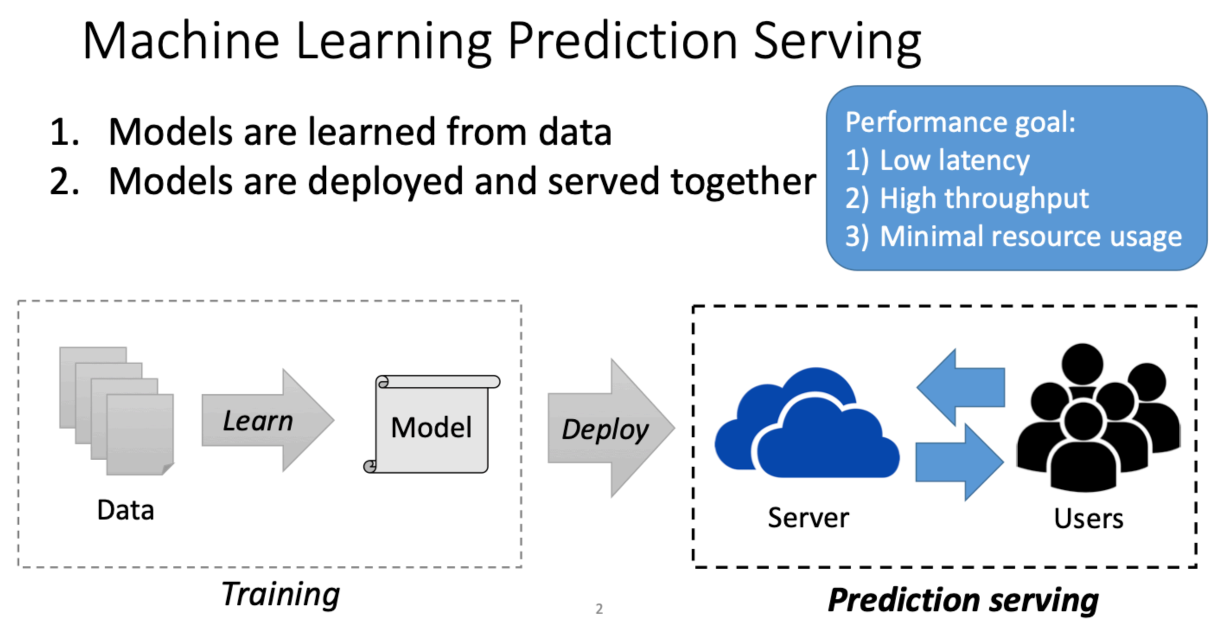
Lecture 8 - Efficient model serving



Machine Learning Prediction Serving

1. Models are learned from data
2. Models are deployed and served together

Performance goal:
1) Low latency
2) High throughput
3) Minimal resource usage

Data

Learn

Model

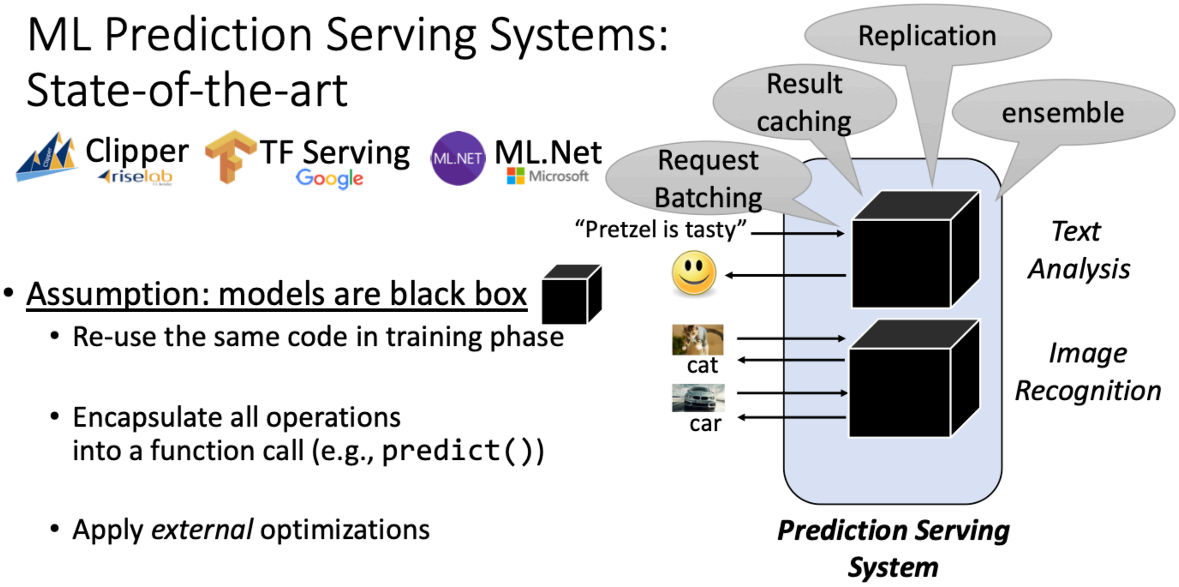
Training

Deploy

Server

Users

Prediction serving



ML Prediction Serving Systems: State-of-the-art

Clipper riselab TF Serving Google ML.Net Microsoft

- Assumption: models are black box
  - Re-use the same code in training phase
  - Encapsulate all operations into a function call (e.g., `predict()`)
  - Apply *external* optimizations

Replication

Result caching

ensemble

Request Batching

"Pretzel is tasty"

cat

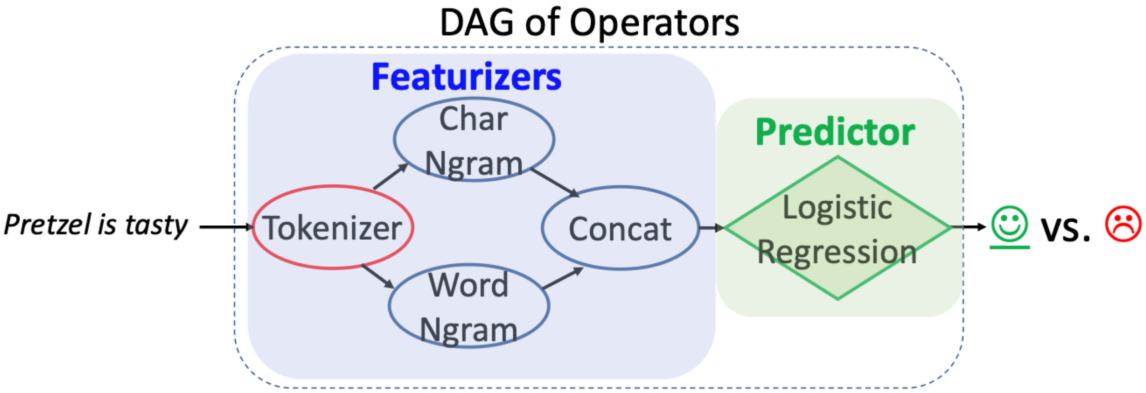
car

Text Analysis

Image Recognition

Prediction Serving System

How do Models Look inside Boxes?



DAG of Operators

Featurizers

Predictor

Pretzel is tasty

Tokenizer

Char Ngram

Word Ngram

Concat

Logistic Regression

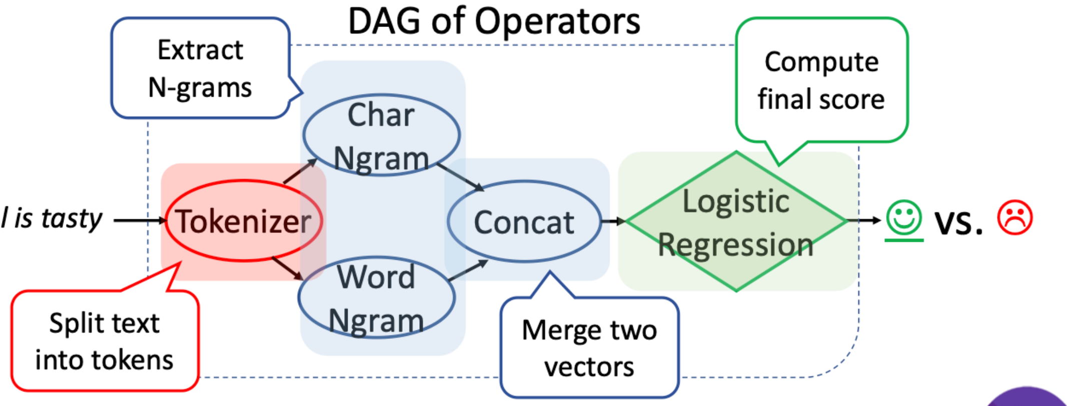
☺ vs. ☹

<Example: Sentiment Analysis>

ML.NET

Microsoft

How do Models Look inside Boxes?



DAG of Operators

Extract N-grams

Compute final score

Pretzel is tasty

Tokenizer

Char Ngram

Word Ngram

Concat

Logistic Regression

☺ vs. ☹

Split text into tokens

Merge two vectors

<Example: Sentiment Analysis>

ML.NET

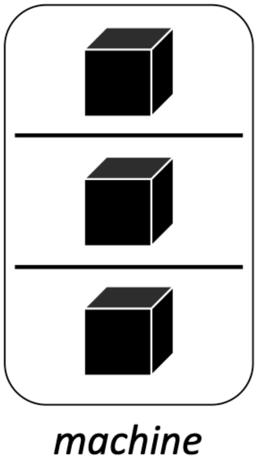
Microsoft

Limitation 1: Resource Waste

- Resources are isolated across Black boxes

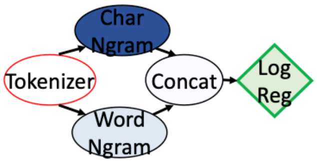
1. Unable to share memory space
   ➔ Waste memory to maintain duplicate objects
   (despite similarities between models)

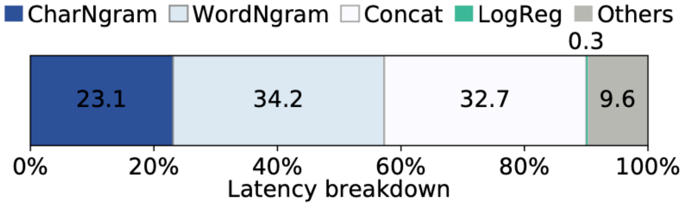
2. No coordination for CPU resources between boxes
   ➔ Serving many models can use too many threads

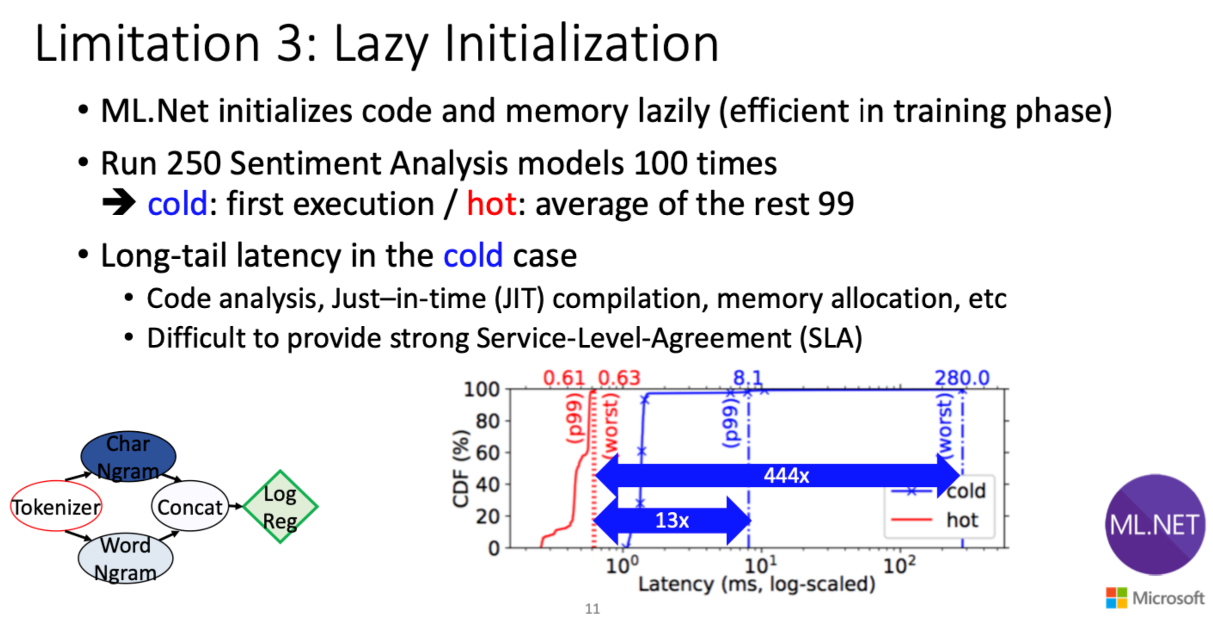


Limitation 2: Inconsideration for Ops' Characteristics

1. Operators have different performance characteristics
   - `Concat` materializes a vector
   - LogReg takes only 0.3% (contrary to the training phase)
2. There can be a better plan if such characteristics are considered
   - Re-use the existing vectors
   - Apply in-place update in LogReg





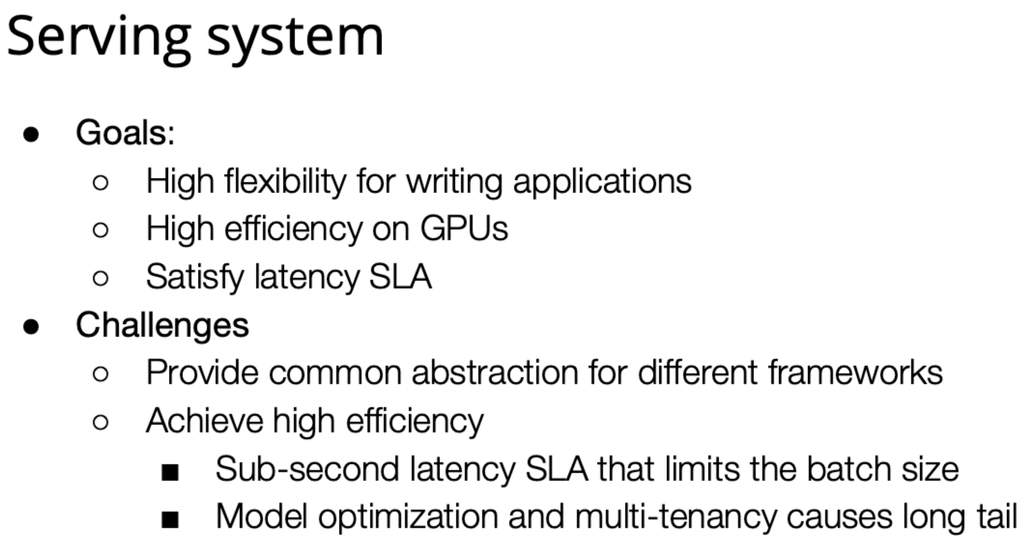


Limitation 3: Lazy Initialization

- ML.Net initializes code and memory lazily (efficient in training phase)
- Run 250 Sentiment Analysis models 100 times
  ➔ cold: first execution / hot: average of the rest 99
- Long-tail latency in the cold case
  - Code analysis, Just–in-time (JIT) compilation, memory allocation, etc
  - Difficult to provide strong Service-Level-Agreement (SLA)

11

Requirements of a serving system



Serving system

- Goals:
  - High flexibility for writing applications
  - High efficiency on GPUs
  - Satisfy latency SLA
- Challenges
  - Provide common abstraction for different frameworks
  - Achieve high efficiency
    - Sub-second latency SLA that limits the batch size
    - Model optimization and multi-tenancy causes long tail

Cascades:

Reading **Rapid Object Detection using a Boosted Cascade of Simple Features** **https://www.cs.cmu.edu/~efros/courses/LBMV07/Papers/viola-cvpr-01.pdf**

Talk about Cascading classifiers:

https://cs.nyu.edu/courses/fall12/CSCI-GA.2560-001/FaceRecognitionBoosting.pdf

Then talk about Willump
https://mlsys.org/media/Slides/mlsys/2020/balla(02-14-30)-02-15-45-1416-willump_a_stat.pdf