Level Co-Occurrence Matrix (GLCM) and Gray-Level Run-Length Matrix (GLRLM) are formed from the CT image. GLCM provides the probability of combined occurrence of two intensity values. From that matrix, features like autocorrelation; cluster prominence, shade, and tendency; contrast; correlation; difference and joint entropy; informational measures of correlation; inverse difference and difference moment (normalized); inverse variance; maximum probability; sum average, entropy, and squares can be extracted. GLRLM represents the run-length of gray level in the CT image, that allows features such as run emphasizes (short and long run, low and high gray level, and combined), gray level and run length nonuniformity, and run percentage.
- High-order statistical features aim to characterize the repeated or nonrepetitive potential patterns inside the tumour region [4]. Two possible approaches are wavelets and Laplace pyramids, where first and second- order statistical features are extracted from the decomposed images.

Datasets and Data Analysis: In this study, three lung CT datasets are considered, involving non-small cell lung cancer (NSCLC) patients:

- The RIDER test/retest dataset [5] provides blind delineations to 31 patients of the RIDER Lung CT dataset [10]. RIDER Lung CT consists of same day repeat scans, where two CT scans were acquired from each patient within 15 minutes. In this study, this dataset was used to assess the reliability of the features.
- The multiple delineation dataset [5] consists of lung CT scans of 21 patients, manually delineated by five oncologists independently. This dataset was used to assess the feature stability.
- The Lung1 dataset [5] consists of lung CT scans of 422 patients, together with manual delineations, clinical and survival data. This dataset was used to assess the prognostic value of the radiomic features.

Radial Harmonic Fourier Moments: Following [8], consider the radial harmonic basis function H_{pq} $(p \in \mathbb{N},\ q \in \mathbb{Z})$, defined in polar coordinates as

$$H_{pq}(r,\varphi) := R_p(r)e^{iq\varphi} \qquad (r \in [0,1],\ \varphi \in [0,2\pi))\,,$$

where

$$R_p(r) := \begin{cases} 1/\sqrt{r}, & \text{if } p = 0, \\ \sqrt{2/r}\cos(\pi p r), & \text{if } p \text{ is even}, \\ \sqrt{2/r}\sin\left(\pi(p+1)r\right), & \text{if } p \text{ is odd}. \end{cases}$$

Radial harmonic basis functions form a complete orthonormal system in the space of the square integrable functions over the unit disk (i.e. in $L^2(\mathbb{D})$) with respect to the usual scalar product

$$\langle F, G\rangle := \frac{1}{2\pi}\int_0^{2\pi}\int_0^1 F(r,\varphi)G^*(r,\varphi)r\,dr\,d\varphi \qquad \left(F, G \in L^2(\mathbb{D})\right).$$

A grayscale image, represented as a function $f \in L^2(\mathbb{D})$ over the unit disk, can be expressed as

$$f(r,\varphi) = \sum_{p=0}^{+\infty}\sum_{q=-\infty}^{+\infty} M_{pq}H_{pq}(r,\varphi) \qquad (r \in [0,1],\ \varphi \in [0,2\pi)),$$

where $M_{pq} := \langle f, H_{pq}\rangle$ is the radial harmonic Fourier moment (RHFM) of order $p \in \mathbb{N}$ and repetition $q \in \mathbb{Z}$. Here, we will consider the partial reconstruction of order $n, m \in \mathbb{N}$ as

$$f(r,\varphi) \approx f_{nm}(r,\varphi) := \sum_{p=0}^{n}\sum_{q=-m}^{m} M_{pq}H_{pq}(r,\varphi) \qquad (r \in [0,1],\ \varphi \in [0,2\pi)).$$

Note that the real application of RHFMs involves discretization and the conversion of the input image to the unit disk, for which we investigated multiple approaches.

Proposed features: We propose the extraction of high-order statistical features based on RHFM. Our motivation is that similar to wavelets and image pyramids, orthogonal moments can also capture higher level patterns on the CT image. Additionally, orthogonal moments possibly provide more flexibility and even adaptivity in some cases. In this study, we applied RHFM slice-wise, and used the reconstructed images (f_{nm}) of order $n = m = 2, 4, 10, 18, 22, 32, 40, 48$. Then, 14 first-order statistical, 17 GLCM and 11 GLRLM-related features are extracted from those reconstructed images, as introduced above. Totally, we investigated 336 RHFM-based features, which we compared to the same amount of wavelet-based features as of [5], where the first level of 3D wavelet decomposition with Coiflet 1 was utilized. The low-order features were extracted using the pyradiomics package [7].

3 Results and Discussion

In this study, the extracted radiomics features of lung cancer are analyzed to evaluate their reliability, stability, and prognostic power, following the workflow proposed in [5]. In the case of reliability, the features from test/retest dataset are examined using intraclass correlation coefficient (ICC), which describes how strongly units in the same group resemble each other. Here, the ICC score indicates not only the degree of correlation but also the agreement between the features from test and retest dataset. In order to evaluate the reliability, we have classified the ICC reliability index into four groups according to the usual guidelines. The four groups are: Poor(ICC<=0.4), Fair(ICC>=0.4 and ICC<0.6), Good (ICC>=0.6 and ICC<0.75) and Excellent (ICC>=0.75 and ICC<=1). Table 1 shows the distribution of the number of RHFM and wavelet features among those groups. The table implies that the RHFM features are more promising compared to wavelets. Next, the stability of features is measured using Friedman

Table 1: Reliability: number of features of different agreement levels based on ICC

Type of Features	**Poor**	**Fair**	**Good**	**Excellent**
RHFM	19	31	50	236
Wavelet	96	68	43	129

test, a non-parametric statistical test which is used to detect the differences of units among multiple groups. Here, the test is applied on the five-feature set extracted from five multiple delineation dataset. It has revealed that there are 89 and 151 stable features for RHFM and wavelet decomposition, respectively, at a 5% significance level. The test shows that the stability of RHFM and wavelet features are comparable. Finally, the prognostic power of extracted features is determined using Cox hazard regression model which makes association between survival times of patients and one or more predictor variables. The regression model applied on Lung1 dataset gives concordance index (CI) indicating the weights of the variable in the prediction of survival time. In our experiment, 319 RHFM and 326 wavelet features are above 0.5 and thus show prognostic value. The mean and median CI is approximately 0.55 and 0.56 for both the RHFM and wavelets, which proves a similar prognostic value. In summary, RHFM provides multiple features that are stable, reliable, and also have prognostic power.

4 Conclusion and Future Work

We investigated the application of orthogonal moments for radiomics analysis of lung CT images of NSCLC patients, and compared the extracted RHFM with state-of-the-art wavelet features. Statistical tests were performed to determine the stability, reliability, and prognostic value of the proposed features, which aspects play important roles in clinical oncology. We conclude that orthogonal moments are promising for radiomics analysis, since they show similar or better behavior compared to wavelets, while they are more flexible and possibly adaptive.

In the future, we plan to further investigate the application of orthogonal moments in radiomics in several respects. This includes the adoption of various orthogonal bases, adaptive transformations, discretization, extension of the models to 3D, and the direct utilization of transformation invariant moments as radiomic features. Further, we plan radiogenomics analysis using orthogonal moments.

References

[1] G. C. Pereira et al. The Role of Imaging in Radiation Therapy Planning: Past, Present, and Future. *BioMed Research International*, 2014:e231090, Apr. 2014.

[2] P. Lambin et al. Predicting outcomes in radiation oncology–multifactorial decision support systems. *Nature Reviews. Clinical Oncology*, 10(1):27–40, Jan. 2013.

[3] R. Li et al. *Radiomics and Radiogenomics : Technical Basis and Clinical Applications*. CRC Press, 2019.

[4] J. Tian et al. *Radiomics and Its Clinical Application: Artificial Intelligence and Medical Big Data*. Academic Press, June 2021.

[5] H. Aerts et al. Decoding tumour phenotype by noninvasive imaging using a quantitative radiomics approach. *Nature Communications*, 5:4006, Aug 2014.

[6] T. P. Coroller et al. CT-based radiomic signature predicts distant metastasis in lung adenocarcinoma. *Radiotherapy and Oncology*, 114(3):345–350, 2015.

[7] J. J. M. van Griethuysen et al. Computational Radiomics System to Decode the Radiographic Phenotype. *Cancer Research*, 77(21):e104–e107, Nov. 2017.

[8] H. Ren et al. Multidistortion-invariant image recognition with radial harmonic Fourier moments. *Journal of the Optical Society of America. A, Optics, Image Science, and Vision*, 20(4):631–637, Apr. 2003.

[9] Y. Liu et al. Accurate quaternion radial harmonic fourier moments for color image reconstruction and object recognition. *Pattern Analysis and Applications*, 23:1–17, 11 2020.

[10] B. Zhao et al. Evaluating variability in tumor measurements from same-day repeat CT scans of patients with non-small cell lung cancer. *Radiology*, 252(1):263–272, 2009.

Design of Hyperledger Fabric Private Data Collections with Formal Concept Analysis

Damaris Jepkurui Kangogo, Imre Kocsis

Abstract: Hyperledger Fabric, a permissioned blockchain-based distributed ledger framework, enables secure collaboration among organizations in a network only shared by them. Its architecture facilitates data partitioning through a number of mechanisms to ensure that specific data segments are replicated only among the relevant subsets of organizations in the network consortium. These include Private Data Collections (PDCs), for scenarios where a specified subset of the organizations who jointly maintain a ledger wants to keep some portion of data only hash-committed to the common ledger. In this paper, we propose a novel use of Formal Concept Analysis (FCA) for the requirement-based design of PDCs.

Keywords: blockchain, Distributed Ledger Technology, Hyperledger Fabric, confidentiality, Formal Concept Analysis

1 Introduction

Distributed Ledger Technology (DLT), is a blockchain-based technological infrastructure originally motivated by Bitcoin digital currency and its derivative technologies. DLT has demonstrated a transformative potential for many traditional business practices, including finance, supply chains, healthcare, and education. The fundamental idea behind the popularity of blockchain-based systems is the notion of a distributed ledger kept up-to-date by a peer-to-peer network in which every node, or peer, possesses an identical copy of the ledger. There is no single node that owns the ledger, meaning there's no central authority to trust concerning the ledger contents. The integrity of the ledger is guaranteed through some form of Byzantine fault- and error-tolerant multiparty consensus mechanism. Transactions submitted to the network are validated, ordered, and grouped into blocks. A hash chain is built over these blocks, forming a chain, hence the name blockchain.

Hyperledger Fabric [1] is a leading DLT, with an architecture that facilitates the creation of bespoke, cross-organizational blockchains (i.e., blockchains where consensus as well as access are *permissioned*). In a Hyperledger Fabric network, *channels* are distributed ledgers managed by a subset of organizations. Whenever a group of organizations needs to transact and exchange data in isolation from the rest of the network, they have the option to create a new channel accessible only to these subsets of organizations. In contrast, for scenarios where transactions must be shared among the channel members while restricting access to some (or all) of the transaction data to only a subset of channel members, Fabric offers Private Data Collections (PDCs). Transactions over data in PDCs write only hash commitments to the channel ledger and maintain the underlying data by a dedicated gossip protocol for data sharing and using a – not blockchain-backed, potentially transient –"sideDB" on the network nodes that participate in the PDC.

2 Data compartmentalization design in Hyperledger Fabric

Introducing the architectural design space of Hyperledger Fabric networks and the approach to Byzantine consensus used in Fabric are beyond the scope of this paper. For our purposes here, a Fabric network is operated by an O set of organizations through their peers dedicated to this task, where for the channel set C of the network, the nodes maintaining each channel as a distributed ledger come from non-empty subsets of O. Smart contracts ("chaincode" in Hyperledger Fabric parlance) are deployed on the channels; smart contracts can read and write their own channel freely (subject to distributed consensus) and, under certain circumstances, can read the contents of other channels (a chaincode executed on an organizational node participating in multiple channels of that organization can "read between channels"). Global transaction ordering and identity services are shared across all channels in a network.

Fabric uses a generic key-value storage model as its ledger abstraction and chaincode is developed against this key-value storage model. Application-level data models have to be translated to this key-value storage model; earlier work has demonstrated this process for the rather involved relational data model for the classic TPC-C benchmark [2] and approaches for the model-driven engineering of data model mappings are under development [3]. Without losing much generality, we treat a Fabric network here as a distributed means to store a *relation*, with attribute set A. Then, assuming that each attribute is maintained only on a single channel, a partitioning of the attribute set to channels can serve as a robust mechanism to fulfill data confidentiality requirements across the organizations: organizations who shall not be privy to the values of certain attributes can be closed out from the circle maintaining that attribute.

In earlier work [4], we proposed the use of Formal Concept Analysis (FCA) to establish the channel structure necessary to adhere to confidentiality requirements, given an attribute set, a set of organizations, and a binary matrix of *positive* data maintenance requirements – that is, which organization is required to participate in the maintenance of which attribute.

However, using channels for data compartmentalization has the drawback that cross-channel atomic (writing) transactions are not straightforward under the Fabric channel model [5] and their performance engineering is largely unexplored. Also, our original positive access modeling approach is too restrictive in the sense it is ill-equipped to express "may see" relations and to handle decision making for newly joining organizations. Of the two, the first is the conceptually more important problem: in a decentralized setting, we still want to maximize the level of (Byzantine fault tolerant) replication of data, up to the level allowed by confidentiality requirements.

Thus, in this paper, we investigate the use of FCA for design for confidentiality in Fabric using PDCs, under the restrictive ("must not see") model.

3 Formal Concept Analysis

Formal Concept Analysis (FCA) is a technique that has been used primarily for knowledge representation, data analysis, and information management [6, 7]. It takes a *formal context*, a binary cross-table of objects and their attributes. From the formal context, *formal concepts* can be extracted: a pair of maximal sets of objects (the *extent* of the concept) that share a maximal set of attributes (the *intent* of the concept). Every object in extent shares every attribute in intent, every attribute in the intent is shared by all objects in the extent and neither the intent nor the extent can be further extended without reducing the other.

A natural subconcept-superconcept relation over the formal concepts of a formal context entails a *concept lattice*. In this lattice, a more specific concept (with fewer objects and more attributes) is "below" a more general concept (with more objects and fewer attributes). Formal concepts of a context are regularly visualized with line diagrams, which enable their expert exploration.

Formal concept analysis has found applications in various domains. Among them, FCA has been applied in the security domain, specifically for modeling access control. The majority of the studies have focused on modeling role-based access control (RBAC) [8, 9]. [10] used FCA to model Chinese wall access control (CWAC).

However, to the best of our knowledge, our use of FCA for modeling data isolation in distributed ledgers is novel.

Table 1: Example context of organizations and "attributes". X denotes "must-not-see".

	Warehouse	District	Customer	Order	O_Line	Stock	Item	N_Order	C_History
Manufacturer			X						X
Distributor			X						X
Retailer									
Courier				X	X	X	X	X	X
Regulatory Agencies			X					X	X

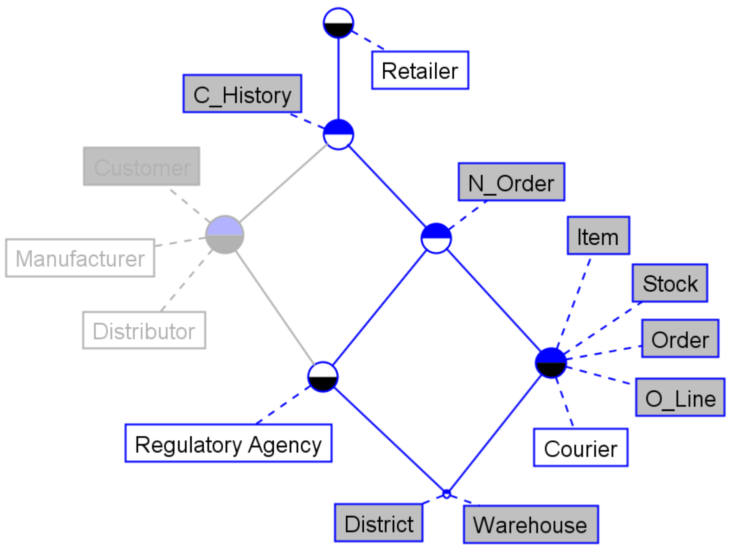


Figure 1: Line diagram of the concept lattice for the context in Table 1

4 Applying FCA for setting up PDCs

As a demonstration, we use a simple supply chain case study (a heavy simplification of [2]) of five stakeholders. As attributes, we use the TPC-C tables (Warehouse, District, Customer, Order, Order_Line, Stock, Item, New_Order, Customer_History). Table 1 defines our formal context; the line diagram visualization of the corresponding formal concept lattice is depicted on Figure 1. (We used the tool Concept Explorer[1] to generate the concept lattice from the formal context.)

To demonstrate the "processing" of the diagram to define PDCs, we highlighted the node `N_Order`, the formal concept node which "introduces" this attribute as a commonly shared one to the set of all objects "below" it – i.e., all object reachable through a downward walk – by omitting the `Retailer` object. (Visual collection of the attributes belonging to a formal concept node is performed the same way through all "upward" walks). The interpretation of the node is that `Regulatory Agency` and `Courier` should not see `N_Order`, which implicates the creation of a PDC over `N_Order` for all *other* organizations. In general, all nodes with at least one attached attribute in the line diagram translate to exactly one PDC this way.

Common FCA algorithms built into FCA tools have the potential to support PDC design further. In particular, *attribute exploration* enumerates attribute implications which follow from the context and queries the user about their general validity (with an option to specify new, counterexample objects). In usual FCA use, attribute exploration on the context and its transposition serve to reveal "missing" (or on contrary, "redundant") attributes and objects; in our application context, it promises to be a useful tool for catching modeling errors.

[1] https://conexp.sourceforge.net/

5 Conclusion

We proposed the use of Formal Concept Analysis to create Private Data Collections in Hyperledger Fabric channels to fulfill confidentiality requirements with a rich structure. As PDCs are handled in a straightforward way in Hyperledger Fabric chaincode, the challenges associated with cross-channel atomic transactions in a channel based Fabric confidentiality-supporting solution do not arise in this setting. At the same time, the performance implications of a truly heavy use of PDCs are poorly understood in Fabric for representative smart contract logic and workloads. Thus, we plan to empirically analyse this aspect, in addition to integrating the integrated use of PDC sets to our upcoming model-based Fabric chaincode data access layer solution.

References

[1] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. D. Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich, S. Muralidharan, C. Murthy, B. Nguyen, M. Sethi, G. Singh, K. Smith, A. Sorniotti, C. Stathakopoulou, M. Vukolic, S. W. Cocco, and J. Yellick. Hyperledger Fabric: A distributed operating system for permissioned blockchains. In R. Oliveira, P. Felber, and Y. C. Hu, editors, *Proceedings of the Thirteenth EuroSys Conference, EuroSys 2018, Porto, Portugal, April 23-26, 2018*, pages 30:1–30:15. ACM, 2018.

[2] A. Klenik and I. Kocsis. Porting a benchmark with a classic workload to blockchain: Tpc-c on hyperledger fabric. In *Proceedings of the 37th ACM/SIGAPP Symposium on Applied Computing*, pages 290–298, 2022.

[3] M. Debreczeni, A. Klenik, and I. Kocsis. Transaction conflict control in hyperledger fabric: A taxonomy, gaps, and design for conflict prevention. *IEEE Access*, 12:18987–19008, 2024.

[4] K. Damaris and K. Imre. Requirement-based, structural design for confidentiality in Hyperledger Fabric. In *PROCEEDINGS OF THE 31ST MINISYMPOSIUM*, pages 78–83. Dept. of Measurement and Information Systems, Budapest University of Technology and Economics, 2023.

[5] E. Androulaki, C. Cachin, A. De Caro, and E. Kokoris-Kogias. Channels: Horizontal scaling and confidentiality on permissioned blockchains. In J. Lopez, J. Zhou, and M. Soriano, editors, *Computer Security*, pages 111–131, Cham, 2018. Springer International Publishing.

[6] B. Ganter and R. Wille. *Formal concept analysis: mathematical foundations*. Springer Science & Business Media, 2012.

[7] F. Škopljanac Mačina and B. Blašković. Formal concept analysis – overview and applications. *Procedia Engineering*, 69:1258–1267, 2014.

[8] C. A. Kumar. Designing role-based access control using formal concept analysis. *Security and Communication Networks*, 6(3):373–383, 2013.

[9] C. A. Kumar, S. C. Mouliswaran, J.-h. Li, and C. Chandrasekar. Role based access control design using triadic concept analysis. *Journal of Central South University*, 23:3183–3191, 2016.

[10] S. C. Mouliswaran, C. A. Kumar, and C. Chandrasekar. Modeling chinese wall access control using formal concept analysis. In *2014 International Conference on Contemporary Computing and Informatics (IC3I)*, pages 811–816. IEEE, 2014.

Identifying security issues in Elixir web applications

Smiljana Knežev, István Bozó and Melinda Tóth

Abstract: The security of software products is extremely important in the era of internet-based applications. Building secure web applications is not straightforward. Several guidelines and tools were developed to support this process. The biggest challenge for those tools is to not overwhelm the developers with false-positive hits. This paper aims to investigate the use of static analysis for accurate vulnerability identification in the case of Elixir web applications.

Keywords: Static analysis, Elixir, security vulnerabilities, XSS

1 Introduction

Elixir [1] became a widely used programming language for developing web-based fault-tolerant, scalable applications. It inherited all the useful properties of the BEAM virtual machine.

Application security become one of the most important metrics of web-based applications. Developing secure applications for a non-expert programmer is challenging, therefore several standards, guidelines and tools support this process [2, 3]. Static analyser tools[1], [2] can help to identify security vulnerabilities in an early stage of development [4].

Developing tools to identify security issues is not straightforward. The most challenging part is to provide valuable useful results. Simple text-based searches can identify several issues quickly, but syntactic, tree-matching-based approaches usually provide more accurate results. However, both approaches result in lots of false-positive hits. Providing too many results for developers requires manual post-analysis of the result, it is time-consuming and error-prone. Semantic analysis-based approaches can help reduce false-positive hits. Control- and data-flow analysis-based approaches can help to produce more accurate results.

Security checkers already exist for Elixir. This paper aims to investigate the role of static analysis in the vulnerability identification process and provide a tool for accurate Elixir security analysis.

2 Background

Erlang [5] and the BEAM virtual machine were designed for building fault-tolerant distributed applications. Its "Let it crash" concept allows processes to fail instead of handling runtime errors and provides an extensive mechanism for handling process failures. Elixir, as a new language on the top of BEAM, also inherited this property. However, security issues can not be handled with process error handling. Therefore, there has been a huge interest in secure application development on the BEAM in recent years. The Security Working Group [6] of the Erlang Ecosystem Foundation (EEF) defined several guidelines for building secure and secure web applications on the BEAM: for Erlang and Elixir.

The RefactorErl tool [7] has an extensive static analyser framework for Erlang and we built a security analyser on top of that framework [8]. It provides data- and control-flow analysis as well. Elixir programs are compiled to the same intermediate source-code representation as Erlang programs (the so-called Abstract Code), therefore we build our Elixir analysis on the top of RefactorErl. We build RefactorErl's Semantic Program Graph from the abstract code and search for the vulnerabilities in that representation. Using this approach we can build semantic analysis-based security checkers.

[1] https://codechecker.readthedocs.io/en/latest/

[2] https://spotbugs.readthedocs.io/en/latest/introduction.html

3 Secure Elixir web applications

Web Application Security Best practices for BEAM languages EEF's Security Working Group [10] describes a special document for best practices for secure web applications running on BEAM. These recommendations are mostly related to the use of the Phoenix [11] framework for web applications.

Cross-site scripting (XSS) is a common security vulnerability where a web application incorporates user input into its generated output without performing validation allowing the user to inject malicious code.

Motivation example The importance of data-flow analysis when identifying and labelling security vulnerabilities like XSS attacks can be demonstrated in a small example. Below is an example of a controller from a sample Phoenix app.

```
defmodule HelloWeb.PageController do
  use HelloWeb, :controller
  def home(conn, _params) do
    render(conn, :home, layout: false)
  end
  def html_resp(conn, %{"i" => i}) do
    html(conn, "<html><head>#{i}</head></html>")
  end
  def html_resp2(conn) do
    var = "<html></html>"
    html(conn, var)
  end
  def html_resp3(conn) do
    html_resp4(conn, "<html></html>")
  end
  defp html_resp4(conn, obj) do
    html(conn, obj)
  end
end
```

We can observe the function `html_resp` taken from the EEF guideline on Common Web Application Vulnerabilities on Cross-Site Scripting [10] and its variations `html_resp2`, `html_resp3`, and `html_resp4`. Sobelow [12] rightfully flags `html_resp` function as XSS in 'html' with High Confidence but also the `html_resp4` function even though we can see it is private and its parameter comes from function `html_resp3` and could be viewed as safe. Also, Sobelow flags function `html_resp2` as XSS in 'html' - Medium Confidence even though the var parameter is defined in the same function as `"<html></html>"`.

Semgrep [9], without specifying custom rules does not flag any of these functions as vulnerable.

Using data-flow analysis we spot that these functions, besides `html_resp` are safe and reduce the false positives. The Algorithm 1 used is an adapted algorithm defined in [13] for detecting OS injections. Location of function calls posing an XSS threat, like `html/2`, are found then data-flow analysis is run on those function's parameters.

Proposed methodology RefactorErl also supports analysis for compiled, BEAM files. Therefore we take the Semantic Program Graph as a base of our analysis and look for specific function calls that may lead to an XSS attack. We analyse the arguments of the function using data-flow analysis. We calculate the set of possible values for the vulnerable arguments and analyse

Algorithm 1 Algorithm for detecting XSS vulnerabilities [13]

```
1: Function get_calls_for_xss
2: Matchers ← [{'Elixir.Phoenix.Controller', [html, 2]}, ...]
3: CandidateFuns ← get_calls_for(Fun,Matchers)
4: FunParamTuples ← new list
5: for C ∈ CandidateFuns do
6:     Param ← get_expr_params_for(C, Matchers, get_all_children(C))
7:     add_to_list({C, Param}, FunParamTuples)
8: end for
9: DataFlowValues ← get_origins(FunParamTuples)
10: FilteredDataFlowValues ← get_unsafe_funs(DataFlowValues)
11: UnknownFuns ← get_funs_no_body(FilteredDataFlowValues)
12: ExportedFuns ← get_exported_funs(FilteredDataFlowValues)
13: return merge(UnknownFuns, ExportedFuns)
```

them. Once we find an input that originated from an unknown function[3] or a non-validated user input[4], we mark the function call as vulnerable. Applying these steps allows us to report `html_resp` as vulnerable and to not report `html_resp2` and `html_resp4` as vulnerable.

Further steps We aim to develop new security checkers specifically for issues described in the EEF guidelines regarding Elixir and Phoenix. Furthermore, we plan to analyse open-source projects and investigate the types of security issues that arise in real-life projects.

Our method marks unknown functions as vulnerable, i.e. functions producing tainted inputs. To improve the accuracy of false positive elimination in vulnerability identification for Elixir, we need to develop an understanding around frequently used library functions, preventing them from being mistakenly marked as vulnerabilities in our analysis tools.

4 Related work

The empirical study looks into vulnerability commits of 25 open-source Elixir projects [14]. It was found that 2% of commits were vulnerability-related. The study also found that in 9 out of 25 projects, one security vulnerability was present. The study suggests further research into the problem is needed and suggests the use of static code analysis tools for more detailed results.

There are several static analysis tools available that support Elixir analysis. Sobelow [12] is the most popular security-focused static analysis tool for Elixir and its web framework Phoenix. The tool provides the confidence of each insecurity, dividing it into three colour-coded categories: green-low, yellow-medium, and high-red. It also rather over-reports than under-reports resulting often in false positives. EEF recommends including Sobelow in a CI/CD pipeline so it can be run every time code is merged into the main branch [10].

Semgrep [9] is an open-source static analysis tool that supports more than 30 programming languages, including Elixir. Semgrep also allows user to enforce their own rules for code standards in a code-like manner. The tool uses pattern-oriented matching technology.

5 Conclusions

Application security has to be considered in all software products. Tools supporting the

[3]functions that are not analysed by RefactorErl, so the SPG of the function is not available

[4]an argument that can be provided by the user through an API call

identification of possible vulnerabilities are highly desired by the developer communities. We focus our research on Elixir web applications and want to provide a tool for reliable security checking based on static analysis. In this paper, we presented our approach to the XSS issue. In the future, we will investigate the different issues listed by the EEF guideline.

References

[1] Saša Jurić. Elixir in Action. *Manning*, 2024.

[2] The OWASP Foundation. Application security verification standard. https://owasp.org/www-pdf-archive/OWASP_Application_Security_Verification_Standard_4.0-en.pdf. Accessed: 29-03-2024.

[3] Carnegie Mellon University, Software Engineering Institute. Cert coding standards. https://wiki.sei.cmu.edu/confluence/display/seccode/SEI+CERT+Coding+Standards. Accessed: 29-03-2024.

[4] G. Ann Campbell and Patroklos P. Papapetrou. SonarQube in Action. *Manning Publications*, 06 2013.

[5] Francesco Cesarini and Simon Thompson. Erlang programming. *O'Reilly*, 06 2009.

[6] Web application security best practices for beam languages. https://erlef.org/wg/security. Accessed: 29-03-2024.

[7] M. Tóth and I. Bozó. Static analysis of complex software systems implemented in Erlang. Central European Functional Programming Summer School – Fourth Summer School, CEFP 2011, *Revisited Selected Lectures, Lecture Notes in Computer Science (LNCS)*, Vol. 7241, pp. 451-514, Springer-Verlag, ISSN: 0302-9743, 2012.

[8] M. Tóth and I. Bozó. Supporting secure coding for Erlang. In *The 39th ACM/SIGAPP Symposium on Applied Computing (SAC '24)*, April 8–12, 2024, Avila, Spain. ACM, New York, NY, USA, 3 pages, https://doi.org/10.1145/3605098.3636185, 2024.

[9] Semgrep. https://github.com/semgrep/semgrep. Accessed: 29-03-2024.

[10] Security working group. https://erlef.github.io/security-wg/web_app_security_best_practices_beam/index. Accessed: 29-03-2024.

[11] Geoffrey Lessel. Pheonix in Action. *Manning*, 2019.

[12] Sobelow. https://github.com/nccgroup/sobelow#installation. Accessed: 29-03-2024.

[13] Baranyai, B., Bozó, I., and Tóth, M. Supporting secure coding with RefactorErl. In *13th Joint Conference on Mathematics and Informatics – MaCS 2020* (2020), pp. 24–25.

[14] Dibyendu Brinto Bose, Kaitlyn Cottrell, and Akond Rahman. Vision for a secure elixir ecosystem: An empirical study of vulnerabilities in Elixir programs. In *Proceedings of the 2022 ACM Southeast Conference*, ACM SE '22, pages 215–218, New York, NY, USA, 2022. Association for Computing Machinery.

Optimizing SAP S/4HANA On-Premise with Cloud-Ready Extensions: a Clean-Core system

Imre Munkácsi, Márta Alexy Angyalné, Tamás Gábor Orosz

Abstract: With the rise of SaaS-based SAP ERP systems in recent years, extension techniques for cloud-based systems have evolved to enable automatic upgrades without disrupting custom developments. These techniques and tools are also available for On-Premise system versions, in which traditionally classical extension tools were used. This paper proposes a novel methodology for applying cloud-ready extension tools and methodologies to On-Premise SAP implementations, aligning with the classical RICEFW categories. Through this approach, organizations can optimize their ERP systems for future scalability, minimize technical debt, and facilitate the adoption of innovation within their operations.

Keywords: SAP S/4HANA On-Premise, Cloud-Ready Extension, Technical-debt, Clean Core, Key-user extensibility

1 Introduction

Organizations have traditionally relied on classic ABAP extensions to meet business requirements, investing significant effort in building custom processes tailored to specific requirements. Consequently, the core ERP system has grown increasingly complex due to extensive custom developments and enhancements, thus increasing the "technical debt" of the system. While classic extensibility has proven powerful and flexible in addressing diverse business needs, it has also contributed to inefficiencies and suboptimal performance. The TCO (Total cost of ownership) of the system has increased, as custom developments requires maintenance, additional testing and encumber the upgradeability of the core system and as a result, slows down the innovation adaptation process, while increasing the technical debt of the ERP system [1]. SAP is shifting away from a monolithic ERP architecture towards a Software-as-a-Service (SaaS) with the S/4HANA Public Edition ERP, but customers are still - and most probably will be - able to license Private Cloud on On-Premise solutions. Present paper introduces a new methodology how the available cloud-ready extension tools and methodologies apply in an On-Premise SAP implementation in alignment with the classical RICEFW categories. With this methodology, new customer requirements can be designed in-sync with the clean-core principle in order to decrease the TCO of the ERP system. The paper explores the full range of cloud-ready solutions and tools present in the SAP S/4HANA 2023 On-Premise FPS0 system version. Given SAP's adoption of a two-year release cycle for major On-Premise and Private Cloud versions, subsequent releases are likely to introduce new tools and modifications related to this subject.

2 Clean core concept

The concept of "keeping the core clean" was introduced by former SAP CTO, Björn Goerke, during his keynote address at TechEd 2018. This concept is currently utilized in SAP's Public Cloud edition, facilitating the automatic deployment of mandatory system upgrades to the client's system twice a year [2]. The Clean Core concept does not mean a modification free system, as most clients require some level of extension. Rather, describes a system where modifications are done in alignment of the Clean Core guidelines and utilizes tools which are provided. From the "Core" elements we are focusing in this paper on the "Extensibility" and "Integration" part, as these are the components that are usually affected by classical extension methods. The fulfillment to the criteria can be established in a green-field SAP implementation

Table 1: Criteria for clean core [3]

Element of Core	Criteria for Clean Core
Software Stack	Software version close to the latest release Partner solutions clean core compliant
Extensibility	Upgrade-stable extensions following prescribed extensibility model Only actively used and well-documented extensions Adherence to general code quality standards and best practices No duplication of SAP standard functionality
Integration	Upgrade stable interfaces Proper monitoring and error resolution capabilities Only actively used and well-documented integration
Data	Complete Correct Used and relevant
Processes	No inconsistent or inefficient processes Leveraging SAP recommended Best Practices
Operations	Planned and executed regularly to maintain alignment with guidelines Opt-in on lifecycle events such as periodic upgrades

at the start of the project, but also can be applied in a brown-field implementation, where a previous SAP system is upgraded to the latest S/4HANA version. In the latter case, the already existing custom modification shall be examined and re-designed to be aligned with the Clean Core concept. Extensibility means added functionality to the software, which extends or changes the standard system behavior e.g. by adding new data fields on database level and exposing it on the UI to capture additional data during operational work. Integration is used to exposing SAP ERP system data to other, third-party applications, or fetching data to ERP from another system. An example for an interface is to download the daily exchange rates of currencies into the system. White-listed APIs are listed and maintained by SAP [4]. This means that SAP has certified and released these APIs and are safe to use according to the Clean Core principle.

3 Cloud-ready extensibility techniques

SAP categorizes these techniques into three groups [5]. All three extensibility options adhere to the Clean Core guidelines, ensuring that the extension remains decoupled from the core code. This decoupling is vital as it guarantees that extensions do not disrupt system upgrades, and contrary, upgrades do not impact extensions.

1. Key-user extensibility tools (Tier 1)

   (a) Custom field app: With custom field, key-users are able to extend standard business object (e.g. product master, sales order, accounting document etc.) with additional fields on database level. The tool supports multiple data types and additional search-help with custom developed Value Help entities. After creating the custom field, the system allows the users to select where the new field can be available: which transactional application, analytical report, form interface, or even legacy GUI based transaction screen appends are created automatically.

   (b) Custom logic app: In the custom logic application, the available business contexts defines the different extension points where custom logic runs, e.g. change the document before save. The tool uses simplified ABAP language syntax and provides sample code for key-users for simple logic.

   (c) Adapt UI app: The tool allowes standard application user interface to be changed and deployed to users. The functionality includes adding/deleting fields, and modifying sections on the UI.
   (d) Create and extend forms: With the help of the tool, key-users are able to change the layout of the forms and create different variants for different use cases.
   (e) Create custom business object: CRUD enabled objects can be created, with the option to deploy it as a standalone application and also as an API.
   (f) Create custom CDS view app: CDS views are the main building blocks of the S/4HANA virtual data modeling solution [6]. CDS views provide a way to define and consume semantically rich data models in a standardized and efficient manner. The custom CDS views are able to re-use whitelisted, standard CDS views. This provides a way to created APIs and datasources for reporting which otherwise are not available in the system.
   (g) Custom query app: This tool provides the capability to create end-user reports and deploy them as multidimensional application, which uses the embedded analytics engine of the system. The users can freely change what field they need both in rows and columns.
   (h) Situation handling - Extended Framework: The situation handling engine uses BOR or Class based workflow events to trigger custom notifications to the users. This includes upcoming deadlines, failed approvals of documents, expiring contracts, stock shortage etc.

2. Developer extensibility (Tier 2) [7]

   (a) Eclipse-based ABAP Development Tool (ADT) - Extend standard CDS views, Create ABAP RESTful Application Programming Model (RAP) based applications [8], Create new APIs
   (b) Business Application Studio on SAP BTP - Deploy Fiori Elements based application, Extend standard application via Adaptation Project
   (c) Visual Studio Code - Deploy Fiori Elements based application, Extend standard application via Adaptation Project

3. Side-by-Side extensibility (Tier 3) - Side-by-side extensions are software applications designed to operate externally from SAP S/4HANA while interfacing with it through standard SAP APIs. The target audience for these extensions are ABAP or non-ABAP (JAVA, Node.js) developers. The SAP Business Technology Platform (BTP) is the preferred platform for creating side-by-side extensions for any SAP solution [9]. The objective is to develop extensions that are loosely connected yet integrate smoothly, enabling them to operate independently from the SAP S/4HANA's operational processes and lifecycle management. Since these applications are separated from the core system and operate on a distinct technology stack compared to the standard ERP system, the core system's maintainability and upgradeability remain unaffected.

4 Align RICEFW with Cloud-Ready extensions

RICEFW is an acronym used in SAP to categorize different types of requirements or objects that are typically encountered during an implementation or extension project. Each letter in RICEFW represents a different type of object [10] Every tool designed for cloud-ready extensions can correspond to one or several RICEFW categories. The RICEFW framework, by itself, does not explicitly pinpoint requirements that overlap - such as a scenario where a new report triggers a workflow, which in turn initiates a printing process. Consequently, to address more complex situations, it is often necessary to employ a combination of multiple tools.

Table 2: Cloud-ready techniques for RICEFW categories

RICEFW category	Cloud-ready extension technique
Reports	Custom CDS view, Custom analytical query, or BTP analytical application
Interfaces	SAP standard white-listed APIs or custom interface via Custom CDS view and deployed as a RESTful API
Conversions	Custom data migration can be done via released APIs or standard Data migration cockpit functionality
Enhancements	Key user extensibility tools: Custom field, Adapt UI and Custom logic Custom object for new business objects Situation handling framework for notification-based extensions Developer extensibility tools: SAP S/4HANA Cloud ABAP environment for RAP applications Side-by-side extension on BTP for substitute applications
Forms	Side-by-side extensibility on BTP with Adobe form service Custom data source with developer extensibility
Workflows	Side-by-side extensibility on BTP with SAP Workflow management

5 Result and discussion

By applying this methodology, all user requirements can be categorized according to RICEFW in the design phase of the project. In the subsequent step, the technical teams should examine each user requirement to identify which extension type can fulfill it.

1. Categorization by RICEFW: This step ensures a structured approach to understanding the project's scope and identifying the specific needs that must be addressed.

2. Evaluation by Technical Teams: Once the requirements are categorized, the technical team examines each user requirement to determine which type of extension would be most appropriate for its realization. This determination is crucial as it directly impacts the extension's complexity and future flexibility.

3. Preference for Lower Tier Extensions: In situations where more than one extension type could satisfy a requirement, the goal is to opt for the extension of the lowest tier.

4. Collaboration for Simplification: This step involves engaging in discussions with business analysts and the initial requestor of the feature. The purpose of these discussions is to simplify the requirement without sacrificing essential business functionality.

5. Avoiding High-tier Extensions: The overarching goal is to implement solutions that fulfill business requirements without resorting to tier 3 or, ideally, even tier 2 extensions.

Although cloud-ready solutions are not mandatory for On-Premise systems, they can be used to get closer to the goals mentioned in the introduction. Accordingly, a big role is left to the development teams when they start the technical modelling of the specific user needs, as they have to be careful to choose techniques that they may have less knowledge about, but which can facilitate the long-term operability of the system. For enterprise decision-makers, the move towards cloud-ready solutions is not just about embracing innovation for the sake of modernization. It's a strategic decision aimed at minimizing technical debt - the accumulation of suboptimal technology choices that can hinder future system upgrades, scalability, and maintenance. Technical debt is a critical consideration in ERP system implementation, as it can significantly impact the total cost of ownership, system flexibility, and the ability to adopt new features over time.

References

[1] OutSystems. (2021, May). The growing threat of technical debt. Retrieved March 30, 2024, from https://www.outsystems.com/1/growing-threat-technical-debt/

[2] RISE with SAP - ERP Clean Core Strategy. (n.d.). SAP. Retrieved March 30, 2024, from https://www.sap.com/products/erp/rise/clean-core.html/

[3] Nancy. (2024, February 22). Certification of Partner Solutions following Clean Core. SAP Community. https://community.sap.com/t5/enterprise-resource-planning-blogs-by-sap/certification-of-partner-solutions-following-clean-core/ba-p/13556247

[4] SAP Business Accelerator Hub. (n.d.). https://api.sap.com/

[5] SAP Extensibility Explorer for SAP S/4HANA Cloud. (n.d.). https://extensibilityexplorer.cfapps.eu10.hana.ondemand.com/ExtensibilityExplorer/

[6] Robson, S. (2013). Agile SAP: introducing flexibility, transparency and speed to SAP implementations. IT Governance Ltd.

[7] Glavanovits, R., Koch, M., Krancz, D., Olzinger, M. (2023). Full Stack Development with SAP. SAP Press.

[8] Baumbusch, L., Jäger, M., Lensch, M. (2022). ABAP RESTful Application Programming Model: The Comprehensive Guide. SAP Press

[9] S. Banda, S. Chandra and C. Aun Gooi, SAP Business Technology Platform (An Introduction) (Rheinwerk Publishing, Quincy, MA, 2022), pp. 28-50.

[10] Hrastnik, R. D. C., Dentzer, R., Hrastnik, J. (2024). Core Data Services for ABAP. SAP Press.

State-of-the-Art Business Intelligence Applications: A Journey Through Time and Technology

Zoltán Ságodi, István Siket

Abstract: The advent of large language models (LLMs) has revolutionized software development, significantly reducing the effort required to create state-of-the-art applications. This study compares the effort involved in developing advanced software before and after integrating LLMs into the development process. By examining traditional methodologies alongside modern AI-enhanced approaches, the research highlights the challenges and efficiencies of each era. Key aspects such as coding time, debugging processes, innovation cycles, and resource allocation are analyzed to provide a comprehensive understanding of the evolution of development practices. The findings demonstrate a significant reduction in development time and an increase in software quality with LLMs. This study underscores the transformative impact of AI on software engineering and offers valuable insights for developers and organizations looking to leverage these advanced tools for future projects.

Keywords: LLM, GPT, Machine Learning, Human effort, Business Intelligence

1 Introduction

In today's technology landscape, Large Language Models (LLMs) are making significant impacts across various fields, including software development. One of the most well-known LLMs is the Generative Pre-Trained Transformer (GPT). LLMs are expertly designed to handle Natural Language (NL) and excel in data processing tasks [1]. Proficiency in understanding NL and processing Big Data opens up numerous applications, such as answering questions based on extensive knowledge bases which can be used in Business Intelligence (BI).

Business intelligence is a critical aspect of modern enterprises, prompting software developers to devise innovative solutions. Developing BI applications involves several complex steps, including Natural Language Processing (NLP), Knowledge Retrieval, and answer generation. Traditionally, creating such sophisticated software requires substantial time and the expertise of multiple developers.

LLMs are well-suited for performing NLP tasks and handling Big Data, making them ideal for developing BI applications. There is a growing body of research evaluating the efficacy of GPT models in Question Answering (QA) scenarios [2, 3]. As Wu et al. highlighted [3], LLMs leverage the knowledge acquired during their pre-training phase. However, to ensure these models remain effective with up-to-date information, various techniques must be employed, as continuous fine-tuning on new data is not feasible.

This work aims to analyze the differences between traditional software development and AI-supported development. The research questions guiding this study are:

- **RQ1:** What are the main similarities and differences between traditional and AI-supported development?
- **RQ2:** Are maintenance efforts different in traditional and AI-supported applications?
- **RQ3:** How reliable are they compared to one another?

By addressing these questions, we aim to provide a comprehensive understanding of how AI, and specifically LLMs, can enhance the software development process, particularly for applications involving complex data and natural language tasks.

2 Methodology

To compare the two development methods we need a traditional and an AI-supported development process. We only evaluate the parts that take actual development, as elements like communicating with the client and describing the final project expectations cannot be done twice. To evaluate traditional methods we use an already finished project developed by traditional means. This project leverages various techniques to provide an NL interface over business reports. In this case, we have the information on which experts were involved and how much time was spent on the project. To evaluate the AI-supported process we re-develop the key components without the additional elements such as front end and input validation since these elements can be reused. We analyze this solution in the same way as before.

3 Analysis of development

In this section, we describe the process of traditional and AI-supported development and the factors that make them easier or more problematic. The key steps we discuss are: Understanding NL questions (NLP), Analyzing the Data (AD), and Creating Responses and Charts (CRC).

3.1 Traditional

NLP: NL related processes, such as translation, can be achieved i.e. by syntax trees, although more complex tasks, such as text understanding require more resources. An additional difficulty is that most of the results in NLP non-AI are based on English texts, therefore, the results are worse for different languages. Developer teams could leverage various heuristics, predefined sentence patterns, and look-up tables combined with syntax tables. The NLP step regarding these difficulties requires a tremendous amount of effort from programmers and other specialists, e.g. linguists. Using predefined sentence patterns, the variable values can be provided by API endpoints where one endpoint represents one particular question.
AD: Usually getting the data requires various database connections and predefined queries. After retrieving the data, various algorithms or even whole data analysis frameworks might be applied. These steps mostly include applying third-party tools.
CRC: Creating responses mostly leverages template sentences and predetermined charts. The generator scripts are developed by the team and mostly use third-party libraries.

3.2 AI-supported

In the AI-supported development, we applied GPT-4 to source code generation [4] and the knowledge retrieval part completely.
NLP: As GPTs are designed to understand NL therefore this step can be skipped.
AD: This step is more complex. Firstly, we have to retrieve the data itself. In an AI-supported system, developers can create database communicators that connect to a database. With AI-pipelines, the required data can be mined, which even includes SQL queries, where hard-coded queries can be swapped for ones that the AI is capable of creating. Therefore, the queries are dynamic and can be applied to any database. Secondly, GPTs are designed to generate data based on the knowledge contained in the training set. We cannot use them directly to provide information about data it did not meet during training. To overcome this issue we can use prompting, especially few-shot learning. This way we can provide examples of how to react to data and what is our required output. Although this leads the model to a better outcome the data is still not fed to the model. We could use the whole data set and provide it to the model, however, this solution would require too much memory. This problem is solved by using Retrieval Augmented Generation (RAG) [5].

CRC: As GPTs are generative models, a GPT model can generate the complete answer, although it cannot provide charts. To overcome this issue we made GPT generate code that creates the charts. These scripts could both use third-party libraries to perform the whole task by themselves, based on the AI itself, although, it mostly prefers third-party libraries too.

Answer to RQ1: What are the main similarities and differences between traditional and AI-supported development?
Both traditional and AI-supported development uses third-party libraries and scripts to perform most of the tasks, however, traditional methods are more rigid and require more time to complete the scripts for every step. AI-supported systems are faster to develop as AI creates scripts on the fly. This also makes AI-supported systems more dynamic.

4 Analysis of software evolution

Maintaining software is a critical aspect that often incurs significant costs and places pressure on the evolving development team. The easier a system is to maintain, the more cost-effective and straightforward it becomes to implement changes to it over time.

Generally, the greater the volume of code to maintain, the more challenging it becomes. However, this difficulty is influenced by various factors. Here, we will focus on the abstract components of systems rather than the source code level, as the maintainability of the source code is highly dependent on its quality.

4.1 Traditional

In a traditional system that relies on predefined sentences, maintenance can involve implementing new algorithms for evaluation and continuously updating scripts, such as Python versions. Introducing new sentence patterns often requires the implementation of the corresponding algorithms and the responses to the new queries.

4.2 AI-supported

In an AI-supported system, where tasks are performed by AI and scripts are generated dynamically, maintenance primarily involves updating the underlying models or prompts. This allows for the automatic generation of new scripts. Due to the adaptive nature of such a model, adding new questions requires no extra effort, as the AI pipeline can seamlessly handle new queries. In our experimental system, the pipeline was designed to run iteratively until all results were obtained. However, we faced challenges in automatically verifying the correctness of the results. There were instances where the outcomes were incorrect or irrelevant to the desired objective.

Answer to RQ2: Are maintenance efforts different in traditional and AI-supported applications?
Traditional systems suffer from maintenance tasks, while AI-supported systems can be updated by changing configuration files to use new prompts and models. Although using new models is easy to implement, creating one is way harder, however, that is outside the scope of this paper.

5 Analysis of reliability

The reliability of a system's provided answers is one of its most crucial requirements. Reliability ensures that the system can generate responses that are both accurate and correct.

5.1 Traditional

The reliability of a traditional system hinges on the quality of the scripts created and the thoroughness of their testing. While we cannot provide exact figures, the results are generally consistent with those observed during evaluations, as the same algorithms are executed. This consistency, however, is heavily dependent on the quality of the evaluation dataset. The overall robustness of the system also relies on rigorous testing, ensuring there are no additional risks.

5.2 AI-supported

The reliability of an AI-supported system also depends on thorough testing, but there is a higher risk of producing incorrect results. Analyzing scripts are executed dynamically, or results are generated by a model. Both approaches are non-deterministic, making it difficult to predict how the system will react. When generating natural language responses based on the analyzed data, there is an additional uncertainty, as large language models (LLMs) can hallucinate and alter the original content, potentially leading to incorrect results.

Answer to RQ3: How reliable are they compared to one another?
The reliability of the traditional system is higher due to the non-deterministic nature of LLMs.

6 Threats to validity

We chose GPT-4 as our large language model (LLM), which raises several data security concerns. Although other open-source LLMs could have been used, potentially impacting the validity of our results, we opted for GPT-4 due to its state-of-the-art capabilities. Additionally, using custom local models would have required significant energy and costly hardware, making GPT-4 a more practical choice for our needs.

Acknowledgements

The research was supported by the project 2019-1.1.1-PIACI-KFI-2019-00449 implemented with the support provided by the National Research, Development and Innovation Fund, financed under the “Support for market-driven R&D and innovation projects” funding scheme.

References

[1] O. M. Alyasiri, D. Akhtom, and M. N. Alrasheedy. An overview of gpt-4’s characteristics through the lens of 10v’s of big data. In *2023 3rd International Conference on Intelligent Cybernetics Technology & Applications (ICICyTA)*, pages 201–206, 2023.

[2] Y. Tan, D. Min, Y. Li, W. Li, N. Hu, Y. Chen, and G. Qi. Can chatgpt replace traditional kbqa models? an in-depth analysis of the question answering performance of the gpt llm family. In *International Semantic Web Conference*, pages 348–367. Springer, 2023.

[3] Y. Wu, A. Henriksson, M. Duneld, and J. Nouri. Towards improving the reliability and transparency of chatgpt for educational question answering. In *European Conference on Technology Enhanced Learning*, pages 475–488. Springer, 2023.

[4] B. Idrisov and T. Schlippe. Program code generation with generative ais. *Algorithms*, 17(2):62, 2024.

[5] J. Li, Y. Yuan, and Z. Zhang. Enhancing llm factual accuracy with rag to counter hallucinations: A case study on domain-specific queries in private knowledge-bases. *arXiv preprint arXiv:2403.10446*, 2024.

Multi Model Recursion for Hungarian Electricity Load Forecasting

Mátyás Sebők

Abstract: Time series analysis and prediction is a difficult and complex problem. Many machine- and deep-learning methods exist with better and better results. This paper I proposes a strategy called Multi Model Recursion. It uses separate deep-learning models per feature that needs predicting. Another improvement is not predicting features which are easily calculated. Having extra models per feature helps in "simulating" a future environment since it predicts exogen variables otherwise unkown.

The Multi Model Recursion developed is an improvement of the commonly used Recursive strategy. The paper compares this method with models and strategies frequently used in the field. The testing dataset is put together from publicly available Hungarian electricity load and weather data. The task was to predict the country's net electricity load for the next 3 hours.

Keywords: time-series, deep-learning, Multi Model Recursion, electricity load forecasting

1 Introduction

Short term electricity load forecasting is useful since the forecasting models can adapt better to the given situation and give more accurate predictions. The better predictions then give the chance for participants to better exploit their resources and minimize their costs. A 3 hour forecast comparison of Hungary's net electricity load shows the different models' strengths at single step forecasting and also gives an idea about their longer range performance.

The difficulty is that while weather data is available at a large resolution, forecasts are not always available the same way. The focus of Multi Model Recursion is to create a simulated environment with the given exogenous variables and their respective models to further enhance the predictions of the target variable. Compared to the regular Recursive strategy, this architecture can optimize better since it doesn't have to directly take into account the exogenous and time-series variables when calculating the cost function.

This paper compares Multi Model Recursion with the regular Recursive strategy using recurrent deep-learning algorithms. It also compares it with the Multi Input Multi Output (*MIMO*) strategy using Convolutional, Temporal Convolutional and LSTM networks. Further compares it with the very powerful Sequence to Sequence (*Seq2Seq*) strategy, which uses an encoder-decoder architecture. To justify the usage of such complex algorithms it also looks at the performance of a machine-learning algorithm known as Random Forests and a Statistical method known as Seasonal Autoregressive Integrated Moving Average (*SARIMA*).

2 Related Work

In time-series forecasting we can discuss different statistical, machine- and deep-learning models and different strategies. All approaches have their own respective advantages, disadvantages and use cases. For electricity load forecasting it was important to choose a strategies which can forecast for multiple time-steps. Examples for these are Recursive, Multi Input Multi Output (*MIMO*) and Sequence to sequence (*Seq2Seq*) methods.

[1] describes the problem of time-series forecasting to predict electricity load and shows the different strategies used in the field. Most machine- and deep-learning models can be adapted to most of the strategies. [2] describes the usage of Convolutional and Temporal Convolutional Networks for time-series forecasting, Convolutional models were used in 1D on the target variable. [3] shows the use of LSTMs for multi-step time-series forecasting confirming it has superior performance to the ARIMA model. [4] shows the use of GRUs for sequence predictions,

GRUs behave similarly to LSTMs but generally it's not possible to tell which will perform better on a given task. [5] shows the usage and tuning of Random Forests, a machine-learning model that can easily be adapted to the MIMO strategy.

3 Multi Model Recursion

The Recursive strategy suffers from some issues that heavily decrease its performance in tasks which have many exogenous variables. Even more so if they are difficult to forecast. It works in a way where it forecasts a single time-step at a time with all its variables. Afterwards, it assumes that its predictions as correct and forecasts the next step with this assumption. The problem is that each variable has to partake in calculation of errors and then in Backpropagation which heavily decreases the performance for the target variable.

Algorithm 1 Multi Model Recursion algorithm, using the same notation as in [1]

```
1:  for $i = 0 \dots n_O - 1$ do                          ▷ $n_O$ is the length of the predictions
2:      for $j = 0 \dots k - 1$ do                       ▷ $k$ is the amount of predicted features
3:          $\hat{y_t}[i, j] := f_j(x_t)$      ▷ predict feature using $f_j$ model, store in $\hat{y_t}$ for timestep
4:          $x_t[n_T, j] := \hat{y_t}[i, j]$              ▷ add new, predicted features to next timestep
5:          if $random(0 \dots 1) < teacher\_forcing$ then
6:              $x_t[n_T, j] := y_t[i, j]$                                      ▷ teacher forcing
7:          end if
8:      end for
9:      $x_t[n_T, k \dots] := g(x_t)$                  ▷ add pre calculated features to next timestep
10:     $x_t := x_t[1..n_T)$     ▷ remove first timestep of input to preserve model input dimensions
11: end for
12: return $\hat{y_t}[\dots, 0]$                                          ▷ return the target variable
```

The designed Multi Model Recursion (*MMRec*) strategy is a deep-learning approach that aims to eliminate the common issues with the regular Recursive strategy. Firstly, it doesn't make forecasts using the deep-learning model when the values are simple to calculate. A good example of this are time/date related variables such as hour, day of the week or holiday identifiers. These values are calculated before the predictions occur and are given to the model at the appropriate time-step. The second difference is that it uses separate models for each remaining variable. In the dataset electricity load (the target variable), precipitation and global radiation were chosen. At the implementation level, these models are combined in a way that backpropagation happens only from the target variable backwards. This means that the optimization only happens based on the variable that is important and is actually used from the output.

4 Dataset

The results were produced with the dataset constructed from the data of OMSZ's Data Publication for weather data and from MAVIR's Data Publication for electricity load data. This concludes in an hourly time-series dataset from the start of 2015 until the end of August 2023.

After examining the importance of variables during Exploratory Data Analysis and employing feature selection methods such as evaluating the Confusion Matrix and using Sequential Automatic Feature Selection with the MIMO strategy and Random Forests the the amount of features used were heavily reduced. The remaining ones were electricity load, precipitation, global radiation and time describing variables.

5 Results

The models and training hyperparameters were optimized using the Grid Search algorithm while splitting the data to 7 pieces. For evaluation the data was split into 10 pieces. Table 1 shows the results of the respective metrics averaged over the 9 train-evaluation pairs and 6 runs. This totals to 54 runs per model-strategy. The mean $\pm$ standard deviation is listed for the results where applicable. MAVIR's predictions are given as is and not computed in this paper.

Multi Model Recursion (*MMRec*) was tested with GRUs predicting electricity load and Convolutional networks predicting exogenous variables. These are the GRU 1L and GRU 2L entries in Table 1. The difference between them is the layer-count used by the GRU models. MMRec FULL GRU is a model using separate GRU models for each of the 3 variables, this makes it slower but perform better since Recurrent networks are relatively good at dealing with time-series data.

Table 1: The evaluational scores for each model and strategy combination, closer to 0 is better

Model and Strategy	**MAE** (*MW*)	**RMSE** (*MW*)	**MAPE** (%)	**MPE** (%)
MAVIR predictions	252.58	300.81	4.97	-4.70
MIMO - RF	69.96 ± 15.29	104.32 ± 24.0	1.42 ± 0.32	0.017 ± 0.25
MIMO - CNN	103.13 ± 21.5	146.69 ± 27.55	2.12 ± 0.46	0.196 ± 0.416
MIMO - TCN	63.92 ± 10.46	92.96 ± 15.57	1.31 ± 0.22	-0.001 ± 0.198
MIMO - LSTM	62.62 ± 6.05	88.97 ± 9.19	1.28 ± 0.13	0.05 ± 0.175
Seq2Seq - GRU	58.75 ± 6.22	84.21 ± 9.83	1.21 ± 0.13	0.08 ± 0.168
Recursive - GRU	94.41 ± 14.41	128.39 ± 17.39	1.94 ± 0.28	0.119 ± 0.744
MMRec - GRU 1L	65.4 ± 7.68	92.43 ± 10.88	1.34 ± 0.17	0.114 ± 0.292
MMRec - GRU 2L	64.79 ± 7.31	90.85 ± 10.27	1.32 ± 0.16	-0.029 ± 0.295
MMRec - FULL GRU	62.17 ± 7.95	88.42 ± 11.25	1.27 ± 0.17	0.098 ± 0.261

SARIMA's performance is not included since after testing it was found that computing the proper parameters for only taking 3 hour predictions takes much longer than other models. If the proper parameters are not found it's performance is below the public predictions of MAVIR.

In terms of performance the Seq2Seq strategy performed best on in this dataset followed closely by the MIMO LSTM and MMRec FULL GRU strategy-model pairs. It's important to keep in mind that the electricity load's mean is around 4900 MW and the standard deviation is around 735 MW for the entire dataset.

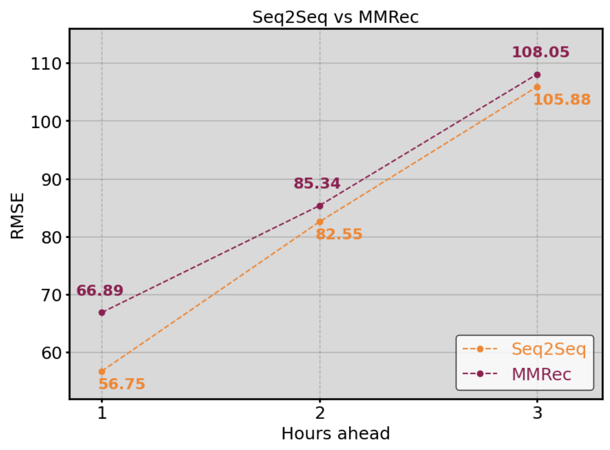


Figure 1: Comparing Seq2Seq and MMRec by RMSE error per hour ahead prediction

6 Conclusion

When taking a look at Figure 1 it's clear that the MMRec strategy's biggest weakness is first hour predictions. This could be due to the implementation since all models tested with the strategy show this behaviour. The advantage of this approach is that when looking at predictions after the first hour, the advantage of the Seq2Seq is much smaller. This shows that MMRec's way of "simulating" the exogenous variables state may be beneficial to predictions.

The improvements upon the regular Recursive method are clear from Table 1. Furthermore it's very simple to use this model in a way where external forecasts are incorporated. In that case the corresponding model can just be ignored for the given step. This is very useful in cases where the forecasts are not consistently available since the exogenous variable predictors can step in when something is not available.

Future usecases for MMRec include wind and solar electricity production predictions since they correlate heavily with weather variables and events. Future tests should include comparing the incorporation of external weather forecasts into both the Seq2Seq and MMRec strategy to compare their performance.

7 Acknowledgments

This work was supported by Eötvös Loránd University via fellowhip from the HALL 23 application. Special thanks to dr. Vincellér Zoltán from the ELTE Faculty of Informatics for supervising the project.

References

[1] Gasparin, A., Lukovic, S., Alippi, C.: Deep learning for time series forecasting: the electric load case. CAAI Trans. Intell. Technol. 7(1), 1–25 (2022). https://doi.org/10.1049/cit2.12060

[2] Shaojie Bai, J. Zico Kolter és Vladlen Koltun. An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling. 2018. arXiv: https://arxiv.org/abs/1803.01271

[3] Masum, S., Liu, Y., Chiverton, J. (2018). Multi-step Time Series Forecasting of Electric Load Using Machine Learning Models. In: Rutkowski, L., Scherer, R., Korytkowski, M., Pedrycz, W., Tadeusiewicz, R., Zurada, J. (eds) Artificial Intelligence and Soft Computing. ICAISC 2018. Lecture Notes in Computer Science(), vol 10841. Springer, Cham. https://doi.org/10.1007/978-3-319-91253-0_15

[4] Guizhu Shen, Qingping Tan, Haoyu Zhang, Ping Zeng, Jianjun Xu, Deep Learning with Gated Recurrent Unit Networks for Financial Sequence Predictions, Procedia Computer Science, Volume 131, 2018, Pages 895-903, ISSN 1877-0509, https://doi.org/10.1016/j.procs.2018.04.298.

[5] Probst P, Wright MN, Boulesteix A-L. Hyperparameters and tuning strategies for random forest. WIREs Data Mining Knowl Discov. 2019; 9:e1301. https://doi.org/10.1002/widm.1301

Convergence of Fog Computing, Blockchain, and Federated Learning for Advancing New Generation Networks

Wilson Valdez Solis

Abstract: The rapid evolution of New Generation Networks (NGN) such as 5G, 6G, and the Internet of Things (IoT), demands innovative approaches to address emerging challenges in scalability, security, and data management. In this sense, the convergence of novel technologies such as Fog Computing (FC), Blockchain (BC), and Federated Learning (FL) serves as means to enhance the capabilities of modern networks, in order to achieve enhanced performance, reliability, and privacy. This paper presents a high-level taxonomy to categorize the synergy of these paradigms in terms of networks architecture, security, privacy, resource management, data analytics, and applications. Through this taxonomy, we aim to provide insights into the potential benefits and challenges of integrating these technologies, and to guide future research and development efforts in building robust and scalable NGN infrastructures.

Keywords: Fog Computing, Edge Computing, Blockchain, Federated Learning, Taxonomy.

1 Introduction

Nowadays, the rapid growth of New Generation Networks (NGN) has catalyzed a paradigm shift, necessitating a profound reevaluation of conventional approaches to address several escalating challenges such as scalability, security, interoperability, and data management. As networks burgeon in complexity and scale, the imperative for innovative solutions becomes ever more pronounced [1, 2].

In response to this imperative, the emergence, development, and convergence of Fog Computing (FC), Blockchain (BC), and Federated Learning (FL) technologies appear to be an important inflection point, heralding a transformative epoch in network architecture and functionality. The synergistic potential of integrating these technologies holds immense promise for NGN to achieve unparalleled advancements in performance, reliability, and data privacy. FC offers decentralized processing prowess, distributing computing power across multiple devices for collaborative data processing while ensuring privacy and security [3]. BC provides immutable transparency, utilizing distributed ledger technology to securely record transactions across a network, ensuring data integrity with cryptographic techniques [4]. FL introduces collaborative learning dynamics, training models across decentralized devices without exchanging data samples, allowing for model improvement while preserving individual data privacy [5].

This paper introduces a high-level overview of the complexities of this convergence, elucidating its multifaceted applications across key domains encompassing network architecture, security protocols, resource allocation, data analytics, and real-world applications. Through a crafted high-level taxonomy, our work seeks to provide the academic community with a structured framework conducive to a nuanced comprehension of the synergies and intricacies inherent of these technologies. By delineating the potential advantages and impediments intrinsic to this convergence, we endeavor to furnish a compass guiding future research and practical pursuits aimed at fortifying the NGN infrastructures, thereby ensuring their resilience and adaptability in navigating the evolving terrain of the digital epoch.

The remainder of this paper is organized as follows. Section presents a short overview of state-of-the-art studies about integrating these technologies. Section provides a high-level taxonomy about integrating FC-BC-FL with their explanation. Finally Section presents conclusions and future research directions.

2 Related work

This section provides an overview of related taxonomies focusing on the integration of these technologies. While there are not explicit taxonomies addressing the integration of all three technologies simultaneously, several studies examine their combination in pairs.

In [6] is introduced a taxonomy about Edge Computing (considering Edge as a similar paradigm to Fog) in FL. Their taxonomy addresses fundamentals, architectures, frameworks, applications, and challenges across various areas, including efficiency and privacy. It also outlines several open issues and future research directions. Notably, the key distinction from our work lies in the absence of BC and other aspects such as security issues, data analytics, among others.

The study in [7] integrates BC, Mobile Edge Computing (MEC), and FL, coining the term FLChain to represent their combination and offering a detailed taxonomy of FLChain design and use cases. It extensively examines aspects like design, communication costs, resource allocation, security, and privacy. Key contributions include the FLChain architecture for EC networks, addressing challenges like communication costs and security, and exploring opportunities in EC applications such as data sharing and content caching. However, while the survey covers MEC, it overlooks FC capabilities and applications in the field.

[8] introduces a BC-FL integration for IoT applications. It provides an overview of architectural features and privacy techniques, establishes taxonomies for both FL and BC in the IoT context, and discusses various types of BC and consensus protocols. Furthermore, it presents a BC-enabled IoT architecture and demonstrates local machine-learning models and FL algorithms. However, the paper overlooks FC capabilities.

Overall, the proposed studies do not directly combine these three technologies into a taxonomy, underscoring this paper's importance.

3 Our proposed FC-BC-FL High Level Taxonomy

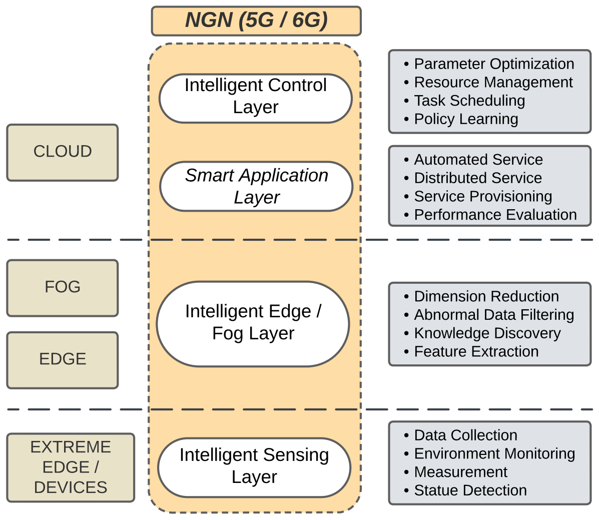


Figure 1: NGN's functional outlook

This section introduces the proposed high-level taxonomy for the integration of FC-BC-FL. The taxonomy is structured based on the functionalities of the NGNs, as depicted in Figure 1. The overview of this perspective emphasizes three key layers:

- *Extreme Edge or Devices Layer:* This layer serves as the intelligent sensing component responsible for data collection, environment monitoring, and measurement of required parameters.

- *Edge and Fog Layer:* This middle layer (Fog/Edge) acts as a sub-control point for the sensing layer variables. Key functionalities include Data Dimension Reduction, Abnormal Data Filtering, Knowledge Discovery, and Feature Extraction.

- *Cloud Layer:* This level represents both the Intelligent Controlling Layer and the Smart Application. The former focuses on Parameter Optimization, Resource Management, Task Scheduling, and Policy Learning. The latter focuses on Automated and Distributed services, Service Provisioning, and Performance Evaluation.

Afterward, we structured the high-level taxonomy as presented in Figure 2. The categories have been established from the Systematic Literature Review (SLR) presented in [9]. The proposed taxonomy includes the following branches:

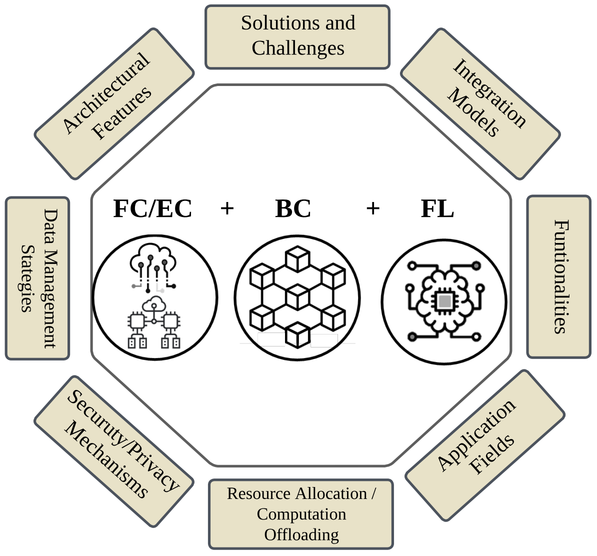


Figure 2: FC-BC-FL High-Level Taxonomy

- *Architectural Features.* This branch describes the potential architectural models empowered by the three technologies that can fit in the NGNs.

- *Data Management Storage.* This branch elucidates how data is managed and stored across various tiers of the NGN architecture. It provides detailed insights into strategies pertaining to Data Collection and Acquisition, Data Preprocessing and Filtering, Data Storage and Management, Data Sharing and Collaboration, as well as Data Privacy and Security.

- *Security/Privacy Mechanisms.* This branch should describe mechanisms regarding security and privacy concerning to Authentication and Authorization, Data Encryption and Decryption, Access Control Policies, Anonymity and Identity Management, and Privacy-Preserving Techniques.

- *Resource Allocation and Computation Offloading mechanisms.* This branch should describe the algorithms and orchestration process for managing resources and making adjustments to enhance the functionality of the NGNs.

- *Integration Models.* This branch should delineate the various types of integration feasible through the different mechanisms provided by FC/EC, BC, and FL technologies, and their combination.
- *Functionalities.* This branch must describe the functionalities that this integration should offer. In this case, we can refer to the outlook presented on Figure 1.
- *Application Fields.* This branch should introduce novel application fields where the advantages of this integration can be exploited. It can encompass both existing and new applications. Besides, it also should describe the types of services offered.
- *Solutions and Challenges.* This branch should outline the solutions attainable through the FC/EC-BC-FL integration, along with the forthcoming challenges, whether using this integration or not.

4 Conclusions and Future Work

In this paper, we have introduced a high-level taxonomy for integrating FC/EC, BC, and FL to meet the evolving requirements and perspectives of the NGNs. This initial taxonomy serves as a foundational framework for further exploration, both in terms of a dedicated taxonomy for the integration of these three technologies and for a more comprehensive taxonomy that builds upon these initial perspectives. Future efforts will focus on delving into the detailed characteristics and elements specific to FC/EC, BC, and FL, to address the evolving demands of the next generation of networks that will shape the future.

Acknowledgements

The research leading to these results has received funding from the national project TKP2021-NVA-09 implemented with the support provided by the Ministry of Culture and Innovation of Hungary from the National Research, Development and Innovation Fund, financed under the TKP2021-NVA funding scheme.

References

[1] W. Chen, X. Lin, J. Lee, A. Toskala, S. Sun, C. F. Chiasserini, and L. Liu. 5g-advanced toward 6g: Past, present, and future. *IEEE Journal on Selected Areas in Communications*, 41(6):1592–1619, 2023.

[2] J. M. Parra-Ullauri, X. Zhang, A. Bravalheri, S. Moazzeni, Y. Wu, R. Nejabati, and D. Simeonidou. Federated analytics for 6g networks: Applications, challenges, and opportunities. *IEEE Network*, 2024.

[3] S. Yi, Z. Hao, Z. Qin, and Q. Li. Fog computing: Platform and applications. In *2015 Third IEEE Workshop on Hot Topics in Web Systems and Technologies (HotWeb)*, pages 73–78. IEEE, 2015.

[4] H.-N. Dai, Z. Zheng, and Y. Zhang. Blockchain for internet of things: A survey. *arXiv preprint arXiv:1906.00245*, 2019.

[5] C. Zhang, Y. Xie, H. Bai, B. Yu, W. Li, and Y. Gao. A survey on federated learning. *Knowledge-Based Systems*, 216:106775, 2021.

[6] H. G. Abreha, M. Hayajneh, and M. A. Serhani. Federated learning in edge computing: a systematic survey. *Sensors*, 22(2):450, 2022.

[7] D. C. Nguyen, M. Ding, Q.-V. Pham, P. N. Pathirana, L. B. Le, A. Seneviratne, J. Li, D. Niyato, and H. V. Poor. Federated learning meets blockchain in edge computing: Opportunities and challenges. *IEEE Internet of Things Journal*, 8(16):12806–12825, 2021.

[8] M. Ali, H. Karimipour, and M. Tariq. Integration of blockchain and federated learning for internet of things: Recent advances and future challenges. *Computers & Security*, 108:102355, 2021.

[9] W. Valdez, J. Parra, and A. Kertesz. Exploring the synergy of fog computing, blockchain, and federated learning for iot applications: A systematic literature review. *Submitted to IEEE Access*, 2024.

Multithreading Atomicity Static Analysis Checkers in Java

Patrik Péter Süli, Judit Knoll and Dr. Zoltán Porkoláb

Abstract: This paper explores the enhancement of Java application thread safety through static analysis. It highlights the significance of Java's concurrency mechanisms and introduces two newly created checkers in SpotBugs by the authors. These checkers are designed to identify unsafe usages of shared resources and improper atomic operations in concurrent Java programming, aiming to mitigate common multithreading issues such as race conditions. By emphasizing consistent locking strategies and the correct use of Java's atomic types, the study offers insights into how to improve the reliability of multithreaded Java applications.

Keywords: java, concurrency, atomic operations, static analysis

1 Introduction

Static analysis in software development can detect several types of issues, such as runtime errors and security violations in the code, without running the program itself, so developers could be informed about bugs in early stages of development. There are many ways to analyze source codes, for example, Control Flow Analysis examines the execution, revealing infinite loops, unreachable codes, and improper usages of control structures (e.g., `if`, `else`, `for`, `while`) [1]; Data Flow Analysis focuses on data tracking to identify issues like uninitialized variables, null pointer dereferences, and potential memory leaks [2]; Pattern-Based Analysis uses predefined rules to look for common issues or antipatterns in the code [3].

Guidelines have been created to assist developers in producing code that is more secure and reliable. The Software Engineering Institute (SEI) has its own, called CERT Coding Standards [4]. It has many rules covering various aspects of coding practices, including memory handling, proper use of concurrency, input validation, and more, with the aim of preventing software vulnerabilities such as buffer overflows, race conditions, and injection attacks. These rules are applicable to different programming languages, and the SEI provides specific sets of rules for C, C++, Java, and other languages.

For Java, the SEI Cert Coding Standard contains several rules for atomicity, which is essential in softwares that work with parallel threads. This paper focuses on two specific rules of these, which are concerned with thread-safe usage of shared data.

In Java, it is crucial to know, that the source code is not directly compiled into machine code. Instead, it is first transformed into platform-independent bytecode, that can be interpreted by the Java Virtual Machine (JVM), and this allows running the Java code on any device with a Java Runtime Environment (JRE) installed [5]. In addition, Java provides high-level concurrency constructs, such as synchronized blocks and methods, locking objects, and concurrent collections.

There are several tools that can analyze code and make suggestions to improve it, one of them is SpotBugs [6] (a fork of the abandoned FindBugs). It is an open source tool that looks for bugs in Java code using Apache BCEL (Byte Code Engineering Library) [7], so it can handle binary *.class* files and understand instructions and methods at the bytecode level. When analyzing classes, SpotBugs uses BCEL to read and understand the structure of the bytecode, looking for specific patterns, coding practices, or known issues.

2 Strategies to use resources thread-safe

When multiple threads run simultaneously and use common resources, it is important to consider using some kind of locking, for example, synchronization on the threads or in the called methods.

Java provides multiple solutions for managing shared resources to avoid data inconsistency or corruption of the state of an object.

A field can be declared as `volatile`, and this guarantees that the Java Memory Model will make sure that all threads see a consistent value for the variable.

If a method or block is marked with the `synchronized` keyword, it can only be accessed by one thread at a time for the same object. In some situations, it could be more flexible to use the `java.util.concurrent.locks` package, which provides a framework for locking and waiting for conditions [8].

Also, the Java Concurrency API was introduced in Java 5, containing the `java.util.concurrent.atomic` package which includes classes that are useful in concurrent programming [9]. The `AtomicInteger`, `AtomicLong`, `AtomicBoolean`, and `AtomicReference` classes are created to perform atomic operations on single variables of integer, long, boolean and object reference types. In addition to this, the package contains concurrent collections that are thread-safe versions of standard Java collections (e.g. `ConcurrentHashMap` of `HashMap`).

```
private AtomicInteger number = new AtomicInteger(0);
public void update(int value) {
    number.set(value);
}
public int addAndGet(int value) {
    return number.addAndGet(value);
}
```

Listing 1: Atomic usage of an AtomicInteger field

If the methods contain only one operation for one atomic variable, the code is correct, because the implementations of the `atomic` package provide atomicity, even if the methods run in different threads. However, if atomic variables are combined, then synchronization is always necessary to create atomic methods that combine multiple resources.

The case is the same when a method contains more than one operation for an atomic variable because these are not atomic overall, and could cause race condition between the threads. So, to handle this, synchronization of code blocks or functions guarantees that multiple threads cannot simultaneously modify or access shared resources.

3 Finding non-atomic operations with static analysis

The *VNA03-J* [10] and *VNA04-J* [11] SEI Cert rules focus on the proper use of locking with synchronization. The authors designed and implemented two checkers into the SpotBugs static analyzer to find unsafe usages of common references between threads, and make sure the proper usage of fields with Java `atomic` type. The checker covering VNA04-J works with references, of which types are not related to the Java Concurrent API, and the other checker ensures the proper usage of `atomic` type-based classes.

The checkers have the same base logic: analyzing functions in a class context and log each method call and field assignment of common objects which are not `synchronized`. If a method contains a `synchronized` block, the detector logs only once for every different object inside the block, no matter how many times they are accessed; because of the synchronization, it is considered an atomic operation. If a private method performs an unsafe operation without proper synchronization, but all the methods that call it have proper synchronization, then the private function does not need another one.

Shared data could be a field of the class, a function argument, or a local variable containing a reference for a shared resource, for example, an element of an atomic collection.

There is a strict constraint in the VNA04-J checker, determining what methods to examine, because only concurrent logic needs to be analyzed. In addition, the Java `atomic` types are ignored. `Thread` instances are sought by the detector and the functions that are passed to threads are then processed, including other methods which are in the call hierarchy, but taking place in the current class. If the detector finds shared data that is used at least once in multiple threads without a consistent locking policy, it marks these instructions as a bug.

The `atomic` typed fields and collections have atomic methods, so the VNA03-J checker must seek multiple or combined usages of these method calls. If a shared data is used more than once, the operations are not atomic, so these accesses will be marked as bug. There may be the possibility that all shared data are used just once in a method, but if combined (for example `a.get().add(b.get())`) then it is a bug too, because these two resource accesses must be atomic. It is important to note that, if a shared data is used by multiple methods and at least one accesses it more than once, all methods that work with it need synchronization for consistent locking.

4 Results

To validate our checkers, first we implemented a large number of unit test cases to eliminate potential bugs and filter out possible false positive cases. After that we evaluated our VNA03-J checker on large, modern, open source Java projects, which applied concurrent solutions to get real-world measurement data. The results (can be seen in Table 1) show that the detector has low hit rate, indicating its suitability for practical, real-world projects, as confirmed by manual review of the marked errors to prevent false-positive alerts.

Table 1: VNA03-J Measurements on large, open source projects

Project	Lines of Java Code	Count of combined access bugs	Count of simple access bugs
Bt [12]	80 714	6	14
MATSim-Libs [13]	154 975	48	18
Jenkins [14]	311 151	0	0
OpenGrok [15]	146 324	0	0

We marked *combined atomic accesses* issues where atomic variables are accessed multiple times in the same function without synchronization and marked cases of *simple atomic accesses*, when the access needs synchronization due of the existence of the combined resource usages in other methods.

```java
private AtomicInteger num = new AtomicInteger(0);
public void conditionalUpdate(int val) {
    if (num.get() < 0) {
        num.set(val); // combined atomic access bug, multiple access of var
    }
}
public int addAndGet(int val) {
    return num.addAndGet(val); // simple atomic access bug
}
```

Listing 2: Example of the relation between the bug types

In the `conditionalUpdate` method there are unsynchronized *combined atomic accesses*, and because of this, the operation in the `addAndGet` method is marked as a *simple atomic access* bug.

5 Conclusion

In concurrent programming, it is crucial to use shared resources in a thread-safe way. To achieve this, it is recommended to use a consistent locking policy, which could be even necessary, when a program works with Java `atomic` based types. Static analysis is a very useful tool to look for mistakes and make sure the developers implement their concurrent logics in a proper way. The authors developed new detectors to the SpotBugs project (an open source static analyzer tool for the Java language) to detect this kind of unsafe resource usage between concurrent threads leading to more reliable applications worldwide.

References

[1] V. H. Halim and Y. Dwi Wardhana Asnar, "Static Code Analyzer for Detecting Web Application Vulnerability Using Control Flow Graphs", 2019 ICoDSE, Pontianak, Indonesia, 2019, pp. 1-6

[2] I. Aghav, V. Tathe, A. Zajriya and M. Emmanuel, "Automated static data flow analysis", 2013 Fourth ICCCNT, Tiruchengode, India, 2013, pp. 1-4

[3] X. Zhang, Y. Zhou and S. H. Tan, "Efficient Pattern-based Static Analysis Approach via Regular-Expression Rules", 2023 IEEE International Conference on SANER, Taipa, Macao, 2023, pp. 132-143

[4] SEI CERT Coding Standards, https://wiki.sei.cmu.edu/confluence/display/seccode/SEI%2bCERT%2bCoding%2bStandards, accessed 03 2024

[5] Java virtual machine, https://www.artima.com/insidejvm/ed2/platindep.html, accessed 03 2024

[6] Official SpotBugs website, https://spotbugs.github.io, accessed 03 2024

[7] Apache Commons BCEL, https://commons.apache.org/proper/commons-bcel/, accessed 03 2024

[8] The Java Language Specification – Java SE 21 Edition, https://docs.oracle.com/javase/specs/jls/se21/jls21.pdf, accessed 03 2024

[9] Package java.util.concurrent, https://docs.oracle.com/javase/8/docs/api/java/util/concurrent/package-summary.html, accessed 03 2024

[10] VNA03-J. Do not assume that a group of calls to independently atomic methods is atomic, https://wiki.sei.cmu.edu/confluence/display/java/VNA03-J.+Do+not+assume+that+a+group+of+calls+to+independently+atomic+methods+is+atomic, accessed 03 2024

[11] VNA04-J. Ensure that calls to chained methods are atomic, https://wiki.sei.cmu.edu/confluence/display/java/VNA04-J.+Ensure+that+calls+to+chained+methods+are+atomic, accessed 03 2024

[12] Bit Torrent, https://github.com/muddlebee/java-bt, accessed 03 2024

[13] MATSim, https://github.com/matsim-org/matsim-libs, accessed 03 2024

[14] Jenkins, https://github.com/jenkinsci/jenkins, accessed 03 2024

[15] OpenGrok, https://github.com/oracle/opengrok, accessed 03 2024

New interval-based training technique to parameter robustness

Attila Szász and Balázs Bánhelyi

Abstract: Today's artificial neural networks appear in many scientific fields and have a wide range of applications, for example, they are widely used for image and speech recognition. Over the years, the accuracy of the networks has continuously improved, but many studies have shown that these networks are also not error-free. Many technologies have been developed to reduce the probability of error, but most of them look for examples of adversities in the input space and try to make neural networks more robust by using them. There is a much less technique to search for aversality in the smaller distances of the weight matrices of the neural network. However, this type of adversity is magnified in that area, where taught neural networks are evaluated with much lower accuracy. In these quantized neural networks, the results differ greatly from the expected result.

Keywords: Neural Network training, Parameter robustness, Interval Arithmetic

1 Introducion

Neural networks have received outstanding attention in recent years, both from users and from the research side. Nowadays, they are used in many fields, such as computer vision and speech recognition. Most of the research is increasing and has gone in the direction of creating more reliable networks [1]. To teach reliable networks, also known as robust networks, researchers have proposed several teaching techniques. The two majority of methods can be classified into 2 groups. The algorithm is based on adversarial learning and many of the adversarial examples optimize the parameters of the networks. Certified teaching-based methods count as inclusion in the network to its outputs, and then the worst case that can be assumed based on the constraints is minimized. In addition to input-focused attacks, there are also network parameter attacks coming to the fore. This type of attack modifies the parameters of the networks and thus provokes hostile behavior. Based on these, teaching methods were developed that incorporate the achievement of stability of the network parameters into the teaching process, which the most common in the literature is Adversarial Weight Perturbation (AWP) [2], became his method. The main disadvantage of the algorithm is that the worst case is extreme underestimates, as a result of which it did not provide full protection against parameter attacks much opposite. During our research, we showed the main weakness of the AWP algorithm and proposed a certified-based learning algorithm that increased the resilience of networks against adversarial parameter attacks.

2 Robust teaching methods

During our investigation, we examined the robustness of the networks from two points of view: input and in terms of parameter robustness. Input robustness is network resilience, referring to its ability to react to changes in the input. In the literature, most of the research focused on this robustness. The parameter is robustness, on the other hand, for resistance to perturbation of the weight and bias parameters to which the network refers. Based on these, a network is called input robust if the input is in a fixed environment, the network produces the same output for all inputs, and parameter robust if the network is in a fixed environment of its parameters. All existing alternative networks result in the same output for the fixed input.

During robust input training, the goal is Θ a parameter configuration of the model finding the one that minimizes the maximum expected in the environment (ϵ) of the input error:

$$\Theta = \arg\min_{\Theta} \mathbb{E}_{x,y} \left[\max_{\check{x} \in B_p(w,\epsilon)} L(\check{x}, y) \right]$$

L is the error function used during teaching, and $B_p(w, \epsilon)$ is the inclusion of the ϵ radius of the input x denotes, based on which $B_p(w, \epsilon) = \{\check{x} : \mid \check{x} - x \mid_p \leq \epsilon\}$

3 Type of methods

The teaching algorithms proposed in the literature can be classified into two groups depending on how the internal maximization task is optimized.

During adversarial training, the network is trained with inputs that maximize the error function in the given environment. One of the most common methods is Projective Gradient Descent (PGD). Based on the input gradient, the algorithm maximizes the error, and then the model optimizes on the input that results in the maximum error parameters. The method can easily get stuck in a local optimum, as a result of which the worst case can be underestimated. Adversarial networks are still vulnerable with stronger attack methods based on propagation and linear programming against.

Certified teaching methods are a reliable inclusion in the online to the outputs, and then optimize the parameters. Many methods are known to include the output, which usually overestimates the set of output values. Due to the strong overestimation, the methods introduce a very strong regularizing effect into the teaching process, the results of which evaluation of networks is simpler, and their demonstrable robustness increases, however, its normal accuracy over the test set decreases. It has been shown in the literature that the algorithm used during the verification of networks produces narrower limits, although they would reduce the strong overestimation, they are much more difficult to optimize and would result in data that would eventually lead to lower-performing networks [3]. Qualified teaching methods have appeared that are precise, but do not expect an unconditional reliable inclusion of the output value set.

The teaching, evaluation and use environment of networks can often be different, and the parameters of the networks must be converted to the type of the current environment. However, the result of the conversion will be a completely new network with the original one, and the operation identical to the network cannot be guaranteed.

Algorithms focusing on input robustness are explicitly input-error space is optimized by adversarial inputs (adversarial training) or propagated constraints (certified training) based on the worst-case scenario. The methods have a common back, on the other hand, the worst case is always locally, the current input, or input calculated based on a group. On the other hand, during parameter robustness, we are looking for parameters (adversary networks) that trigger gestalt behavior, which are the complete error over the training set is maximized, thus a model-level, global worst result in a case. Based on the above, the question of parameter perturbation of networks can be a useful and important addition to methods that only focus on input perturbation. The professional teaching method used in the literature is illustrated in Table 1.

4 IA and PGD combined based method

Our method is a certified algorithm (unsound) that is more resilient in networks against parameter attacks such as AWP. The essence of the method is that the output value sets are included in the interval of the parameters of the network calculated in addition to its value, giving a reliable but grossly overestimated value, worst case according to meters. To compensate for gross overestimation, the $\check{x}$ component of the subtask is optimized with PGD. Because PGD does not guarantee the global optimum, the limits calculated with $\check{x}$ will certainly not contain a global worst-case (unsound method), but a precise solution is expected to be provided, which is sufficient for teaching.

Input-Certified algorithms often use propagation methods to compensate for the naive overestimation due to the dependency problem of interval arithmetic. In addition to compressing

Table 1: Classification of robust training methods

$\breve{x}$ (*Input*)	$\breve{\theta}$ (*Network*)	Class	Implementation
PGD	fixed	Input-Adversarial	PGD
IA	fixed	Input-Certified (sound)	IA, IBP, ...
PGD+IA	fixed	Input-Certified (unsound)	TAPS
fixed	IA	Model-Certified	-
fixed	PGD	Model-Adversarial	-
IA	IA	Certified-Sound	this work
IA	PGD	Certified-Unsound	-
PGD	IA		this work
PGD	PGD	Adversarial	AWP

the networks, the algorithms significantly reduce the length of the calculation chain, and they reduce the overestimation error. Our algorithm for the parameters of the networks calculates the output value set in addition to its inclusion there by making propagation impossible methods. When including the parameters, the radius of the intervals is determined relatively, taking into account the absolute value of the parameter with which we can properly weight the required interval widths. With this technique, the degree of overestimation can be kept at an acceptable level, while the robustness of the neural network improves.

5 Conclusion

A robust learning algorithm based on the parameters of neural networks was prepared. In the lecture, it was compared with the techniques available in the literature, which shows that nets were not taught using this technique before. Based on the results, better results can be achieved on medium-sized networks than previous techniques. Unfortunately, on large networks, the interval technique becomes unusable due to overestimations, but we hope that this disadvantage can be reduced with statistical procedures.

References

[1] Zombori, Dániel, Bánhelyi, Balázs, Csendes, Tibor, Megyeri, István and Jelasity, Márk: Fooling a Complete Neural Network Verifier, *International Conference on Learning Representations*,2021.

[2] Wu, Yihan, Bojchevski, Aleksandar and Huang, Heng:Adversarial Weight Perturbation Improves Generalization in Graph Neural Networks, *NeurIPS*, 2020.

[3] Szász, Attila and Bánhelyi, Balázs: Effective inclusion methods for verification of ReLU neural networks, *Annales Mathematicae et Informaticae*, 2024.

Evaluating GPT-4 on a real Python bug dataset

Norbert Vándor

Abstract: Python is currently one of the most popular languages. In a previous work, we created PyBugHive, a Python database consisting of 151 manually validated bugs from 11 projects. The entries in the database contain the bug report's summary, the patch that fixes and the test cases that expose the given bug. We also demonstrated a use case involving a large language model, GPT-3.5[1]. We used multiple prompts to see whether GPT-3.5 could detect, or even fix the bugs present in the database. In the case of bug finding, the results were mediocre, and in the case of bug fixing, they were negligible. As a follow-up, we present an updated benchmark, now with GPT-4 and updated prompts.

Keywords: bug dataset, benchmark, Python, real bugs, LLM

1 Introduction

In a previous work, we introduced PyBugHive, a meticulously curated dataset of Python bugs derived from well-known open-source projects. PyBugHive aims to fill the void between Python's popularity and the scarcity of high-quality research datasets, providing researchers with a dependable resource for their experiments. Modeled after influential bug databases like Defects4J and BugsJS, PyBugHive includes buggy code, associated failing test cases, fixed code, and detailed installation instructions to replicate the bugs.

To showcase PyBugHive's utility, we presented a use case involving GPT-3.5, a large language model. We wanted to find out two things: whether the AI could find the bugs contained within the database, and whether it could provide a fix for them. To achieve this, we used multiple prompts: 4 for bug finding, and 1 for bug fixing. The results were mediocre at best. As a follow-up to our previous work, this paper aims to evaluate a large language model again: this time GPT-4.

Bug datasets can be used in a variety of ways with both static and dynamic analysis methods. For example, it is widely used in defect prediction [1]. Vince et al. [2] and Vancsics et al. [3] used Defects4J in spectrum-based fault localization. Several bug datasets were used in the work of Jiang et al. [4], who evaluated ten code language models on four benchmarks, including Defects4J and QuixBugs. Kang et al. [5] used Defects4J to evaluate their framework LIBRO in automatically generating tests from bug reports.

In the field of automated program repair, bug datasets are commonly used [6]. QuixBugs is often employed in automatic program repair: Ye et al. [7] studied the effectiveness of 10 program repair tools on the part of QuixBugs. Prenner et al. [8] used OpenAI's Codex [9] for automated program repair. They concluded that although Codex is not trained for automatic program repair, it is quite efficient, and Codex works better for Python code than for Java. Sobania et al. [10] used QuixBugs to evaluate ChatGPT's bug-fixing performance. They concluded that ChatGPT's bug-fixing capability is comparable to the widely used deep learning techniques CoCoNut and Codex and is even superior to the results of standard program repair techniques.

The rest of the paper is organized as follows. We provide a short summary of PyBugHive in Section . The methodology we used for evaluating GPT-4's capabilities is presented in Section . We provide the results in Section . Finally, we draw conclusions in Section .

[1]`https:\chat.openai.com`

2 Summary of PyBugHive

PyBugHive is a manually curated and validated, reproducible Python bug dataset. It consists of 151 bugs collected from 192 files of 11 open-source GitHub projects. All of the collected bugs fulfill a number of restrictive criteria. For example, the issue the bug is discovered in must have a bug label and must be fixed in a single commit.

PyBugHive contains not just the bugs themselves, but also the commit and patch information, and the necessary steps to install, test, and verify them. We also provided a CLI program, which can automatically do these steps.

3 Methodology

To evaluate GPT-4, we decided to update the prompts used in the previous paper (which is currently under publication).

We have to give two different kinds of prompts. *System prompts*, which set the context for the AI, are used by models like GPT to steer the model's reaction in a particular direction before it interacts with the user. However, the real inputs and questions utilized to communicate with the AI are called *user prompts*.

It can be difficult to choose the right prompts because the answers that the AI provides can differ greatly based on the user prompts that are used. To achieve the best outcomes, we employed two distinct system prompts for the two jobs.

3.1 Bug searching

The first system prompt was used in bug searching. It instructed the model to act like a senior developer conducting a comprehensive code review. We devised multiple user prompts and tested them to see which worked best for our situation. In the end, we chose the following two prompts:

> **Prompt 1**: "Does the code contain any major bugs? Whenever you generate an answer, please explain the reasoning and assumptions behind it. If possible, use specific examples or evidence with associated code samples to support your answer of why the code is buggy or not. Moreover, please address any potential ambiguities or limitations in your answer, in order to provide a more complete and accurate response. Begin your answer with a yes or no."
>
> **Prompt 2**: "Does the code contain any major bugs? I am going to provide a template for your output. Everything between [[and]] is a placeholder. Any time that you generate text, try to fit the output into the placeholder that I list. Preserve the formatting and overall template that I provide. This is the template: Answer: [[YES or NO]]"

3.2 Bug fixing

The second system prompt, which was used in bug fixing, instructed the AI to act like an expert developer doing a review of a code snippet which was marked buggy by another developer.

The user prompt was the following:

> **Prompt**: "Another expert said that the following code contains bugs, and made a github issue. Based on that, fix the following code's bugs. You have to return the entire code, with your fixes included. The resulting code should be the last part of your answer. Do not include comments in your code! The title of the mentioned github issue is the following: {the title of the issue}."

Despite the specification of the prompt, the AI sometimes returned more than what was asked, included comments, changed the code indentation, etc. This was a problem in the previous paper as well, meaning the new prompt did not fix the issue. Consequently, every result had to be manually checked and cleaned. After that, we manually compared the original patch with the one the AI created. We marked each patch with the following:

Fix: if the patch provided by the AI correctly fixes the problem. **Partial fix**: if the patch provided by the AI only partially fixes the problem. **Minor change**: if the patch provided by the AI modified parts of the code, which was not a part of the original bug, but was a correct suggestion nonetheless. **No change**: if the AI returned the original code, unmodified. And **Failure**: if the patch provided by the AI failed to fix the issue.

4 Results

In this section, we provide the results of our evaluation.

4.1 Bug searching

Table 1: Results of the evaluation

Project Name	# Buggy Samples	Prompt 1 Yes	Prompt 1 No	Prompt 2 Yes	Prompt 2 No	# Has Yes	# Multi Yes	# Corr. Expl.	Failure	No change	Minor change	Partial fix	Fix
black	38	1	37	1	37	1	1	0	37	0	0	0	1
cookiecutter	2	0	2	2	0	2	0	1	1	0	0	0	1
discord.py	2	0	2	2	0	2	0	0	2	0	0	0	0
freqtrade	15	0	15	1	14	1	0	0	15	0	0	0	0
jax	17	0	17	1	16	1	0	0	16	0	0	1	0
numpy	1	0	1	0	1	0	0	0	1	0	0	0	0
pandas	45	2	43	6	39	6	2	0	42	0	0	1	2
poetry	11	0	11	2	9	2	0	1	11	0	0	0	0
salt	7	2	5	3	4	3	2	0	6	0	0	0	1
scrapy	2	0	2	1	1	1	0	0	2	0	0	0	0
spaCy	11	0	11	3	8	3	0	0	10	0	0	0	1
$\sum$	151	5	146	22	129	22	5	2	143	0	0	2	6

The first half of Table 1 summarizes the results obtained with GPT-4 grouped by projects. The total number of code snippets shown to the model with known bugs is shown in the buggy samples column. As was previously mentioned, we asked the model to identify any bugs in the given source code using two separate prompts. The table shows how many times the model answered a prompt with a *yes* (the code is buggy) or a *no* (the code is clean). The number of instances in which the model answered *yes* to at least one of the prompts is displayed in the following column. The number of instances where more than one prompt resulted in a *yes* response is listed in the last but one column. The last column lists the number of bugs where the model answered with a *yes* that also had correct explanations.

As can be seen, both prompts resulted in very few *yes* responses: there were only 22 cases, where at least one of the prompts resulted in a *yes*. Out of those 22 cases, there were only 5 where both prompts resulted in a *yes*. Also, the model gave correct explanations for why a given code snippet is buggy in only 2 cases. Interestingly, both of those explanations came from Prompt 2, which is supposed to be a simple *yes* or *no* question. Additionally, these are not the only cases, where the AI ignored the template of the prompt, but with the other cases, the explanation given was wrong.

In comparison, GPT-3.5 was able to detect 67 bugs, by providing at least one *yes* answer. Out of those cases, there were 11 with more than one *yes* result.

4.2 Bug fixing

The second half of Table 1 shows the results of bug fixing, with the labels discussed in in the header. As can be seen, most fixing attempts resulted in a failure. However, in 2 cases, the model provided a partial fix, and in 6 cases, it provided a correct fix. In comparison, GPT-3.5 was able to correctly fix only 1 issue, and partially fix another.

5 Conclusion

We followed up on one of our previous paper and evaluated GPT-4's bug searching and fixing capabilities on PyBugHive, our Python bug database. In addition to upgrading GPT's model, we also updated the prompts.

In the case of bug searching, the model correctly marked 14% of the bugs in at least one prompt and 3% in both prompts. In 2 cases, it also provided a correct explanation (despite not instructed to do so). In the case of bug fixing, almost all attempts resulted in failure. However, it could partially fix 1.3%, and completely fix 3.9% of the bugs. As mentioned in Section , GPT-3.5 had wildly different results: the old model was able to find more bugs but fix less.

References

[1] G. Antal, Z. Tóth, P. Hegedűs, and R. Ferenc. Enhanced bug prediction in javascript programs with hybrid call-graph based invocation metrics. *Technologies*, 9(1):3, 2020.

[2] D. Vince, A. Szatmári, Á. Kiss, and A. Beszedes. Division by zero: Threats and effects in spectrum-based fault localization formulas. In *2022 IEEE 22nd International Conference on Software Quality, Reliability and Security (QRS)*, pages 221–230. IEEE, 2022.

[3] B. Vancsics, F. Horváth, A. Szatmári, and Á. Beszédes. Fault localization using function call frequencies. *Journal of Systems and Software*, 193:111429, 2022.

[4] N. Jiang, K. Liu, T. Lutellier, and L. Tan. Impact of code language models on automated program repair. *arXiv preprint arXiv:2302.05020*, 2023.

[5] S. Kang, J. Yoon, and S. Yoo. Large language models are few-shot testers: Exploring llm-based general bug reproduction. In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*, pages 2312–2323. IEEE, 2023.

[6] S. Saha, R. k. Saha, and M. r. Prasad. Harnessing evolution for multi-hunk program repair. In *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*, pages 13–24, 2019.

[7] H. Ye, M. Martinez, T. Durieux, and M. Monperrus. A comprehensive study of automatic program repair on the quixbugs benchmark. *Journal of Systems and Software*, 171:110825, 2021.

[8] J. A. Prenner and R. Robbes. Automatic program repair with openai's codex: Evaluating quixbugs. *arXiv preprint arXiv:2111.03922*, 2021.

[9] M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. d. O. Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

[10] D. Sobania, M. Briesch, C. Hanna, and J. Petke. An analysis of the automatic bug fixing performance of chatgpt. *arXiv preprint arXiv:2301.08653*, 2023.

Knowledge Graph Powered LSTM in Stock Investment Decision Making

Ronglin Zuo, Bálint Molnár

Abstract: There are a lot of factors related to the trend of the stock market, not just the price at which people buy and sell. Some significant decisions of enterprises, new policies, or regulations of the country or city will impact the stock price. All those kinds of information exist in the daily news data. However, some news didn't directly mention the company's name, on the contrary, it may mention some area, industry, the name of an executive, etc. Nowadays researches' shortcomings are they ignore or can't let those news relate to any stock. This paper propose to use Knowledge graph to build stock information, dig deeper relationships, and find more relevant news. Used LSTM with news sentiment to predict future trends. Moreover, introduced further potential research directions and discussed the influence of trend prediction on the stock market.

Keywords: Finance, Stock Movement, LSTM, Knowledge Graph

1 Introduction

Artificial intelligence has made our lives more convenient, saves more time, and energy. Intelligent services can also give us so-called "perfect" answers fast and accurately without being distracted by additional human emotions[5]. This is because machine can process and analyze a large amount of data quickly. Applying this advantage to stock investment can help people process a large amount of information, estimate whether a stock will rise or fall, and give recommendations instantly. Financial firms based on AI technology use existing data to analyze and predict future trends[4]. However, many problems still need to be solved and improved in prediction[3]. Many factors affect the stock price. News aspects include the recent international situation, new government regulations, state support for a new industry, or companies involving scandals, such as quality issues or fraud. That real-time information will impact those correlated stocks, which can cause them to rise or fall[6]. Of course, there are many other factors. Exchange rate, house price, oil price, etc [2]. The number of stocks is enormous, and there are many connections between them, those connections affect each other or have common characteristics that shouldn't be ignored. However, the current research on price prediction does not include those connections, they only use one or two potential direct influencing factors to predict the trend and price, and even use the price to predict the price directly, ignoring the relationship between influencing factors and price, just like predict a person's weight only concerns how many slices of bread he ate yesterday and his previous weight. Exploring and storing the connections is unsuitable for traditional databases. By contrast, using knowledge graph can better show the relationships between them. This paper mainly builds stock-related information structure, predict trends, and the impact of news information on stock prices. Moreover, based on this research we give other extended research directions and prove their possibility and importance.

2 Research work

Use the existing open-source financial database Tushare Pro data platform [1] to obtain real-time financial news data, and conduct research by using each stock's information, and daily stock prices.

2.1 *Comparison of Chinese word segmentation tools*

This paper compares the following Chinese word segmentation tools:pyltp, Tsinghua, NLPIR, Elasticsearch, Jieba, and Pkuseg-python. Discussed their corpus and suitable situation and compared their results and accuracy. The best performance is pkuseg-python, which can achieve higher accuracy than other word segmentation tools for specific news feeds.

2.2 *Build dictionaries and analysis sentiment*

Sentiment classification of news is usually based on the emotional dictionary-based and machine learning-based approaches. We built a stopwords dictionary, user word dictionary, and financial sentiment dictionary according to existing open source dictionaries, did sentiment analysis, and used sentiment analysis API (SnowNLP sentiment analysis, Baidu Sentiment Analysis API(NLP-Python-SDK), cnsenti Chinese sentiment analysis library) and com-pared their result.

2.3 *Build knowledge graph and use LSTM to predict future stock trend*

Established a basic knowledge graph by extracting important information about each stock, and then associated news. For each stock, sum sentiment values for its directly associated and indirectly associated news sentiment. Use LSTM to predict future stock prices. Compaird directly use price to predict, use news sentiment only contain the corresponding company's name, use news sentiment contain any associated information.

2.4 *Further research probability and discussion*

Positive and negative analysis of financial emotional words: positive and negative analysis of emotional words in the financial field and generation of positive vocabulary dictionaries and negative vocabulary dictionaries. And their two-side effect analysis. Automatic detection of lexica: It can automatically detect new lexica and verify its credibility, and can identify existing lexica that was segmented wrong. The impact level and last time of positive news and negative news on stock trends(not easy). Other influencing factors analysis: exchange rate, house price, oil price, etc. Data analysis and confidence research on factors that affect stock prices: Find factors that affect stock prices and put them in the database and use them together for analyzing stock trends. Buy-in and sold-out opportunity analysis: Can use moving average convergence divergence(MACD) and Bollinger Bands, etc. Increase the correctness of investment actions. Auto quantitative trading. Future discussion of the predicted trend effect of the stock market: If everyone can predict stock price. If the prediction working well everyone can use it. If predict well, but only investment company can use it.

3 Conclusion

In my opinion, the accuracy of the estimates is not ideal. It also proves that it is not enough to only use news to predict stock prices even already dug deeper relation of news and stocks, and more influencing factors need to be added as judgment conditions. Moreover, the analysis of news data, especially the processing of financial news, needs to be improved. News keeps pace with the times, and in terms of word segmentation, it is also necessary to identify and summarize new lexical at all times. The sentiment vocabulary judgment of news also needs to be strengthened.

Acknowledgements

This research was supported the Thematic Excellence Programme TKP2021-NVA-29 (National Challenges Subprogramme) funding scheme, and by the COST Action CA19130 - “Fintech and Artificial Intelligence in Finance Towards a transparent financial industry” (FinAI).

References

[1] Tushare Pro, February 2024. [Online; accessed 11. Feb. 2024].

[2] David M. Cutler, James M. Poterba, and Lawrence H. Summers. What Moves Stock Prices? *NBER*, March 1988.

[3] John W. Goodell, Satish Kumar, Weng Marc Lim, and Debidutta Pattnaik. Artificial intelligence and machine learning in finance: Identifying foundations, themes, and research clusters from bibliometric analysis. *Journal of Behavioral and Experimental Finance*, 32:100577, December 2021.

[4] Debidutta Pattnaik, Sougata Ray, and Raghu Raman. Applications of artificial intelligence and machine learning in the financial services industry: A bibliometric review. *Heliyon*, 10(1):e23492, January 2024.

[5] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach (4th Edition)*. Pearson, 2020.

[6] B. Shravankumar and Vadlamani Ravi. A survey of the applications of text mining in financial domain. *Knowl. Based Syst.*, 114:128–147, 2016.

List of Authors

Angyalné Alexy, Márta: Eötvös Loránd University, Budapest, Hungary,
E-mail: `abalord02@inf.elte.hu`

Arafat Md, Easin: Eötvös Loránd University, Budapest, Hungary,
E-mail: `arafatmdeasin@inf.elte.hu`

Asuah, Georgina: Eötvös Loránd University, Budapest, Hungary,
E-mail: `asuahgeorgina@inf.elte.hu`

Ádám, Zsófia: Budapest University of Technology and Economics, Hungary,
E-mail: `adamzsofi@edu.bme.hu`

Bánhelyi, Balázs: University of Szeged, Hungary,
E-mail: `banhelyi@inf.u-szeged.hu`

Bognár, Gergő: Eötvös Loránd University, Budapest, Hungary,
E-mail: `bogqaai@inf.elte.hu`

Bozó, István: Eötvös Loránd University, Budapest, Hungary,
E-mail: `bozo_i@inf.elte.hu`

Erdei, Zsófia: Eötvös Loránd University, Budapest, Hungary,
E-mail: `zsanart@inf.elte.hu`

Farkas, Martin: Budapest University of Technology and Economics, Hungary,
E-mail: `martin.farkas@edu.bme.hu`

Ferenczi, Daniel: Eötvös Loránd University, Budapest, Hungary,
E-mail: `danielf@inf.elte.hu`

Fridli, Sándor: Eötvös Loránd University, Budapest, Hungary,
E-mail: `fridli@inf.elte.hu`

Gera, Imre: University of Szeged, Hungary,
E-mail: `gerai@inf.u-szeged.hu`

Hajder, Levente: Eötvös Loránd University, Budapest, Hungary,
E-mail: `hajder.levente@gmail.com`

Heinc, Emília: University of Szeged, Hungary,
E-mail: `heincze@inf.u-szeged.hu`

Hoque, A. H. M. Sajedul: Eötvös Loránd University, Budapest, Hungary,
E-mail: `sajed@inf.elte.hu`

Kangogo, Damaris Jepkurui: Budapest University of Technology and Economics, Hungary,
E-mail: `dkangogo@edu.bme.hu`

Knežev, Smiljana: Eötvös Loránd University, Budapest, Hungary,
E-mail: `smiljana.knezev@inf.elte.hu`

Knoll, Judit: Eötvös Loránd University, Budapest, Hungary,
E-mail: `judit.knoll@sigmatechnology.com`

Kocsis, Imre: Budapest University of Technology and Economics, Hungary,
E-mail: `kocsis.imre@vik.bme.hu`

London, András: University of Szeged, Hungary,
E-mail: `london@inf.u-szeged.hu`

Micskei, Zoltán: Budapest University of Technology and Economics, Hungary,
E-mail: `micskei.zoltan@vik.bme.hu`

Molnár, Bálint: Eötvös Loránd University, Budapest, Hungary,
E-mail: `molnarba@inf.elte.hu`

Munkácsi, Imre: Eötvös Loránd University, Budapest, Hungary,
E-mail: `munkacsiimre@inf.elte.hu`

Orosz, Tamás: Eötvös Loránd University, Budapest, Hungary,
E-mail: `orosztamas@inf.elte.hu`

Péter, Bertalan Zoltán: Budapest University of Technology and Economics, Hungary,
E-mail: `bpeter@edu.bme.hu`

Porkoláb, Zoltán: Eötvös Loránd University, Budapest, Hungary,
E-mail: `gsd@inf.elte.hu`

Ságodi, Zoltán: University of Szeged, Hungary,
E-mail: `sagodiz@inf.u-szeged.hu`

Sebők, Mátyás: Eötvös Loránd University, Budapest, Hungary,
E-mail: `sebokmatyas01@gmail.com`

Siket, István: University of Szeged, Hungary,
E-mail: `siket@inf.u-szeged.hu`

Solis, Wilson Valdez: University of Szeged, Hungary,
E-mail: `wilson@inf.u-szeged.hu`

Süli, Patrik Péter: Eötvös Loránd University, Budapest, Hungary,
E-mail: `spatrik95@gmail.com`

Szász, Attila: University of Szeged, Hungary,
E-mail: `szasz@inf.u-szeged.hu`

Tarlan, Ahadli: Eötvös Loránd University, Budapest, Hungary,
E-mail: `arlanahad@gmail.com`

Tóth, Melinda: Faculty of Informatics, Eötvös Loránd University, Budapest, Hungary,
E-mail: `tothmelinda@elte.hu`

Vándor, Norbert: University of Szeged, Hungary,
E-mail: `vandor@inf.u-szeged.hu`

Zuo, Ronglin: Eötvös Loránd University, Budapest, Hungary,
E-mail: `oscrtq@inf.elte.hu`

Notes