Ruslan Sakevych xlionell@gmail.com
github.com/lionell 1-650-732-8050

Education

- **University of Washington** Seattle, USA
  M.S. in Computer Science; GPA: 3.93; Est. grad: 2025 (part-time)
- **Taras Shevchenko National University** Kyiv, Ukraine
  M.S. in Computer Science; GPA: 3.94
- **Taras Shevchenko National University** Kyiv, Ukraine
  B.S. in Computer Science; GPA: 3.98

Experience

- **Google** Seattle, WA
  Software Engineer *May 2020 - Present*
  - Lead development of the critical service responsible for microarchitecture selection for executables.
  - Evaluated Reinforcement Learning approaches for solving abovementioned problem and prototyped the best one.
  - Performed multiple one-off experiments that identified problematic non-cross platform code in the codebase.
  - Attempted to learn neural embeddings for the Google's build graph that would work as distance oracles.
  - Anticipated poor system integration due to volatility of core metrics used for control.
  - Prototyped time-series forecasting (SARIMA) solution to the abovementioned problem.
  - Joined Continuous Integration & Release Infrastructure (CI&R Infra) team.
  - Researched online log clustering problem and developed algorithm that outperformed state-of-the-art LogParser.
  - Performed data modeling around A/B experimentation framework that GCP uses to track VM experiments.
  - Incorporated various data sources (like Linux kernel subsystems and in-house hypervisor) into VM pipelines.
  - Joined GCE VMX Tools and Telemetry team responsible for core services on top of hypervisor and tooling around.
- **Facebook** Seattle, WA
  Software Engineer *Oct 2019 - Present*
  - Joined Test Infrastructure team which later became Continuous Integration Analytics team (CIA).
  - Designed and implemented real-time service for Probabilistic Flakiness Score computation via Bayesian inference.
  - Scaled PFS service CPU utilization 10x, latency 20x, number of replicas 7.5x via profiling/caching/parallelism.
  - Details: https://engineering.fb.com/2020/12/10/developer-tools/probabilistic-flakiness
  - Replaced statistical model with stochastic ML one for sizing(memory-wise) CI workflows.
  - New Job Sizer model reduced memory utilization by 30% and out-of-memory rate by 17%.
  - Achieved another 12% improvement in MSLE by doing feature engineering for Job Sizer model.
  - Prototyped an ML based model for the CI workflow ETA estimation.
  - It reduced MSLE by 50% and allows to control confidence levels of the prediction (Monte-Carlo dropout layer).
- **Google** Sunnyvale, CA
  Software Engineering Intern *May 2019 - Aug 2019*
  - Designed and implemented an automatic build memory regression finder that operated at Google scale.
  - Created a developer dashboard for troubleshooting memory regression issues.
  - Was able to automatically(via dashboard) pinpoint culprit changes for recent major regressions.

- **Google** Sunnyvale, CA
  Software Engineering Intern *Aug 2018 - Nov 2018*
  - Migrated old ML pipeline onto Tensorflow-backed framework TFX. Experience with data processing pipelines.
  - Implemented parallel N-ary search algorithm. Sped culprit finder up 16x times (on millions of changes).
  - Designed more sophisticated and robust parallel batching algorithm. Reduced tail request latency 3x times.
  - Mined build graph of the whole Google using MapReduce. Did an attack on dependency set similarity problem.
- **Facebook** London, UK
  Software Engineer Intern *Jan 2018 - Mar 2018*
  - Rearchitected Hack parser to be reactive, allowing parsing to be inlined with the computation of the result.
  - 25% parse time reduction for the Hack type-checker (using most of the file contents) on the full-codebase.
  - Up to 50% speed up for tools that use less information(facts extraction) on hundreds of thousands of files.
  - Developed a toolset to analyze and remove unnecessary build dependencies, resulting in 2x speed up.
- **Microsoft** Redmond, WA
  Software Developer Intern *Jul 2017 - Oct 2017*
  - Engineered a new workflow to automate raw telemetry data aggregation and transformation.
  - System monitors execution of user-defined query and publishes results back to data warehouse.
  - Used for intermediate metrics aggregation to reduce data volumes and speed up queries.
- **Google** Sunnyvale, CA
  Software Engineering Intern *Apr 2017 - Jul 2017*
  - Research on build/test time prediction. Performed data analysis, model evaluation and feature engineering.
  - Created tools for ML models debugging/visualization and core service efficiency evaluation.
  - Investigated and mitigated incidents in complex build infrastructure at Google scale.
- **Google** Mountain View, CA
  Software Engineering Intern *May 2016 - Aug 2016*
  - Engineered a service that clasterizes build targets to reduce overall resources usage.
  - Performed evaluation of different batching strategies: memory, run-time optimization.
  - Trained ML models to predict build memory usage and avoid out of memory errors.

Skills

- **ML**: DNN, NLP, Stochastic models, Bayesian inference, Tensorflow, TFX, PyTorch, PyTorch Lightning
- **Systems**: Distributed systems, Build/Test infrastructure, CI internals, Build systems, Programming languages
- **Languages**: Python, Java, C/C++, Go, PHP/Hack, OCaml, C#, SQL, Bash, TypeScript, JavaScript
- **Containers**: Docker, Docker Swarm, Kubernetes, RunC, Linux namespaces, Google's Borg, Facebook's Tupperware
- **Data processing**: MapReduce, Apache Beam, Hive, Hadoop, Google Flume, Facebook DataSwarm, Google Cloud SQL, Spark SQL, Microsoft Kusto
- **Misc.**: Algorithms & Data structures, Linux internals, Capture the Flag competitions, Competitive programming

Projects

- **PARCS**: Communication Sequential Processes (CSP) like approach for language agnostic distributed computing.
- **PARCS autodiscovery**: LAN service autodiscovery for nodes in PARCS cluster based on UDP broadcasting.
- **Smart Pacmans**: Training NN using genetic algorithms (no gradient). See https://lionell.github.io/smart-pacmans
- **Resolution Theorem Prover**: First-order logic theorem prover that utilizes sequential method under the hood.
- **Huffman+RLE**: Small (1% on average) optimization for Huffman algorithm followed by run-length-encoding.
- **Aqua Lang**: Imperative data processing language that uses raw relational algebra operators. Opposite to SQL (declarative).
- **Parallel PageRank**: Implemented and compared PageRank on top of MPI and OpenMP when applied to Wikipedia page graph.