Erik-Cristian Seulean

Email : erikseulean@gmail.com
Mobile : +44-780-496-3809
linkedin.com/in/erikseulean/

Education

- **University of St. Andrews** — United Kingdom
  MSc Statistics — *Distinction, Deans' List, grade 17.9/20, September 2022*
  - **Modules**: Markov Chains, Bayesian Statistics, Multivariate Analysis, Statistical Modelling using GLM/GAMs, Quantitative risk analysis, Machine Learning
  - **Dissertation**: An Ensemble of HMM and KNN for signal generation in horse betting markets. Grade 19.3/20 - Highest grade in my cohort
- **University of Sheffield** — Remote, United Kingdom
  Graduate Certificate in Statistics — *Distinction, grade 90/100 July 2021*
  - **Modules**: Mathematical Methods for Statistics, Probability and Probability Distributions, Basic Statistics
- **Politechnica University of Timisoara** — Timisoara, Romania
  B.S.E in Computer Science — *July, 2014*
  - **Modules**: Software Engineering, Computer Architecture, Algorithms, Software Development Fundamentals, Design Patterns, Programming Languages, Databases

Experience

- **Bloomberg LP** — London, United Kingdom
  Senior Software Engineer — *2019 - September 2021*
  - **Taxonomy and Metadata System**: Infrastructure team developing a taxonomy and metadata system to support various asset classes at Bloomberg. Python, GraphQL, Redis, RabbitMQ, Solr, RDF, PostgreSQL, Data Pipeline
    - **Architected ingestion pipelines** that handle ingestion of metadata for datasets such as commodity index tickers.
      - **Add support for idxmddb tickers, and help with the migration. Make sure that tickers originating in both databases are fine, have the option to have tickers that are duplicated but not released, to help engineers check them.**
      - **Add extra storage to support extra usecases available and reduce the pressure on the existing solutions. Research on the best way to store the data and find out how to store the data to get the best performance.**
      - **Rework to move slow ingestion parts offline and preprocessed. (Eg. move expandsources from being a step in the ingestion pipeline to being done offline.)**
    - Added **support for sharding** to increase the speed of the ingestion pipeline. Split up the initial ingestion into 8 shards that are running distributed across an entire cluster.
      - **Initially, the ingestion was done based on one RDF file, which deemed slow. I added support for sharding, which means that a file was split into multiple shards that could be ingested by multiple processes running in parallel on different machines.**
    - Implemented node overlay and concatenation executed on RDF graphs.
    - Worked on **rearchitecting the ingestion pipeline** from a synchronous pipeline to a file based ingestion.
      - **Initially everything in the pipeline was synchronous, using RabbitMQ. Messages would fly from one worker to another through a queue. This was problematic over time as sometimes workers ended up being slower and queues would fill up. Generally the team was slowed down by this because they had to take into account the performance concerns of everything that they were working on. The idea was to make every worker instead of write to a queue,**

output a file, with all the changes and have the next worker in line pick up that file. The end goal (Golden city) was to have this working using notifications.

- ∗ Migrated the codebase from Python 2 to Python 3 **decreasing the overall response time of services by 15%**
  - · **Faster due to Infrastructure team reworking the services and the way the objects were being deserialized in python. For the python 2 version, every parameter on an object required a round trip call into C code, while for python 3, everything was deserialized with one single call to C code.**
- ∗ Implemented services in GraphQL to avoid overfetching and underfetching on the existing Bloomberg specific microservice framework.
  - · **The RPC calls that we were making would always either fetch more data or the user would be required to make more than one call to actually get the data. Change this by implementing some services in GraphQL rather than RPC and allow the user to specify exactly what they need.**
  - · **Remove the need for client bespoke queries.**
- ∗ Led **intergration of the metadata system with timeseries datasets**, a company wide effort to support non-tickerized data.
  - · **Separate team that was formed with me (part of the taxonomy team) and two others part of the data team. The goal was to get the nonsecuritized data into athena (datasets) and timeseries that were not tickerized.**
- ∗ Worked on a side project, debugging tool, to help find metadata for given entities regardless of the source of provenience, using a **topological sort algorithm on service APIs and RDF Graphs**.
  - · **Given an entity, and all the existing sources of data, find everything that you can about that entity, from various systems that we own and expand further. This helps find discrepancies between the data or missing data. Uses**
- ∗ Led the **research into adoption of triplestores** at Bloomberg, an effort across 6 departments towards choosing the right solution to cover the usecases presented at Bloomberg. Researched on solutions such as Virtuoso or TerminusDB.
  - · **Various teams having data in RDF format but generally no good storage for it.**
  - · **Initially consider only open-source free solutions and see if they work.**
  - · **Split the work in a team of 7, research different solutions.**
  - · **Virtuoso was not good enough because (the open source version) didn't support replication and this is not a valid SLA for a datastorage at Bloomberg.**
  - · **TerminusDB loads everything in memory to achieve good query performances, but this solution wasn't useful either.**
- ∗ Organized weekly department-wide design discussions, to share knowledge regarding ongoing work.
- ∗ Team lead backup for a team of 7 engineers.

- **Bloomberg LP** London, United Kingdom
  Software Engineer *2016 - 2019*
  - ○ **Real Time Analytics**: Application team, maintaining a large variety of analytics, integrated in different products across Bloomberg. Processing several billion trades per day. C++, Distributed, Realtime
    - ∗ Implemented Average Volume at Time for non equities and Top Exchanges in the past X minutes.
      - · **Average volume at time - for a request with given time interval and a ticker, bucketize based on the time interval starting from the beginning of the trading day and find the trades that felt in each interval, and calculate the average volume.**

            - Implemented a **load testing facility** that is used to simulate the impact of market open on the system to assess performance and capacity. I identified a bottleneck in the system and by fixing it, subscriptions are being processed 3 times faster.
                - **At the begining of the day, when trading hours open, we were subscribing to tickerplant data. This subscriptions required a request sent on our end and we realised they are slow. The reason was that the service handling the request was dispatching one request at a time when it had capacity to use more threads in the threadpool.**
            - **Increased availability** of an analytic by implementing fallback to disk in case the database connection cannot be established. This prevents the 1 minute downtime per week when the database cluster is being rebooted.
        - **Recommendation Disclosures and Trade Ideas**: Python, Typescript, Microservices, Event Sourcing
            - Implemented asynchronous messaging using RabbitMQ to allow distribution of recommendations. Added support to publish messages, implemented consumers and set up the middleware.
                - **Share a recommendation with others.**
            - Reduced the codebase by 8% by removing around 100.000 lines of dead or unused code.

Skills

- **Coding**
    - **attrs-strict https://github.com/bloomberg/attrs-strict**: Created and open-sourced a library that allows runtime validation for attrs dataclasses in Python.
    - **bloomberg-bas-middleware**: Maintainer and creator of an inner source library used for Python services at Bloomberg. Created Flask-like middlewares that are used in over 10.000 services running in Python 3 at Bloomberg. Any new service has by default the functionality opted-in.
        - **Early adopter of new Python 3 services. This made me think that we could standardize best practices, given that we were expecting all the future services in the company to change to Python 3 and use these services.**
        - **Added plug-in functionalities, such as request logging and timing**
        - **Uncaught exception handling**
        - **Some syntactic sugar improvements**
        - **Create a dependency injection manager that allows testing the handlers by injecting mock resouce instances (eg. database connections).**
        - **Leveraged Python's MRO(method resolution order) and allowed independent components to be composed together**
        - **By default newly created services have some of the features enabled by default while the others are opt-in.**
        - **Latest work on supporting asynchronous logging of events (either through a separate thread or using some hub existing for c++).**
    - **Python Guild**: Part of a group of 18 engineers leading the direction and adoption of Python at Bloomberg, led the Design Reviews Working group, helping teams to bring projects to a broader audience.
    - **ML 101**: Participated in a 6 month hands on Machine Learning training. The timeseries clustering project developed in a team of 3 was one of the 2 projects out of 15 to be showcased in front of senior management at Bloomberg.
- **Public speaking for large audiences**
    - **"How to migratate services to Python 3"** - Attendance of over 200 engineers.
    - **"Modeling the commodity world"** - Presentation on the architecture of the system I worked on and the different architectural decisions taken in the past years. Attendance of over 150 engineers.
- **Mentorship**
    - Mentored in workshops to help engineers migrate their services to Python 3.
    - Mentored an intern during the summer internship program.

    - Mentored new joiners in the team and helped people contribute to inner-source projects at Bloomberg

- **Architecture and technology**: Large scale distributed systems, Data pipelines, Microservices, GraphQL, PostgreSQL, Redis, Solr, RabbitMQ, Event Sourcing, Git, RDF
- **Languages**: Python, Javascript, Typescript, SQL, C#
- **Mathematics**: Probability theory, Markov chains, Bayesian models, GLMs, statistical inference
- **Machine Learning**: Regression, Trees, Ensemble Methods, Boosting, Neural networks, unsupervised learning
- **Languages spoken**: I speak English(IELTS Grade 8.0) and Romanian fluently. Currently living in London.

Personal Projects

- **G-Research speed coding challenge**: (October 2021) Second place in Scotland at the university coding challenge organised by G-Research. Competed against teams of 4.
- **Cryptocurrency analysis https://github.com/erikseulean/cryptoscraping**: (January 2018) Scraped cryptocurrency data from different sources and created a library that allows to easily analyse prices, market cap and volume. I implemented and plotted the Ichimoku indicator and created correlation plots between google trends and different cryptocurrencies. Python: Numpy, Pandas, Matplotlib, Multiprocessing, Jupyter
- **ColorBlocks**: (November 2015) Awarded 2nd prize competing against 12 teams at HackTM, the largest Hackathon in Eastern Europe lasting 50 hours straight. I was part of a team of three and we developed a cross-platform puzzle game for mobile and tablet devices. Available on Google Play as ColorBlocks. LUA
- **Cycling Community Platform**: (October 2012) Awarded 2nd prize competing against 20 teams in Microsoft Excite, three week competition. Developed a mobile and web based platform for the local cycling community. I was part of a team of three students. C#.NET, Javascript, SQL