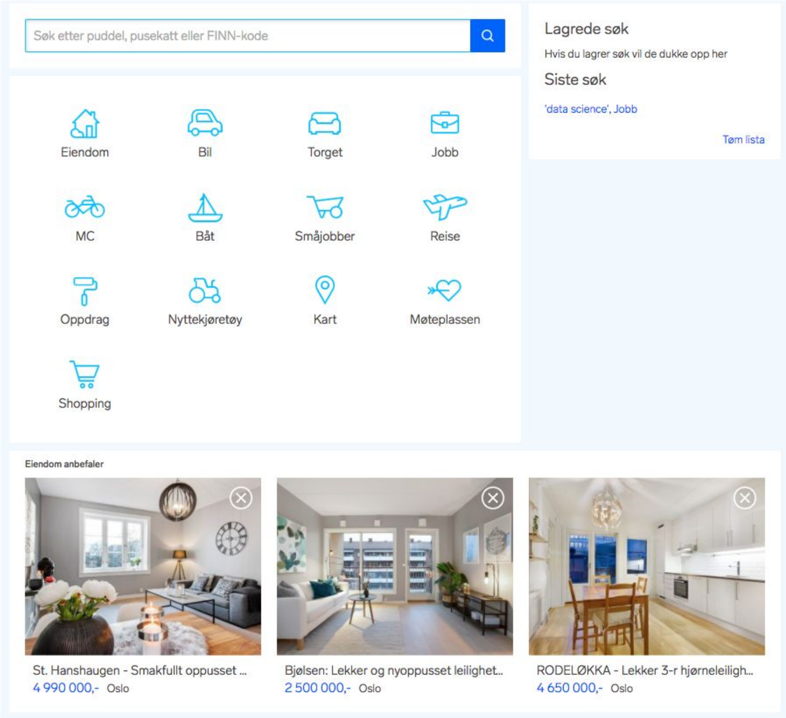
Recommendations @ FINN.no

Building and testing models

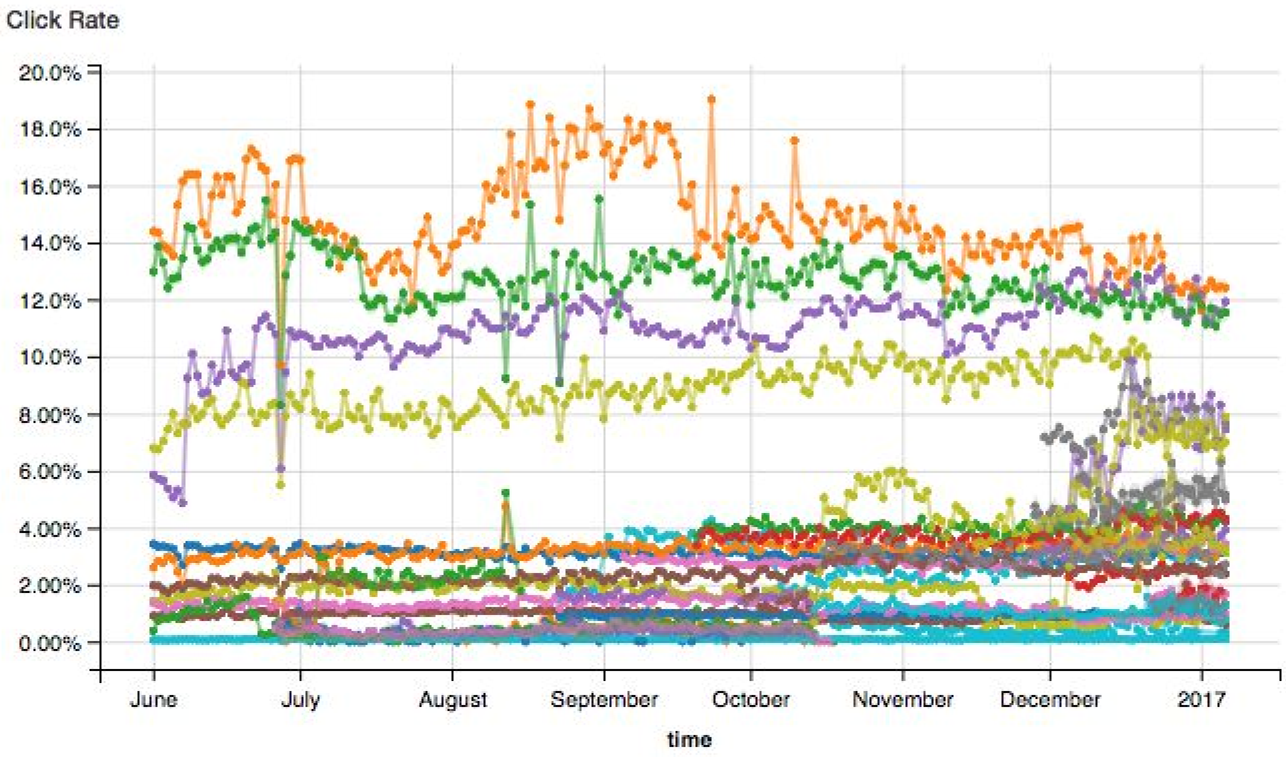
15 million daily recommendations

Internal



External

Click Rates



Click Rate
20.0%
18.0%
16.0%
14.0%
12.0%
10.0%
8.00%
6.00%
4.00%
2.00%
0.00%
June
July
August
September
October
November
December
2017
time

What is a successful Recommendation?

- CLICK
- CONVERSION
- DIVERSITY

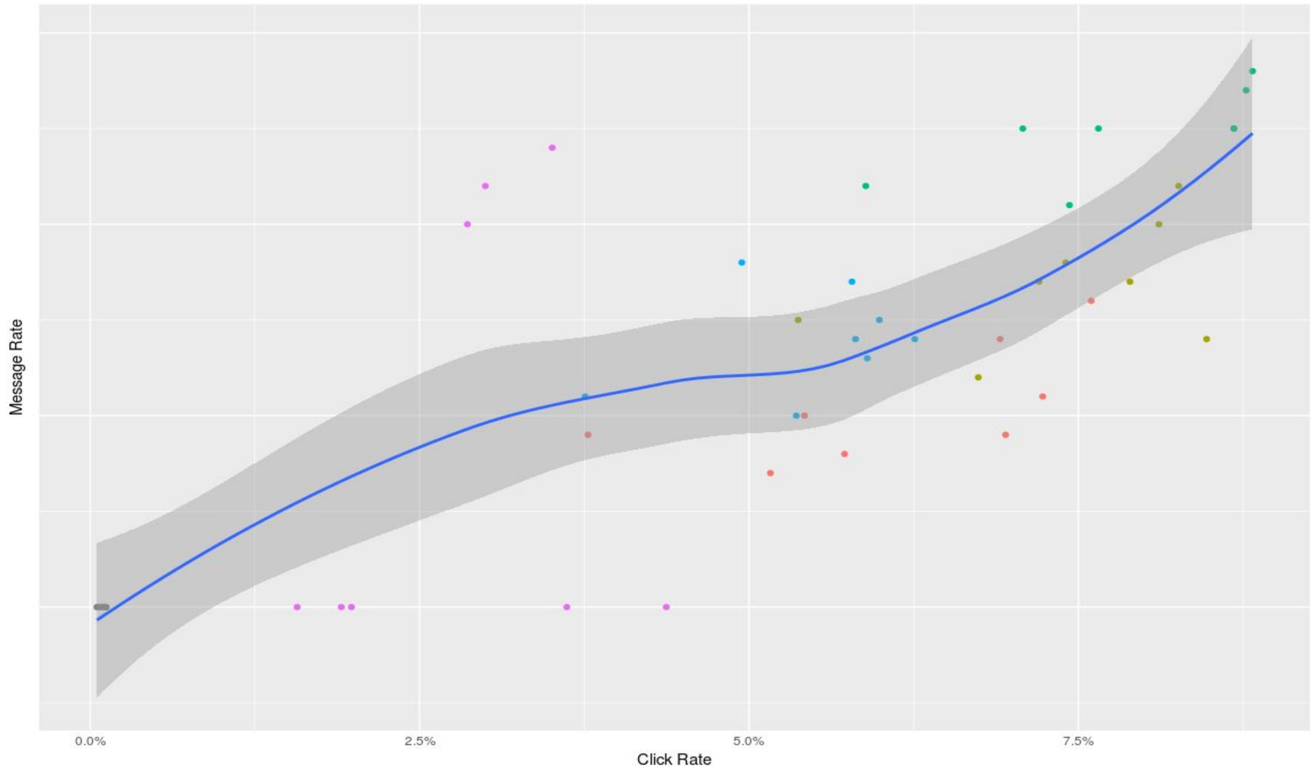
What is a successful Recommendation?

- CLICK
- CONVERSION
- DIVERSITY

FOCUS ON CLICK RATES:

$$\frac{\#\,\text{CLICKS}}{\#\,\text{VISITS}}$$

Click Rates vs Conversion



Test Procedure

1. Idea
2. Offline test (cheap)
3. Online test

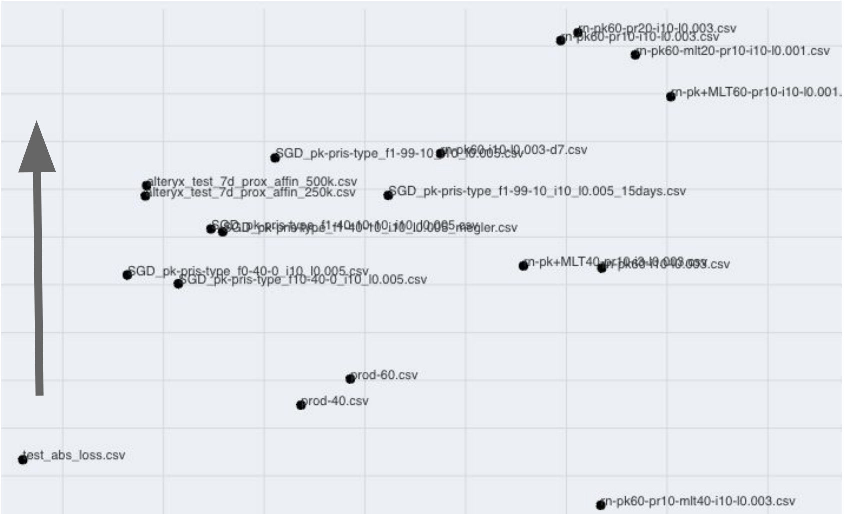
Offline Test

- Squared Errors are **great** for **optimizing**
- Squared Errors are **terrible** for **relevance testing**

- A metric we found with good CTR correlation:

Hitrate@6:

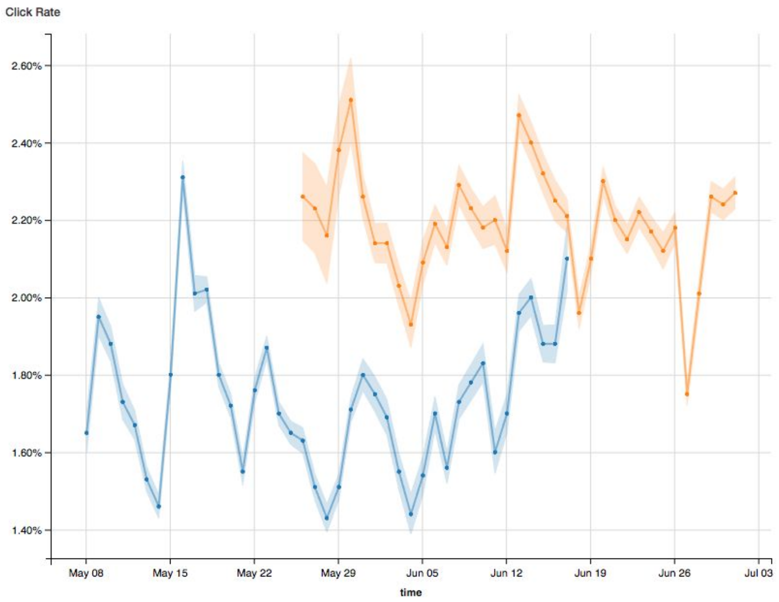
Recommended 6 items (not seen before). How many is visited by the user afterwards?

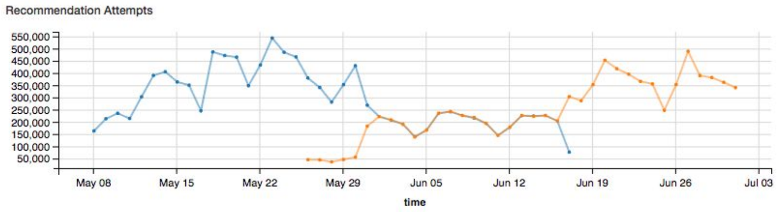


In the end... Only Online tests matter

Online Test: A/B test

- Turn on 10% traffic on challenger
- Does it seem to work? Increase
- Turn off worst model “when it seems obvious”
- If in doubt, keep it simple





Many simplifications:

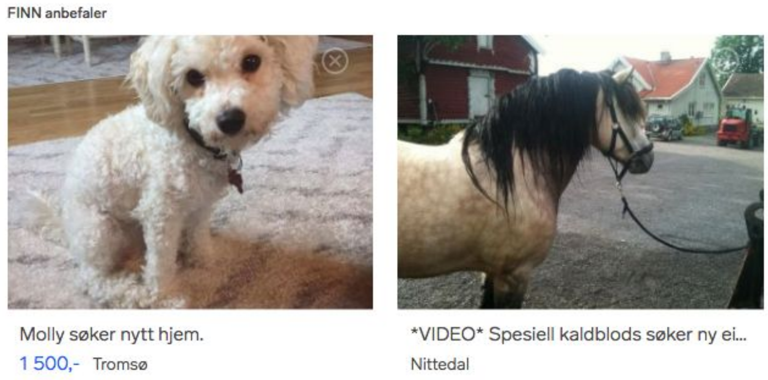
- No-sticky user
- Rest of finn is dynamic

Models

- Popular
- Content / Search
- Collaborative Filtering
- Factorization Machines

Previously Tried

- Popularity
- Random
- Jaccard Similarity
- Search-Based



A hard optimization problem

PROB(CLICK | SHOW SIMON.ITEM 1
ON FRONTPAGE
AT MON 10.00)

A simpler optimization problem

"$P(r_{ui})$"

$$r_{ui} = \begin{cases} 1 & \text{if RELEVANT} \\ 0 & \text{ELSE} \end{cases}$$

The optimization problem

SUM ALL USERS, ITEMS

IS IT RELEVANT?

$$\min_{\theta} \sum_{u,i} \left(r_{ui} - \hat{r}_{ui}(\theta) \right)^2$$

$\hat{r}_{ui}(\theta)$ IS YOUR MODEL.

During Recommendation...

ERIK	0.4	0.1	0.9
MARIA	0.06	0.93	0.1

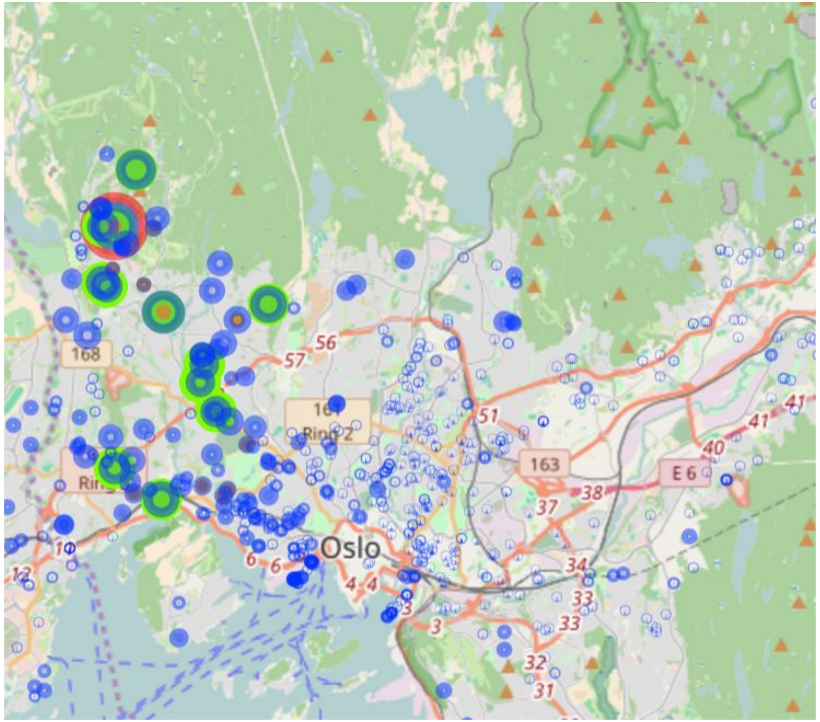
During Recommendation...

ERIK	0.4	0.1	0.9
MARIA	0.05	0.93	0.1

Collaborative Filtering

- Main engine: ALS from Spark
- Based on pageviews (ish)
- Works well out-of-box, better when tuned
- Use latent variables

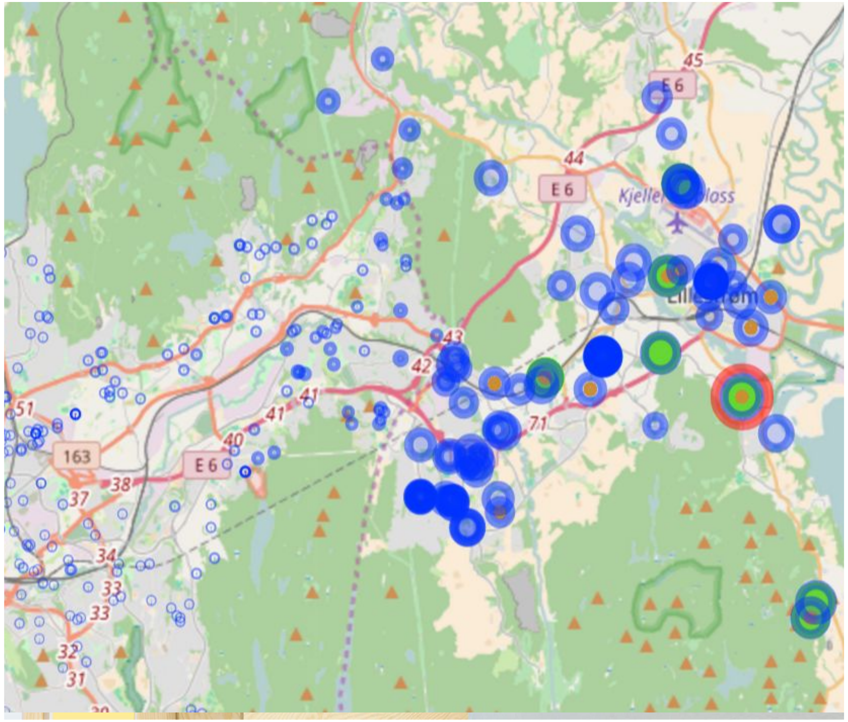
$$f(\text{USER}, \text{ITEM}) = X_u^T Y_i = \begin{pmatrix} 0.3 \\ 0.7 \\ 0.4 \\ 0.1 \end{pmatrix} \cdot \begin{pmatrix} 0.2 \\ 0.3 \\ 0.1 \\ 0.0 \end{pmatrix}$$



Collaborative Filtering

- Can also be used for “More Like This”
- Exchange user-parameters with candidate item-parameters

$$f(\text{ITEM}_2, \text{ITEM}) = X_{i_2}^T Y_i = \begin{pmatrix} 0.3 \\ 0.7 \\ 0.4 \\ 0.1 \end{pmatrix} \cdot \begin{pmatrix} 0.2 \\ 0.3 \\ 0.1 \\ 0.0 \end{pmatrix}$$



Some downsides:

- Cold-start
- Alternative signals
- Little flexibility

Alternative Models

- ALS in Spark has limitations on functional form
- Possible solutions:
  - Ensemble models
  - Pre- and post-processing
  - Content Models
  - Hybrid models

Final: Factorization Machines / SVD++

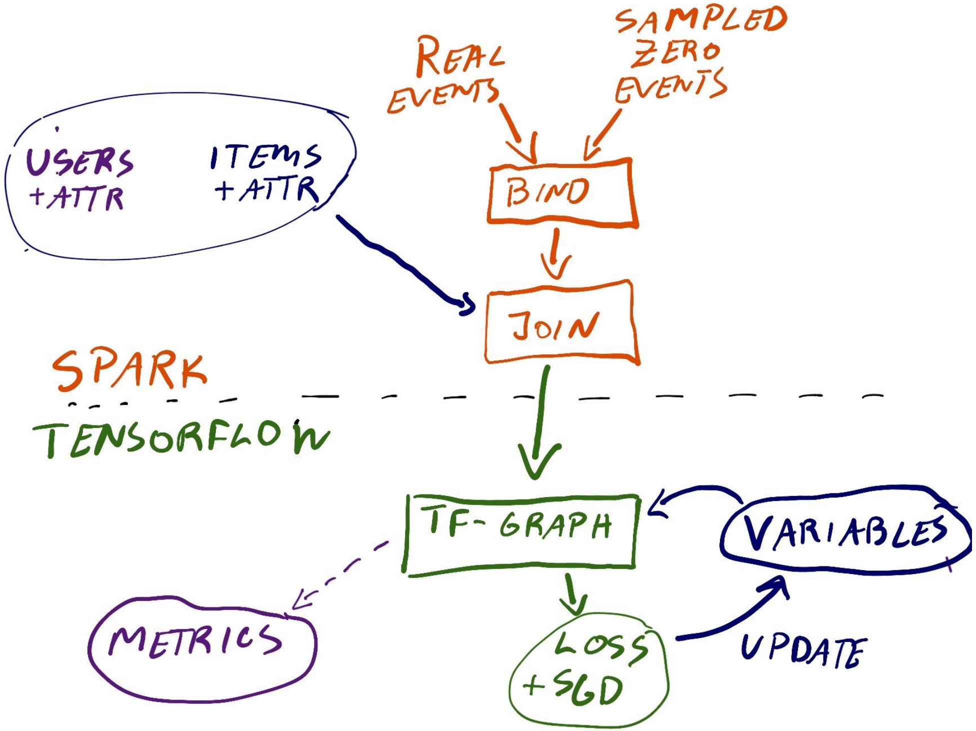
Why Tensorflow?

- Needed to build flexible models
- Need to fit parameters using Stochastic Gradient Descent
- Need to be able to run on large datasets

Tensorflow

- Auto-Differentiation
- “Out-of-box” Optimizers
- High degree of flexibility of functional form
- Support for distributed learning
- A bit too low-level here and there
- Alternative: Keras

Batchwise Training Procedure



Factorization Machines

EMBEDDED VECTORS

$$\text{SCORE} = \text{dot}(\text{USER}, \text{ITEM}) + \text{dot}(\text{USER}, \text{ITEM-POSTCODE}) + \text{dot}(\text{AGE\_USER}, \text{ITEM}) + \text{bias}_{\text{ITEM}} + \text{bias}_{\text{USER}}$$

Factorization Machines

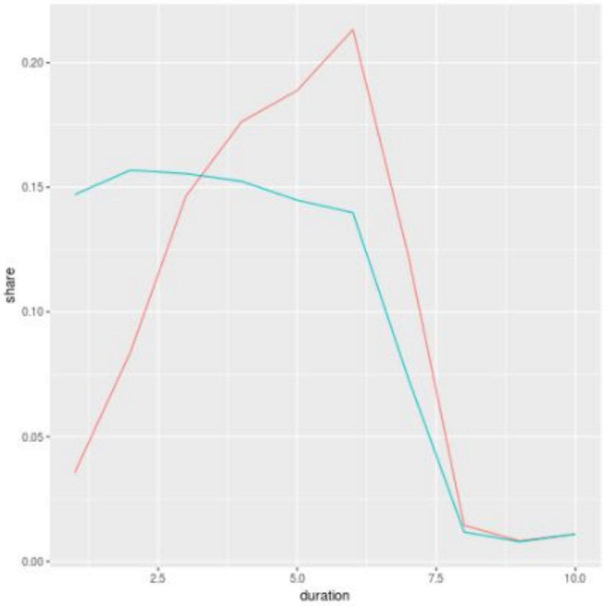
EMBEDDED VECTORS

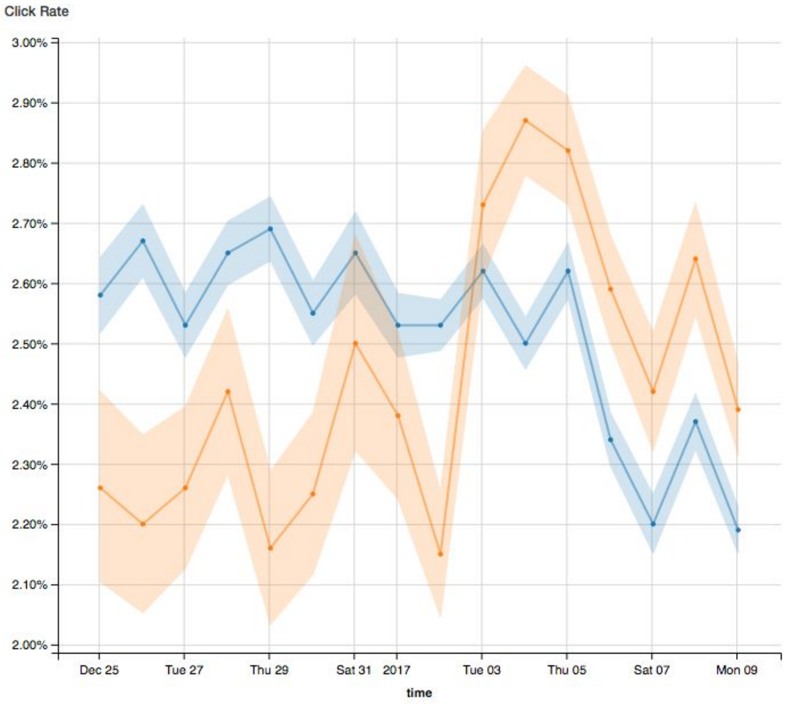
$$\text{SCORE} = \text{dot}(\text{USER}, \text{ITEM}) + \text{dot}(\text{USER}, \text{ITEM-POSTCODE}) + \text{dot}(\text{AGE\_USER}, \text{ITEM}) + \text{bias}_{\text{ITEM}} + \text{bias}_{\text{USER}}$$

$$\text{dot}(X, Y) := \text{TANH}(X^T Y)$$

Results

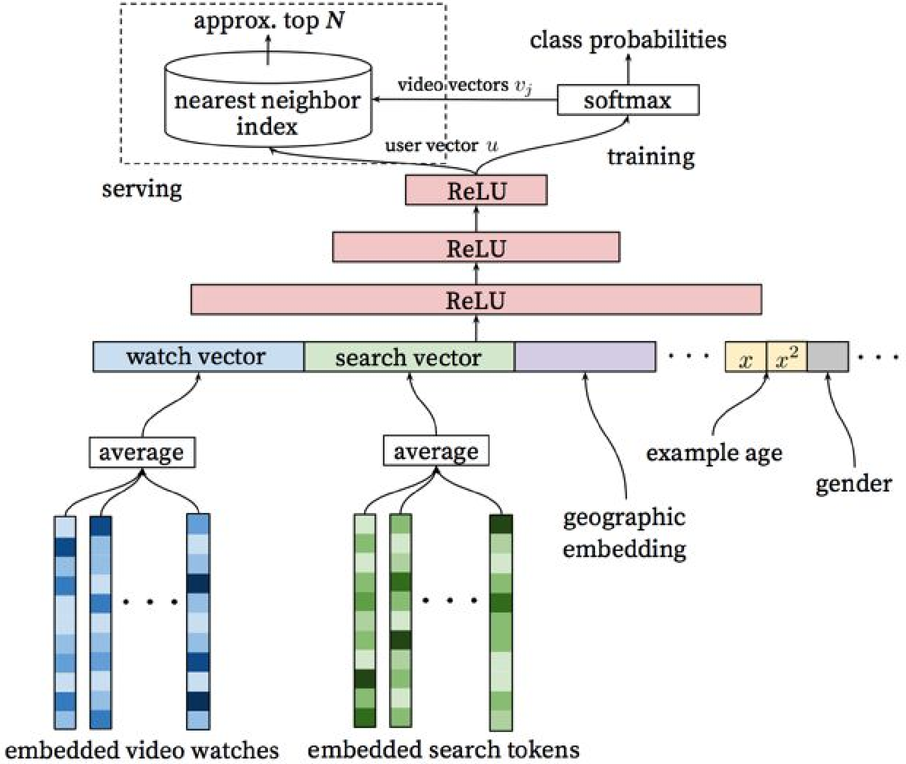
- Good offline results (2x ALS Spark)
- Online: A couple of versions out: On par with ALS-models
- No cold-start problem





Going forward

- Build and tune more similar models
  - Negative Signals
  - Image / Text
  - Search Queries
- Try out new prediction functions
  - Higher degrees, DL
- Solve as classification
- Hyperparameter Tuning
  - (16 parameters + 3 to 4 for each variable!!)
- Combine different recommenders across verticals



Covington 2016

simen.eide@finn.no
@simeneide

We're hiring - hjemmehos.finn.no