Informatics PhD projects at King's College London, AY 24-25 — Foundations of computing

The PhD project proposals listed below will be considered for 2024/25 studentships available in the Department of Informatics to start 1 October 2024 or later during the 2024/25 academic year. Please note that this list is not inclusive and potential applicants can alternatively identify and contact appropriate supervisors outlining their background and research interests or proposing their own project ideas.

The PhD projects are listed in two groups. In the first group are the projects with allocated studentships: each project in this group has one allocated studentship. The remaining studentships will be considered for the projects listed in the second group. The number of those remaining studentships is smaller than the number of the projects in the second group. The allocation of studentships will be based on the merits of individual applications. Applications for PhD studies in the Department of Informatics, for all listed projects as well as for other projects agreed with supervisors, are also welcome from students applying for other funding (within other studentship schemes) and from self-funded students. See also this list of funding opportunities available at King's for post-graduate research in Computer Science.

Scholarship Allocated

(Back to Top)

- Reliability and verification of software for scientific and ML computing
- The complexity of database query evaluation

Reliability and verification of software for scientific and ML computing

Supervisor: Dr Karine Even-Mendoza, Dr Hana Chockler, Dr Hector Menendez Benito (1st, 2nd and 3rd).

Areas: Machine Learning (ML), Artificial Intelligence (AI), Systems (SE, programming, autonomous systems, robotics, ...), Foundations of computing, Computing Applications

(Back to Scholarship Allocated)

Project Description

Project Description. The long-standing challenge of ensuring the reliability of machine learning implementations (ML) and programming language (PL) libraries, particularly in floating-point and arithmetic computation, has been a focus of attention within programming languages, formal methods, and AI research communities. Historically, researchers have often tended to confine the scope of their research to small-scale problems rather than real-world computer systems. However, modern applications increasingly rely on mathematical computations, such as Alexa voice recognition (using discrete Fourier transform) and autonomous cars. This heightened dependence on accurate computation amplifies concerns about the impact of inaccuracies on system reliability. This project addresses low-level implementations heavily reliant on floating-point computations—a crucial but insufficiently tested area. The precision of these computations significantly influences the reliability of ML and PL libraries and compilers. The project aims to enhance the testing of software systems, specifically focusing on the reliability of software using ML and PL libraries in their algorithms. To achieve this, the student will: - Develop methods to assess the quality of test cases, applying them when attempting to detect miscompilations (silent errors during translation into machine code) and logical bugs in floating-point optimizations and library implementations. - Investigate approaches for testing software libraries and their compilers meaningfully in the context of mathematical and numerical procedures. - Explore fault localization approaches and other techniques to pinpoint detected bugs, ensuring clarity in distinguishing actual bugs from potential issues related to the testing mechanism. The student will employ static and dynamic code analysis, code generation for testing, and testing strategies (like differential testing). After designing a system for meaningful testing of ML and PL libraries, the student will extensively evaluate its bug detection capabilities. The emphasis is on ensuring the identified issues are genuine bugs rather than stemming from the testing methodology. The student will actively engage with the software engineering community, reporting any bugs uncovered during the evaluation process. The above will include investigations of novel ways to design tests and testing campaigns and deal better with coverage of specific functionalities in the compilers and their PL and ML libraries. Context. Compilers and their software libraries, widely used complex programs, are the bridge between software (written in English-look-alike programming language) and machine code (consisting of 0s and 1s). They give us the means to write complex and sophisticated yet efficient algorithms in healthcare, finance, transportation (and more) using mathematical, ML and AI components, empowering today's engineers and relieving them of conceptual high-level tasks. Consequently, compiler bugs broadly impact software, and library defects affect ML and AI trustworthiness. C standard libraries give us the power to compute values of the sinuous function in just one line, and ML libraries allow us to run reinforcement learning with several lines of code. However, ensuring correct translation is complex, as it involves reasoning about the program code's connection to its machine code translation. One of the most expensive yet neglected errors is associated with the floating-point data types: essential

types representing numerical data in software, particularly vital in ML and AI implementations; these, in many cases, led to significant financial losses and jeopardised lives. Yet, testing support is often insufficient, commonly limited to the detection of logical faults in lines of code written by the user or crashes when executing machine code because of the testing mathematical code complexity.

References

[1] K. Even-Mendoza, A. Sharma, A. F. Donaldson, and C. Cadar. 2023. GrayC: Greybox Fuzzing of Compilers and Analysers for C. ISSTA 2023: 1219—1231. https://doi.org/10.1145/3597926.3598130

[2] K. Even-Mendoza, C. Cadar, and A. F. Donaldson, CSMITHEDGE: more effective compiler testing by handling undefined behaviour less conservatively. Empir Software Eng 27, 129 (2022). https://doi.org/10.1007/s10664-022-10146-1.

[3] MLighter is an ongoing project with a webpage: http://mlighter.freedevelop.org

[4] K. Even-Mendoza, A. E. J. Hyvarinen, H. Chockler and N. Sharygina: Lattice-based SMT for Program Verification. MEMOCODE 2019 : 16:1-16:11.

[5] H. Chockler, K. Even, and E. Yahav. Finding rare numerical stability errors in concurrent computations. ISSTA, 2013, pages 12—22. (alphabetic order)

[6] J. M. Zhang, M. Harman, L. Ma and Y. Liu, "Machine Learning Testing: Survey, Landscapes and Horizons", in IEEE Transactions on Software Engineering, vol. 48, no. 1, pp. 1-36, 1 Jan. 2022, doi: 10.1109/TSE.2019.2962027.

[7] J. Chen, J. Patra, M. Pradel, Y. Xiong, H. Zhang, D. Hao, and L. Zhang. 2020. A Survey of Compiler Testing. ACM Comput. Surv. 53, 1, Article 4 (January 2021), 36 pages. https://doi.org/10.1145/3363562

[8] X. Yang, Y. Chen, E. Eide, and J. Regehr. 2011. Finding and understanding bugs in C compilers. SIGPLAN Not. 46, 6 (June 2011), 283—294. https://doi.org/10.1145/1993316.1993532

[9] A. F. Donaldson, H. Evrard, and P. Thomson. 2020. Putting randomized compiler testing into production (experience report). In Proceedings of the 34th European Conference on Object-Oriented Programming (ECOOP 2020). Schloss Dagstuhl-Leibniz-Zentrum fur Informatik. https://drops.dagstuhl.de/opus/volltexte/2020/13179/

[10] V. Livinskii, D. Babokin, and J. Regehr. 2020. Random testing for C and C++ compilers with YARPGen. Proc. ACM Program. Lang. 4, OOPSLA, Article 196 (November 2020), 25 pages. https://doi.org/10.1145/3428264

[11] C. Murphy, K. Shen, and G. Kaiser. 2009. Automatic system testing of programs without test oracles. In Proceedings of the eighteenth international symposium on Software Testing and Analysis (ISSTA '09). Association for Computing Machinery, New York, NY, USA, 189—200. https://doi.org/10.1145/1572272.1572295

[12] A Google self-driving car caused a crash for the first time. 2016.

https://www.theverge.com/2016/2/29/11134344/google-self-driving-car-crash-report
[13] SGD: commonly used during the training process to update the weights of the neural network based on the gradients of the loss function with respect to the weights.
https://keras.io/api/optimizers/sgd/

The complexity of database query evaluation

Supervisor: Hubie Chen

Areas: Foundations of computing

(Back to Scholarship Allocated)

Project Description

Evaluating queries on databases is a primary means of extracting information from databases. This project aims to study facets of the complexity of and algorithms for database query evaluation. Our study will draw on tools, techniques, and ideas from areas such as decomposition methods, graph theory, logic, finite model theory, database theory, and parameterized complexity theory. For this project, a strong interest in and background in mathematical aspects of computing is expected.

References

Refer to the articles listed at https://dblp.org/pid/83/3627.html -- particularly those appearing in PODS, ICDT, and LICS.

Scholarship Not Allocated

(Back to Top)

- Software Verification and Nominal Dependent Type Theory
- Nominal Specification and Verification Environments
- Game-theoretic models in cryptoeconomics: incentives, mechanisms and blockchain dynamics
- Learning emergent behaviors in multi-agent systems: game theory and chaos dynamics for artificial intelligence
- Dealing with imperfect rationality in computational systems
- Understanding Software Security: Unveiling Vulnerabilities through Binary-based Testing Strategies
- Dense subgraph detection and breaking
- String Sanitization with Applications to Internet of Things Data
- Indexing text data: practical and (near)-optimal schemes.

Software Verification and Nominal Dependent Type Theory

Supervisor: Maribel Fernandez

Areas: Foundations of computing

(Back to Scholarship Not Allocated)

Project Description

Dependent Type Theory is a mathematical tool to write formal specifications and prove the correctness of software implementations. The proof assistants used to certify the correctness of programs (such as Coq), are based on dependently typed higher-order abstract syntax. The goal of this project is to explore alternative foundations for proof assistants using nominal techniques. The nominal approach has roots in set theory and has been successfully used to specify programming languages. This project will focus on the combination of dependent types and nominal syntax and explore the connections between the nominal approach and the higher-order syntax approach used in current proof assistants.

References

Typed Nominal Rewriting, Elliot Fairweather and Maribel Fernandez, ACM Transactions on Computational Logic, vol.19, number 1, 2018.

Nominal Specification and Verification Environments

Supervisor: Maribel Fernandez

Areas: Foundations of computing

(Back to Scholarship Not Allocated)

Project Description

Software verification techniques have been successfully used to prove correctness of low-level programs, but verification of high-level programming languages is challenging: it requires reasoning about name binding (e.g., visible/hidden channel names, scoping rules defining local and global variable names, parameter passing and substitution of values for variables). A standard approach to deal with name binding in verification tasks is to replace names with numerical codes (de Bruijn indices). While this avoids some of the difficulties of reasoning about names in programs, conducting a formalisation in de Bruijn style is labour-intensive and imposes a significant overhead to comprehending and reusing proofs. Nominal techniques offer an alternative, user-friendly approach, which does not require to replace names with codes. We aim to apply novel nominal techniques to simplify the handling of names and binders in programming languages and verification tasks (e.g., verification of blockchain languages). For this, we aim to develop a core nominal calculus and use it as a basis to enrich with nominal features two successful verification frameworks: Maude and K.

Game-theoretic models in cryptoeconomics: incentives, mechanisms and blockchain dynamics

Supervisor: Stefanos Leonardos

Areas: Machine Learning (ML), Artificial Intelligence (AI), Foundations of computing

(Back to Scholarship Not Allocated)

Project Description

This project targets students interested in advancing cutting-edge research at the intersection of game theory and cryptoeconomics. The project's aim is to model and analyze blockchain-enabled economies through a game-theoretic lens. Special focus will be placed on transaction fee markets, miner extractable value (MEV) incentives, proposer-builder separation in Ethereum block creation, MEV-boost auctions, transaction censorship, attacks in decentralized exchanges, and related phenomena. The study will explore cryptoeconomic mechanisms, dissecting participant incentives, and designing mechanisms to optimize blockchain performance. Due to the dynamic nature of these systems, the project will employ elements from algorithmic game theory and dynamical systems, alongside standard tools from economics, computer science, and machine learning. Successful candidates will develop game-theoretic models, conduct rigorous mathematical analyses, and run simulations to validate theoretical predictions in real-world applications, bridging the gap between academia and industry.

References

1. Buterin, V, Reijsbergen, D, Leonardos, S, Piliouras, G. Incentives in Ethereum's hybrid Casper protocol. Int J Network Mgmt. 2020; 30:e2098. https://doi.org/10.1002/nem.2098
2. Leonardos, S, Reijsbergen, D, Piliouras, G. Weighted voting on the blockchain: Improving consensus in proof of stake protocols. Int J Network Mgmt. 2020; 30:e2093. https://doi.org/10.1002/nem.2093
3. Leonardos, N., Leonardos, S., Piliouras, G. (2020). Oceanic Games: Centralization Risks and Incentives in Blockchain Mining. In: Pardalos, P., Kotsireas, I., Guo, Y., Knottenbelt, W. (eds) Mathematical Research for Blockchain Economy. Springer Proceedings in Business and Economics. Springer, Cham. https://doi.org/10.1007/978-3-030-37110-4_13
4. Leonardos, S., Monnot, B., Reijsbergen, D., Skoulakis, E., and Piliouras, G. (2021). Dynamical analysis of the EIP-1559 Ethereum fee market. In Proceedings of the 3rd ACM Conference on Advances in Financial Technologies (AFT '21). Association for Computing Machinery, New York, NY, USA, 114—126. https://doi.org/10.1145/3479722.3480993
5. D. Reijsbergen, S. Sridhar, B. Monnot, S. Leonardos, S. Skoulakis and G. Piliouras, "Transaction Fees on a Honeymoon: Ethereum's EIP-1559 One Month Later," 2021 IEEE International Conference on Blockchain (Blockchain), Melbourne, Australia, 2021, pp. 196-204, doi: 10.1109/Blockchain53845.2021.00034.
6. Koki, C., Leonardos, S., and Piliouras, G. (2022). Exploring the predictability of cryptocurrencies via Bayesian hidden Markov models, Research in International Business and Finance, Volume 59, 101554, doi: 10.1016/j.ribaf.2021.101554.

7. Leonardos, S., Reijsbergen, D., Monnot, B., and Piliouras, G., "Optimality Despite Chaos in Fee Markets", *arXiv e-prints*, 2022. doi:10.48550/arXiv.2212.07175.

Learning emergent behaviors in multi-agent systems: game theory and chaos dynamics for artificial intelligence

Supervisor: Stefanos Leonardos

Areas: Artificial Intelligence (AI), Machine Learning (ML), Foundations of computing

(Back to Scholarship Not Allocated)

Project Description

This project targets students who are interested in cutting-edge research at the intersection of multi-agent systems, game theory and learning dynamics, with applications in economics, machine learning, and artificial intelligence. The project's objective is to explore the intricate patterns of multi-agent systems through a game-theoretic lens, emphasizing on learning dynamics, chaos theory, and their applications. Special focus will be placed on understanding the emergent behaviors in algorithmic decision-making processes that continuously evolve over time. In this context, the study will explore phase-transitions in strategic interactions, analyze or develop novel algorithms, and quantify their implications on coordination and competition in real-world systems. The analysis will use tools from game theory, mathematics and the theory of dynamical systems, to develop, study and apply learning algorithms in complex multi-agent systems. Successful applicants will have the chance to shape the future of learning systems, bridging theoretical advancements with practical applications with the frameworks of machine learning and artificial intelligence.

References

1. Leonardos, S., and Piliouras, G. (2022). Exploration-exploitation in multi-agent learning: Catastrophe theory meets game theory, Artificial Intelligence, Volume 304, 103653, doi:10.1016/j.artint.2021.103653.
2. Leonardos, S., Piliouras, G., and Spendlove, K. (2021). Exploration-Exploitation in Multi-Agent Competition: Convergence with Bounded Rationality, in Advances in Neural Information Processing Systems, volume 34, pp. 26318--26331, Curran Associates, Inc., https://proceedings.neurips.cc/paper_files/paper/2021/file/dd1970fb03877a235d530476eb727dab-Paper.pdf.
3. Leonardos, S., Overman, W., Panageas I., and Piliouras, G. (2022). Global Convergence of Multi-Agent Policy Gradient in Markov Potential Games, in International Conference on Learning Representations (ICLR 2022), https://openreview.net/forum?id=gfwON7rAm4.
4. Leonardos, S., Sakos, J., Courcoubetis, C. and Piliouras, G. (2023). Catastrophe by Design in Population Games: A Mechanism to Destabilize Inefficient Locked-in Technologies. ACM Trans. Econ. Comput. 11, 1—2, Article 1 (June 2023), 36 pages. doi:10.1145/3583782
5. Cheung, Y.K., Leonardos, S., and Piliouras, G. (2021). Learning in Markets: Greed Leads to Chaos but Following the Price is Right. Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence Main Track, pp. 111-117, doi:10.24963/ijcai.2021/16.

Dealing with imperfect rationality in computational systems

Supervisor: Carmine Ventre

Areas: Foundations of computing, Artificial Intelligence (AI)

(Back to Scholarship Not Allocated)

Project Description

Computational/AI systems often collect their input from humans. For example, parents are asked to input their preferences over primary schools before a centralised algorithm allocates children to schools. Should the AI trust the input provided by parents who may try to game the system? Should the parents trust that the AI system has optimised their interests? Would it be safe to run the algorithm with a potentially misleading input? Algorithmic Game Theory (AGT) is a research field that attempts to add safety and trustworthiness to AI systems vis-a-vis strategic reasoning. With its set of symbolic tools, one aims to align the goals of the AI system (e.g., the allocation algorithm above) with those of the agents (e.g., the parents above) involved. The AI will then be safe, in that we can analytically predict end states of the system, and trustworthy, since no rational agent will attempt to misguide the system and the system will work on truthful inputs. One assumption underlying much of the work in AGT is, however, pretty limiting: agents need to be fully rational. This is unrealistic in many real-life scenarios; we, in fact, have empirical evidence that people often misunderstand the incentives and try to game the system even when it is against their own interest. Moreover, modern software agents, often built on top of AI tools, are seldom able to perfectly optimise their rewards. This project will look at novel approaches to deal with imperfect rationality, including analysis of known AI systems and the design of novel ones. This will involve theoretical work that builds on the recent advances on mechanism design for imperfectly rational agents (namely obvious strategyproofness and not obvious manipulability) to include more complex domains and the modelling of further behavioural biases in mechanism design. Prospective applicants are encouraged to consult the publications of Prof Ventre at https://kclpure.kcl.ac.uk/portal/en/persons/carmine.ventre/publications/.

Understanding Software Security: Unveiling Vulnerabilities through Binary-based Testing Strategies

Supervisor: Dr Karine Even-Mendoza, Dr Hector Menendez Benito

Areas: Machine Learning (ML), Foundations of computing, Cybersecurity, Systems (SE, programming, autonomous systems, robotics, ...), Computing Applications

(Back to Scholarship Not Allocated)

Project Description

Software ecosystems rely on the way operating systems distribute resources. By creating the address space, the process space, the threads and the security tokens of the running program, the system provides an execution context that changes depending on the kernel version or even the compiler used to execute the programs. The integration of a program into a systematic environment that evolves depending on kernel and compilation version might not imply security vulnerabilities, but in the presence of crashes, the exploitability of the system will directly depend on how it deals with resources, as obfuscations proved [2]. Under these conditions, there are a few strategies that can provide some light on ways to identify these vulnerabilities. The first one is to employ semantic equivalent transformations to the software and study the behavioural changes in the system. The second is to study the decompilation of the final PE or EFL file and investigate how it changes under different compilation options. The third is to employ various testing strategies, such as differential testing, to analyse how the environment is changing the execution, often tracked through profiling strategies. These three strategies will define the three parts of the thesis. Part 1: Process Resources. The student will work by extending the previous work on the security of obfuscations [2]. The extension will focus on the way the heap and the stack are affected in terms of the address space and the managed resources. With this information, the student will better understand the exploitability of specific parts of the system and work on potential mitigations that can support the system's security. Part 2: Compiler's configurations. Compilers optimise code by adding transformations that reduce the way the process collects and manipulates resources. It is also affected by the scheduler. The student will catalogue the effect of optimisations in software, especially bugs, and how they change their nature and exploitability when the system is more vulnerable. Based on these principles, the student will extrapolate the previous knowledge on exploitability to the compilers' context. Part 3: Testing improvements. Based on the previous discoveries about how the system interacts with processes and how compilers and contexts change this, this last part of the thesis focuses on changing the ways testing is applied in the context of vulnerabilities with the aim of making it more focused to unmask the risk that the context and the compiler can associate with the execution of the files.

References

[1] A. Dakhama, K. Even-Mendoza, W.B. Langdon, H. Menendez, and J. Petke (2023). SearchGEM5: Towards Reliable gem5 with Search Based Software Testing and Large Language Models. Symposium on Search Based Software Engineering (SSBSE). https://tinyurl.com/2u2aeb4r

[2] H. D. Menendez and G. Suarez-Tangil. 2022. ObfSec: Measuring the security of obfuscations from a testing perspective. Expert Syst. Appl. 210, C (Dec 2022).

https://doi.org/10.1016/j.eswa.2022.118298

[3] K. Even-Mendoza, C. Cadar, and A. F. Donaldson. 2022. Csmithedge: more effective compiler testing by handling undefined behaviour less conservatively. Empirical Softw. Engg., 27, 6, (Nov. 2022), 35 pages. doi: 10.1007/s10664-022-10146-1.

[4] X. Hou, Y. Zhao, Y. Liu, Z. Yang, K. Wang, L. Li, X. Luo, D. Lo, J. Grundy, H. Wang. Large language models for software engineering: A systematic literature review. arXiv preprint arXiv:2308.10620. 2023 Aug 21

[5] V. Le, M. Afshari, and Z. Su. 2014. Compiler validation via equivalence modulo inputs. In PLDI '14. ACM, New York, NY, USA, 216—226. https://doi.org/10.1145/2594291.2594334

[6] E. T. Barr, M. Harman, P. McMinn, M. Shahbaz and S. Yoo, "The Oracle Problem in Software Testing: A Survey," in IEEE Transactions on Software Engineering, vol. 41, no. 5, pp. 507-525, 1 May 2015, doi: 10.1109/TSE.2014.2372785.

[7] X. Yang, Y. Chen, E. Eide, and J. Regehr. 2011. Finding and understanding bugs in c compilers. In Proceedings of the 32nd ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI '11). Association for Computing Machinery, San Jose, California, USA, 283—294. isbn: 9781450306638. doi: 10.1145/199349 8.1993532.

[8] J. Regehr. Finding Bugs in C and C++ Compilers using YARPGen. SIGPLAN. PL Perspectives. https://blog.sigplan.org/2021/01/14/finding-bugs-in-c-and-c-compilers-using-yarpgen/

[9] AFL Michal Zalewski, "Technical "whitepaper" for afl-fuzz," https://lcamtuf.coredump.cx/afl/technical_details.txt

Dense subgraph detection and breaking

Supervisor: Grigorios Loukides

Areas: Data science, Foundations of computing

(Back to Scholarship Not Allocated)

Project Description

Graphs naturally model relationships between entities in domains ranging from social networks to communication networks and the web. In all these domains, a fundamental analysis task is dense subgraph discovery. This task aims at identifying parts of the graph that are cohesive. A clique, for example, is such a cohesive subgraph, while there are several other notions that relax the notion of clique, allowing for more efficient discovery. Dense subgraph discovery is important in a multitude of applications, such as detecting communities in social networks and preventing money laundering. Beyond discovery, studying how dense subgraphs may be broken with minimal graph distortion is also relevant. It is practically useful in maintaining communities in social networks, assessing resilience to attacks or errors in communication networks, and enabling social network users to prevent discrimination. The main goal of this project is to develop algorithms for detecting and for breaking dense subgraphs. There is also possibility to propose new notions of dense subgraphs, or to develop algorithms employing existing notions for complex types of graphs. Candidates with strong background in algorithm design and optimization and with excellent programming skills (preferably in C++) are welcome.

References

Huiping Chen, Alessio Conte, Roberto Grossi, Grigorios Loukides, Solon P. Pissis, and Michelle Sweering. 2021. On Breaking Truss-Based Communities. In Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining (KDD '21). Association for Computing Machinery, New York, NY, USA, 117—126
https://dl.acm.org/doi/pdf/10.1145/3447548.3467365

String Sanitization with Applications to Internet of Things Data

Supervisor: Dr Grigorios Loukides, Professor Luca Vigano

Areas: Cybersecurity, Data science, Foundations of computing, Computing Applications

(Back to Scholarship Not Allocated)

Project Description

The overall aim of the project is to develop and evaluate a robust and efficient approach that allows organisations and businesses to protect the privacy of data represented as strings. The project will consider the protection of aggregated data (event sequences), as well as string databases, and it will also address the interrelated issues of usefulness, security, and scalability. It aims to develop a methodology (model, algorithms, protocols) for sanitising (i.e., transforming) data that is: (I) privacy- preserving, by designing and applying a privacy model along with algorithms for sanitising string data. (II) Utility-preserving, by designing measures and tools for quantifying the level of usefulness of data that must be traded-off for achieving privacy. (III) Secure and scalable, by designing efficient protocols that allow multiple parties to protect their data securely and jointly. The methodology will be evaluated on data from the Internet of Things (IoT) domain.

References

Bernardini et al. Hide and Mine in Strings: Hardness, Algorithms, and Experiments. IEEE TKDE 2023. https://ieeexplore.ieee.org/document/9732522

Indexing text data: practical and (near)-optimal schemes.

Supervisor: Grigorios Loukides

Areas: Foundations of computing, Data science, Computing Applications, Natural Language Processing (NLP), Machine Learning (ML)

(Back to Scholarship Not Allocated)

Project Description

In many real-world database systems, a large fraction of the data is represented by strings: sequences of letters over some alphabet. This is because strings can easily encode data arising from different sources. It is often crucial to represent such string datasets in a compact form but also to simultaneously enable fast pattern matching queries. This is the classic text indexing problem. Unfortunately, however, most (if not all) widely-used indexes (e.g., suffix tree, suffix array, or their compressed counterparts) are not optimized for all four measures (index space, construction space, query time, construction time) simultaneously, as it is difficult to have the best of all four worlds. The topic seeks to take an important step towards designing new indexes that offer good performance in all four measures. One promising direction to do this is to explore specific application-driven special cases of the problem, such as when we have at hand a lower bound ℓ on the length of the queried patterns or when we have extra knowledge about them or the text (e.g., given by a machine learning model or text properties such as the fact that it is repetitive). The candidates should have strong knowledge in algorithms and programming (C++).

References

Ayad et al. Text Indexing for Long Patterns: Anchors are All you Need. Proceedings of the VLDB Endowment (PVLDB) 2023. https://www.vldb.org/pvldb/vol16/p2117-loukides.pdf
Loukides et al. Bidirectional String Anchors for Improved Text Indexing and Top-K Similarity Search. IEEE TKDE 2023. https://ieeexplore.ieee.org/document/10018284