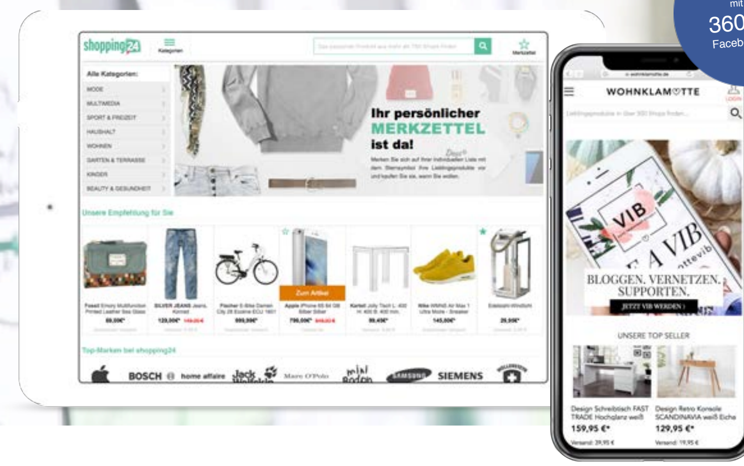
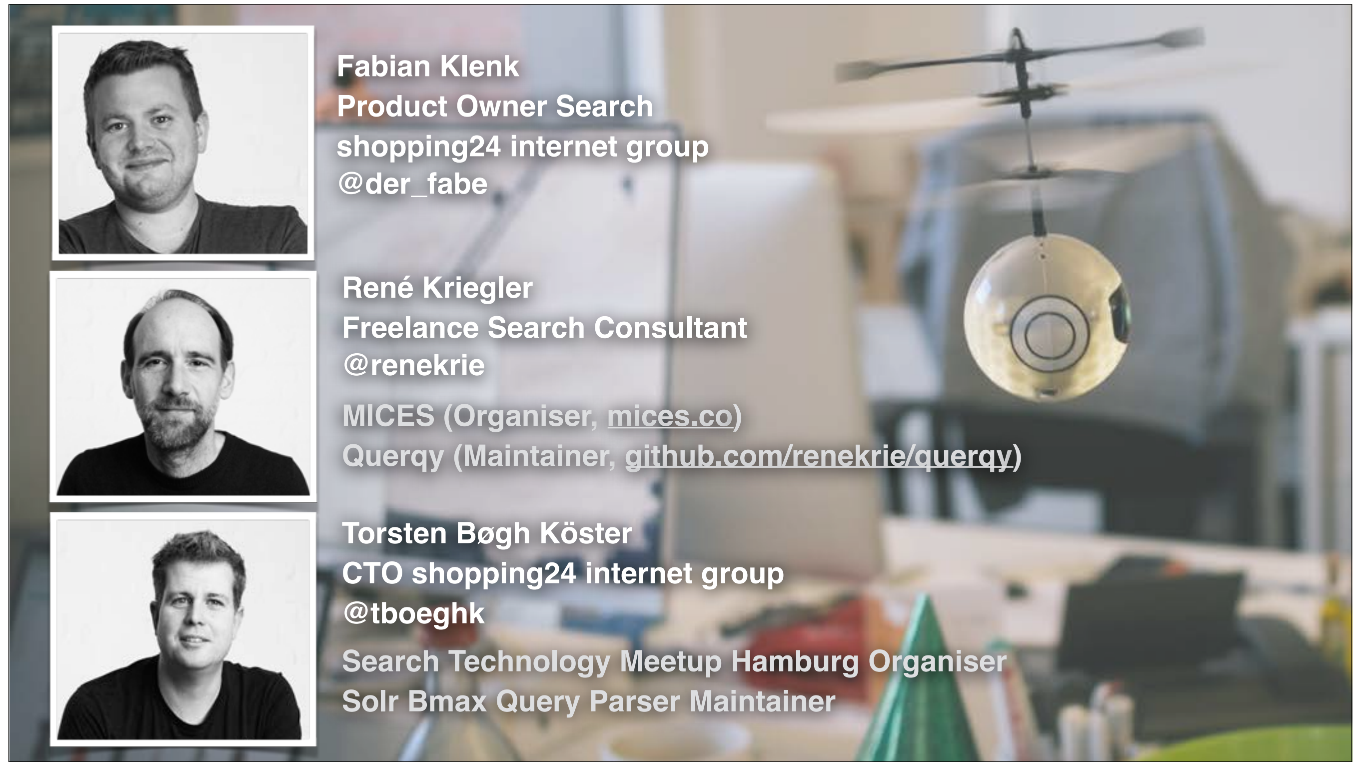
shopping24 internet group

Learning Learning to Rank

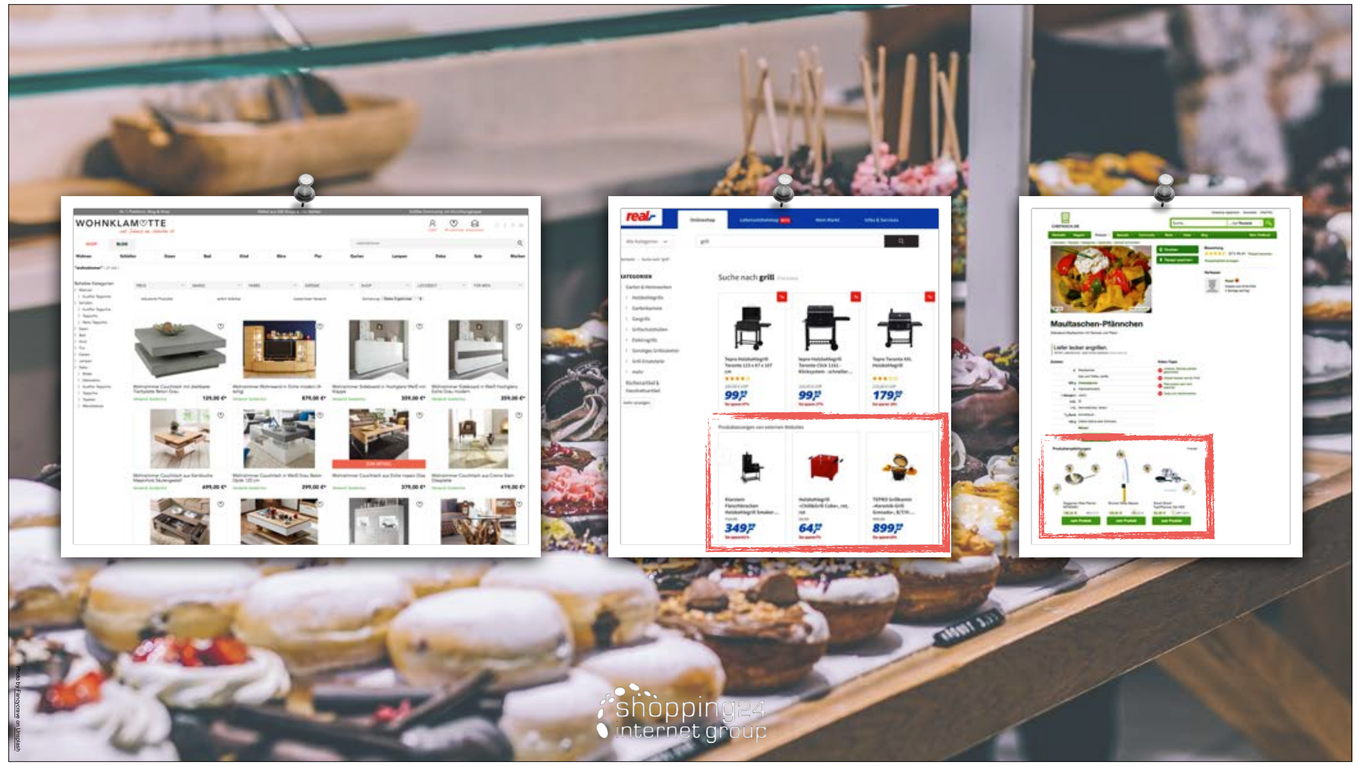
Social Media Liebling mit über 360.000 Facebook Fans



Mehrfach bester Arbeitgeber Deutschlands im Handel und Konsum kununu Arbeitgeber-Ranking

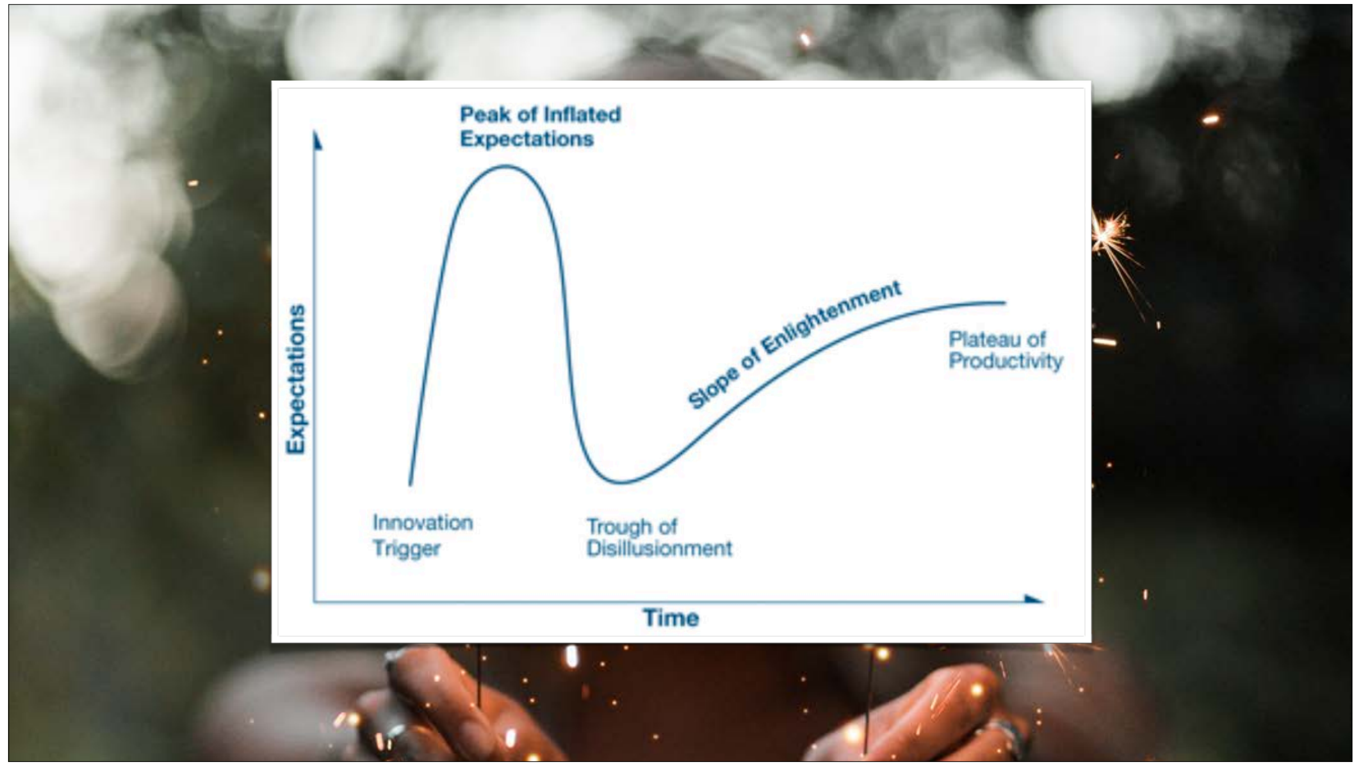


- Here we are with **three different views on Learning To Rank**
  - Fabian: business view
  - René: feature engineering, IR consultant
  - Torsten: ops & management view



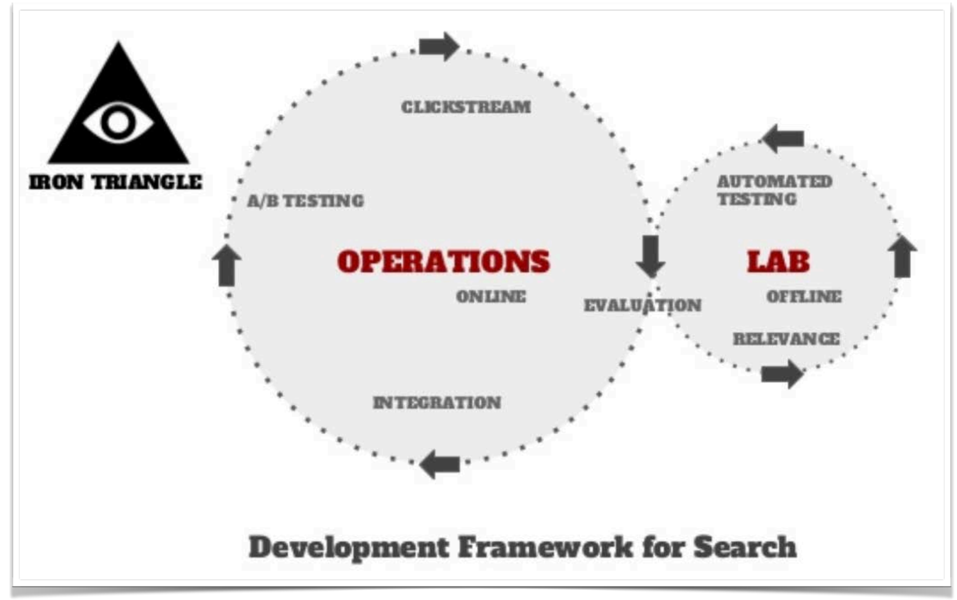
- Shopping24 is part of the OTTO group
  - Not a shop, Google calls us a „**comparison shopping service**“
  - We ship traffic to e-commerce shops
  - We get paid per click on a product (CPC)
- **Three business models**
  - Paid search advertising, 95% search traffic
  - Search widget integrated in other websites
  - Semantic widget integration for content sites.

- Search @Shopping24:
    - Apache Solr as search engine
    - >65M products in each Solr collection, ~ 20 collections
    - ~ 30% products change daily
    - 8M unique search terms per month
    - **Ranking based on exponentially discounted clicks** …
    - … which is basically a **self-fulfilling prophecy**



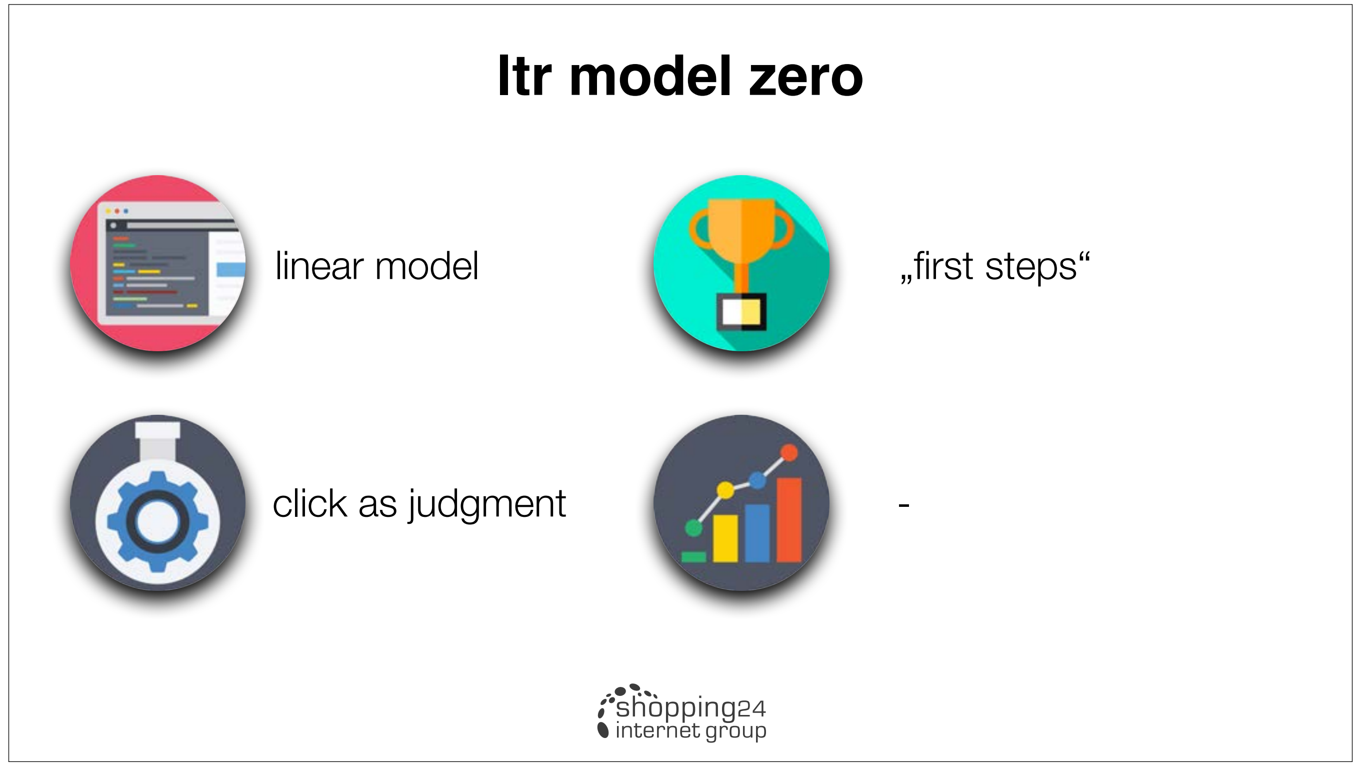
- Machine Learning seems to be at the **peak of the hype cycle**
  - Results may vary from company to company
  - Even inside a company expectation vary
- So: **Expectation management** towards C-Level is important
  - as well towards team members
  - it’s not magic and it’s not self-learning

- Our major goal was to **eliminate the self-fulfilling prophecy**
  - Ranking should be product-ID independent
  - Clicks should serve as judgement only
- Learning To Rank Goals
  - Agnostic to paused or blacklisted products (find products alike)
  - Higher click out rate through more relevant products
  - Higher revenue due to higher click out value

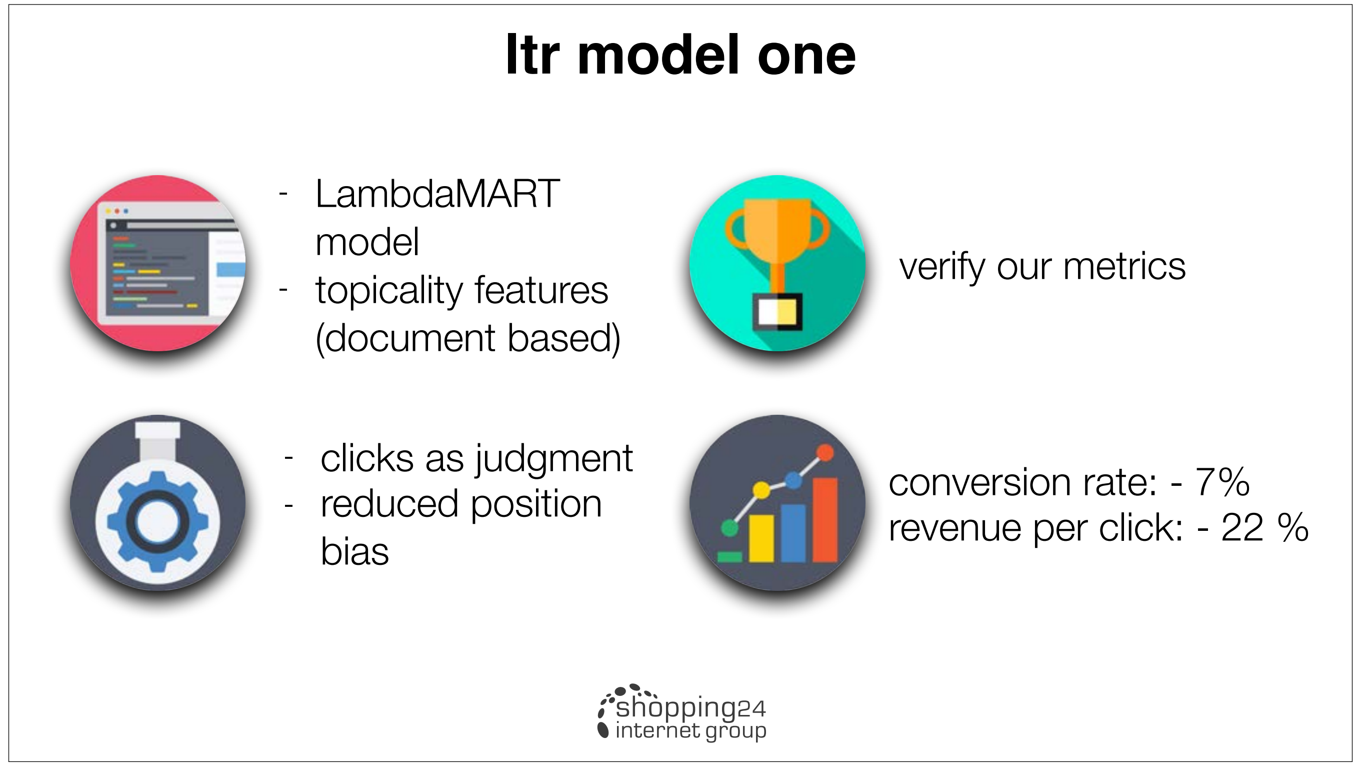


Peter Fries – „*Search Quality - A Business-Friendly Perspective*“
Talk @ Haystack 2018

- Peter Fries presented this simple yet effective development framework for search
    - Have your offline development cycle spin way faster than your online cycle
    - Validate your offline metrics through online a/b-Tests
- You cannot stress this enough: Before launching a machine learning project, have your offline feedback cycle and offline metrics ready
- See: „Best Practices of ML engineering“: http://martin.zinkevich.org/rules_of_ml/rules_of_ml.pdf

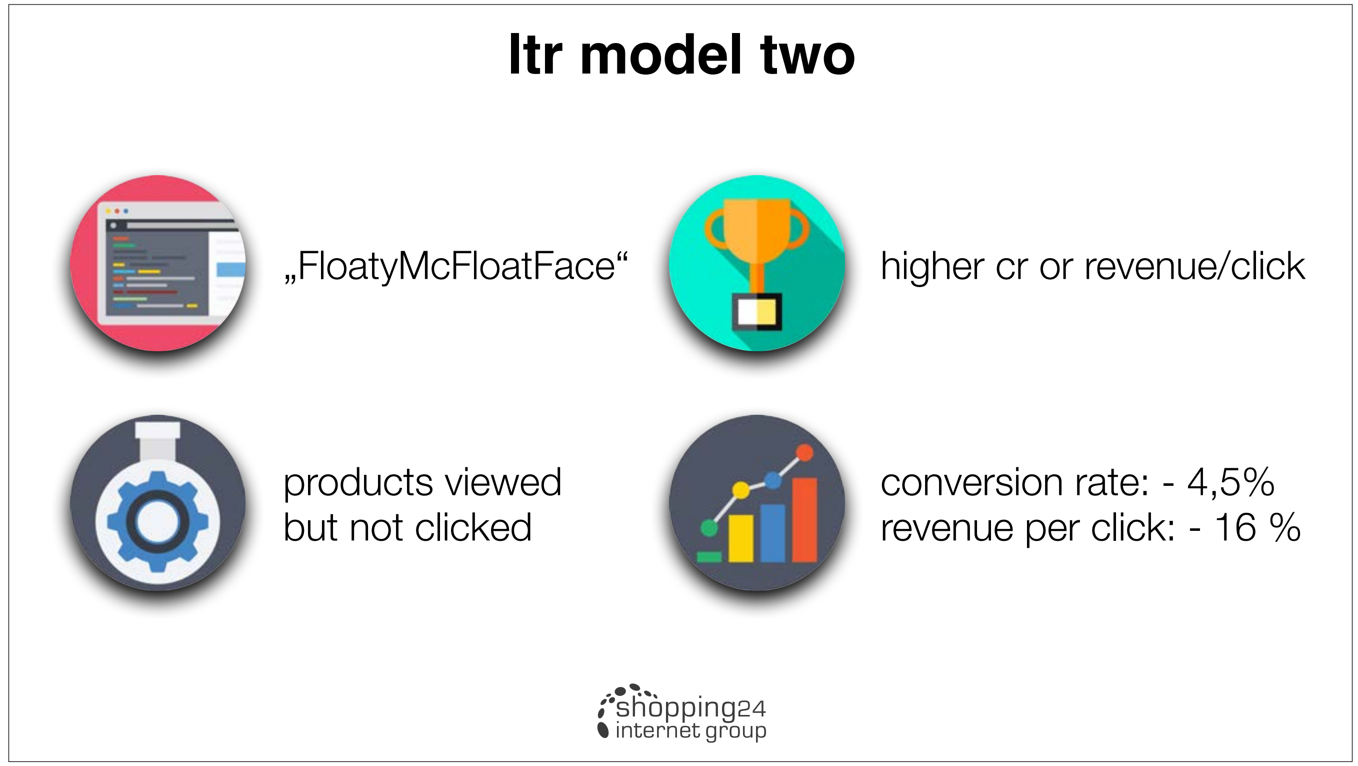


- Let me walk you through some of the major models we built
- Four points of interest
    - Computational changes
    - Jugdmental changes
    - Model and a/b-test goals
    - Overall results
- Model Zero
    - Didn’t work at all, not even test-worthy
    - First steps in collecting relevant data
    - Did not aggregate any clicks
        - as we did not have them in place

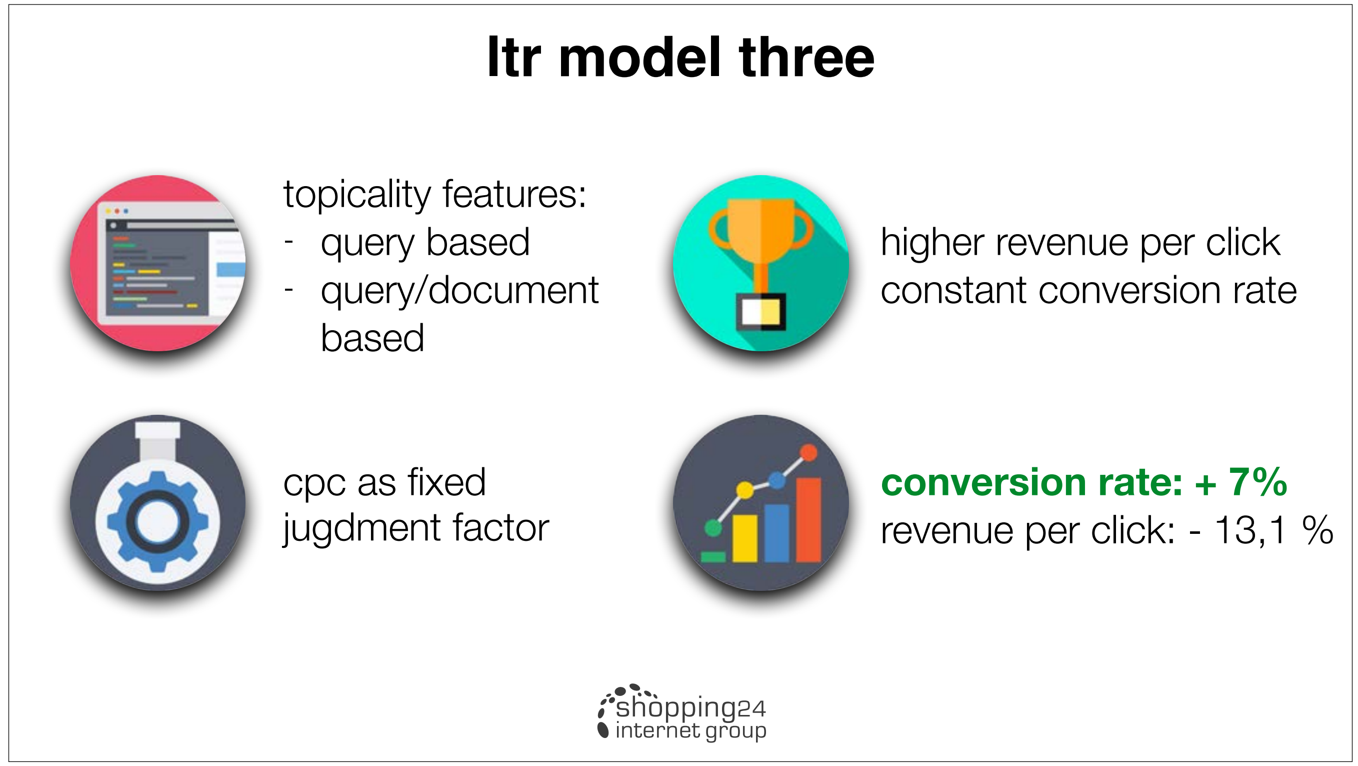


- Model One
    - First model to hit users in an a/b-Test
    - LambdaMART model (Multiple Additive Regression Trees)
    - Major goal was to conclude offline and online metrics
- Not each product has the same click revenue
    - Suprisingly we had a lot of products with an lower cpc above the fold

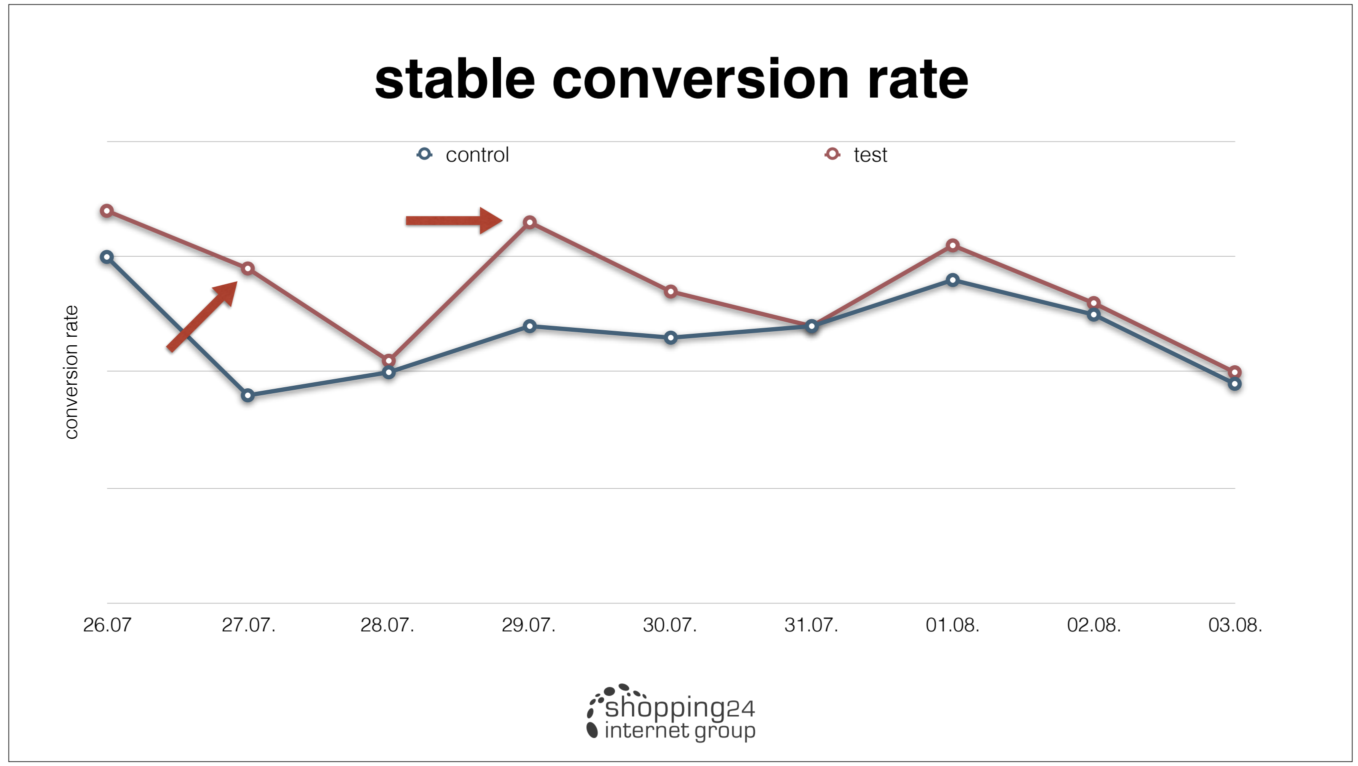
https://medium.com/@nikhilbd/intuitive-explanation-of-learning-to-rank-and-ranknet-lambdarank-and-lambdamart-fe1e17fac418



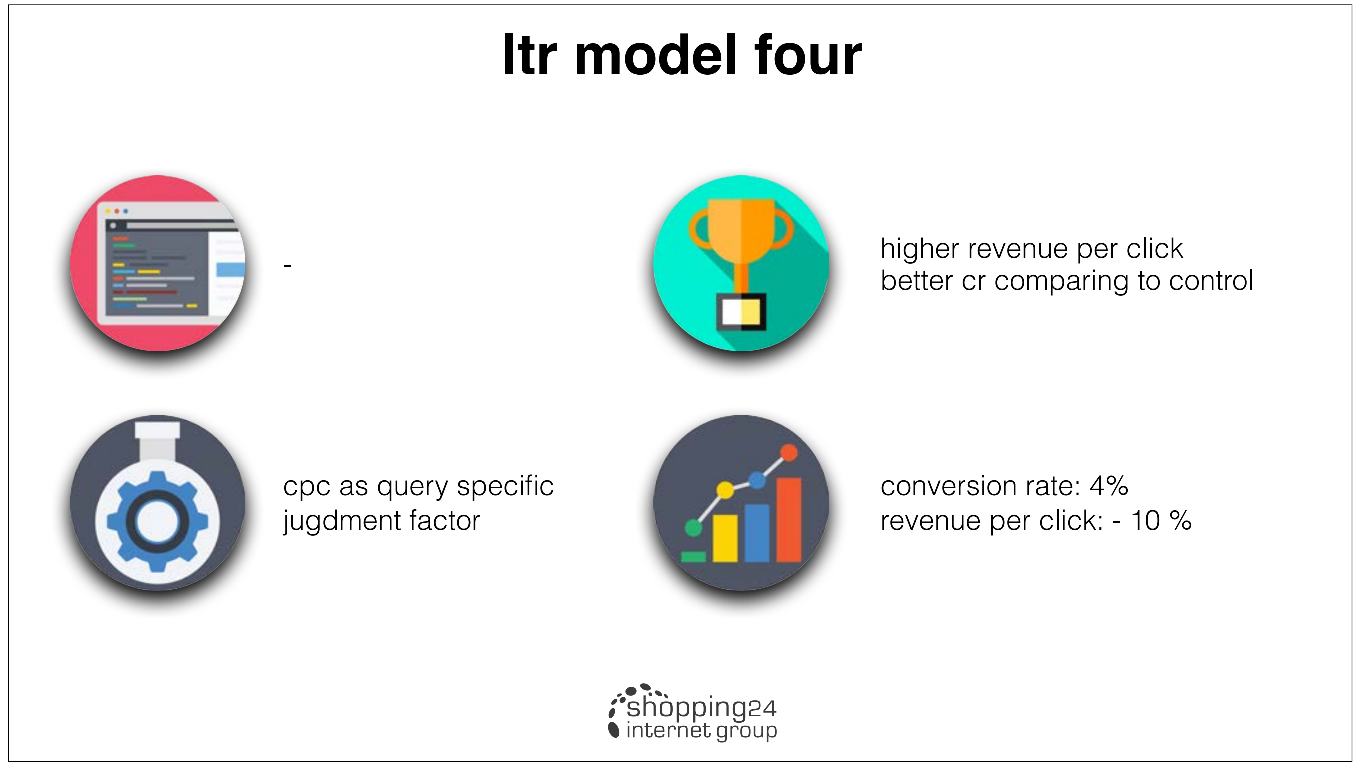
- Model Two
  - Very unsatisfied with graded judgment lists as input into Ranklib
  - Implemented „FloatyMcFloatFace“ to handle float judgments directly
  - Added products viewed but not clicked as counterpart to products clicked
  - Aimed for higher conversion rate and / or revenue per click



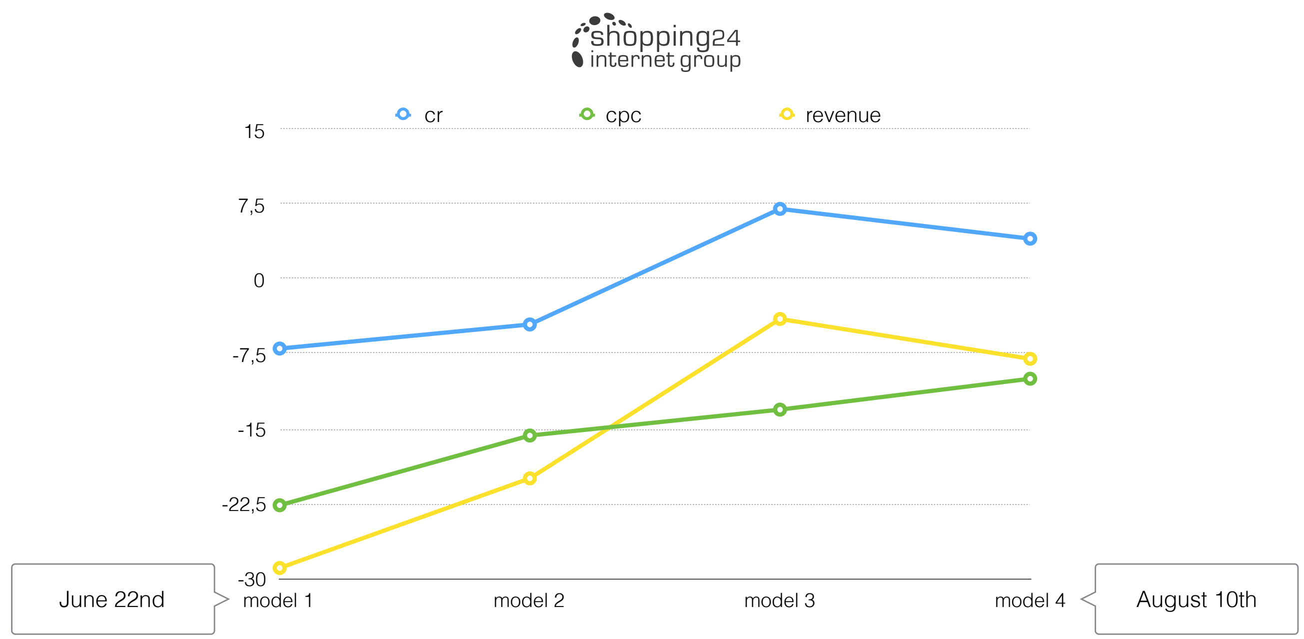
- Model Three
  - Implemented topicality features
  - Used the current product cpc as a fixed jugdment factor
- Saw a better and more stable conversion rate!



- Main goal - to be independent for paused or blacklisted products.
- Saw a better and more stable conversion rate!
- Very promisingly
- A important partner had paused a huge amount of products on day 2



- Model four
    - Focus on judgment tweaking towards higher revenue per click
    - No feature changes



comparing the different models

- Overall comparism if the four models in online a/b test
- Steady increase in at least one kpi
- Timeline: 6 weeks
-

Joining the project as a search relevance consultant

shopping 24 has had an advanced search team for many years but still asked for support:

- choice of **LTR model**
- deriving **judgments** from clicks
- preparing judgments for **RankLib**
- LTR **feature engineering**

- Judgments: dealing with position bias, distinction between seen and unseen documents for zero-click documents
- Judgments in RankLib: graded judgments vs. continous
- Features: Started with: 'Can we just turn ranking factors into features?'

A model for organising LTR features in e-commerce search

Philip Kotler, Kevin Lane
Marketing Management
1997
Peter Morville
Ambient Findability, 2005

Search as part of the 'Buying Decision Process'

Documents in e-commerce search describe a single item - each document is a ‘proxy’ for a concrete thing that we could touch/ examine in a shop

A model for organising LTR features in e-commerce search

Ranking factors in e-commerce search

Topicality - identify the product (type) that the user is searching for (‘laptop’ vs ‘laptop backpack’)

User’s relevance criteria (e-commerce/non-ecommerce)

Seller’s interests (maximise profit)

A model for organising LTR features in e-commerce search

Feature		Topicality	User's interest	Seller's interest
Query matches product title		+		
Query matches brand		+		
product for kids		+	(↓)	
price vs. clicks			↑	
product sold by shop X			↑	↓
CPC				↑
...				

Features grouped by type of ranking factor

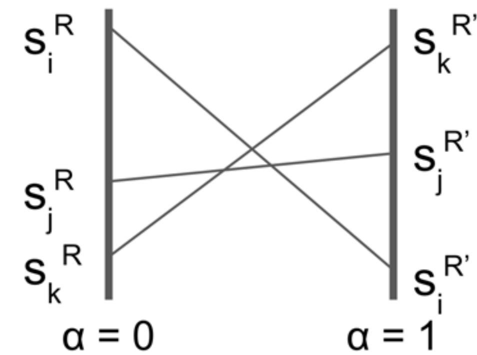
A model for organising LTR features in e-commerce search

Feature	Topicality	User's interest	Seller's interest
Query matches product title	+		
Query matches brand	+		
product for kids	+	(↓)	
price vs. clicks		↑	
product sold by shop X		↑	↓
CPC			↑
...			

Multi-objective optimisation! - start with features related to single objective!

Features grouped by type of ranking factor

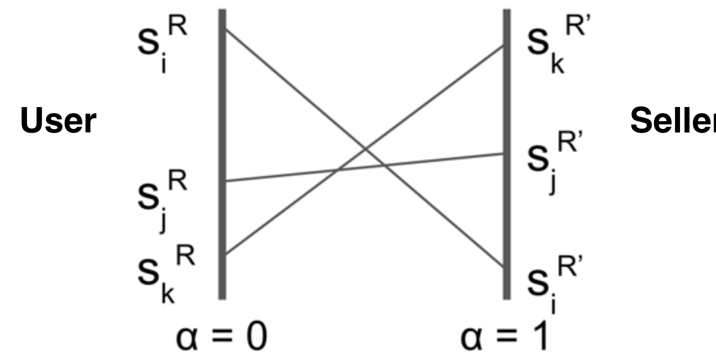
Combining objectives



Optimally combining two rankers. NDCG changes only at crossing points. The two vertical lines represent the sorted list of scores output by Ranker R and R', respectively.

Wu, Q., Burges, C., Svore, K., Gao, J.: *Adapting Boosting for Information Retrieval Measures* (2010)

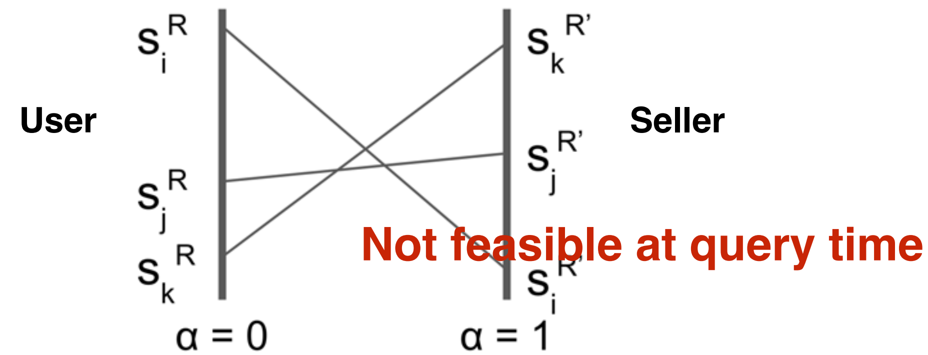
Combining objectives



Optimally combining two rankers. NDCG changes only at crossing points. The two vertical lines represent the sorted list of scores output by Ranker R and R', respectively.

Wu, Q., Burges, C., Svore, K., Gao, J.: *Adapting Boosting for Information Retrieval Measures* (2010)

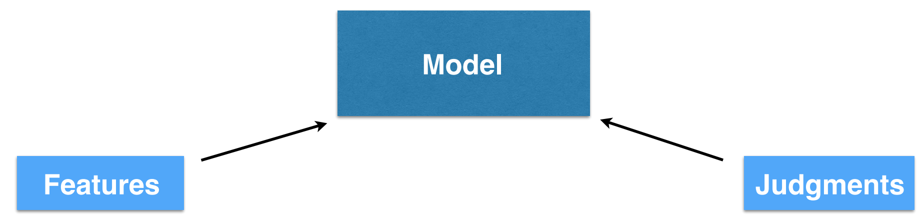
Combining objectives



Optimally combining two rankers. NDCG changes only at crossing points. The two vertical lines represent the sorted list of scores output by Ranker R and R', respectively.

Wu, Q., Burges, C., Svore, K., Gao, J.: *Adapting Boosting for Information Retrieval Measures* (2010)

Combining objectives at training time



Topicality
User's Interest

Normalised click data NC

Seller's interest

CPC

Calculate joint judgment over NC and CPC using ranker combination approach

See also: Doug Turnbull
Optimizing User-Product Matching Marketplaces
https://bit.ly/2P38dld

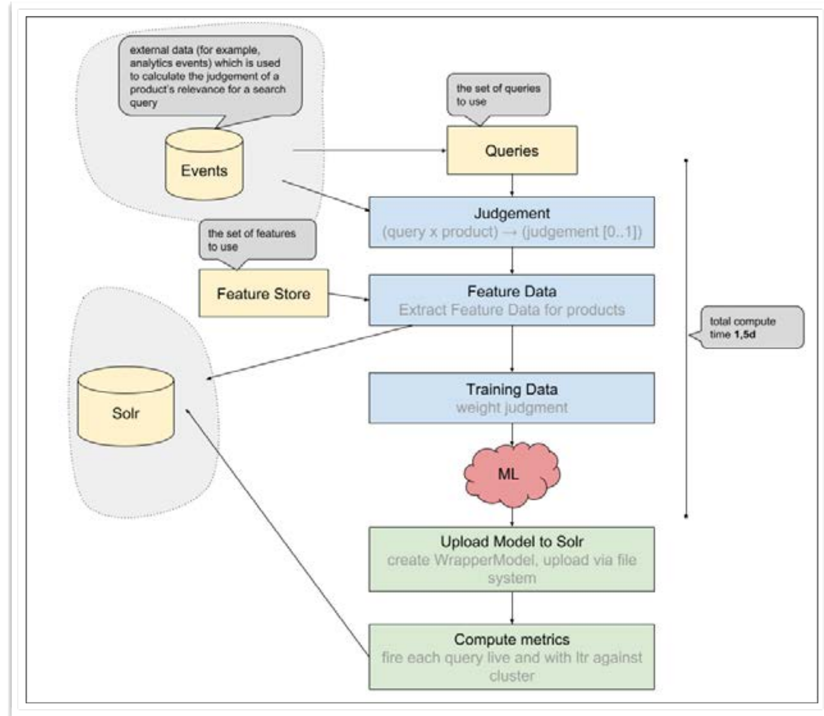
Joining the project as a search relevance consultant

shopping 24 has had an advanced search team for many years but still asked for support:

- choice of **LTR model**
- deriving **judgments** from clicks
- preparing judgments for **RankLib**
- LTR **feature engineering**

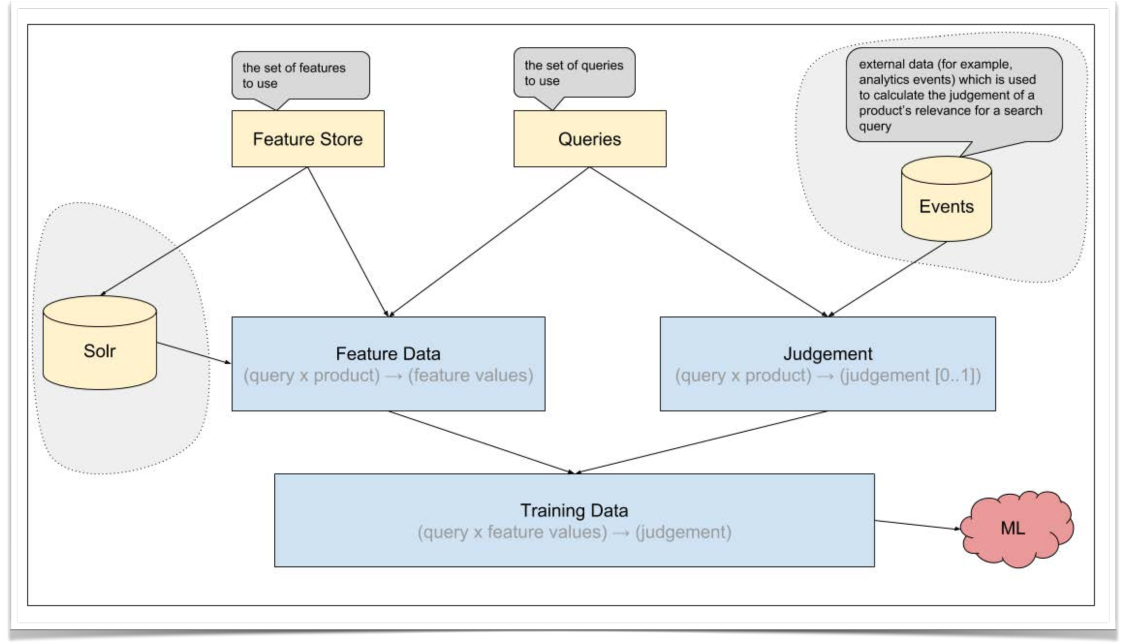
- Search relevance consultant to bring in IR knowledge that would be hard/take long to build in search team

- Scaling learning to rank processes
  - In order to get offline metrics to work, you need to compute models faster and in parallel
  - Ideally you compute a model and receive an email with it’s overall metrics
- Building a model in RankLib is not a problem
  - Modified RankLib to handle float judgments („FloatyMcFloatFace“)
  - Data collection, normalization and cleansing is tedious
  - All models built based on erroneous data (different problems)



Linear LTR model and metric computation

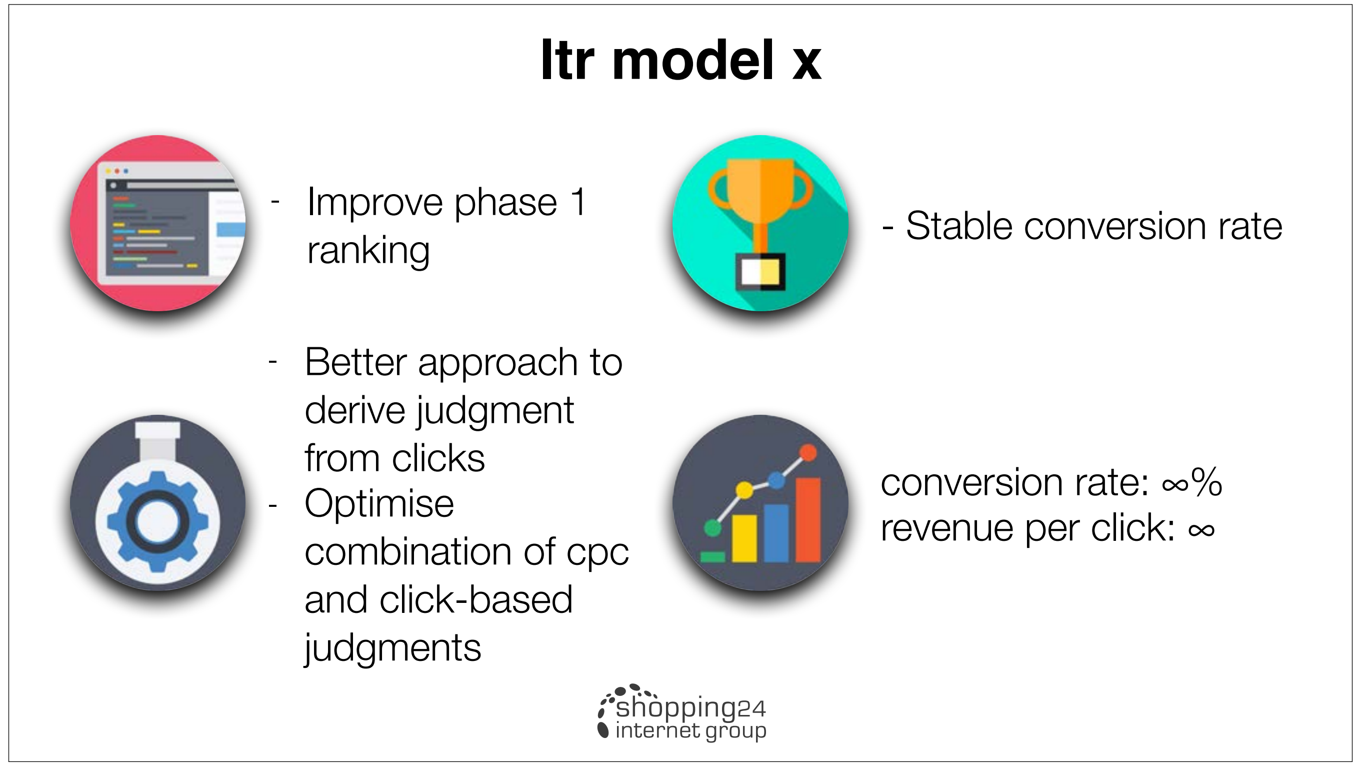
- Linear model computation
  - **4 main artifacts** (query set, judgment, feature data and final training data)
  - Took **1,5 days to compute** for each model
  - Judgment computation and feature gathering very costly
  - Unfortunately not (yet) scalable via CPU or GPU
  - „Easy“ to process as batch job in Kubernetes
- WrapperModel in Solr eases pain of Zookeeper file size limit
  - Distribute models via file systems to all nodes



safe point LTR model computation

shopping24 internet group

- When iterating models …
  - … change one thing at a time (features or judgment)
  - In linear computation mode all artifacts have to be re-computed
- Better: use „safe-points“ to continue work with pre-computed artifacts
  - Split feature data from judgment computation
  - Store artifacts for a given configuration in S3 (or CEPH)
  - Way faster overall compute time
- Example: When working on features, use pre-computed judgment and query set to build training data
- Periodically rebuild everything



- Further explorations
- LTR is applied as re-ranking in Solr (and Elasticsearch or Vespa)
    - So-called Phase 2 ranking
    - Top n documents get re-ranked
    - Phase 1 ranking choses those documents
    - Need to improve phase 1 ranking
- Are clicks recorded from our previous rankings a valid judgment?
    - A different ranking approach will lead to worse metrics
    - Are we optimizing a local maximum?
    - How can we start ranking „outside the box“?


@der_fabe I @renekrie I @tboeghk
Photo by Emily Morter on Unsplash