Search and Recommendation

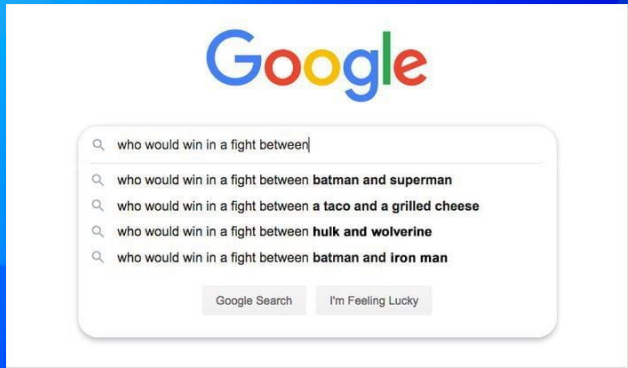
Hao Sheng

August 10th, 2023

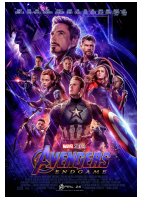
Agenda

- **Introduction**
- **Lecturing:** Search Engine System
- **Break-out:** Recommendation system in daily life
- **Lecturing**: Advanced Topics (Part I)
- **Lab:** Recommendation system notebook II
- **Lecturing**: Advanced Topics (Part II)

Search Engine System

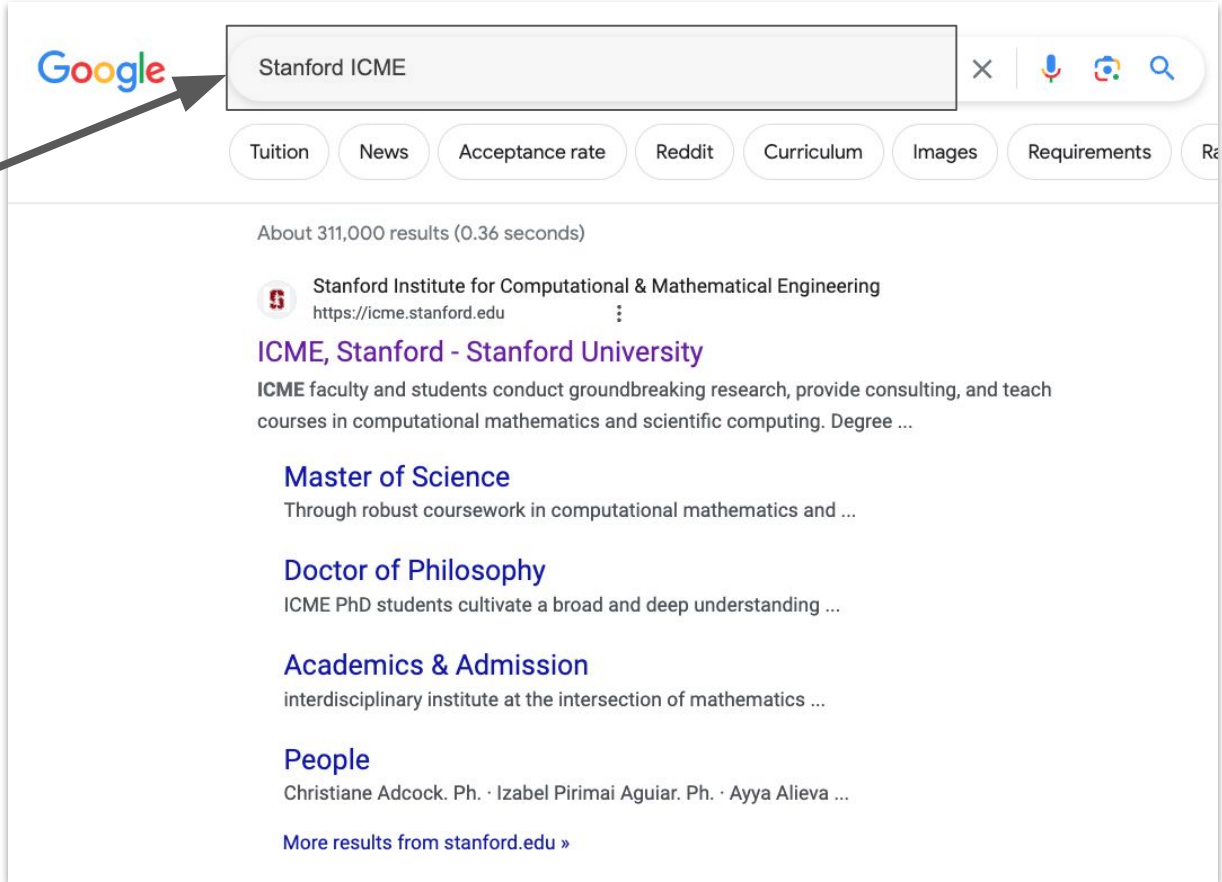


Quick Recap: Movie Recommendation System

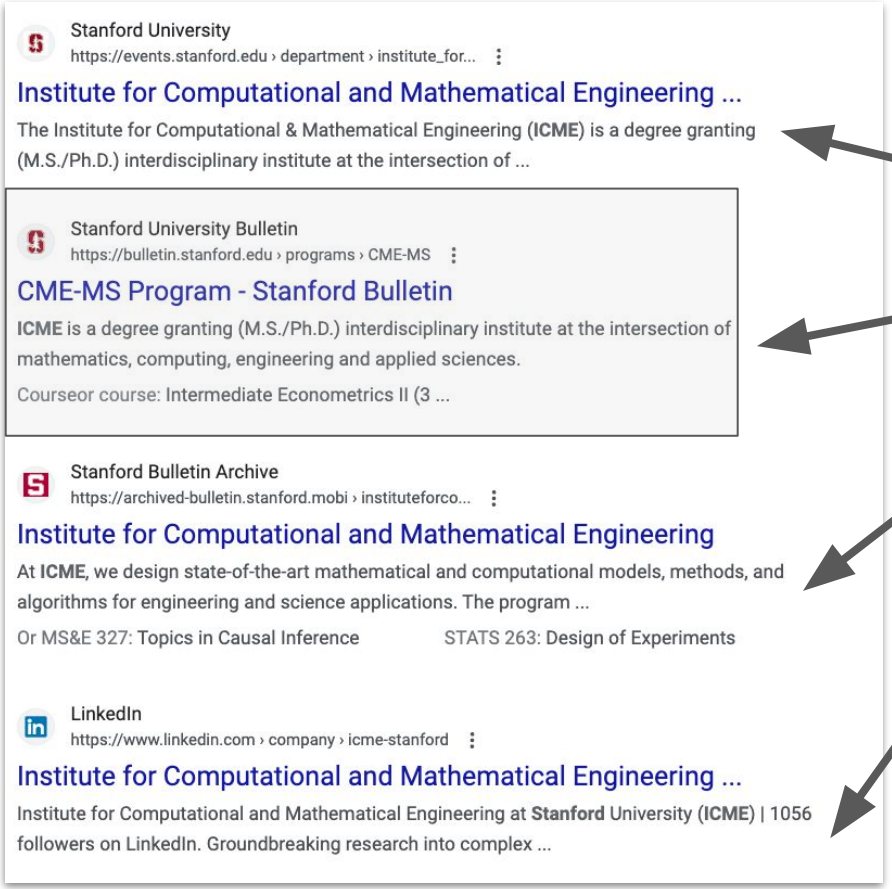
						
Sara	5	3		2	2	2
Jesper	4	3	4		3	3
Therese	5	2	5	2	1	1
Helle	3	5	3		1	1
Pietro	3	3	3	2	4	5
Ekaterina	2	3	2	3	5	5

Search Engine: First Glance

Search Query



Search Engine: First Glance



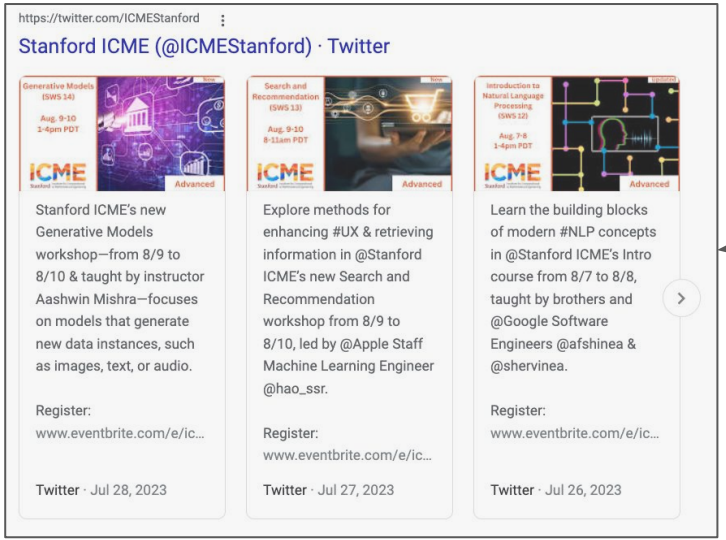
Search Engine Results (Websites)

Search Engine: First Glance

Stanford Online
https://online.stanford.edu › programs › computationa...

Computational and Mathematical Engineering MS Degree

The Institute for Computational and Mathematical Engineering (**ICME**) is a degree granting institute at the intersection of mathematics, computing, engineering ...



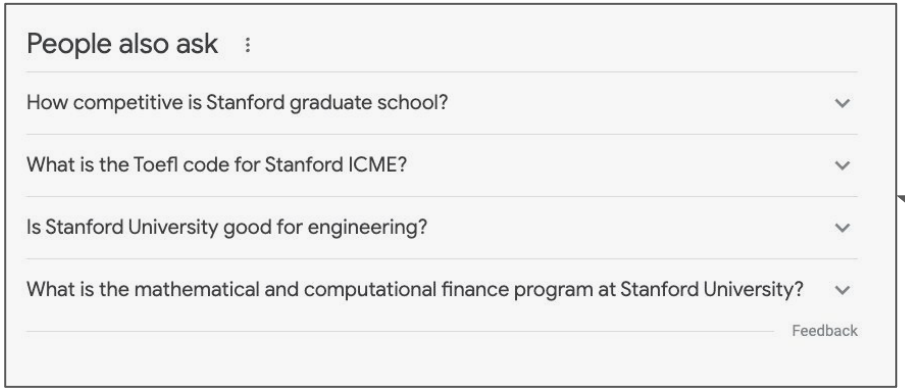
Search Engine Results (Tweets)

Stanford University
https://events.stanford.edu › department › institute_for...

Institute for Computational and Mathematical Engineering ...

The Institute for Computational & Mathematical Engineering (**ICME**) is a degree granting (M.S./Ph.D.) interdisciplinary institute at the intersection of ...

Search Engine: First Glance



Stanford Online
https://online.stanford.edu › programs › computationa...

Computational and Mathematical Engineering MS Degree

The Institute for Computational and Mathematical Engineering (**ICME**) is a degree granting institute at the intersection of mathematics, computing, engineering ...

Search Engine Results (Tweets)

Search Engine v.s. Movie Recommendation

- It has a search bar!
- The items are mal-defined at the first glance.
- User does not simply rate the search results!

Search Engine as a Recommendation System

“Recommender systems (RSs) are software tools and techniques that provide suggestions for items that are most likely of interest to a particular user. ”

--- *Introduction to Recommender Systems Handbook*

Search Engine as a Recommendation System

“Recommender systems (RSs) are software tools and techniques that provide suggestions for items that are most likely of interest to a particular user. ”

--- *Introduction to Recommender Systems Handbook*

Search Engine is a **Recommendation System!**

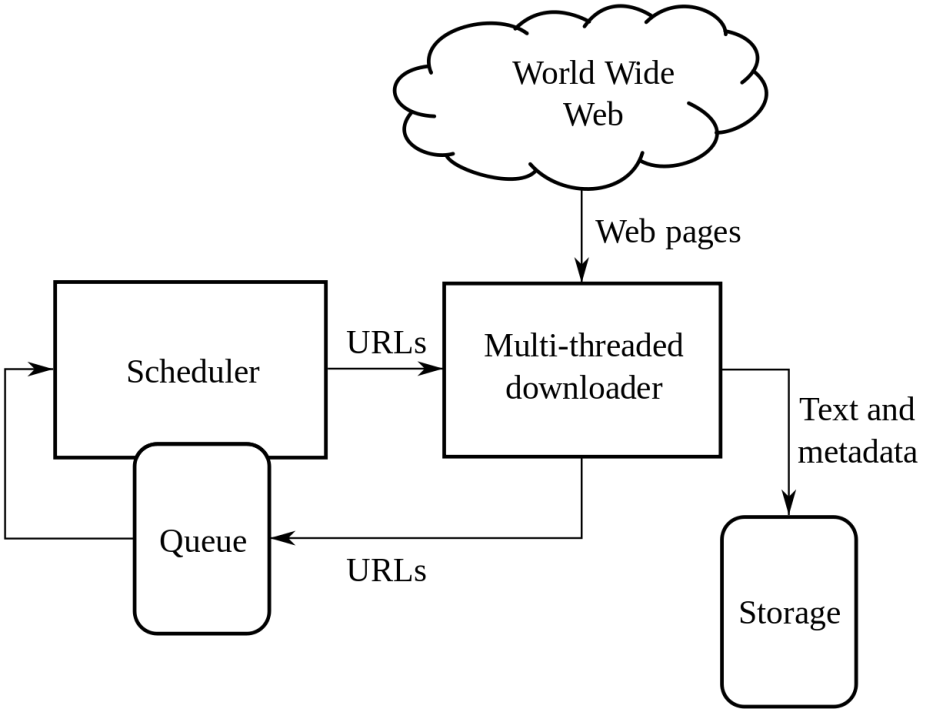
Search Engine: Learning Goals

- It has a search bar!
  - -> How to incorporate the user intention?
- The items are mal-defined at the first glance.
  - -> How to crawl the internet and store the items?
- User does not simply rate the search results!
  - -> How to assign user-item rating with user data?

Search Engine: Web Crawler

Where to get the search results (items)?

The internet!



Search Engine: Web Crawler - Recursion

Where to get the search results (items)?

The internet!

```
class Spider:
    name = 'icme_spider'
    start_urls = 'https://icme.stanford.edu/'
    parsed_urls = []

    def parse(self, url: str):
        self.parse_url.append(url)
        for next_url in Website(url):
            self.parse(next_url)
```

Search Engine: Web Crawler - Recursion

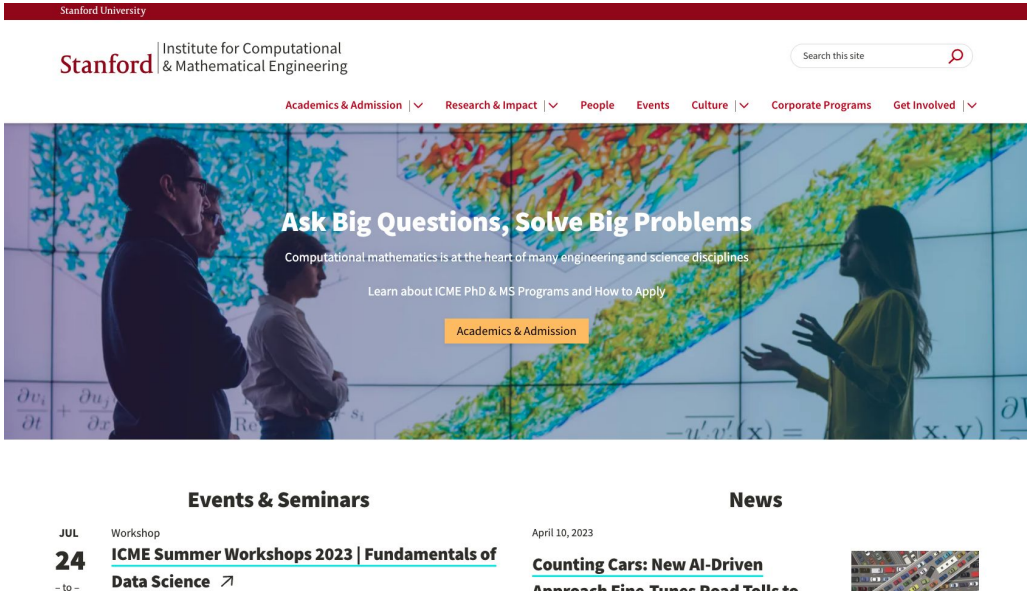
parsed_urls = [“https://icme.stanford.edu”]

```
class Spider:
    name = 'icme_spider'
    start_urls = 'https://icme.stanford.edu/'
    parsed_urls = []

    def parse(self, url: str):
        self.parse_url.append(url)
        for next_url in Website(url):
            self.parse(next_url)
```

Search Engine: Web Crawler - Recursion

parsed_urls = [“https://icme.stanford.edu”]

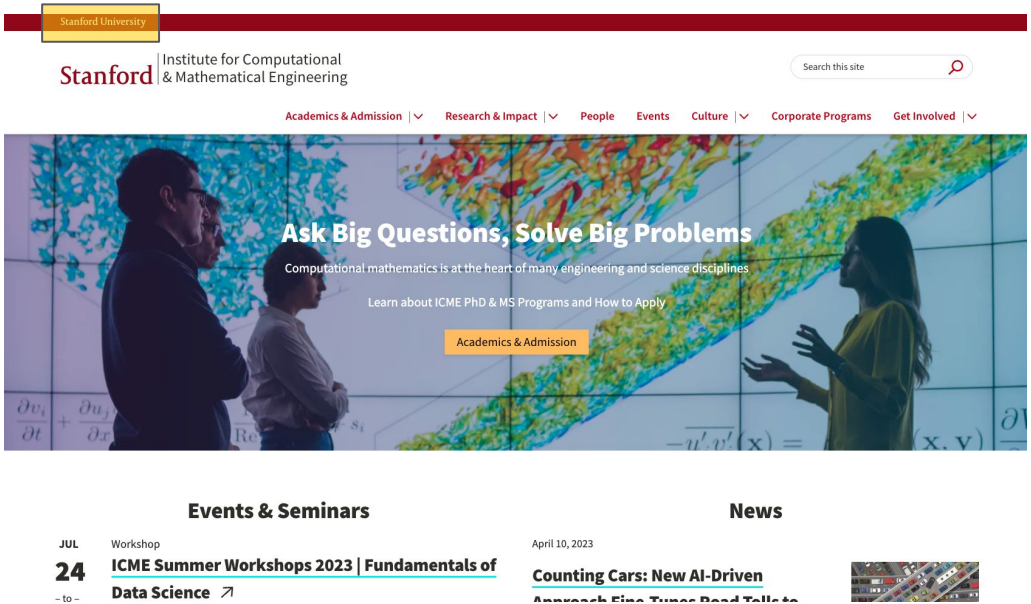
```
class Spider:
    name = 'icme_spider'
    start_urls = 'https://icme.stanford.edu/'
    parsed_urls = []

    def parse(self, url: str):
        self.parse_url.append(url)
        for next_url in Website(url):
            self.parse(next_url)
```



Search Engine: Web Crawler - Recursion

parsed_urls = [“https://icme.stanford.edu”]

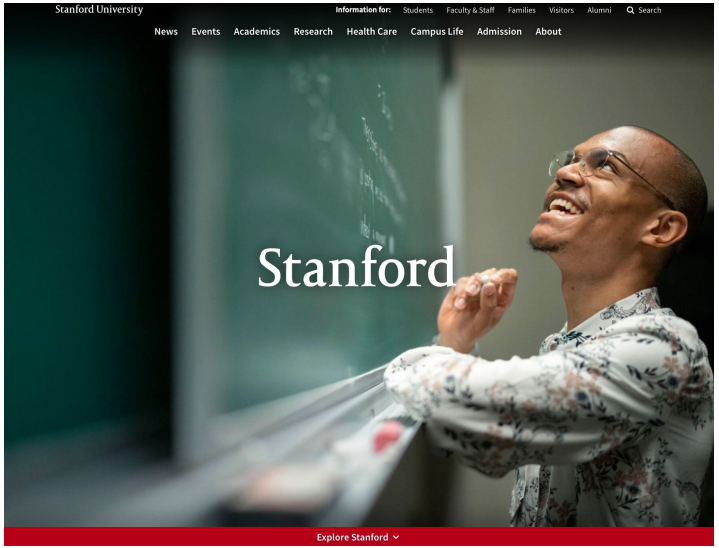
```
class Spider:
    name = 'icme_spider'
    start_urls = 'https://icme.stanford.edu/'
    parsed_urls = []

    def parse(self, url: str):
        self.parse_url.append(url)
        for next_url in Website(url):
            self.parse(next_url)
```



Search Engine: Web Crawler - Recursion

parsed_urls = [“https://icme.stanford.edu”]

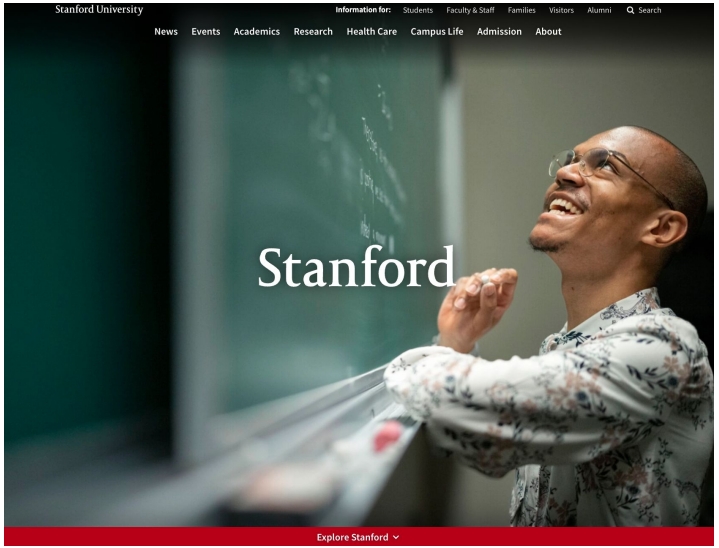
```
class Spider:
    name = 'icme_spider'
    start_urls = 'https://icme.stanford.edu/'
    parsed_urls = []

    def parse(self, url: str):
        self.parse_url.append(url)
        for next_url in Website(url):
            self.parse(next_url)
```



Search Engine: Web Crawler - Recursion

```
class Spider:
    name = 'icme_spider'
    start_urls = 'https://icme.stanford.edu/'
    parsed_urls = []

    def parse(self, url: str):
        self.parse_url.append(url)
        for next_url in Website(url):
            self.parse(next_url)
```

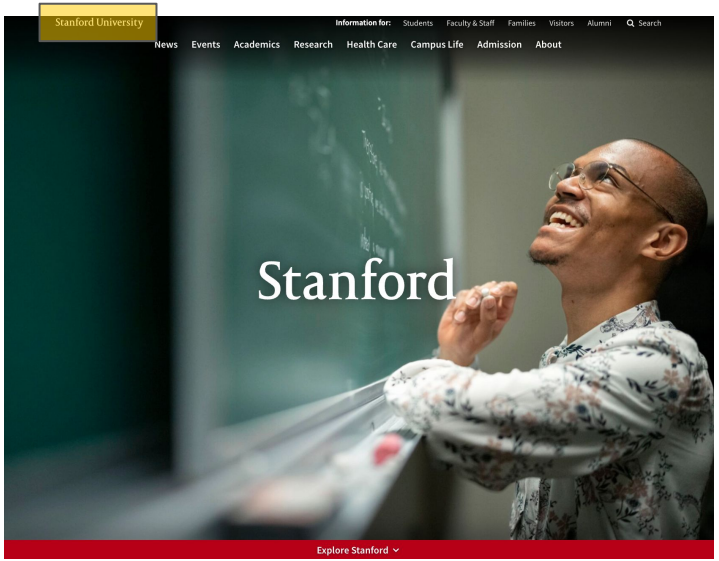
parsed_urls = [“https://icme.stanford.edu”,
“https://www.stanford.edu/”
]



Search Engine: Web Crawler - Recursion

```
class Spider:
    name = 'icme_spider'
    start_urls = 'https://icme.stanford.edu/'
    parsed_urls = []

    def parse(self, url: str):
        self.parse_url.append(url)
        for next_url in Website(url):
            self.parse(next_url)
```

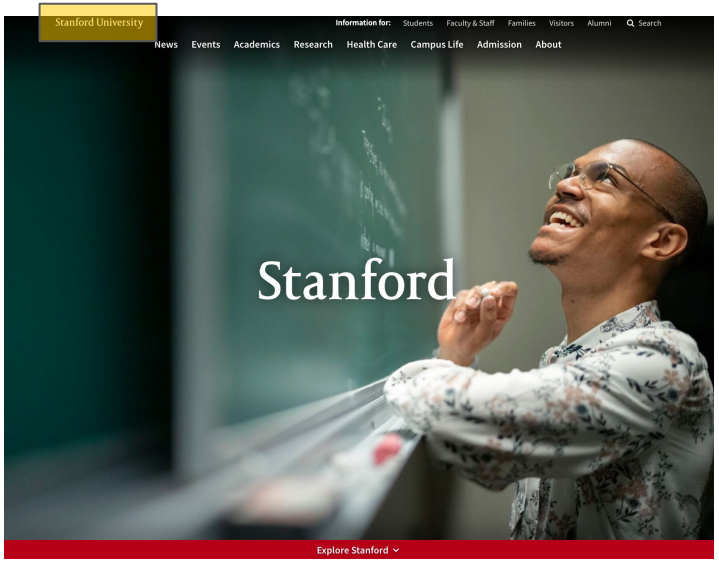
parsed_urls = [“https://icme.stanford.edu”,
“https://www.stanford.edu/”
]



Search Engine: Web Crawler - Recursion

```
class Spider:
    name = 'icme_spider'
    start_urls = 'https://icme.stanford.edu/'
    parsed_urls = []

    def parse(self, url: str):
        self.parse_url.append(url)
        for next_url in Website(url):
            self.parse(next_url)
```

parsed_urls = [“https://icme.stanford.edu”,
“https://www.stanford.edu/”
]

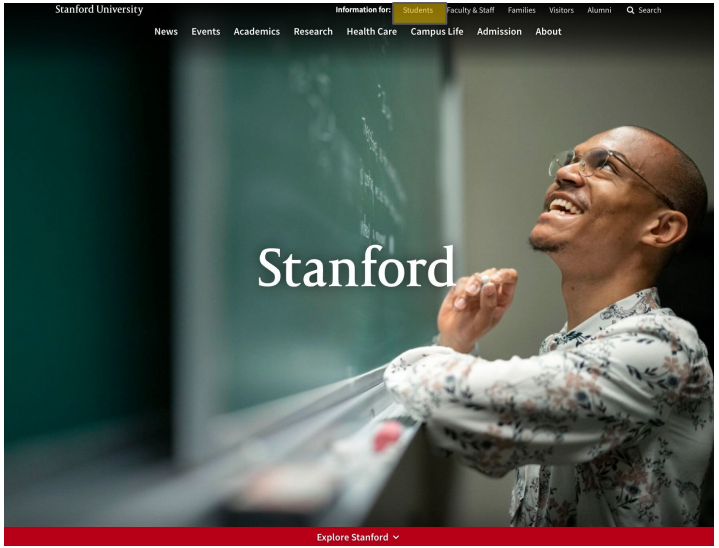


Oh no, it is *“https://www.stanford.edu/”* again -- we had a bug in the code!

Search Engine: Web Crawler - Recursion

```
class Spider:
    name = 'icme_spider'
    start_urls = 'https://icme.stanford.edu/'
    parsed_urls = []

    def parse(self, url: str):
        self.parse_url.append(url)
        for next_url in Website(url):
            if next_url not in self.parsed_urls:
                self.parse(next_url)
```

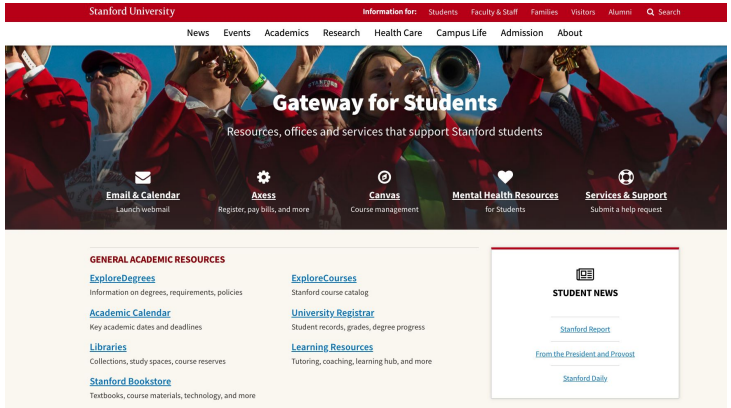
parsed_urls = [“https://icme.stanford.edu”,
“https://www.stanford.edu/”
]



Search Engine: Web Crawler - Recursion

```
class Spider:
    name = 'icme_spider'
    start_urls = 'https://icme.stanford.edu/'
    parsed_urls = []

    def parse(self, url: str):
        self.parse_url.append(url)
        for next_url in Website(url):
            if next_url not in self.parsed_urls:
                self.parse(next_url)
```

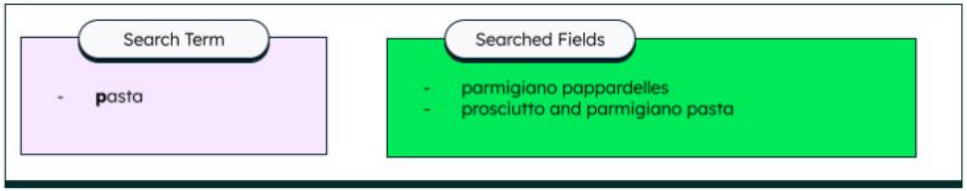
parsed_urls = [“https://icme.stanford.edu”,
“https://www.stanford.edu/”
“https://www.stanford.edu/student-gateway/”
]



Search Engine: Full-text Search

- It has a search bar!
  - **-> How to incorporate the user intention?**
- The items are mal-defined at the first glance.
  - -> How to crawl the internet and store the items?
- User does not simply rate the search results!
  - -> How to assign user-item rating with user data?

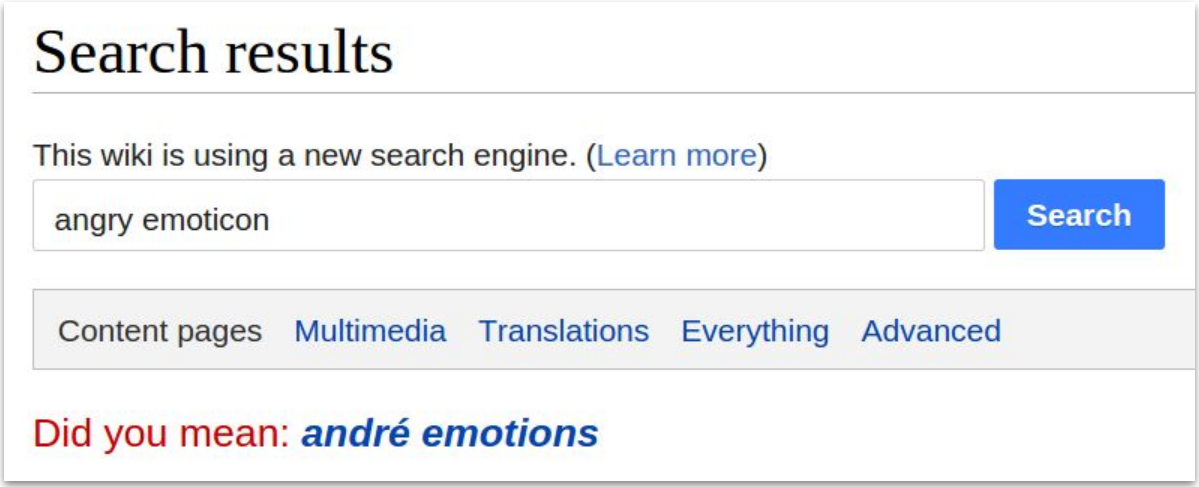
Search Engine: String Search



Average: O(n+m); Worst case: O(mn)

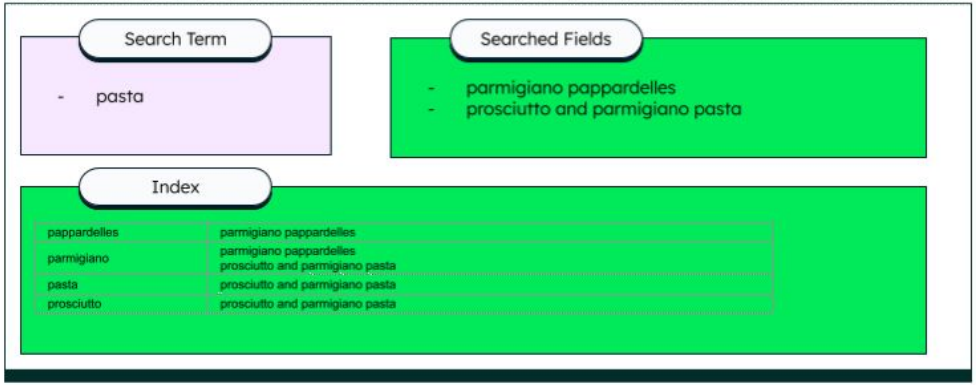
Rabin-Karp algorithm, which looks for matching substrings, is fast and easy to implement.
Knuth-Morris-Pratt algorithm looks for all instances of a matching character, increasing the speed for multiple matches in a string.

Search Engine: Fuzzy Search

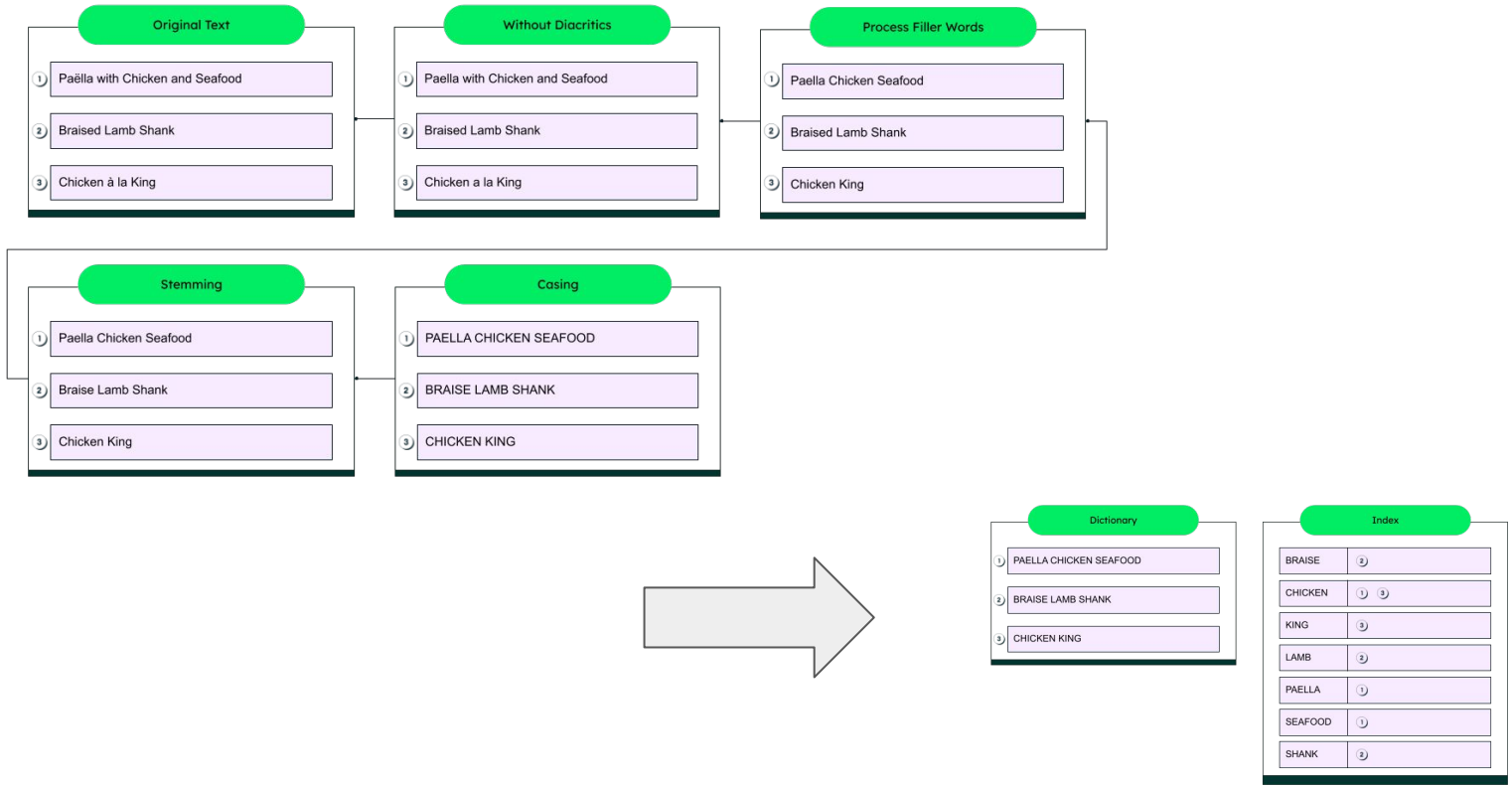


- insertion: *cot* → *co**a**t*
- deletion: *co**a**t* → *cot*
- substitution: *co**a**t* → *co**s**t*

Search Engine: Full-text Search - Inverted Index



Search Engine: Full-text Search - Inverted Index



Search Engine: User-item Rating

- It has a search bar!
  - -> How to incorporate the user intention?
- The items are mal-defined at the first glance.
  - -> How to crawl the internet and store the items?
- User does not simply rate the search results!
  - -> How to assign user-item rating with user data?

Search Engine: PageRank

- So far, web pages are treated as individual documents.
- But there are *hyperlinks* between them!

Search Engine: PageRank

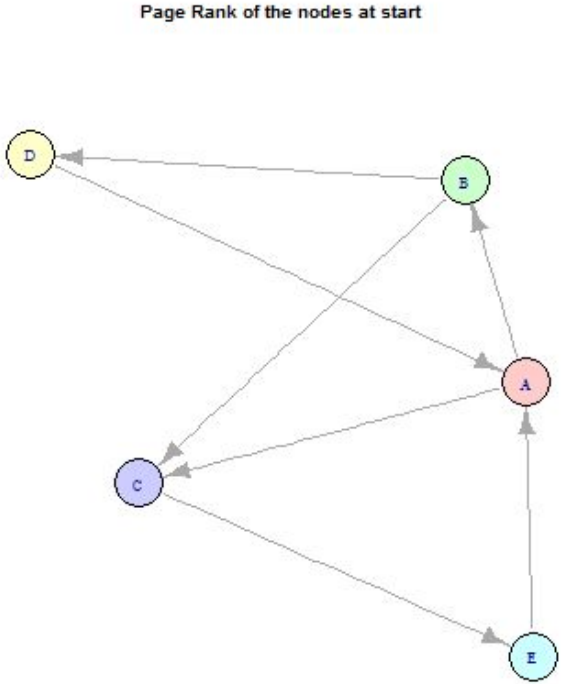
$$PR(p_i) = \frac{1-d}{N} + d \sum_{p_j \in M(p_i)} \frac{PR(p_j)}{L(p_j)}$$

$p_1, p_2, \ldots, p_N$ are the pages under consideration

$M(p_i)$ is the set of pages that link to p_i

$L(p_j)$ is the number of outbound links on page p_j

	Rank
A	0.2
B	0.2
C	0.2
D	0.2
E	0.2



Search Engine: User-item Rating from User Behavior

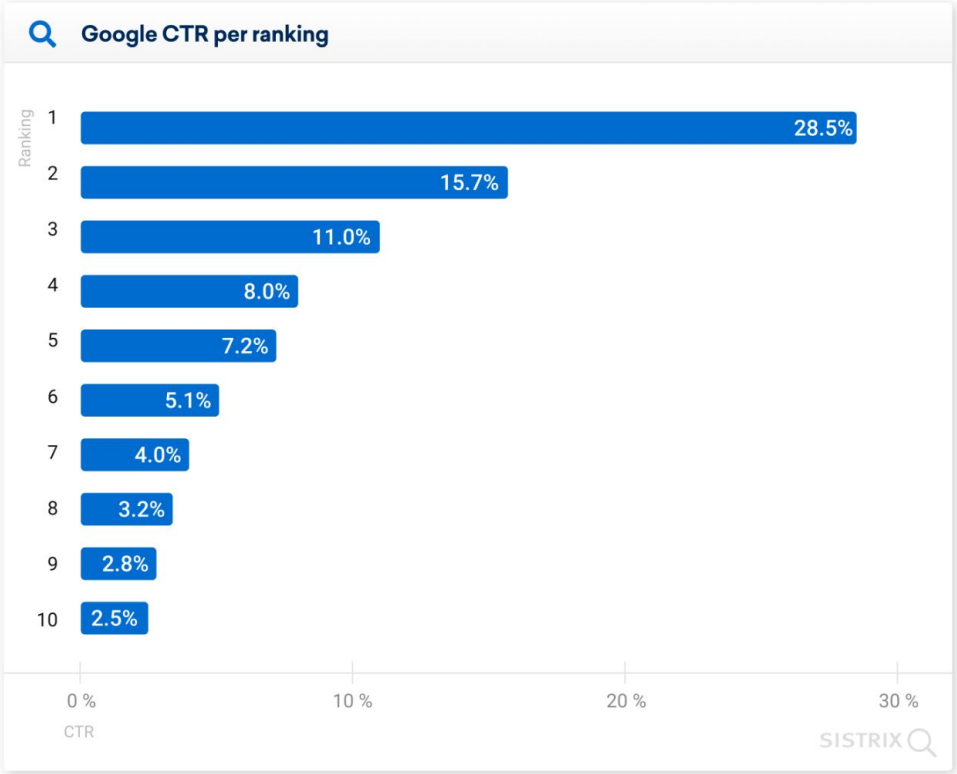
- We have millions of user clicking on some Search Engine Results every day.
- Can we assign click or not as ratings?
  - Yes and no

Search Engine: User-item Rating - Clicks

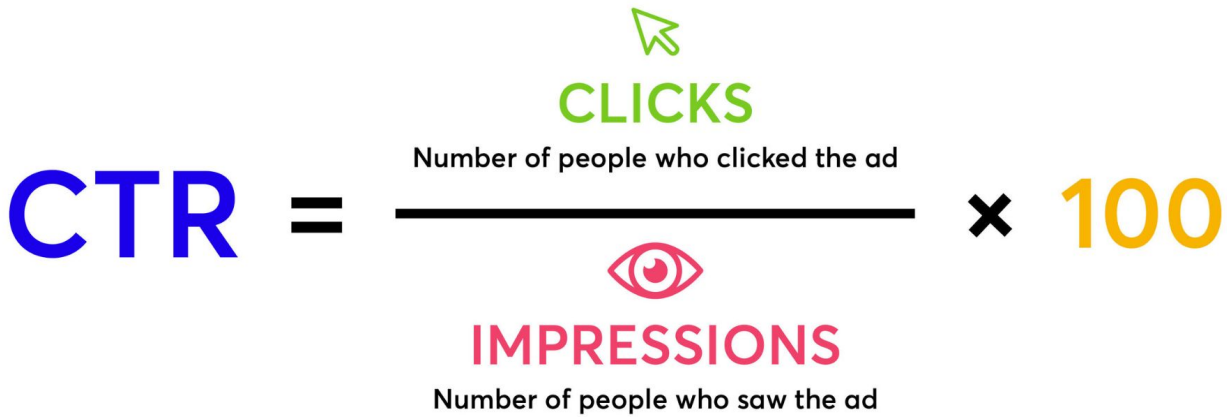
2005 2014

Source: The Evolution of Google Search Results Pages, Mediative, 2014

Search Engine: User-item Rating - Clicks



Search Engine: Evaluation - Clicks



Search Engine: Recap of Online Evaluation

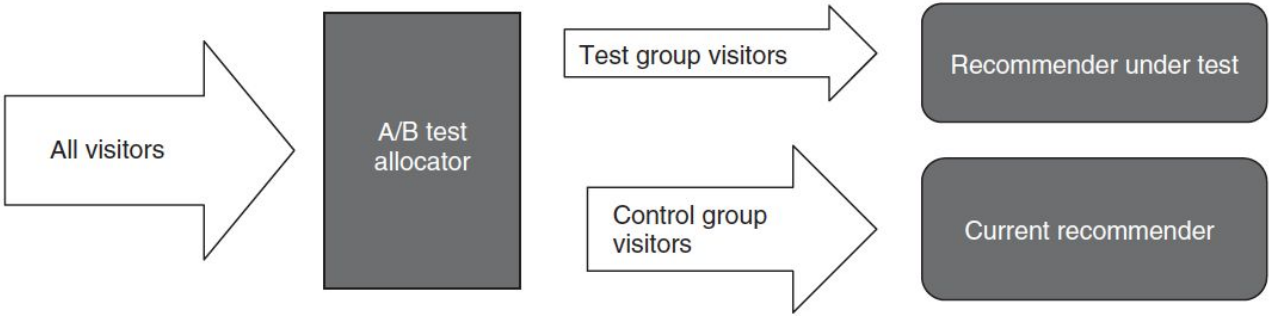


Figure 9.16 In an A/B test, visitors are split into two groups: the test group that sees the new feature and a control group that continues as usual.

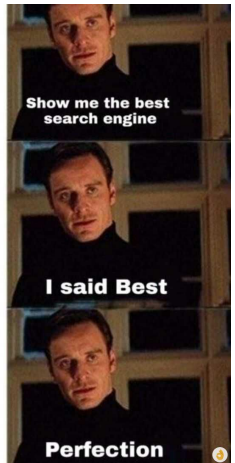
Search Engine: How to measure user satisfaction and success?

- **Q:** For Search Engine, how to measure the user satisfaction and success?
  - Any potential bias?
- **Q:** How about the recommendation system you chose yesterday?

Search Engine: Time-to-long-click (TTLC)

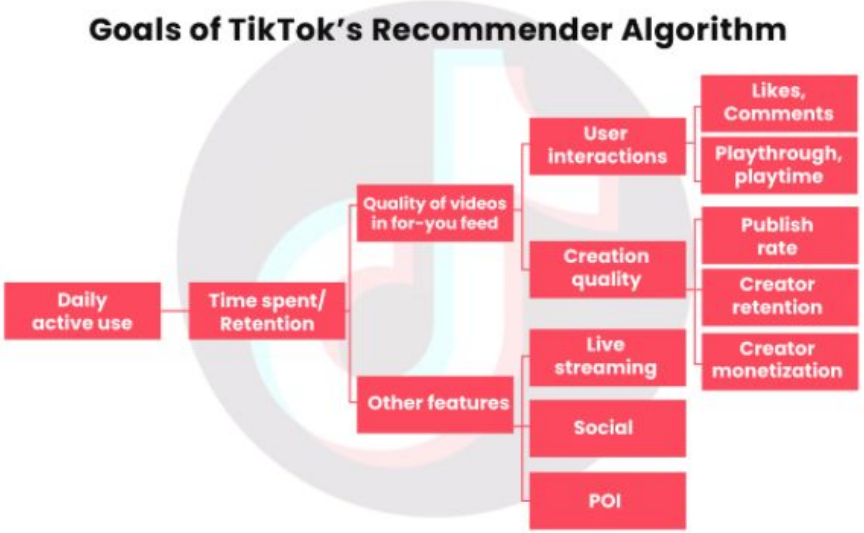
- Long-click: When a user performs a search, clicks through on a result and remains on that site for a long time.
  - Anti-pattern: Pogo-sticking
- Domain specific
- Knowledge panels and direct answers

bing

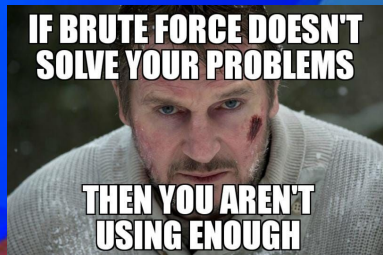


MOM

Tiktok: Multi-target



Advanced Topics of Recommendation System



Advanced Topics of Recommendation System

- Deep learning
- Scale up and speed up
- Cold start
- Responsive
- Social impact

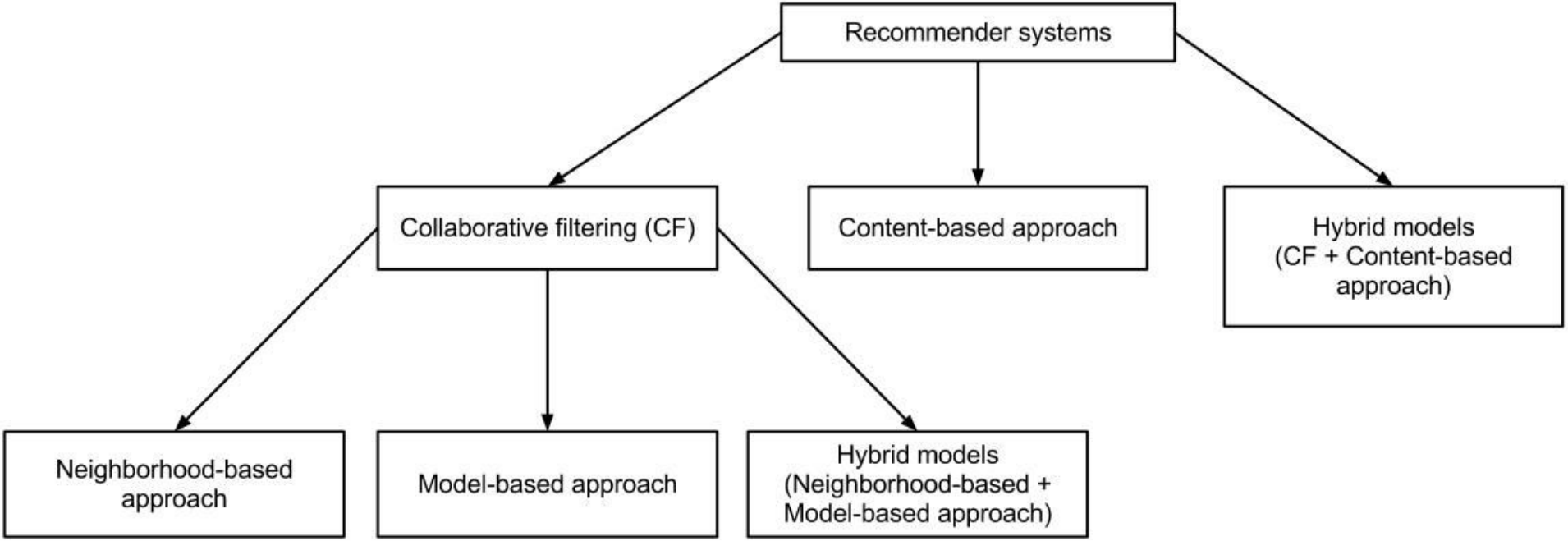
Advanced Topics of Recommendation System

- Deep learning

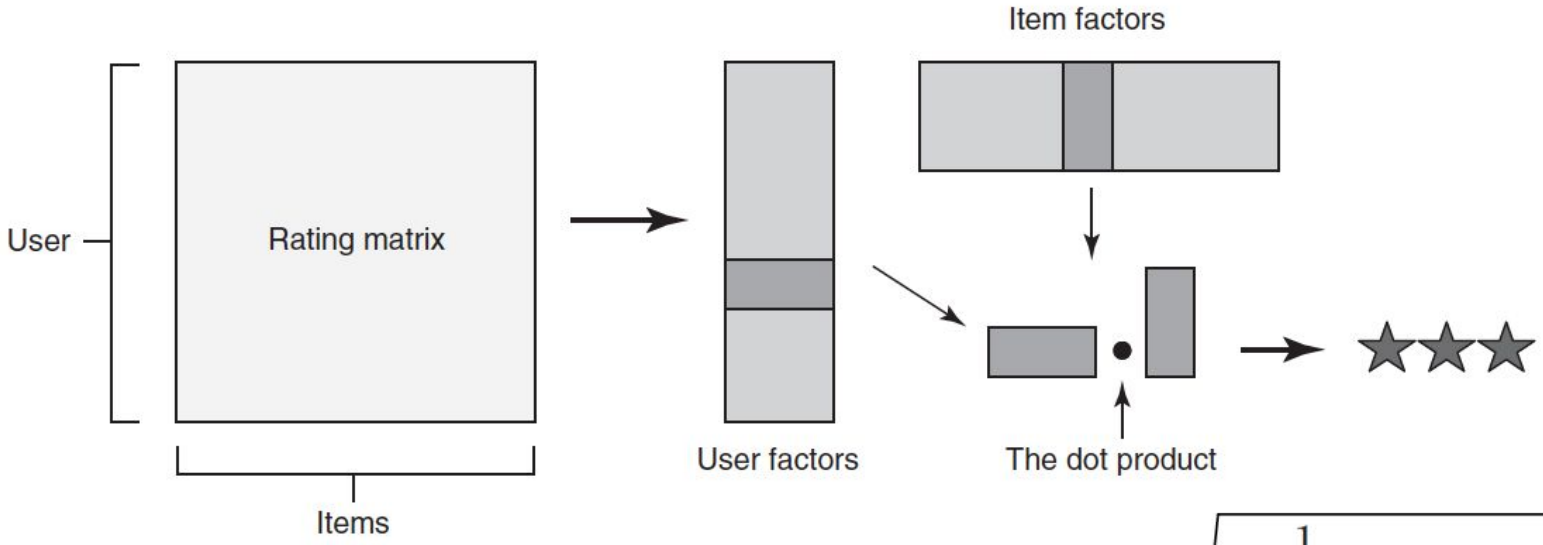
Break!

- Scale up and speed up
- LLM
- Social impact

Recommendation System w/ Deep Learning: Recap



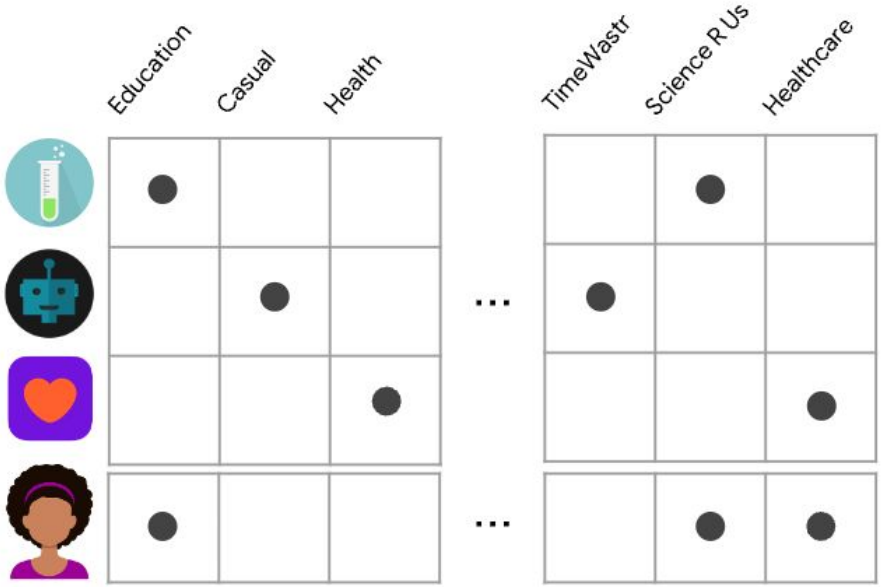
Recommendation System w/ Deep Learning: Recap



$$RMSE = \sqrt{\frac{1}{|known|} \sum_{(u,i) \in known} (r_{ui} - u_u v_i)^2}$$

- Rating/score can be modeled as a product of user vector and item vector.

Recommendation System w/ Deep Learning: Recap



- There are well-defined features (of both users and items) that can be used for the prediction.

RS w/ Deep Learning: Factorization Machines (2010)

- Factorization Machines (FM): Let’s combine the best of the both worlds!

Reformulate Collaborative Filtering

	i_1	i_2	i_3
u_1	2	4	
u_2		1	
u_3	3		5

							y
$x^{(0)}$	1	0	0	1	0	0	2
$x^{(1)}$	1	0	0	0	1	0	4
$x^{(2)}$	0	1	0	0	1	0	1
$x^{(3)}$	0	0	1	1	0	0	3
$x^{(4)}$	0	0	1	0	0	1	5

Observed Ratings

Users

Items

Reformulate Collaborative Filtering

	i_1	i_2	i_3
u_1	2	4	
u_2		1	
u_3	3		5

							y
$x^{(0)}$	1	0	0	1	0	0	2
$x^{(1)}$	1	0	0	0	1	0	4
$x^{(2)}$	0	1	0	0	1	0	1
$x^{(3)}$	0	0	1	1	0	0	3
$x^{(4)}$	0	0	1	0	0	1	5

Observed Ratings

Users Items

$$\hat{r}_{1,1} = w_0 + u_1^T v_1$$

$$\hat{r}_{1,1} = w_0 + \left(\begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} (u_1, u_2, u_3) \right)^T \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} (v_1, v_2, v_3)$$

Reformulate Collaborative Filtering

	i_1	i_2	i_3
u_1	2	4	
u_2		1	
u_3	3		5

							y
$x^{(0)}$	1	0	0	1	0	0	2
$x^{(1)}$	1	0	0	0	1	0	4
$x^{(2)}$	0	1	0	0	1	0	1
$x^{(3)}$	0	0	1	1	0	0	3
$x^{(4)}$	0	0	1	0	0	1	5

Observed Ratings

Users Items

$$\hat{r}_{1,1} = w_0 + u_1^T v_1$$

$$\hat{r}_{1,1} = w_0 + \sum_{i=1}^{n} \sum_{j=i+1}^{n} x_i^{(0)} x_j^{(0)} u_i^T v_j$$

Reformulate Content Filtering

	i_1	i_2	i_3
u_1	2	4	
u_2		1	
u_3	3		5

E.g. News Popularity

	i_1	i_2	i_3	a_1	a_2	y
$x^{(0)}$	1	0	0	2.0	0.0	2
$x^{(1)}$	0	1	0	1.5	0.5	4
$x^{(2)}$	0	1	0	1.5	0.5	1
$x^{(3)}$	1	0	0	2.0	0.0	3
$x^{(4)}$	0	0	1	3.2	1.7	5

Observed Ratings

Items

Auxiliary Features

$$\hat{r}_{1,1} = w_0 + w_1 a_1 + w_2 a_2$$

Back to Factorization Machines (FM)

	i_1	i_2	i_3
u_1	2	4	
u_2		1	
u_3	3		5

$x^{(0)}$	1	0	0	1	0	0	2.0	0.0	2
$x^{(1)}$	1	0	0	0	1	0	1.5	0.5	4
$x^{(2)}$	0	1	0	0	1	0	0.0	1.0	1
$x^{(3)}$	0	0	1	1	0	0	0.3	0.7	3
$x^{(4)}$	0	0	1	0	0	1	3.2	1.7	5

Observed Ratings

Users | Items | Auxiliary Features

$$\hat{r} = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} x_i x_j v_i^T v_j$$

Back to Factorization Machines (FM)

$$\hat{r} = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} x_i x_j v_i^T v_j$$

- w_0: Global bias
- w_i: Weights of features
- v_i: Feature factor i

=> Supervised learning paradigm!!

Back to Factorization Machines (FM)

$$\hat{r} = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} x_i x_j v_i^T v_j$$

- w_0: Global bias
- w_i: Weights of features
- v_i: Feature factor i

=> Supervised learning paradigm!!

Rating Matrix

Collaborative Filtering

FM

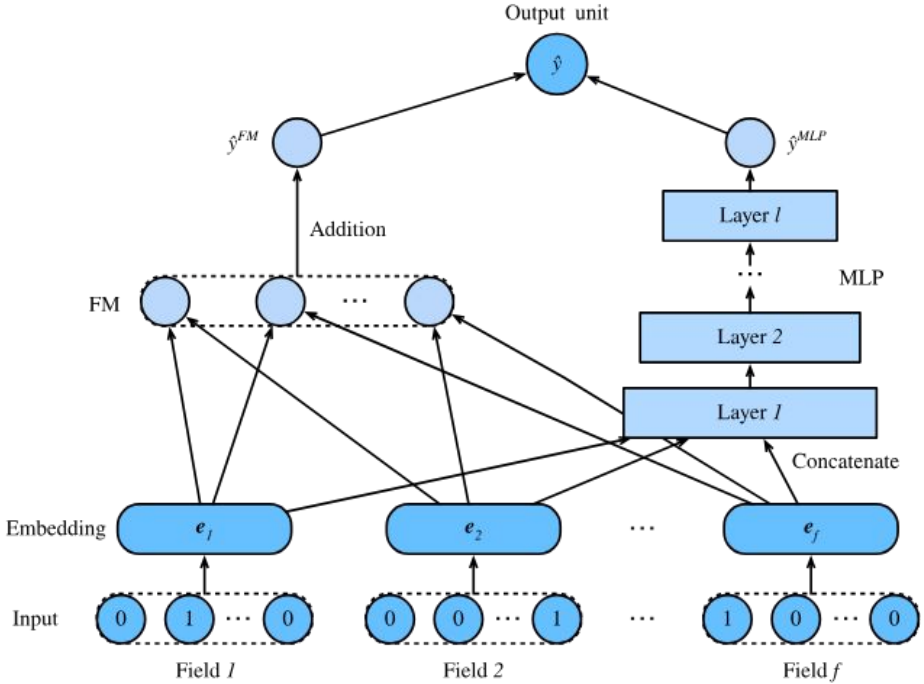
Welcome to the Deep-world

Factorization Machines (FM): Pros & Cons

- Capture user-item feature interaction
- Efficient for sparse data

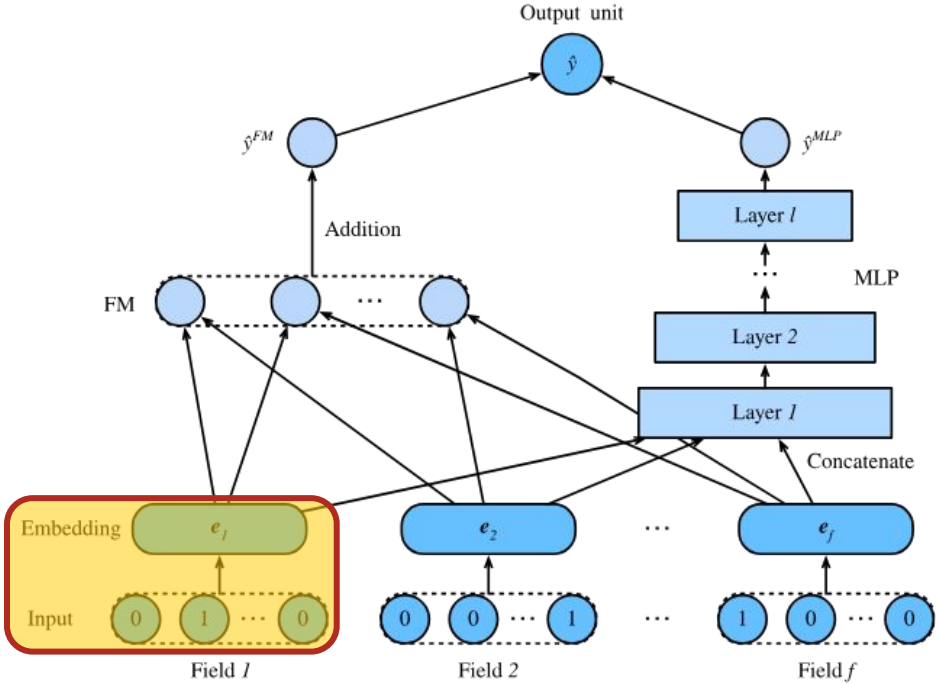
- Non-linear patterns

DeepFM (2017)


Output unit
$\hat{y}$
$\hat{y}^{FM}$
$\hat{y}^{MLP}$
Addition
Layer l
MLP
FM
Layer 2
Layer 1
Concatenate
Embedding
e_1
e_2
e_f
Input
0 1 ⋯ 0
0 0 ⋯ 1
1 0 ⋯ 0
Field 1
Field 2
Field f

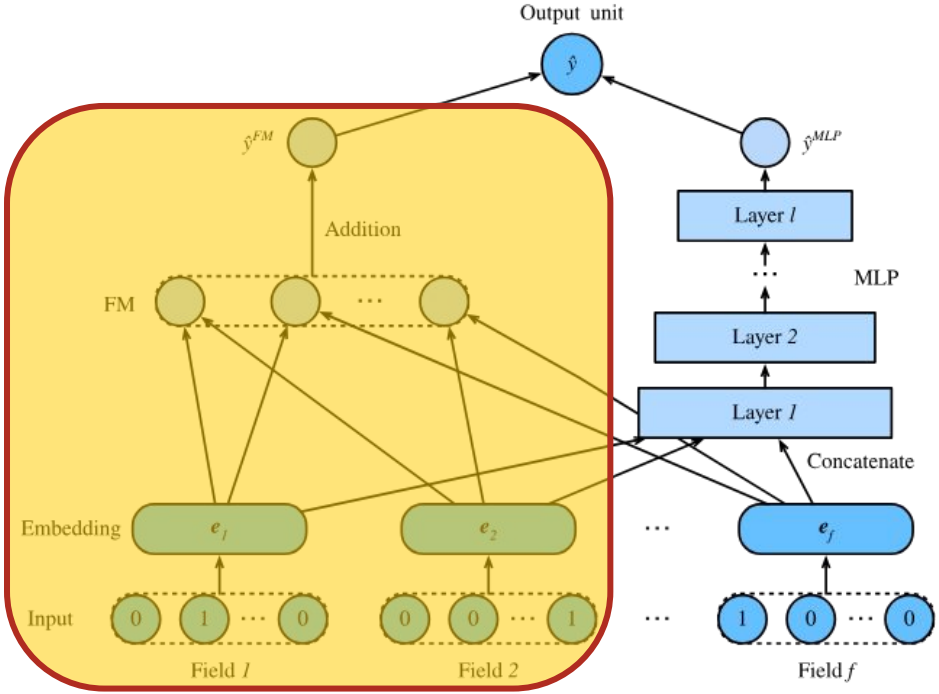
DeepFM (2017)

Factors = Embeddings

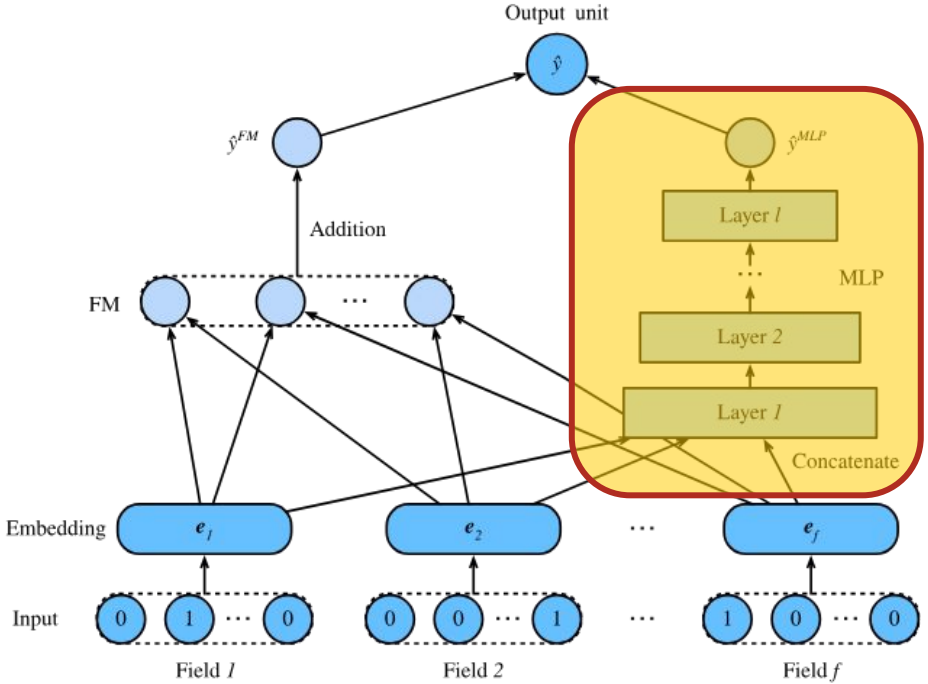


DeepFM (2017)

Factorization Machines

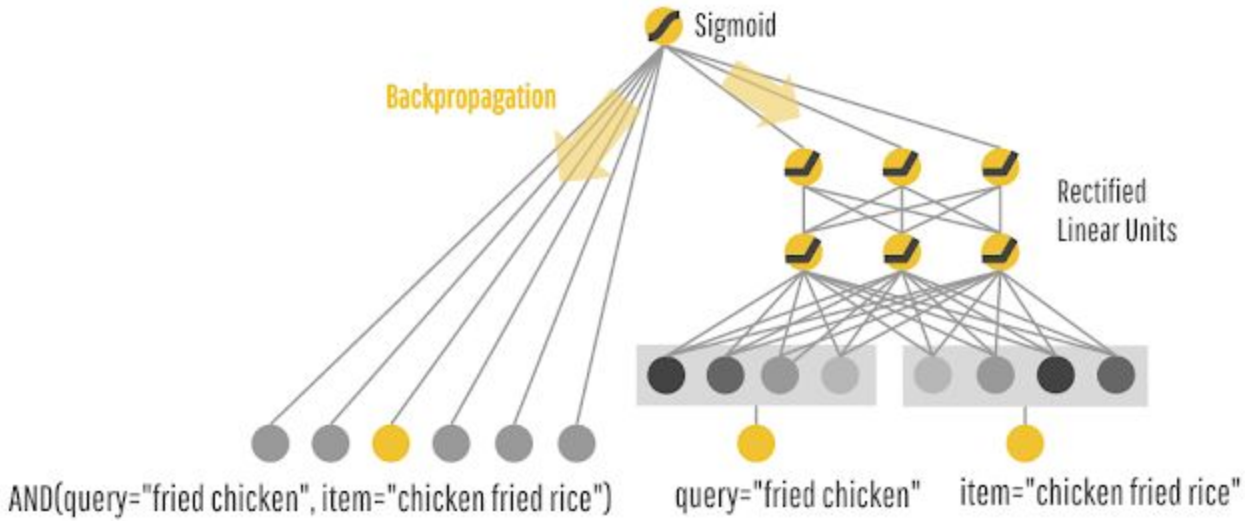


DeepFM (2017)

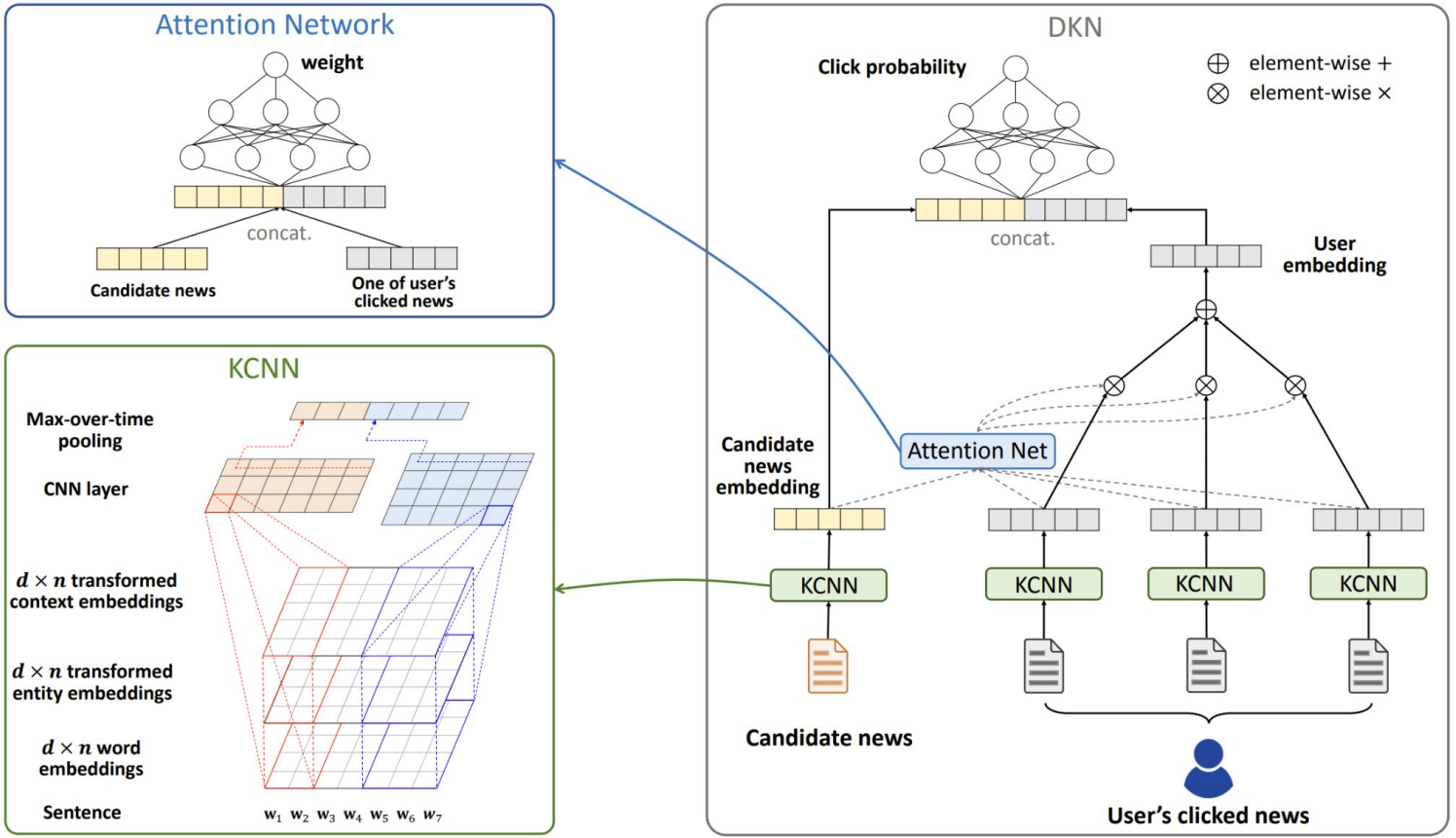


Deep MLP

Wide & Deep (2016)



Deep Knowledge-Aware Network (2018)



Feature Engineering for Deep Recommendation

- Sparse Features: Tag, device, etc.
- Dense Features: Age, income, # of videos watched, etc.
- **Sparse Features** are the first-class citizens.

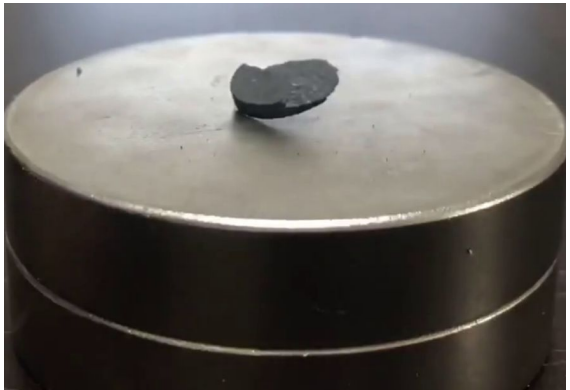
Feature Engineering for Deep Recommendation

- **Sparse Features** are the first-class citizens.
  - Dense feature requires more parameterization.
    - Age -> like this YouTube Video
      - Assumption: Age = 9 -> Age = 10 has a same effect of Age = 43 -> Age = 44

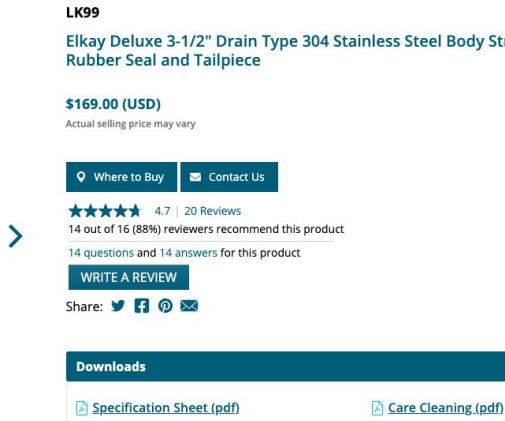
Feature Engineering for Deep Recommendation

- **Sparse Features** are the first-class citizens.
  - Dense feature requires more parameterization.
  - Sparse features are easy to generate cross-features.
    - Hashing tricks: Only keep K more frequent tuples of the combinations.

#lk99 #superconductor



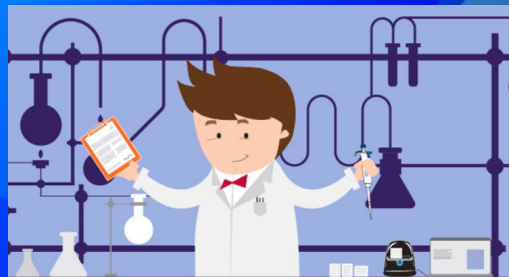
#lk99 #drinking-solution



Feature Engineering for Deep Recommendation

- **Sparse Features** are the first-class citizens.
  - Dense feature requires more parameterization.
  - Sparse features are easy to generate cross-features.
  - * Easy for online training/serving

Lab Time!



Advanced Topics (Part II)

Advanced Topics of Recommendation System

- Deep learning
- **Scale up and speed up**
- Cold start
- Responsive
- Social impact

Scale up & Speed up: Speed Matters

*All other things being equal ... our experiments demonstrate that slowing down the search results page by **100 to 400** milliseconds has a measurable impact on the number of searches per user of **-0.2%** to **-0.6%**. That's 0.2% to 0.6% fewer searches for changes under half a second!*

-- Google Research Blog

Speed-up: Recommendation Funnel

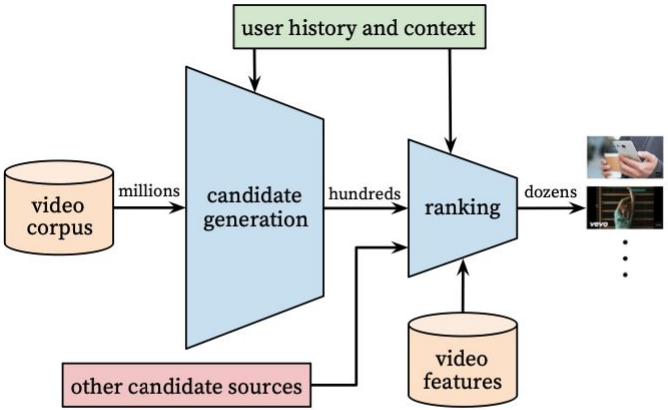
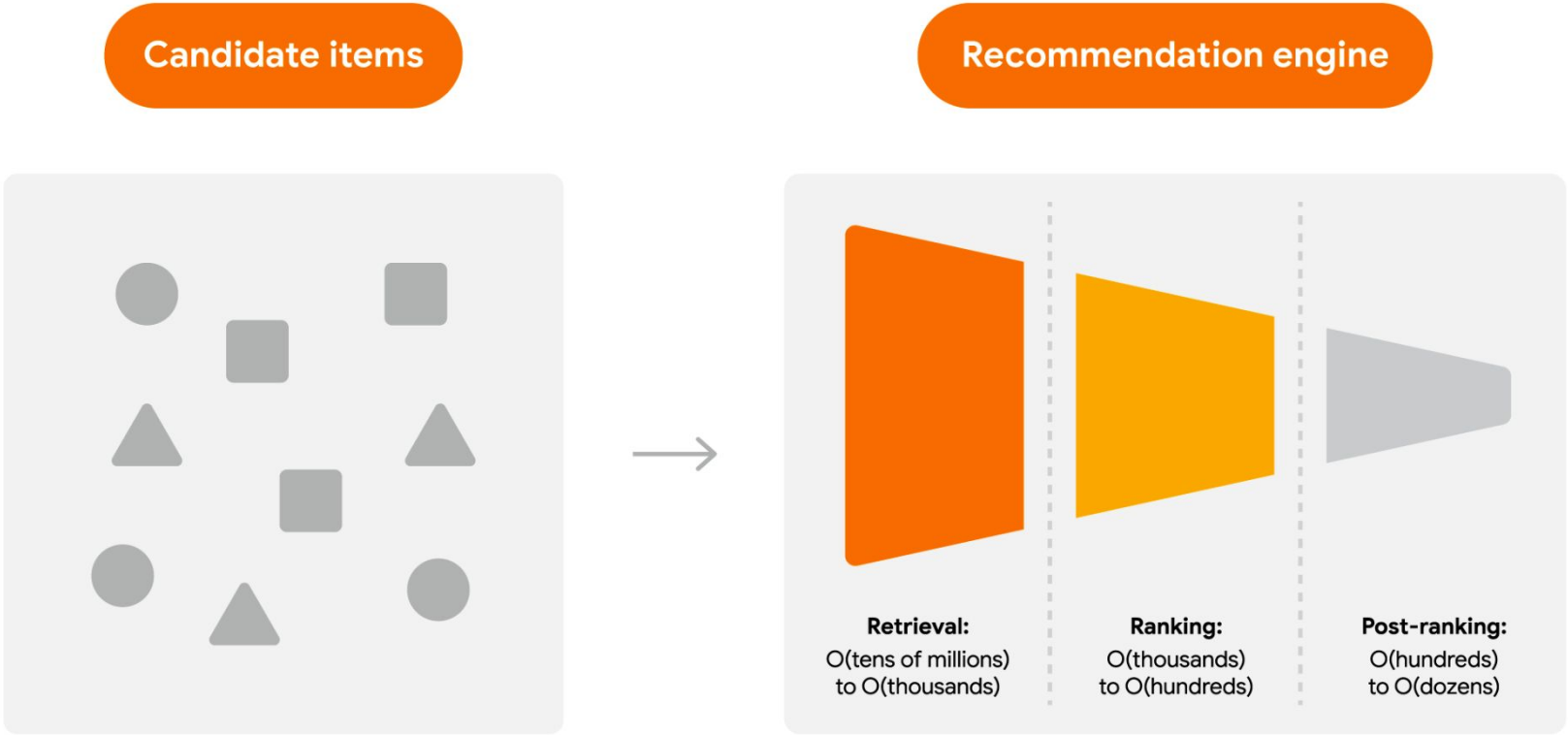


Figure 2: Recommendation system architecture demonstrating the "funnel" where candidate videos are retrieved and ranked before presenting only a few to the user.

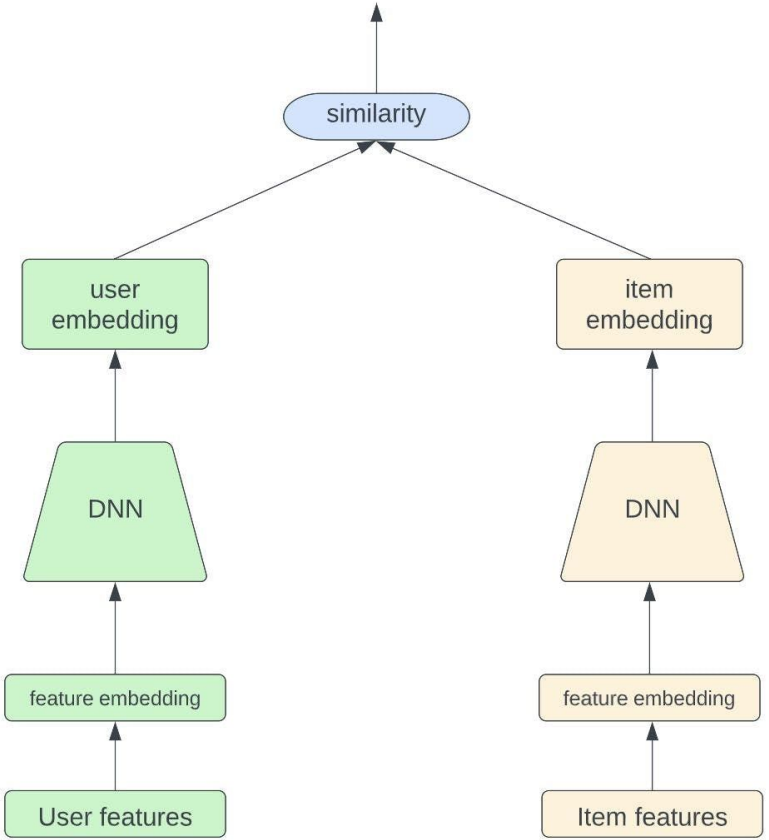
- Retrieval / Candidate-gen
- Ranking / Sort
- Re-rank

Covington, Paul, Jay Adams, and Emre Sargin. "Deep neural networks for youtube recommendations." Proceedings of the 10th ACM conference on recommender systems. 2016.

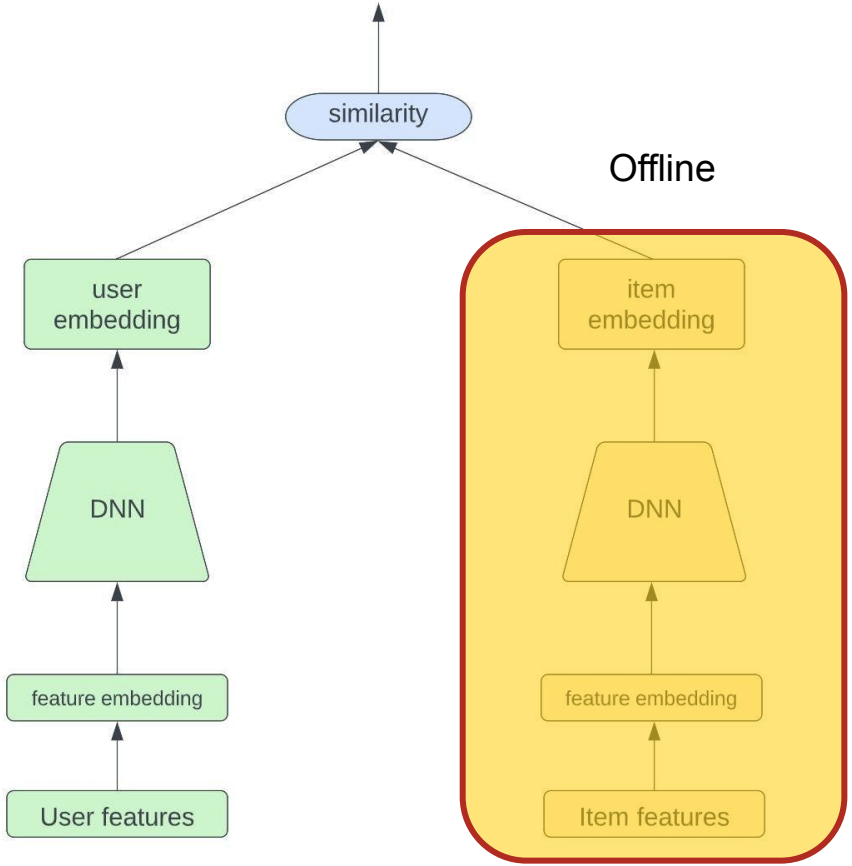
Speed-up: Recommendation Funnel

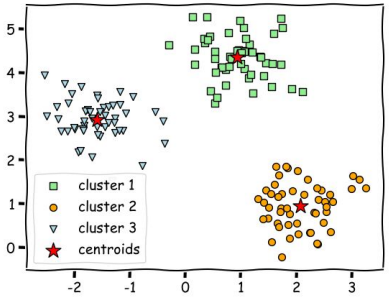


Speed-up: Two-tower Retrieval



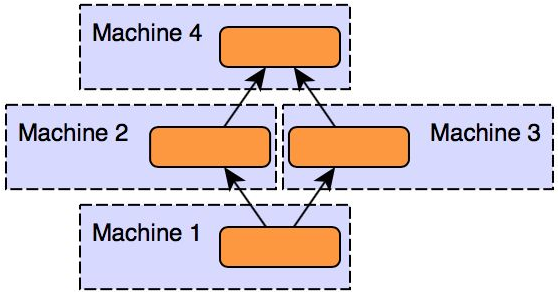
Speed-up: Two-tower Retrieval



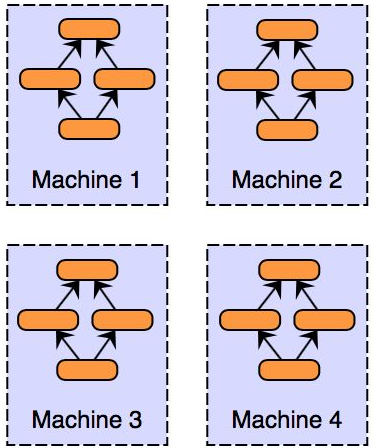


Scale-up: Parameter Server (PS)

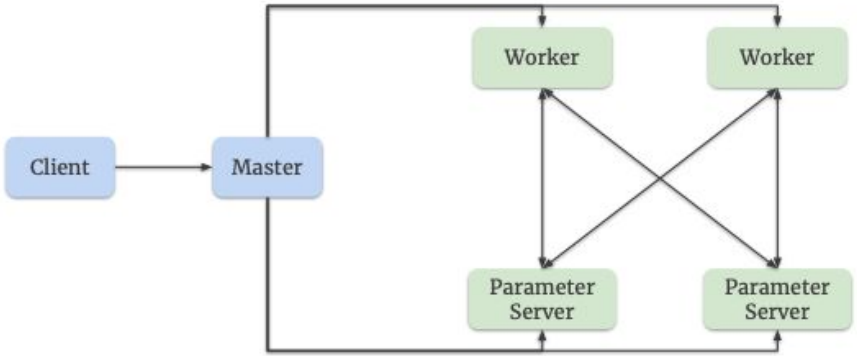
Model Parallelism



Data Parallelism



Scale-up: Parameter Server (PS)



K, V
storage

Li, Mu, et al. "Communication efficient distributed machine learning with the parameter server." Advances in Neural Information Processing Systems 27 (2014).

Algorithm 1 Distributed Subgradient Descent

Task Scheduler:

issue LoadData() to all workers
for iteration $t = 0, \ldots, T$ **do**
    issue WORKERITERATE(t) to all workers.
end for

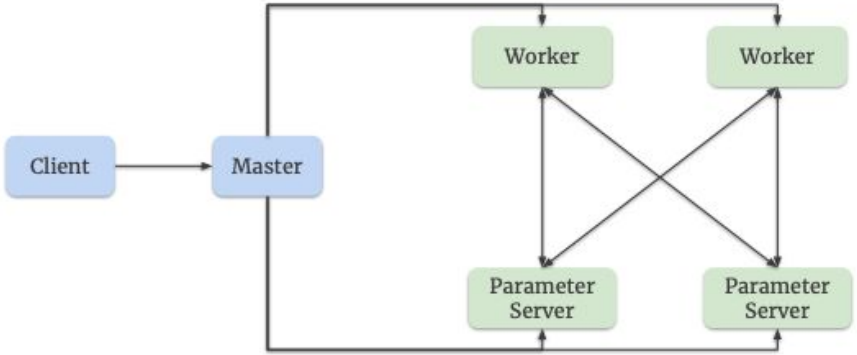
Worker $r = 1, \ldots, m$:

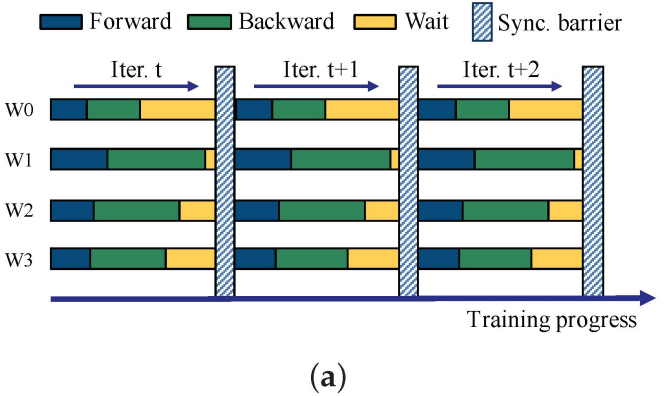
function LOADDATA()
    load a part of training data $\{y_{i_k}, x_{i_k}\}_{k=1}^{n_r}$
    pull the working set $w_r^{(0)}$ from servers
end function
function WORKERITERATE(t)
    gradient $g_r^{(t)} \leftarrow \sum_{k=1}^{n_r} \partial \ell(x_{i_k}, y_{i_k}, w_r^{(t)})$
    push $g_r^{(t)}$ to servers
    pull $w_r^{(t+1)}$ from servers
end function

Servers:

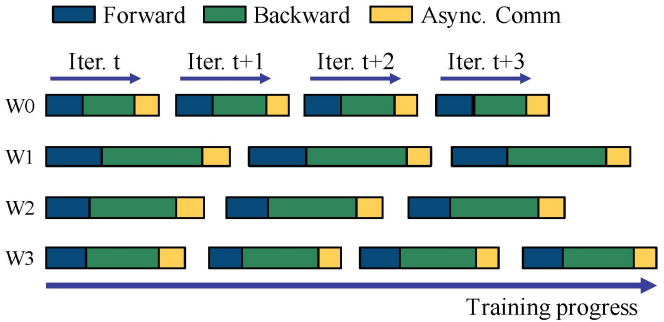
function SERVERITERATE(t)
    aggregate $g^{(t)} \leftarrow \sum_{r=1}^{m} g_r^{(t)}$
    $w^{(t+1)} \leftarrow w^{(t)} - \eta \left(g^{(t)} + \partial \Omega(w^{(t)})\right)$
end function

Scale-up: Parameter Server (PS)





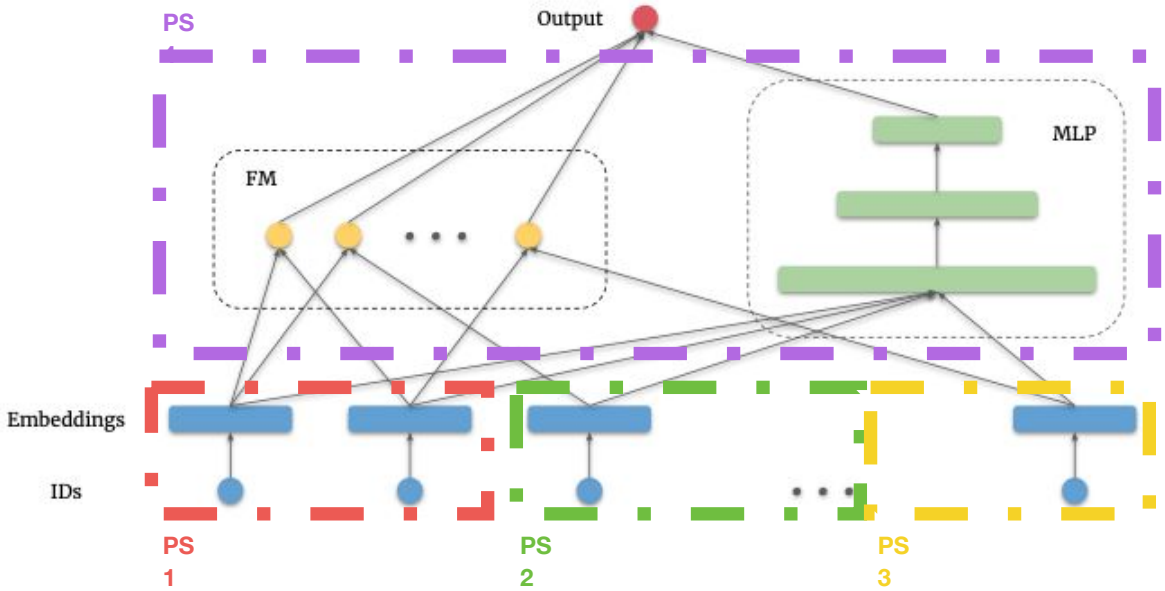
(a)



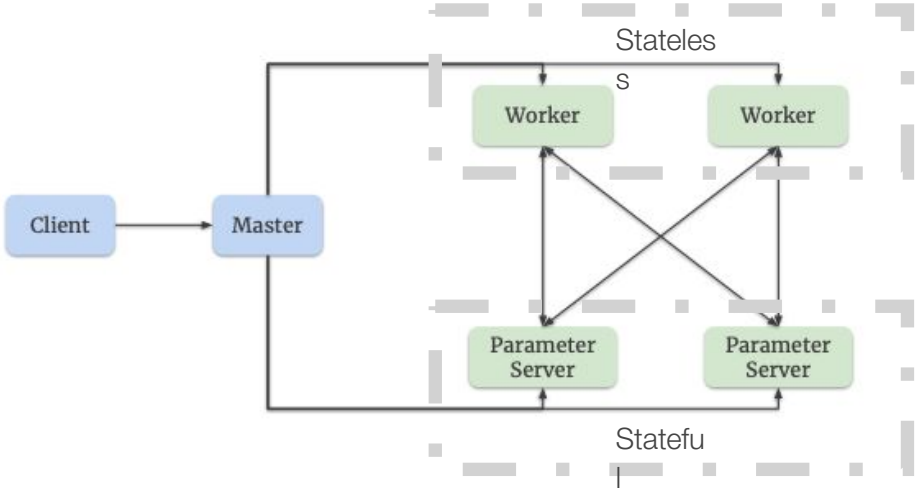
(b)

Li, Mu, et al. "Communication efficient distributed machine learning with the parameter server." Advances in Neural Information Processing Systems 27 (2014).

Scale-up: Parameter Server (PS)

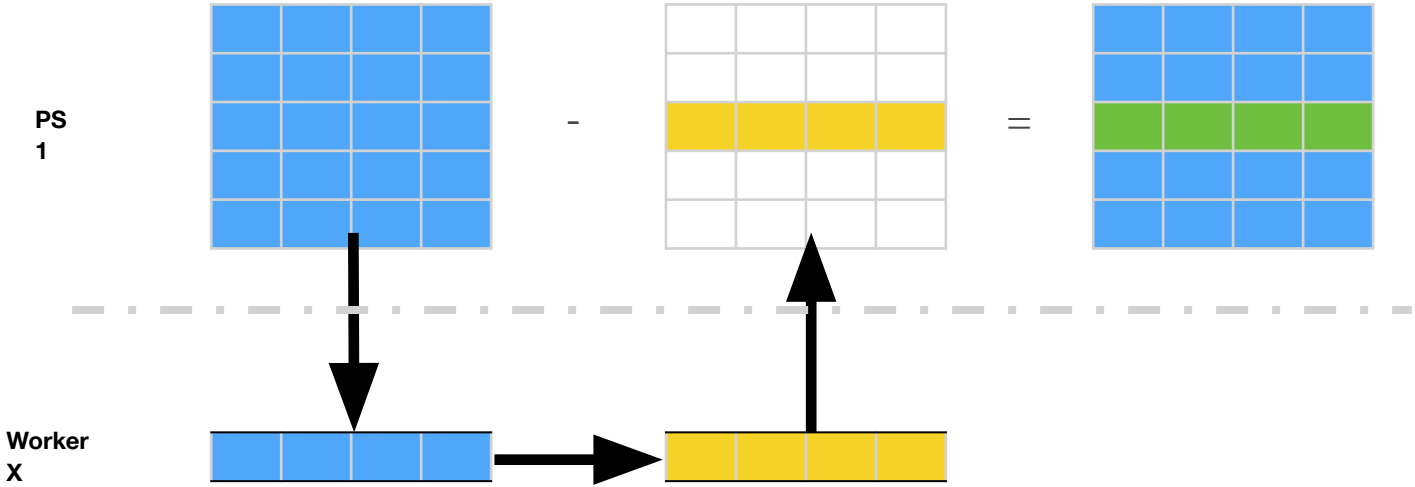


Scale-up: Parameter Server (PS)

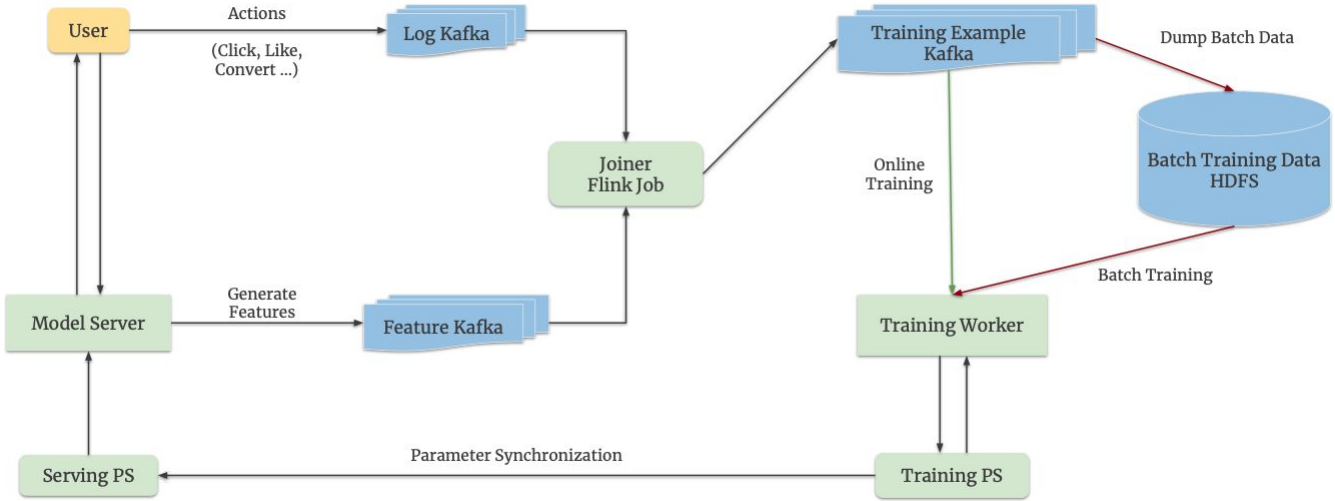


Li, Mu, et al. "Communication efficient distributed machine learning with the parameter server." Advances in Neural Information Processing Systems 27 (2014).

Scale-up: Parameter Server (PS)



Augmenting recommendation systems with LLMs (Large Language Model)



Liu, Zhuoran, et al. "Monolith: Real Time Recommendation System With Collisionless Embedding Table." arXiv preprint arXiv:2209.07663 (2022).

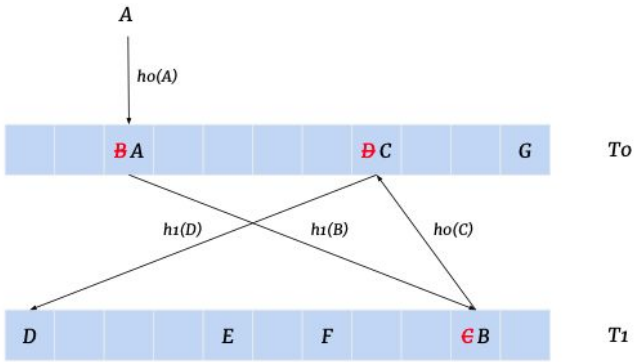
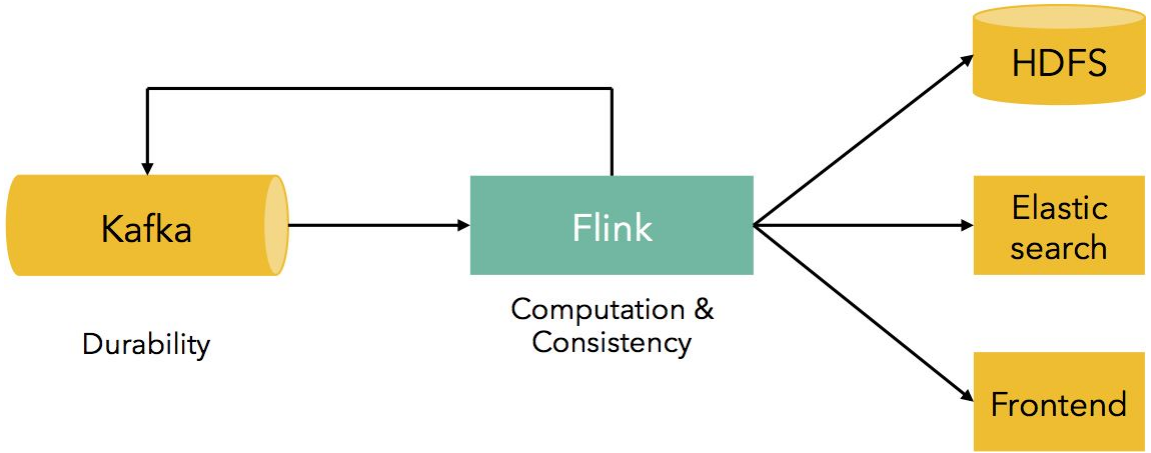
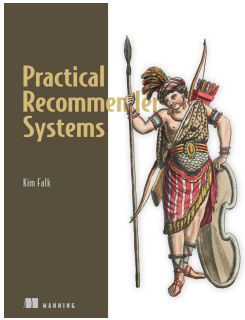


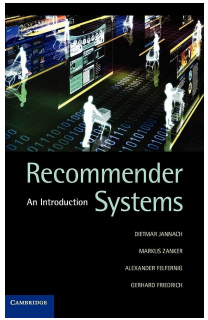
Figure 3: Cuckoo HashMap.

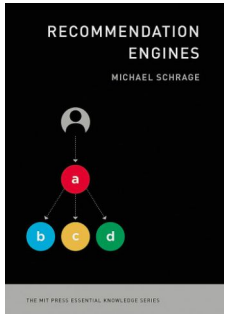
Liu, Zhuoran, et al. "Monolith: Real Time Recommendation System With Collisionless Embedding Table." arXiv preprint arXiv:2209.07663 (2022).

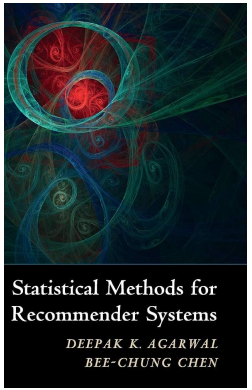

HDFS
Kafka
Flink
Elastic search
Frontend
Durability
Computation & Consistency

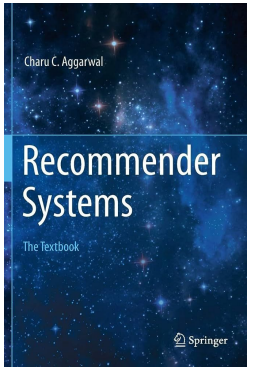
Reference







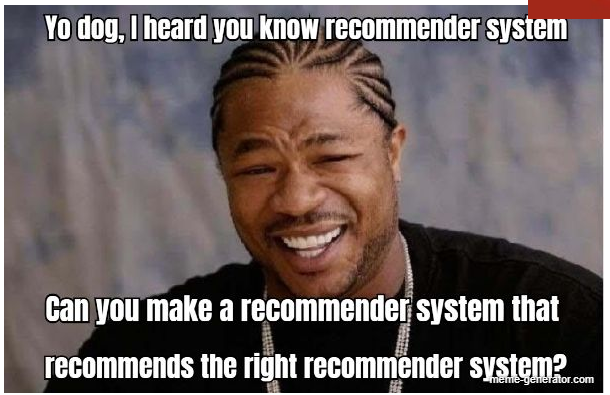




	PRS	IRS	RE	SERS	RST
Related Chap.	Chap. 7-11.	Chap. 2, 7	Chap. 1-5.	Chap. 1, 2, 4.	Chap 1-4.
Hao's Rating	5	3	4	2	4


RECOMMENDER SYSTEMS
RECOMMENDER SYSTEMS EVERYWHERE


SHUT UP AND TAKE MY MONEY!


Yo dog, I heard you know recommender system
Can you make a recommender system that recommends the right recommender system?
meme-generator.com


I SPEND MORE TIME LOOKING FOR A MOVIE TO WATCH
THAN WATCHING IT
zipmeme


CUSTOMERS WHO BOUGHT THIS ITEM:
ALSO BOUGHT:
piccolo


ME
I SEE NO REASON TO VACCINATE.
SOCIAL MEDIA ECHO CHAMBER
ANTI-VAXXER

How likely are you to recommend Windows 10 to a friend or colleague?

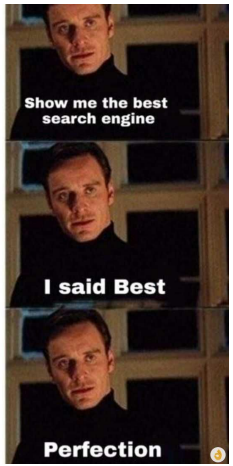
(•) 1 ( ) 2 ( ) 3 ( ) 4 ( ) 5

Not at all likely — Extremely likely

Please explain why you gave this score.

I need you to understand that people don't have conversations where they randomly recommend operating systems to one another

bing



MOM

Page 1 of google

Page 2 of google