A Configurable Streaming Prediction System: a Use-Case in End-of-Life.

Vincent Major, Jager Hartman, Yin Aphinyanaphongs
Department of Population Health, NYU Langone Health

FAC Symposium, Stanford, CA.
September 18, 2019

Background

Timely notification of risk delivered to an experienced physician can prompt transition to a high-risk pathway. However, there are a number of critical challenges to solve:

- The intervention must be proven effective,
- Risk estimation must be accurate, and
- Timely delivered to the care team.

System architecture tends to be an after-thought however, the underlying system is crucial to operationalize a system. Some literature exists that examine requirements on execution time and computational resources, for example event stream processing [1]. However, applications in healthcare are very limited despite the unique constraints.

Use-Case

We present a configurable streaming prediction system within a use-case of predicting short-term end-of-life. The pathway of interventions for patients at risk of dying can involve supportive care, palliative care, and hospice, depending on the individual patient's wishes. Unfortunately, these interventions are often initiated too late [2]. Instead, predictions may prompt the care team to consider the bigger picture, consult with the team, and evaluate the care plan.

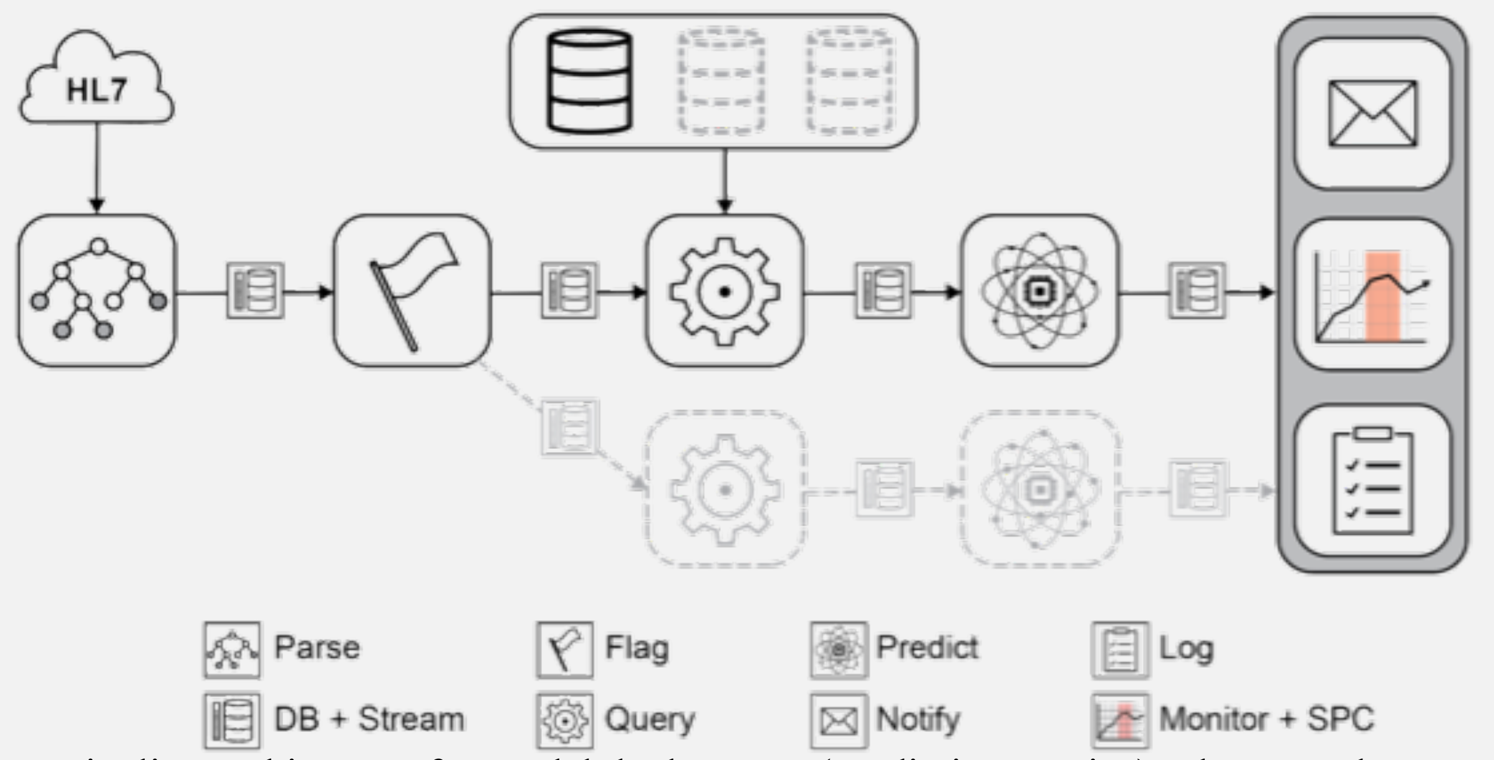

Figure 1. Current pipeline architecture for model deployment (prediction service). The greyed out services represent the modularity of this workflow—new models and new data sources can be added with ease.

Service	Throughput
Parse	8,029,217
Flag	873,429
Query	45,819
Predict	39,108
Notify	88

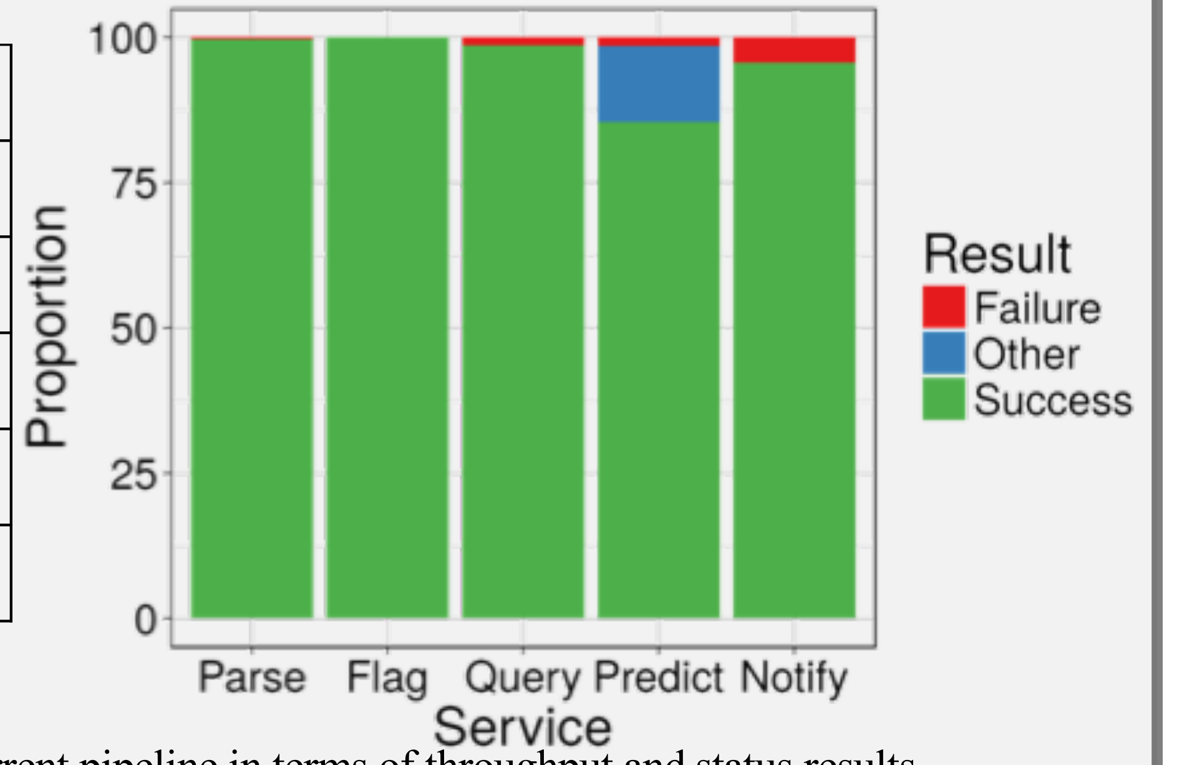

Table 1 and Figure 2. Performance of current pipeline in terms of throughput and status results.

Methods

Overview

We present a streaming prediction infrastructure consisting of containerized microservices (built as Docker containers), each connected with a message queue. The core pipeline consists of 5 steps:

1. **parse,**
2. **flag,**
3. **query,**
4. **predict,**
5. **notify.**

Pipeline

1. HL7 ADT messages are **parsed** for patient-level IDs and message type.
2. Each parsed message is compared against a buffer of recent messages to **flag** newly admitted patients.
3. **Queries** patient data including demographics, encounters, diagnoses, procedures, medications, and labs.
4. Raw data is processed, fragmented into time bins, and aggregated into features enabling **prediction**.
5. High-risk cases (predicted probability > threshold) are pushed into the EMR to **notify** the care team.

Microservices

Microservices enable more simple configuration and continuous integration, as each services is independently constructed from the others. In addition, microservices can easily scale to demand with parallelization or to accommodate alternative workflows (for example, multiple prediction models running concurrently; Figure 1).

Streaming Message Queue

We employ message queues to connect each microservice in our pipeline using Mongo streams. Each message links two microservices. The microservices communicate via messages where the parent service publishes and the child service subscribes to a message queue. Also the messages are stored to enable "playback" , monitoring tools for general system function, and model performance monitoring.

Infrastructure Requirements

The external requirements of such a workflow include:

1. a real-time source of 'trigger' messages,
2. a database of patient data,
3. a method to push a piece of data into the EMR, and
4. a predictive model with preprocessing code.

Naturally, the exact requirements vary but, in our use-case of prediction at admission, we leverage:

1. a real-time source of patient movements (HL7 ADT messages)
2. a clinical data warehouse,
3. a web service that pushes a value into a custom field in the EMR, and
4. a previously learned and validated end-of-life model learnt on retrospective data.

Results

Pipeline

The prediction and monitoring infrastructure are depicted in Figure 1. Each step represents a microservice built as a Docker image which are limited to 1 core and 128–256 MB of RAM each. The system has been live for 8.5 months and we have observed no need to parallelize any service due to saturation.

Execution Time

Considering the 8.5 months spanning October 18, 2019 to June 30, 2019, the median [IQR] total execution time is 1.3 [0.88, 19.8] minutes. The vast majority of the total execution time is the query service, followed by the predict service, where the other three have 75th percentile execution-times under 5 seconds.

Throughput

Considering the same 8.5 months, the throughput of each service is described in Table 1. Since each message triggers the next service only if certain criteria are met, the throughput degrades through the system. The parse service parses over 900,000 messages a month whereas the predict service only completes 4,600 a month.

Status results

Omitting messages that fail to meet inclusion criteria, each service, except notify, has reported a failure rate less than 2%, with notify less than 5% as depicted in Figure 2. The query and predict services can fail with the absence of necessary data and the notify service can fail due to web service outages.

Database Downtime

Any clinical database inevitably has downtime. In the case of clinical data warehouses, nightly downtime aggregates new data from the clinical database. In our case, this process typically takes ~6 hours starting shortly after midnight. During this time execution and read locks prevent connection and queries so incoming events queue up and are executed in order once the database resumes. The run-through typically finishes within 30 minutes.

Conclusion

We present a streaming architecture that can deliver predictions in near real-time. Containerized microservices connected with a message queue of Mongo streams has proven to be a powerful method to separate processing steps required for deployment of machine learning models. Moreover, the typical execution time within minutes with < 5% failure rate satisfies the constraints of our application.

The described architecture is mostly application agnostic and should fit many different predictive analytics workflows. The three external data connections are relatively common in commercial EMR systems and should be easily replicable at other institutions. At this time, several auxiliary models are being implemented in the same pipeline with minor adaptations and we expect to scale the infrastructure to other projects.

Discussion

Pipeline

This architecture is scalable, modular, and sufficiently fast for our use-case. Moreover, additional models can be added to this pipeline by adding auxiliary workflow branches as described in Figure 1, that are triggered at different timepoints, for different patients, using different data or different models. Our approach also allows for data to be "replayed" for simulation. We plan to utilize similar infrastructures for testing and implementation of additional applications of machine learning into the clinical workflow.

Limitations

The major bottleneck of this application was the **query** service. This service was the slowest service on average and the clinical database also experienced downtime due to nightly Extract-Transform-Load (ETL) jobs and regular maintenance.

Outputs

There are two outputs of the system: logs and notifications. Monitoring of the produced logs enables analysis at both the system level using statistical process control and patient level evaluation of predictive discrimination [3]. The notifications are the key output that drives the downstream clinical workflow. Improved clinical outcomes requires additional downstream consideration of the format, timing, person, channel, workflow, intervention, and intervention efficacy [4, 5].

Next Steps

We plan to add auxiliary models to this pipeline, the first will have a delay to incorporate some data from the day of admission, the second will use only text data to enable predictions on all patients regardless of their structured clinical history. We also plan to update models for improved performance and generalization to other hospitals in our system.

References

1. Duenas, J. C., Navarro, J. M., Parada G., H. A., Andion, J. & Cuadrado, F. Applying Event Stream Processing to Network Online Failure Prediction. *IEEE Commun. Mag.* **56**, 166–170 (2018).
2. Hui, D. *et al.* Impact of timing and setting of palliative care referral on quality of end-of-life care in cancer patients. *Cancer* **120**, 1743–1749 (2014).
3. Chen, I., Johansson, F. D. & Sontag, D. Why Is My Classifier Discriminatory? in *Advances in Neural Information Processing Systems 31* (eds. Bengio, S. et al.) 3539–3550 (Curran Associates, Inc., 2018).
4. Bates, D. W. *et al.* Ten Commandments for Effective Clinical Decision Support: Making the Practice of Evidence-based Medicine a Reality. *Journal of the American Medical Informatics Association* **10**, 523–530 (2003).
5. Liu, J., Wyatt, J. C. & Altman, D. G. Decision tools in health care: focus on the problem, not the solution. *BMC Med. Inform. Decis. Mak.* **6**, 4 (2006).