Natural Language Processing

Recommendation Systems

From the engineering point of view

yuliats@cs.washington.edu

PAUL G. ALLEN SCHOOL
OF COMPUTER SCIENCE & ENGINEERING

Readings

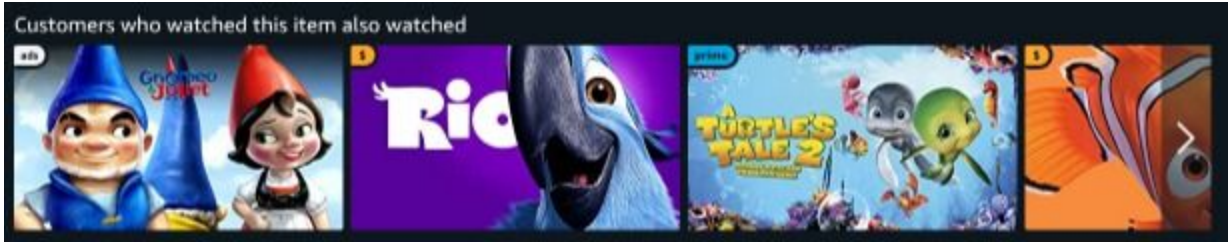
- Dan Jurafsky, Stanford University, CS 124, Recommender Systems lectures

What recommender systems are?

Can you think of examples?

What recommender systems are?

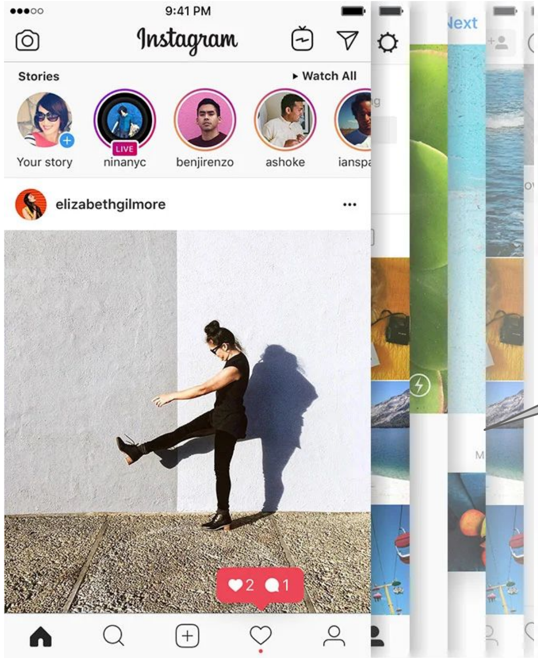
The most obvious example is "similar items" on shopping sites, or media platforms



Relatively simple, in most cases not "personalized", i.e. all users see the same recommendation

What recommender systems are?

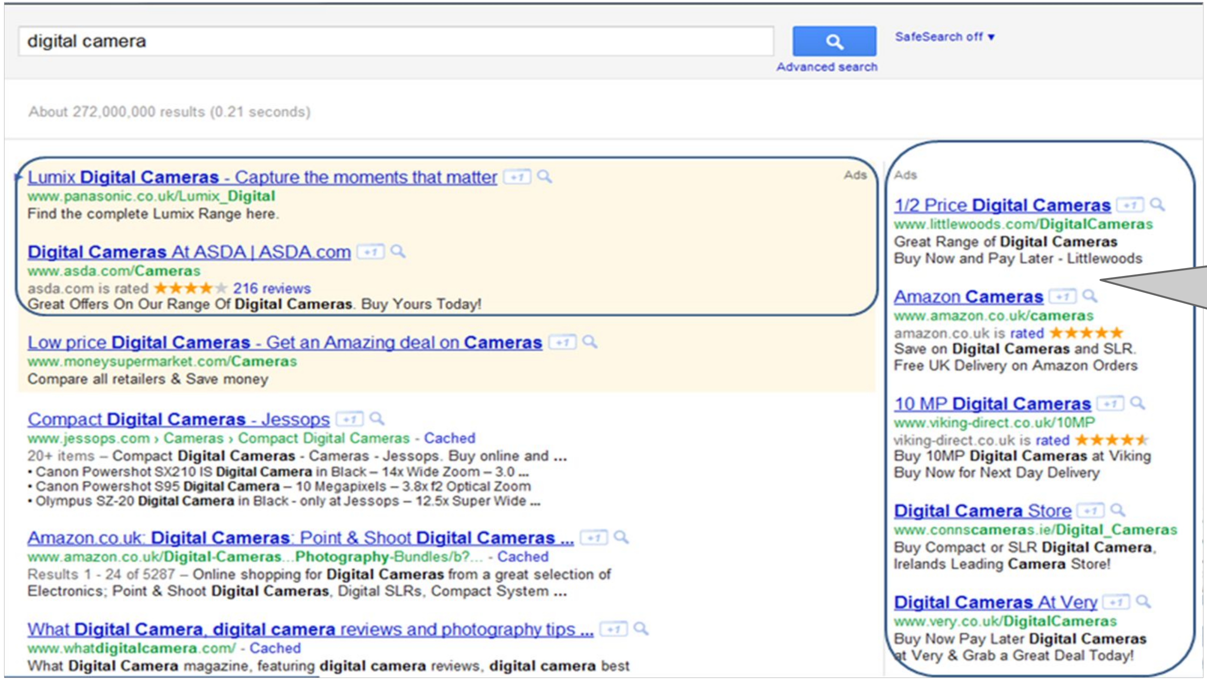
Social media feeds



Personalized for every user

What recommender systems are?

Search Ads

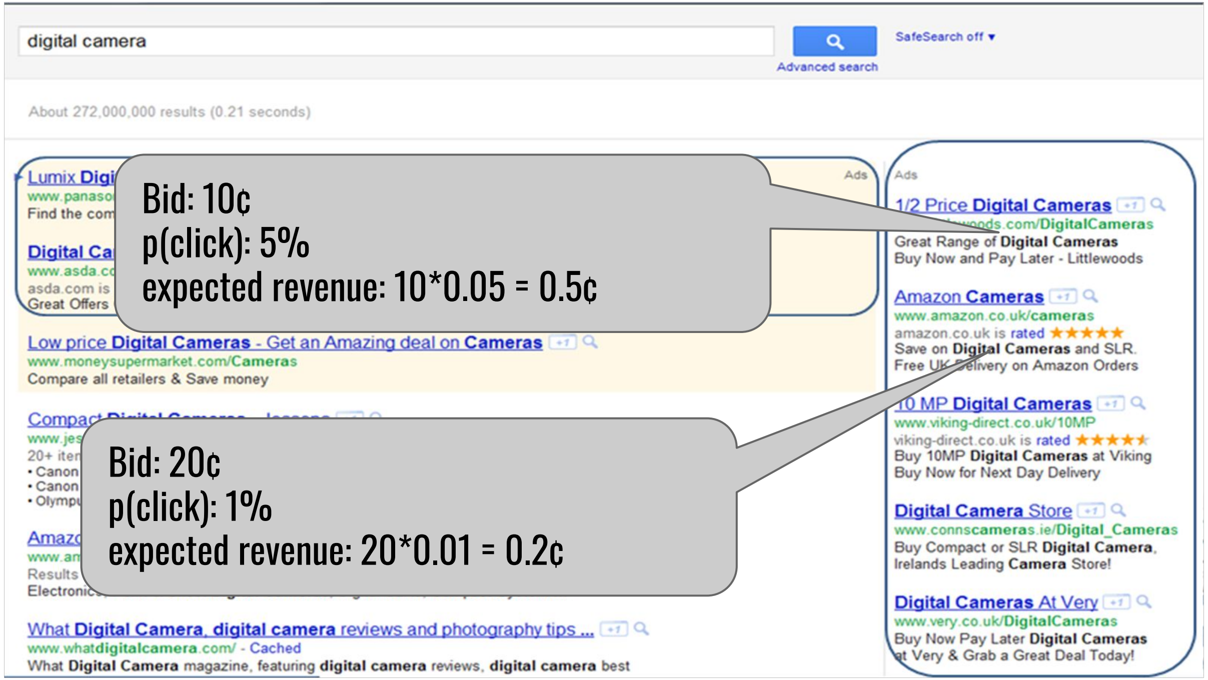


Customized for user and query

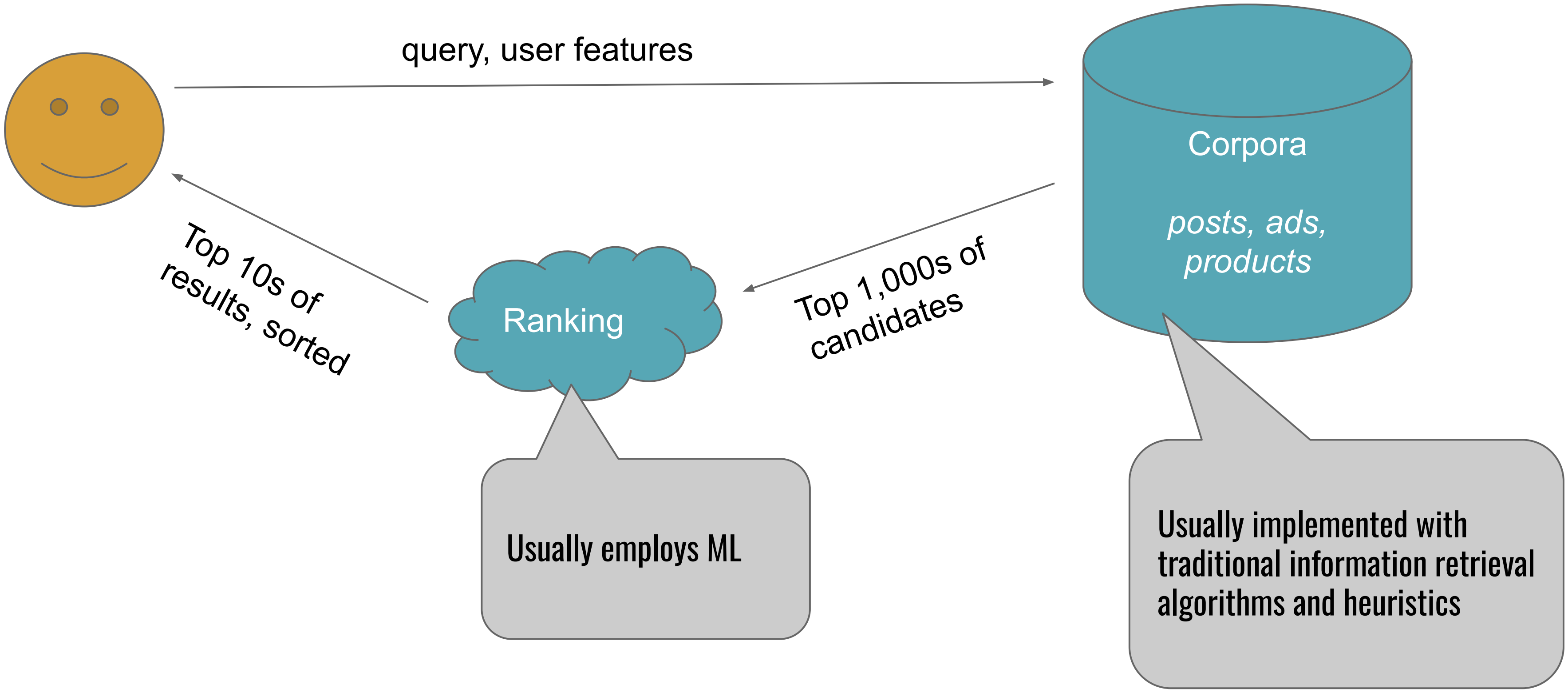
Complicated auction computation

Search ads auction

Predicted Click-Through-Rate (pCTR)



Recommender system task definition



Serving scale challenges

Typical usage stats for popular social media platforms:

- Over a billion of users
- Peaks at >500K requests per second
- Has >1,000 candidates to score for each request
- Needs to perform >500M inferences per second to serve the global user traffic

Serving infrastructure costs many millions of dollars. Must be deployed all over the world, including in countries under sanctions.

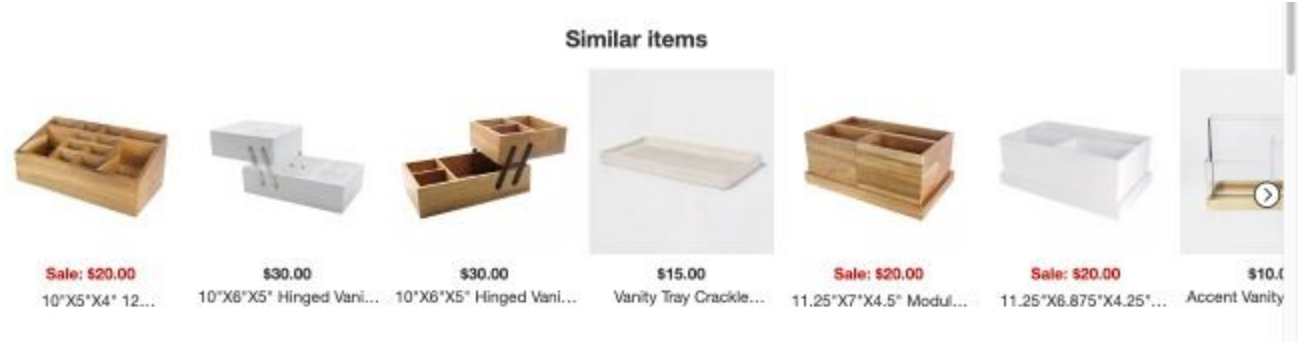
Cannot naively use big NLP models, such as BERT, GPT-3, etc.

Objective function

When using ML you need to figure out the objective function

For shopping recommendations it is relatively simple:

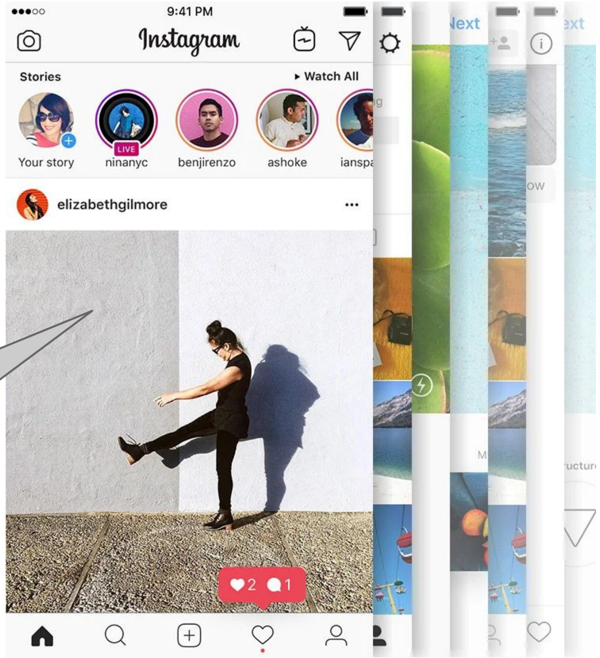
What items are purchased after viewing this item



Objective function

How do you define the objective function for ranking social media posts?

How would a human select the best TikTok videos for me to watch?
What would you need to know about me?



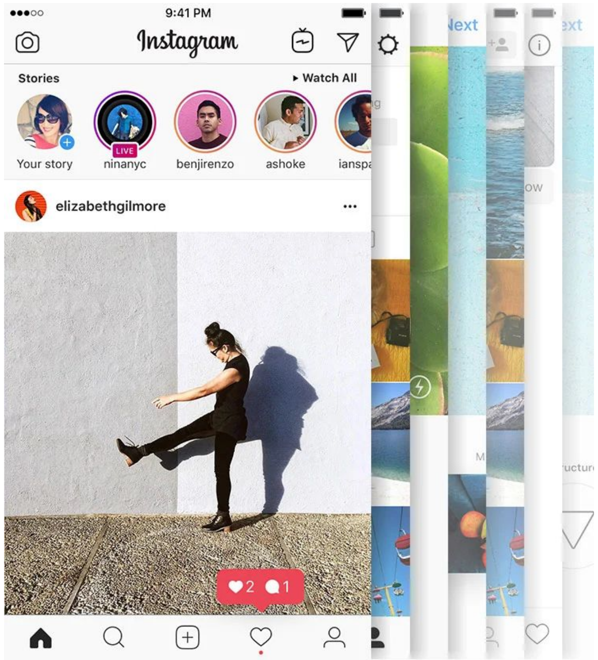
Objective function

How do you define the objective function for ranking social media posts?

What are we optimizing for?

Possible candidates

- Maximize the time spent in the app scrolling the posts
- User spent more time reading the post
- Maximize user engagement, such as "like" button clicks, comments/replies



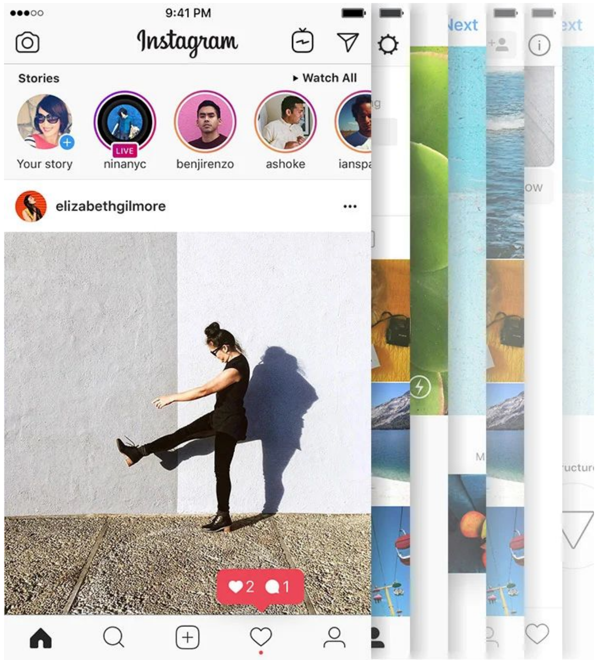
Objective function

Usually requires complicated models with lots of features and multi-objective functions, which are kept in secret.

If users spend a few minutes longer per day in the app, the company will make many billions more $$$ in advertising revenues

It's important to optimize for long-term user happiness.

Companies fight hard to hire ML researchers and infrastructure people



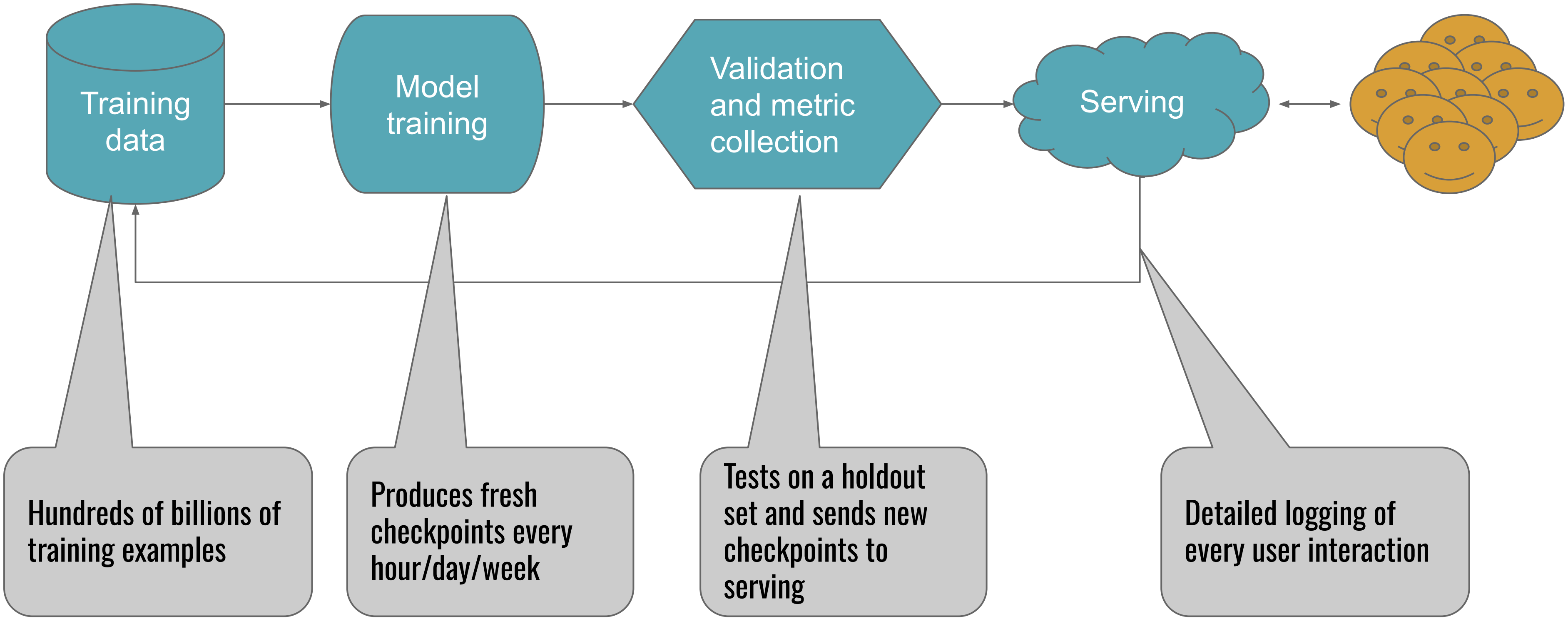
Non-stationary problem

User behavior and the meaning of keywords changes very rapidly.

- World news
  - Science, politics, wars, pandemics, economic and social issues
- Popular events
  - Black Friday, Back-to-School, Oscars, Grammys, Worldcup, Super Bowl
- Trends
  - Fashion, Flashmobs, New art releases

Model quality degrades within days

Online training

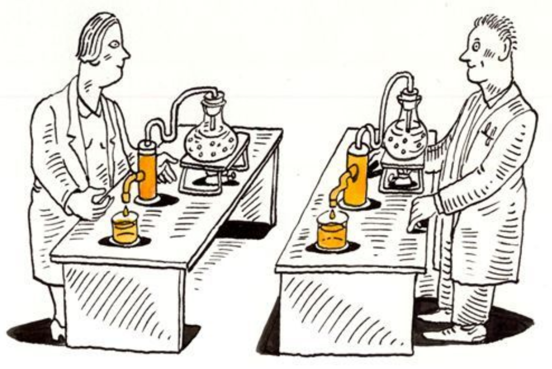


Common issues with model quality

- Model instability
  - Hyperparameters, such as learning rate, are tuned very close to the breaking point to pick up user behavior changes very quickly
  - Usually requires some form of weight normalization and gradient clipping

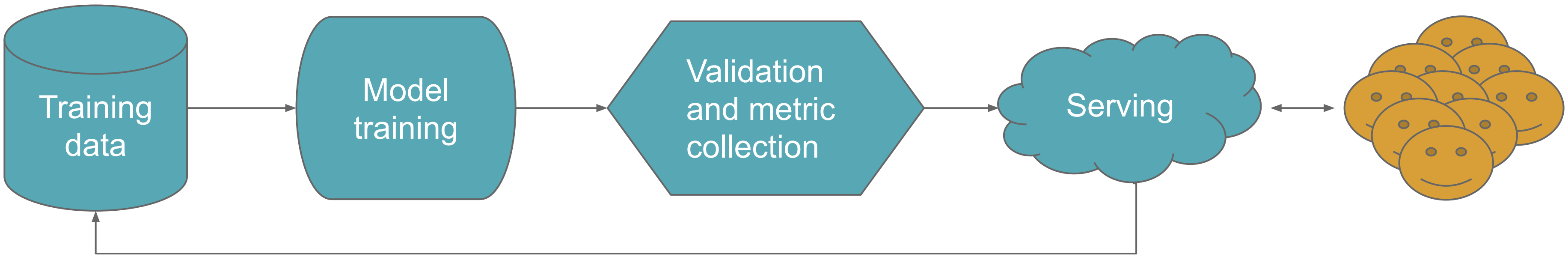
Common issues with model quality

- Reproducibility
  - A retrain of a model should yield the same accuracy up to 0.01%, otherwise ML researchers can't experiment with new models
  - Due to the distributed nature of training infrastructure the training sample visitation is not deterministic
  - Non-stochastic properties of Deep Neural Networks
  - See [Reproducibility in Deep Learning and Smooth Activations]



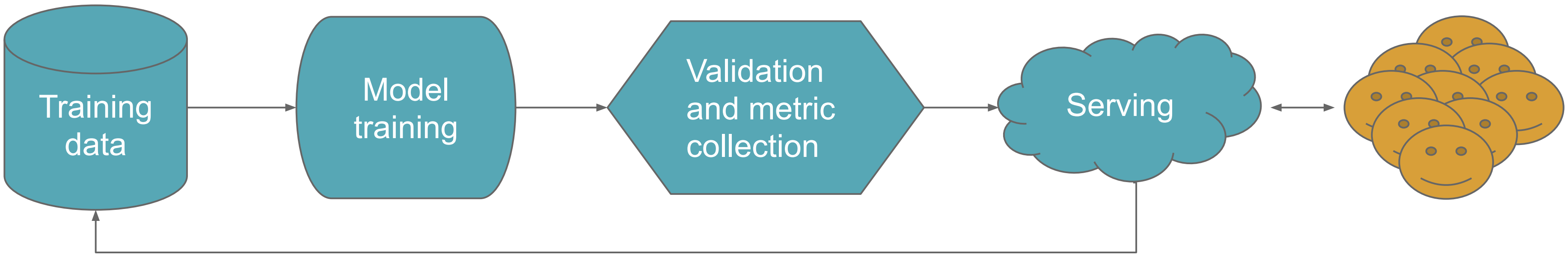
Common issues with model quality

- Ingestion of bad data + ripple effects
  - Happens when an upstream system misbehaves, e.g. the "buy" button gets broken for several hours, which skews the training data
  - The model starts making wrong predictions and new logs get poisoned as well
  - This creates a ripple effect that is very hard to deal with. Usually requires reverting to older checkpoints and isolating large ranges of training data.



Common issues with model quality

- Feedback loop problem
  - A freshly-trained model performs poorly when it starts serving user traffic for the first time
  - It has not seen enough bad examples, because it was trained only on data that was filtered by previous iterations of the model



Training data

Input data

User:

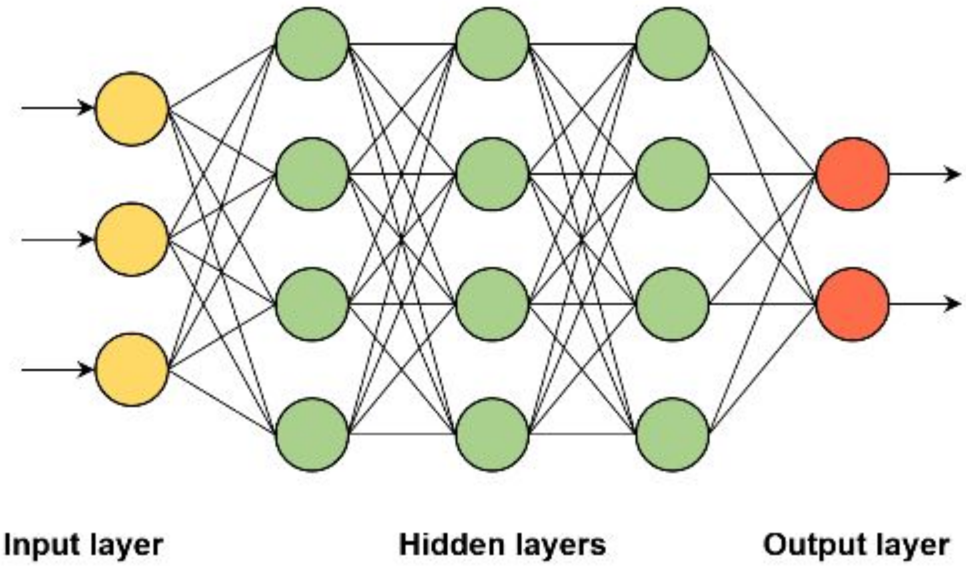
User preferences,

Language, Region,

Browser, device, ...

Post/Ad/Product:

Creators, title,
content, price, rating



Labels

User interactions:

Post viewed

Time spent viewing the post

Like button clicked

Comment left

Training data

Input data

User:

User preferences,

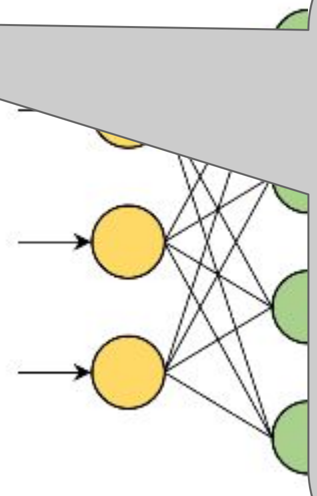
Language, Region,

Browser, device, ...

Post/Ad/Product:

Creators, title,

content, price, rating



Input layer

Hundreds of features, a lot of textual information.

Need to make sure that features are available at serving time. E.g. whether the user clicked on the post is not a good feature.

Deep Neural Networks work with vectors of floating point numbers, not with text, especially not with variable-length features

Heavy usage of embedding tables
If computation capacity allows in serving sequences can be processed using large language models, such as Bert, GPT-2/3

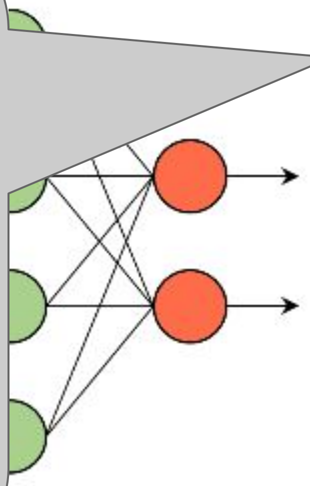
Embeddings are also useful for memorizing non-textual information

Training data

Labels are often binary indicator whether an event happened or not.
E.g. whether the user clicked on the "like" button.

However, the model outputs the probability of an event.
E.g. the probability that the user clicks on the like button is 0.0343

This means that the losses are always high, even when the model is well trained.



Output layer

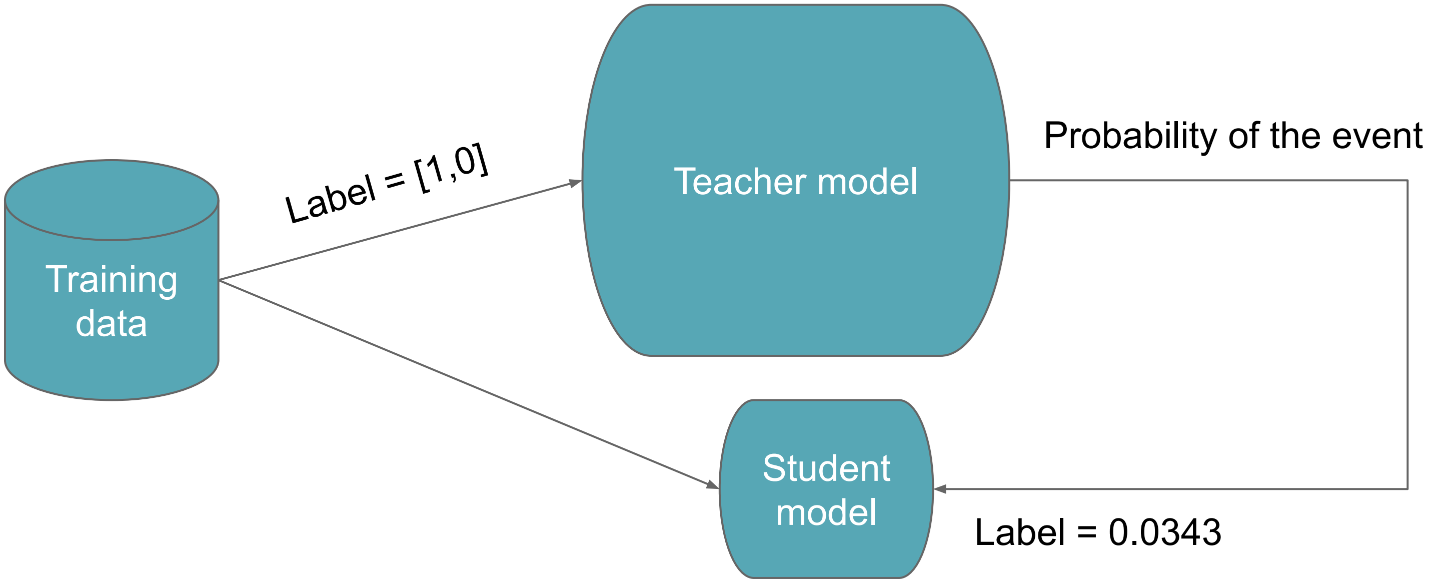
Labels

User interactions:

- *Post viewed*
- *Time spent viewing the post*
- *Like button clicked*
- *Comment left*

Teacher - student

We can train a much bigger model that is unviable for serving and use it as a "teacher" for a production model.



Splitting models

User model:

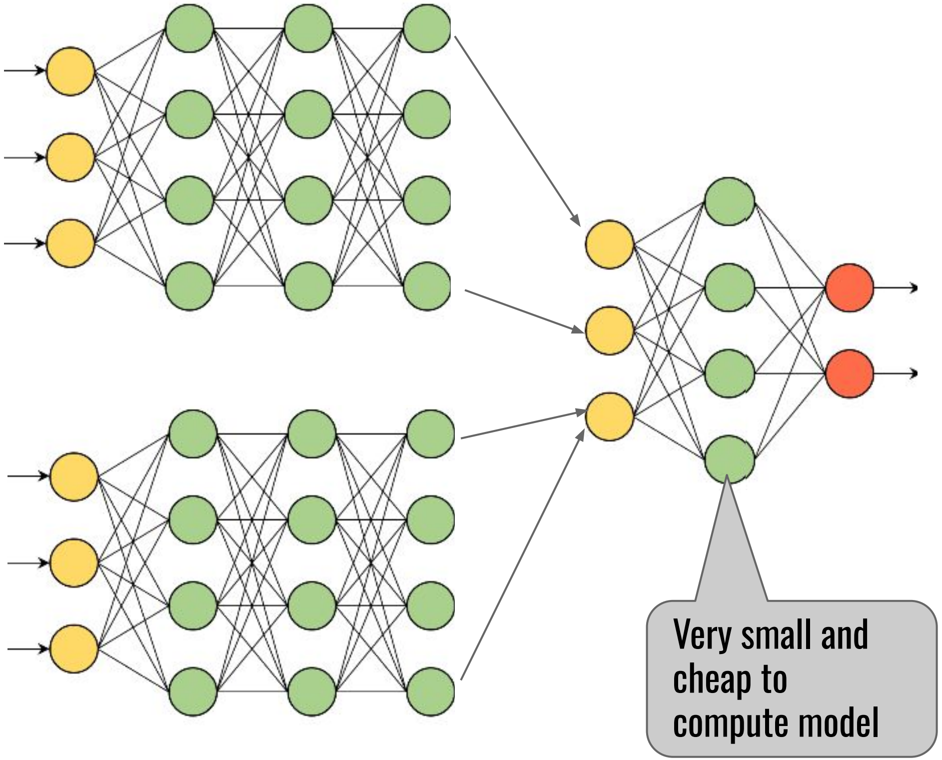
User preferences,

Language, Region,

Browser, device, ...

Post/Ad/Product model:

Creators, title, content, price, rating



Labels

User interactions:

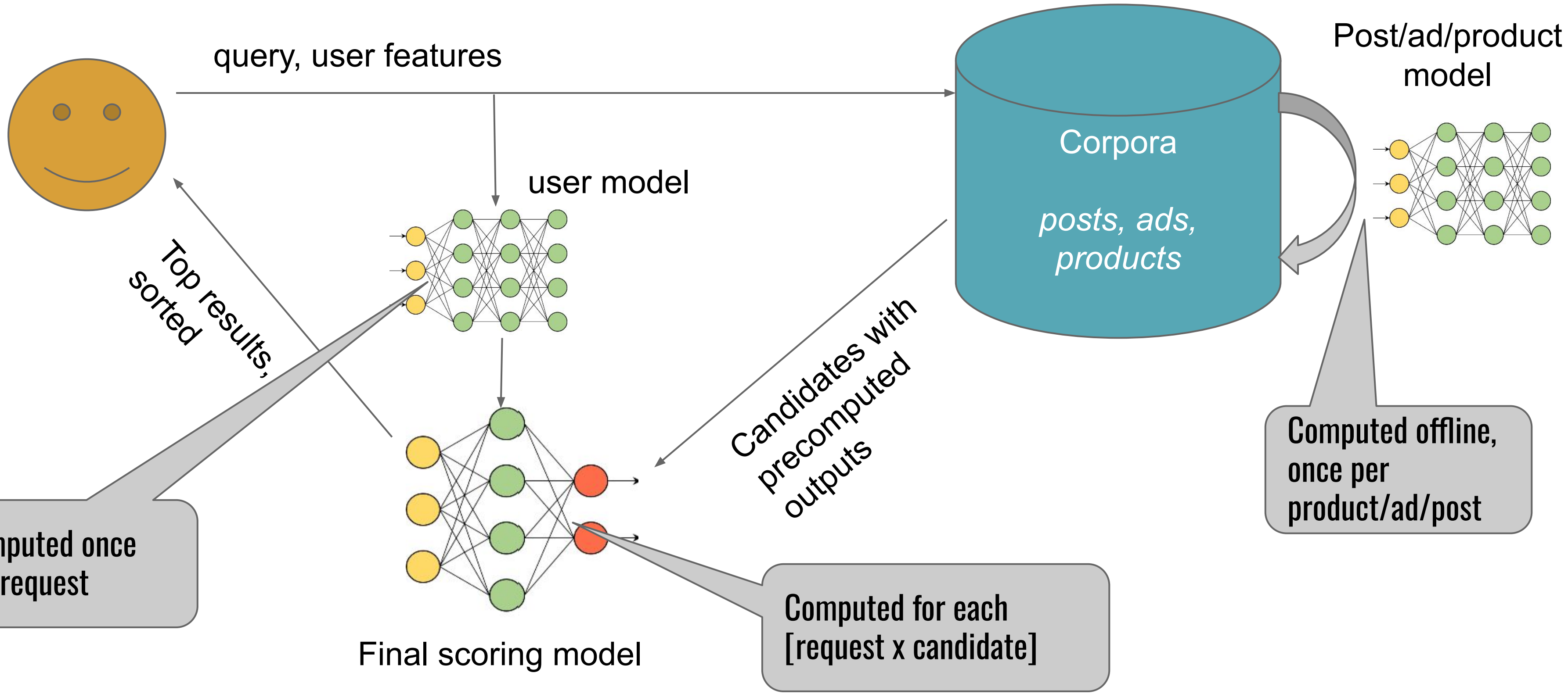
Post viewed

Time spent viewing the post

Like button clicked

Comment left

Splitting models



Splitting models

- Greatly reduces serving cost
- Usually worsens the quality, because user and candidate post features aren't mixed in the neural network
- Adds a lot to system complexity

Parting words

We briefly discussed some interesting issues with building and deploying ML models

Here are some topics that we haven't talked about:

- Privacy issues and protections
- How to evaluate model quality
- Additional model distillation techniques
- Special optimizers and other techniques for asynchronous distributed training
- How to deal with late arriving data
- How to maintain models that train for months
- How to maintain feature definitions
- And many other topics

Thank you!

Questions?