Module 8 Discussion:

***Recommender systems come in all shapes and sizes regarding both data and algorithmic requirements.***

***Provide use cases of two distinct recommendation systems: one that is well suited for batch processing and another that is well suited for stream processing. What distinguishes these systems in terms of their implementation requirements? What software architecture or stack would you recommend for each, and why?***

***If you have examples from your experience, please provide them to illustrate your argument.***

A recommender system that might be well served by batch processing is a recommendation engine for doctors. It can load batches of doctor reviews, wait times, pricing information, and other factors in on a daily, weekly, or monthly basis and help give recommendations to people searching for a specialist – I worked on a project revolving around that for a major payer in the southern US in 2019.

A good recommender system for a streaming process might be finding connections on LinkedIn or Facebook – places where quality connections can lead to higher use of the platform. When you select “Add Connection” or “Add Friend” on LinkedIn or Facebook respectively, oftentimes you will get a recommendation for other people with whom you want to connect. That’s based on a real-time (streaming) recommendation saying “you liked this person, other people with similar connections to you both might be Person A”, and recommend person A as a new LinkedIn connection or Facebook friend.

The batch processing tends to have more lax requirements since it can be done as an entire batch – it can take advantage of the MapReduce framework and has discrete blocks of data to work with. I might recommend a multi-node system for batch processing that partitions data. I am partial to cloud architecture, so I might recommend using AWS or GCP for batch processing data, specifically GCP’s Dataflow or AWS Batch.

Streaming processing must take into account that there are no “finished” blocks of data, so must account for this by using a log or some other such checkpoint mechanism to ensure there is a point to revert to from a failure (which we should treat as inevitable instead of as a possibility). Apache Flink can be deployed using AWS Kinesis and AWS EMR (their cloud-based MapReduce offering).