ML Design Pattern: Ranking

Geoff Hulten

Setup for a ranking system

- Goal of Classification – find correct label
- Goal of Regression – predict correct number
- Goal of Ranking – sort samples in correct order
  - Pointwise – regression for relevance score
  - Pairwise – which response is better
  - Listwise – 1 to N ranking

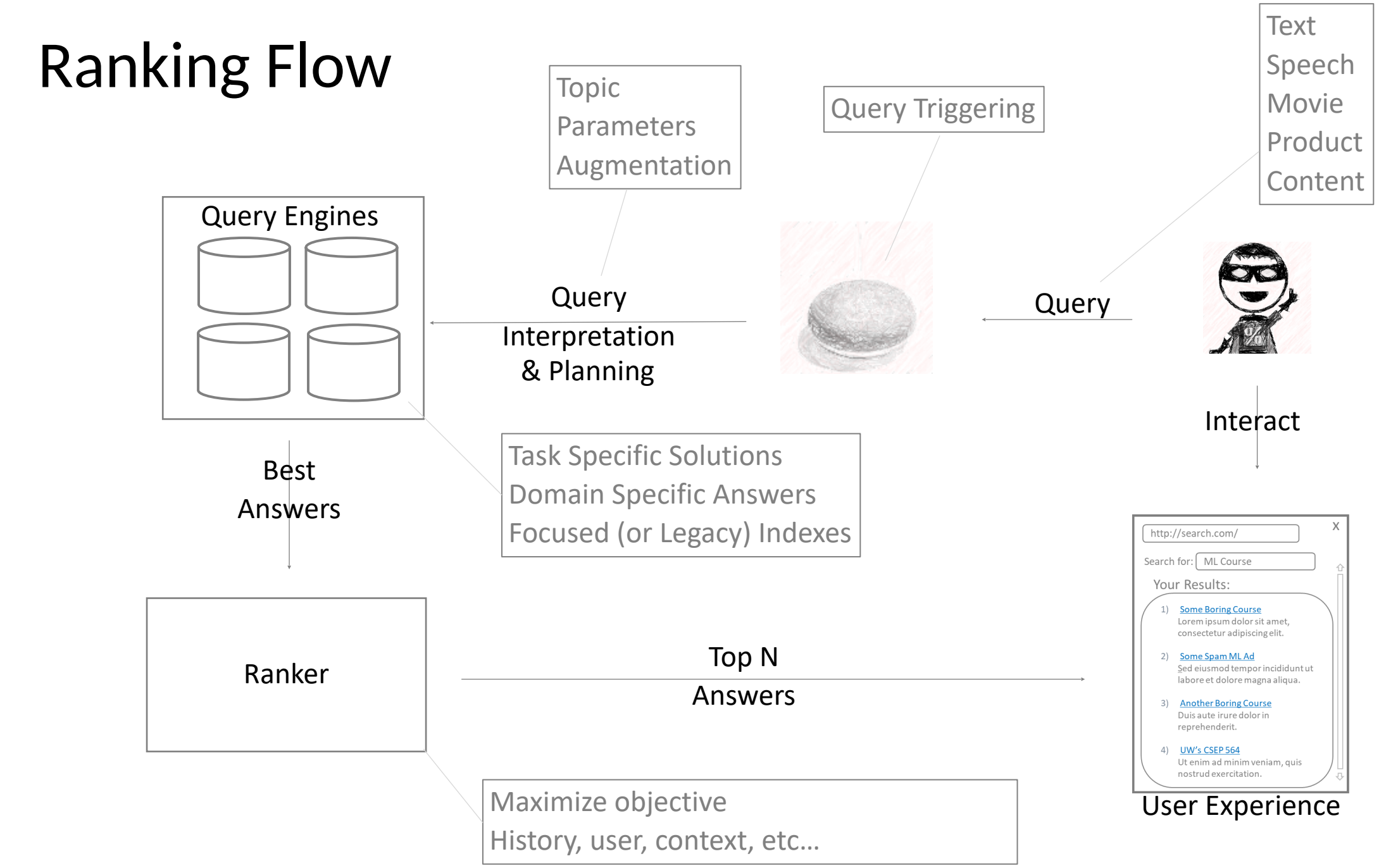
$$F\left(x_{query},\ x_{item}\right) = relevance$$

$$F\left(x_{query},\ x_{item1}, x_{item2}\ \right) = p(x_{item1} is\ higher)$$

$$F\left(x_{query},\ [\ x_{items}\ ]\right) = [\ scores\ ]$$

- Reasons for Ranking
  - Search Results
  - Ad Targeting
  - Movie recommendation
  - Skills for assistant
  - Designs
  - Digital Market place

Ranking Flow

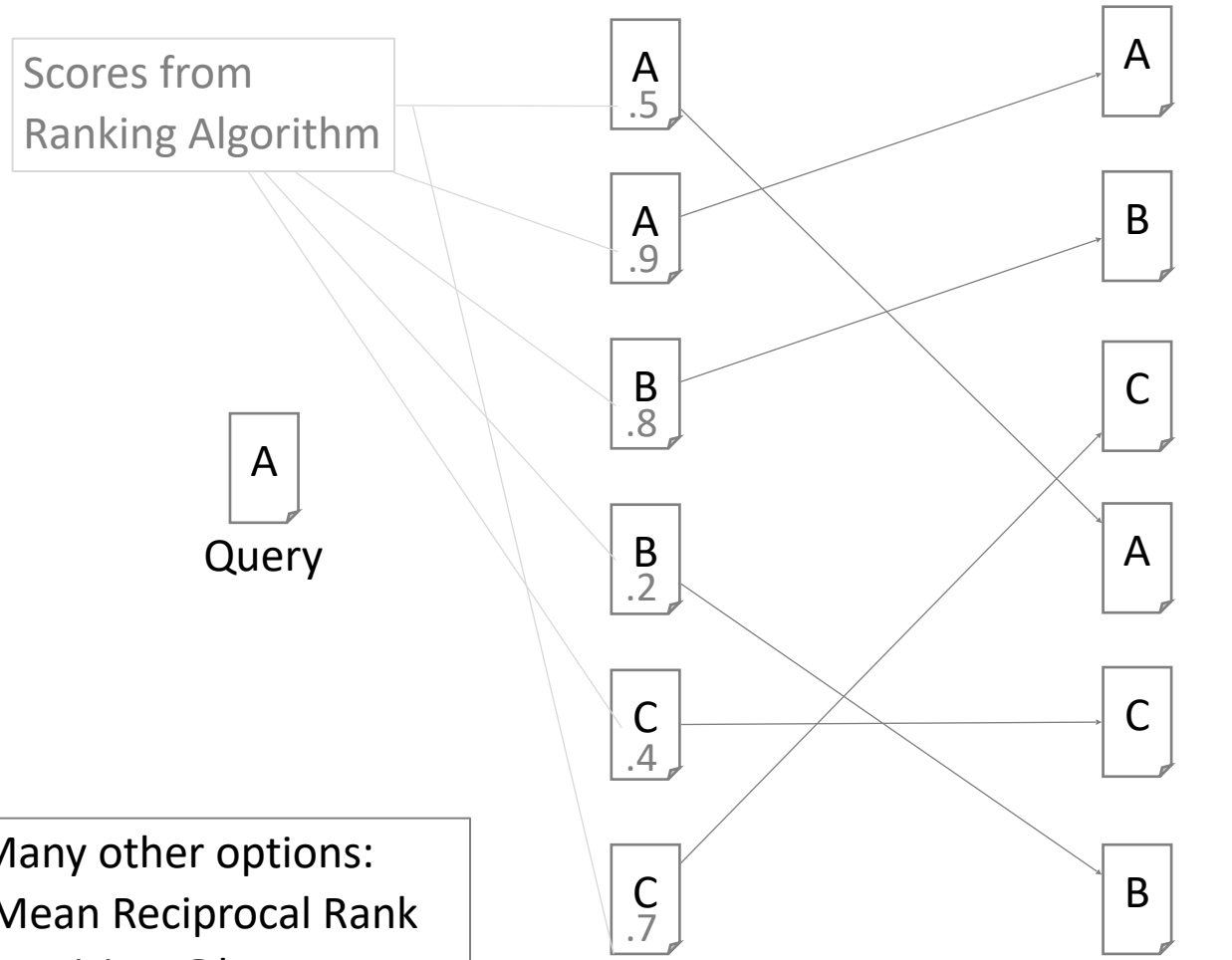


One way of Evaluating Ranking: Mean Average Precision

$$AveP(q) = \frac{\sum_{i=i}^{N} (Precision_{@i} * IsRelavant(i))}{\#Relevant}$$

$$Q = Test\ Queries$$

$$MAP = \frac{\sum_{q \in Q} AveP(q)}{|Q|}$$



$$(1.0 * 1) + (0.5 * 0) + (0.33 * 0) + (0.5 * 1) + (0.4 * 0) + (0.33 * 0)$$

$$AveP = \frac{1.5}{2} = 0.75$$

Many other options:
- Mean Reciprocal Rank
- Precision @k
- Clickthrough Rate
- User Outcomes
- And etc…

If:

Ranker put both As @ top $AveP = 1$

Ranker put both As @ end $AveP \sim .18$

Ranking algorithm sketch (RankNet)

- Training data:

  Set of: Query → $\{ < item_1, rel_1 >, \ldots, < item_n, rel_n > \}$

$$P_{ij} \equiv P\left(item_i \rhd item_j\right) \equiv \frac{1}{1 + e^{(rel_i - rel_j)}}$$

$$C = -\bar{P}_{ij} \log P_{ij} - (1 - \bar{P}_{ij}) \log(1 - P_{ij})$$

1 if i should rank above j else 0

- While not converged:
  - Iterate over training data
    - Apply current model to all items
    - For every pair of items, i, j
      - Adjust the model weights to make them 'more correctly ordered'

https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/MSR-TR-2010-82.pdf

Getting Training data for ranking models

Corpus Centric

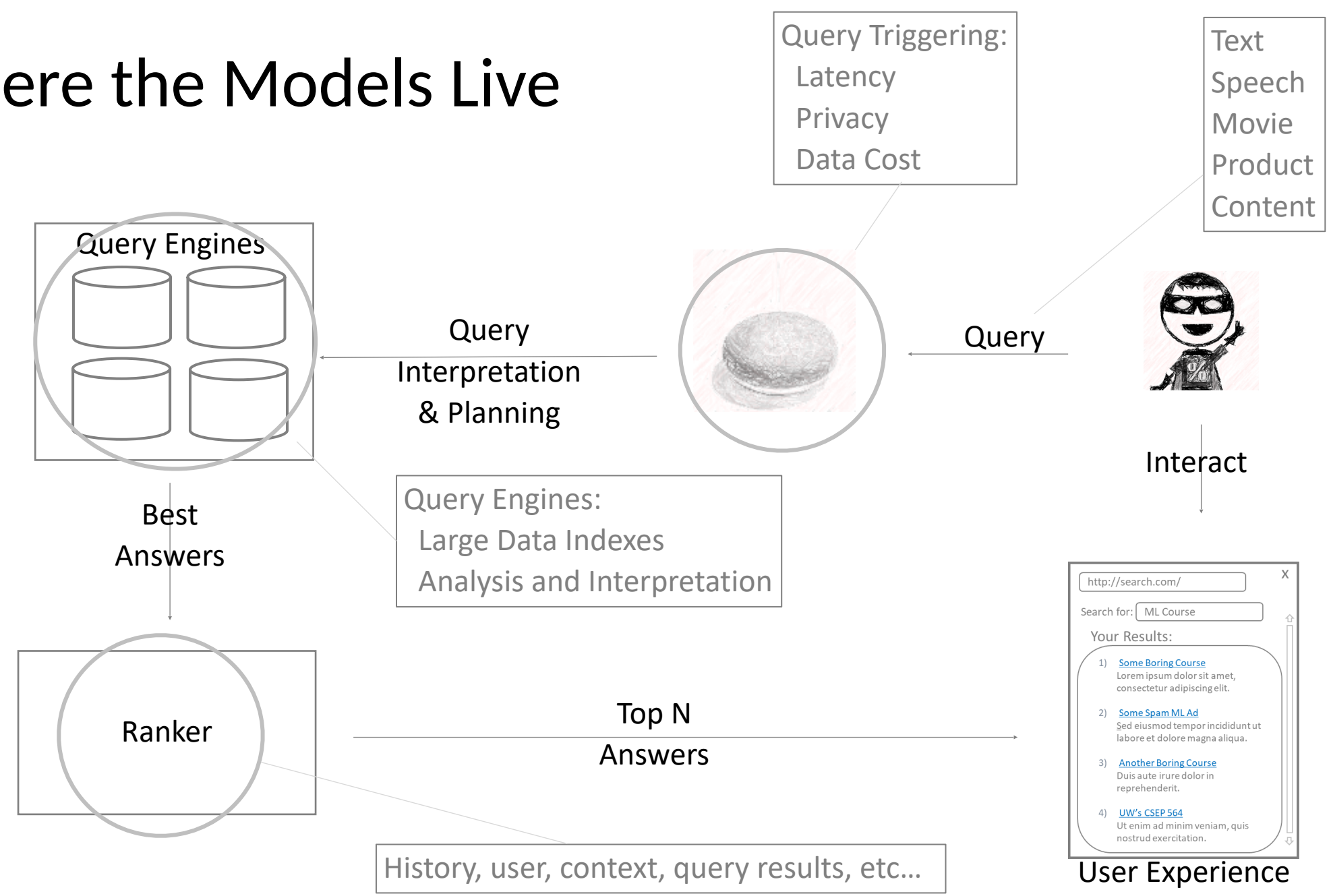
Sample queries from the system and:

- Pay labelers to find relevant (good) answers
- Pay labelers to grade the responses the system gives
- Do Active Learning

Closed Loop

- Record the interactions users have
  - Click through rate
  - Outcomes they achieve
- Explore for ranking training
  - ϵ greedy – show a random answer ϵ percent of the time
- Explore for query engine training
  - ϵ^2 greedy – let random engine show random answers some percent of the time

Where the Models Live

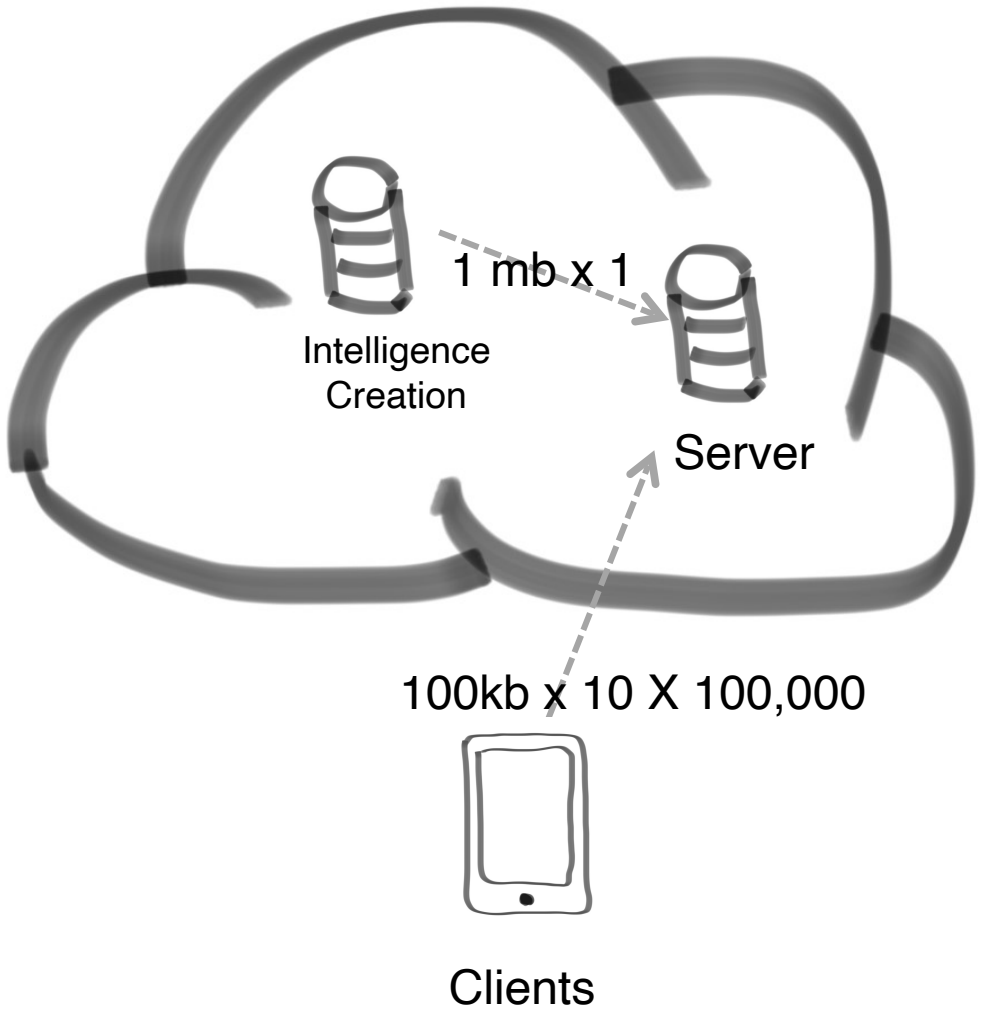


What does it matter where intelligence lives?

- Latency in Updating
  - Quality is evolving quickly
  - Problem is evolving quickly
  - Risk of costly mistakes
- Latency in Execution
  - Slowing the experience
  - The right answer changes too fast
- Cost of operation
  - Cost of distributing intelligence
  - Cost of executing intelligence
- Offline operation
  - Work without Internet?
  - Keep it out of Abuser’s hands…
- PRIVACY

Where Intelligence Lives

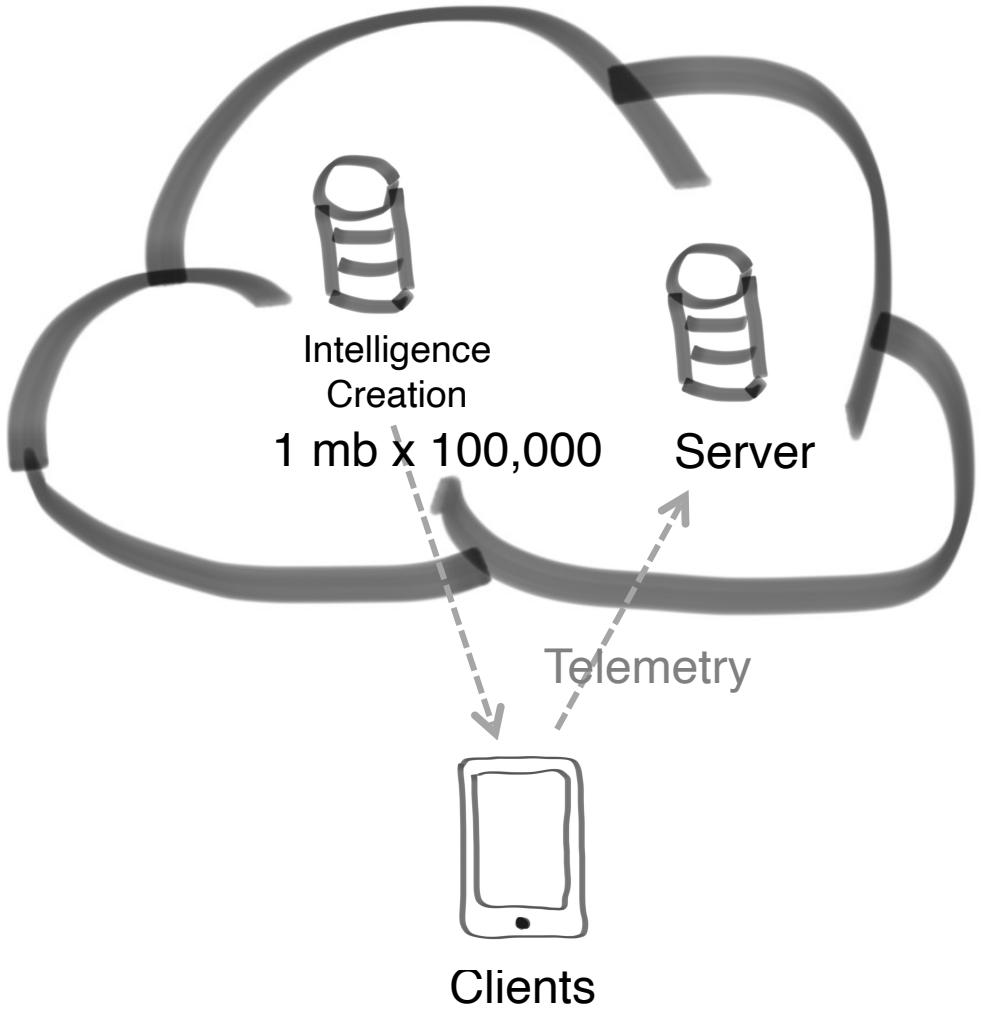
Lives in Service



Total: 100,001 mb + compute

1 MB Model
Daily Update
100k Users
100kb/Call
10 Calls/Day

Lives on Client

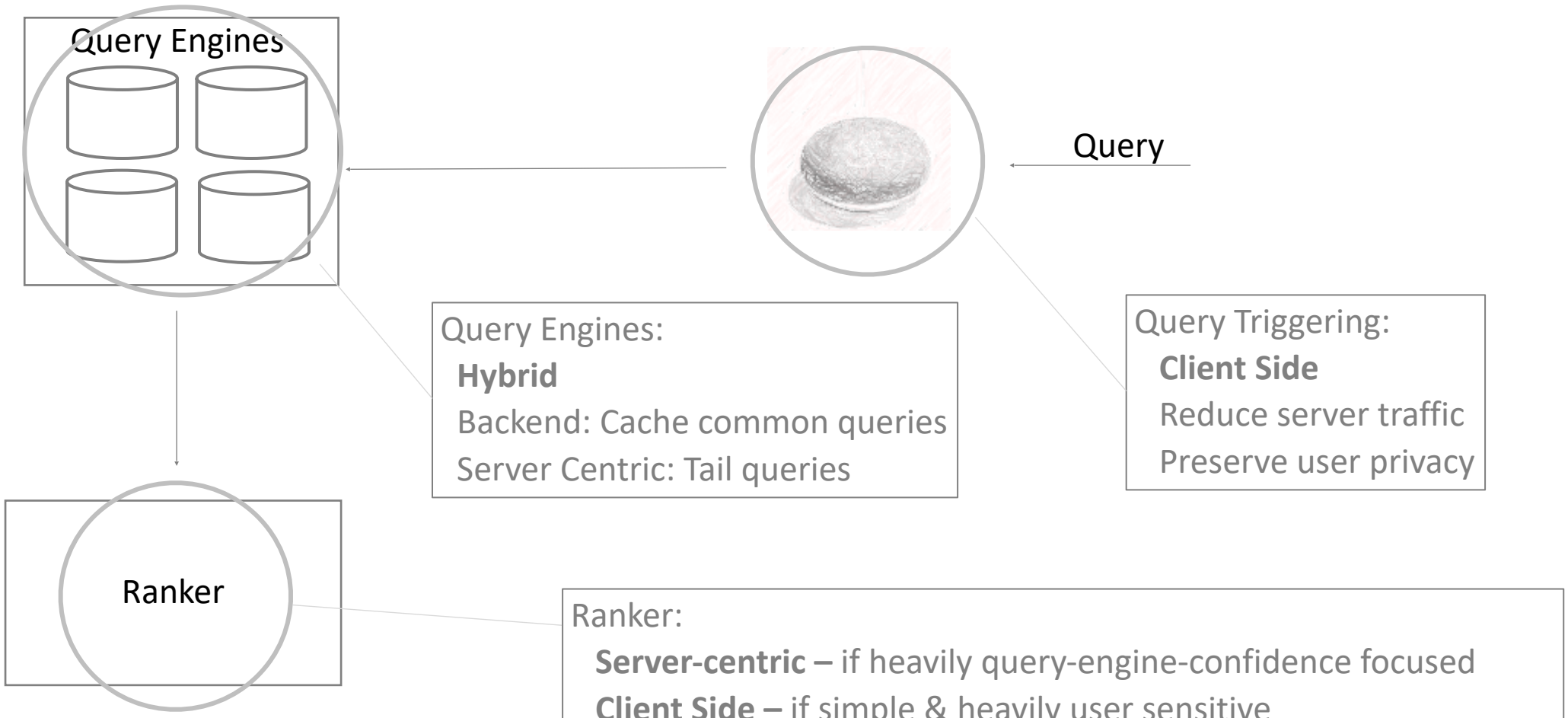


Total: 100,000 mb + Telemetry

Places Intelligence can Live

Where it Lives	Latency in Updating	Latency in Execution	Cost of Operation	Offline?
Static in Product	Poor	Excellent	Cheap	Yes
Client Side	Variable	Excellent	Based on update rate	Yes
Server-Centric	Good	Internet Roundtrip	Can be high	No
Back-end	Variable	Variable	Variable	Partial
Hybrid	??	??	??	??

Where the Models Live



Deploying and Lighting Up (Online Evaluation)

- Single Deployment
  - All users see all updates ‘at once’
  - Simple
  - Relies on great offline tests
  - Risk of costly/hard-to-find mistakes.
- Silent Intelligence
  - Run two versions at once
  - Ensure online is same as offline
  - Gives time to see ‘new’ contexts
  - Latency. No interactions.
- Controlled Rollout
  - Several live at once, transition slowly
  - Lets you observe user interactions
  - Overhead to build and manage
  - Adds latency.
- Flighting Intelligence (A/B test)
  - Deploy options, track till one better
  - Connects accuracy to true objective
  - Overhead to build and manage
  - Latency. Hard to confirm small gains.

Summary Ranking Based

- Ranking sorts possible responses in the correct order based on a query
- Loss metrics for ranking include:
  - Mean Average Precision
  - Mean reciprocal rank
  - Precision @k
  - Clickthrough rate
  - Outcomes
  - And more…
- Choosing where your models lives can have a large impact on cost / effectiveness, options include:
  - Static
  - client side
  - server centric
  - back end
  - hybrid
- Ranking can be used corpus centric or with a closed loop