Samantha - the Engine of Data-Driven Intelligent Systems

Version 1.0.0, Qian@GroupLens

Introduction

- what's samantha
  - A generic recommender and predictor server. Machine learning is the core, but much more than that, particularly for the purpose of recommendation/ranking/candidate generation/evaluation.
  - MIT licence, oriented to production use (online field experiments in research and typical industrial use)
- the amazing features of samantha, what samantha can do in simple words: full-fledged, self-contained server that can be used in production right away with one configuration file
  - data management, including offline and online, in (indexing) and out (post-processing), bootstrap once for all
  - model management, abstract model operations
  - data processing pipeline, data expanding and feature extraction framework
  - state-of-the-art models: collaborative filtering, matrix factorization, knn, trees, boosting and reinforcement
  - experimental framework for A/B and bucket testing
  - feedback (for online learning/optimization) and evaluation (for experimenting) loops among application front-end, application back-end server and samantha
  - abstract model parameter server, i.e. extensible variable and index spaces
  - generic oracle-based optimization framework/solver with classic solvers
  - flexible model dependency, e.g. model ensemble, stacking, boosting
  - schedulers for regular model rebuilding or backup
  - control and customize all these components through one centralized configuration file
- why do I develop samantha
  - user-centered research
  - offline to online trend
  - lack of good ones, many but no ideal
- the goal (what to support in the near future): the engine of data-driven intelligent systems
  - text modeling
  - model chaining/composition
  - scale up to clusters: learning algorithm and offline data storage
- philosophy of samantha design: "minimum viable product"

Interfaces, Components, and Data Flow

Samantha's capabilities are provided to clients through two types of HTTP interfaces/requests. This is the very front/boundary of samantha, through which the outside systems interact with. One type is *management* requests, including model management and configuration management. The other type -

realtime indexing/serving requests - provides recommendation or prediction and the capability of indexing data in real-time. Figure 1 illustrates the processing flow of samantha receiving different types of requests.

1. global configuration management

GET /config
POST /config/reload

2. model management

POST /$engine/predictor/model
POST /$engine/retriever/model
POST /$engine/ranker/model
POST /$engine/evaluate

3. recommendation and prediction serving

POST /$engine/recommendation
POST /$engine/prediction

4. data indexing

POST /$engine/index/data

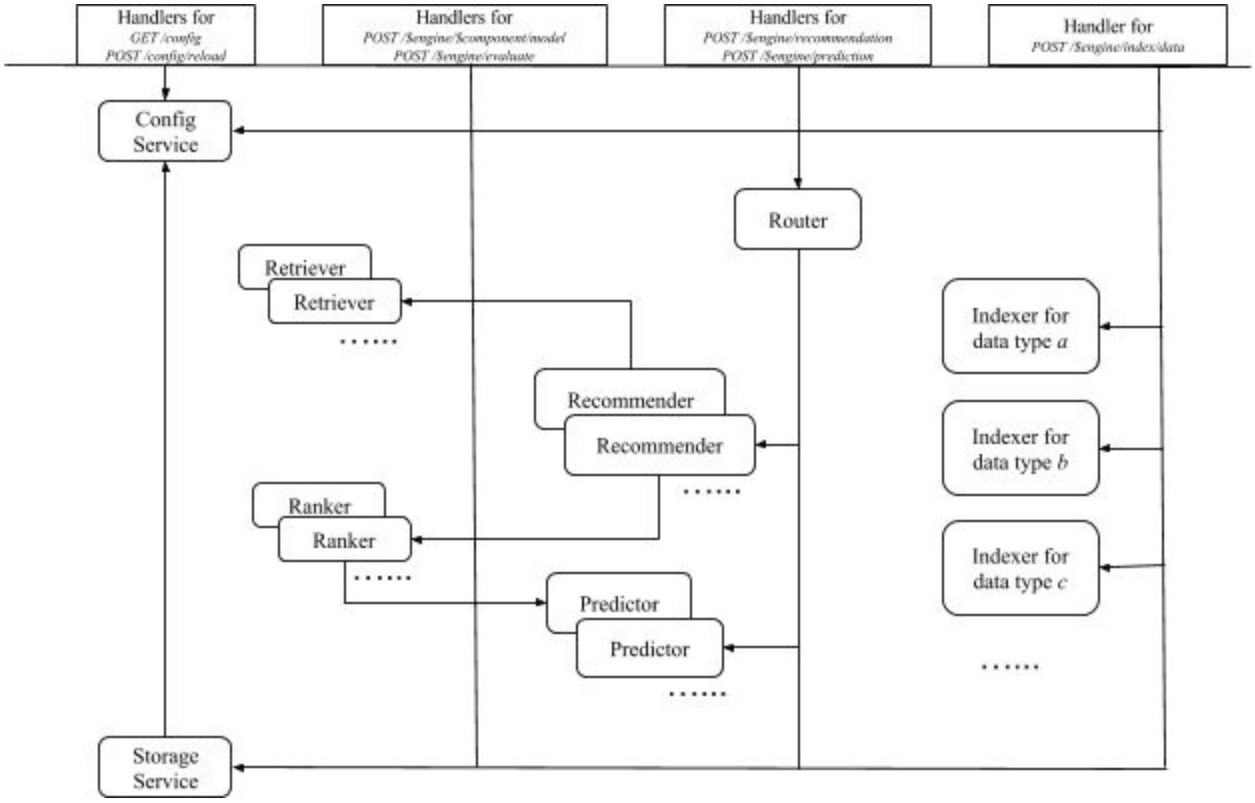

Figure 1. Samantha Data/Request Processing Flows

Design Patterns/Principles

- dependency injection
    - strongly and weakly follow the pattern

    - inject as needed especially for services
    - actual working class and config class
- singleton
- server-oriented and be mindful of the running phase
    - system initiation
    - request serving

Current Technology Stack

- java, play framework
    - configuration
    - dynamic loading and compiling
    - sbt/scala building
    - handlers vs. system logics
- http server/service-oriented programming
- JSON
- Redis and Elasticsearch
- Database and local file-based storage (extensible to distributed file storage, e.g. HDFS)

Related Work

- compare with recommender tools, lenskit, mahout, libfm etc.
- compare with machine learning tools, weka, libsvm, scikit-learn etc.
- compare with machine learning systems: xgboost, mxnet, etc.
- compare with machine learning server prediction.io

Dive into Samantha

Easy/Minimum Setup (*play in activator*)

Engines and Key Components

- recommender vs. predictor engine
- predictor engine component
    - indexer
    - retriever
    - indexer
    - router
    - evaluator
    - scheduler
- additional recommender component
    - ranker

    - recommender

Indexers for Data In/Out

- JSON, DAO
- real-time data and offline data indexing
- grouped data points output

Retriever for Candidate Generation

Learning Data, Learning Model and Optimization Solver

- oracle-based optimization framework
- parametric model
- loss function: normal pointwise loss and listwise ranking loss
- (online) optimization method using stochastic oracles

Light-weight Feature Extraction Framework

Heavy-weight Data Expanding

Model Service and Management

- model defaults

Flexible Component and Model Dependency

- Component Construction
    - system initiation phase
    - request serving phase
- model ensemble/stacking
- recommender depending on another recommender/reinforcement learning

Evaluator and A/B or Bucket Testing

- Metrics
- Experiment Group Assignment/Tagging

Online Learning and Evaluation Loops

- group tagging
- from data to (feature extracted) instance

Scheduler for Offline Jobs

Extensibility

Distributed Protocol Design and Lightweight Core

Shallow Dependencies/Flattened Software Layers

Extended Interface Protocol through Request

Scalability

Index/Variable Space (Parameter Server)

Extensible DAO

Redis and Elasticsearch Integration

Summary: Production Ready with One Configuration File

Case Studies/Examples

Model and Algorithm Research

- feature-level model ensemble
- model-level model ensemble
- boosting and GB-CENT

Application Integration and User Research

- main integration threads
    - prediction/recommendation request
    - data indexing request
- once-for-all bootstrap
- A/B or bucket testing (between-subjects assignment)

Integrating with Other Systems (Extension/Plugin)

- xgboost
- lenskit
- wikibrain

- etc.

Conclusion and Outlook

- what samantha provides
- what's the roadmap

Acknowledgement

Appendix

Installation/Setup

List of Models/Algorithms

Dive into Classes/Interfaces

References