From User Actions to Better Rankings

Challenges of using search quality feedback for LTR

Agnes van Belle

textkernel

Amsterdam, the Netherlands

Search at Textkernel

- Core product: semantic searching/matching solution
    - For HR companies
    - Searching/match between vacancies and CVs
    - (Customized) SAAS & local installation
    - CVs come from businesses

powered by textkernel

Search at CareerBuilder

- Textkernel merged in 2015 with **CareerBuilder**
  - Vacancy search for consumers
  - CV search for businesses (SAAS)
    - Single source of millions of CVs, from people that applied to vacancies on their website

powered by textkernel

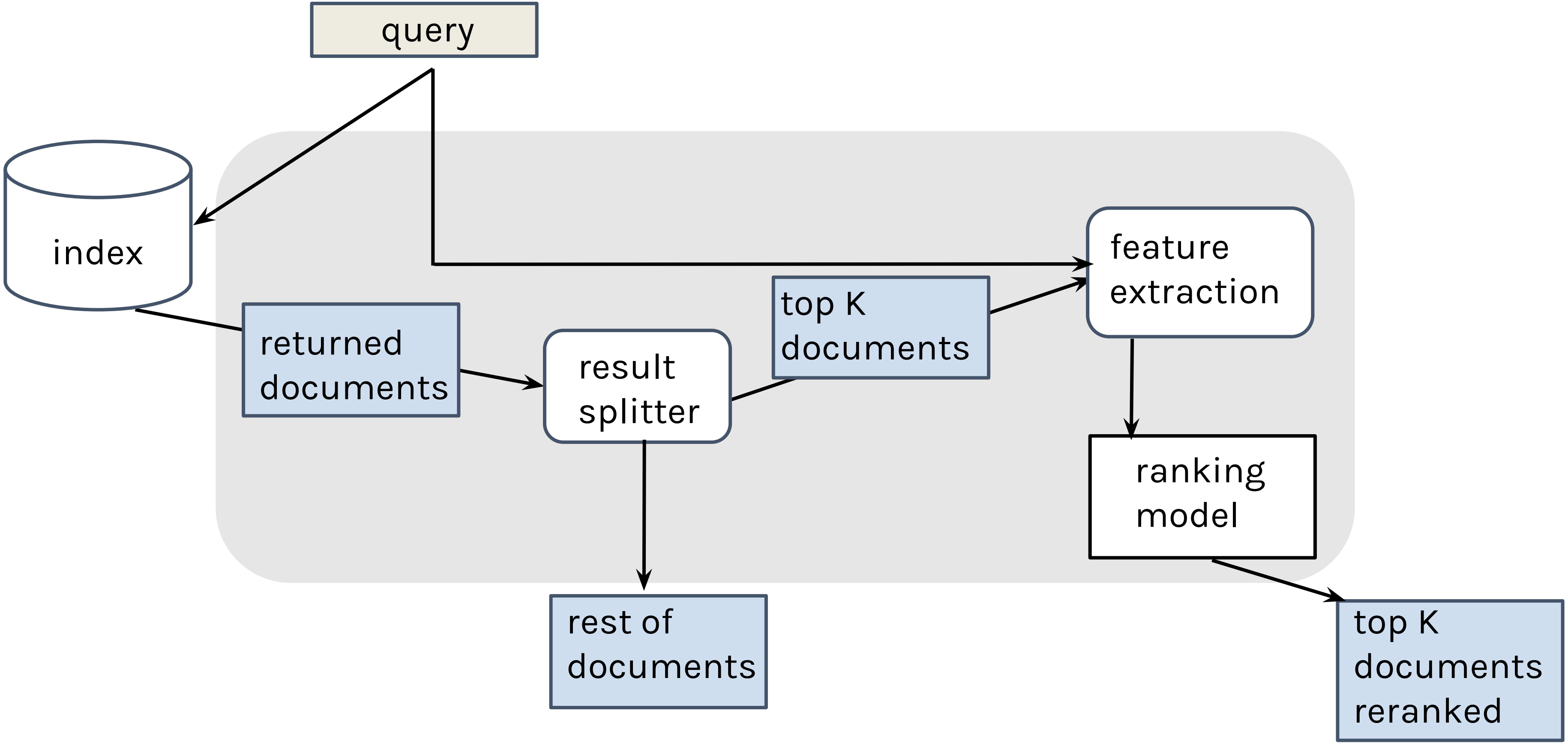
Intuition of LTR in HR field

- “Education will be a less important match, the more years of experience a candidate has”
- “We should weight location matches less when finding candidates in IT”

Learning to rank

- Learn a parameterized ranking model
- That optimizes ranking order
  - Per customer
- We implemented an integration for this in both Textkernels and CareerBuilders search products

LTR integration



powered by textkernel

LTR model training: necessary input

- Machine Learning from user feedback
- Input: set of {query, lists of assessed documents}
    - Each document has a relevance indication from feedback

Feedback types: cost/benefit intuitions

- Explicit feedback
  - Reliable
  - Time-consuming
- Implicit feedback
  - Noisy
  - Comes cheap in huge quantities

Two projects

- Textkernel search product customer
  - Explicit feedback
    - Single customer
    - They have lots of users (recruiters)
- CareerBuilder resume search
  - Implicit feedback
    - Was already action logging implemented

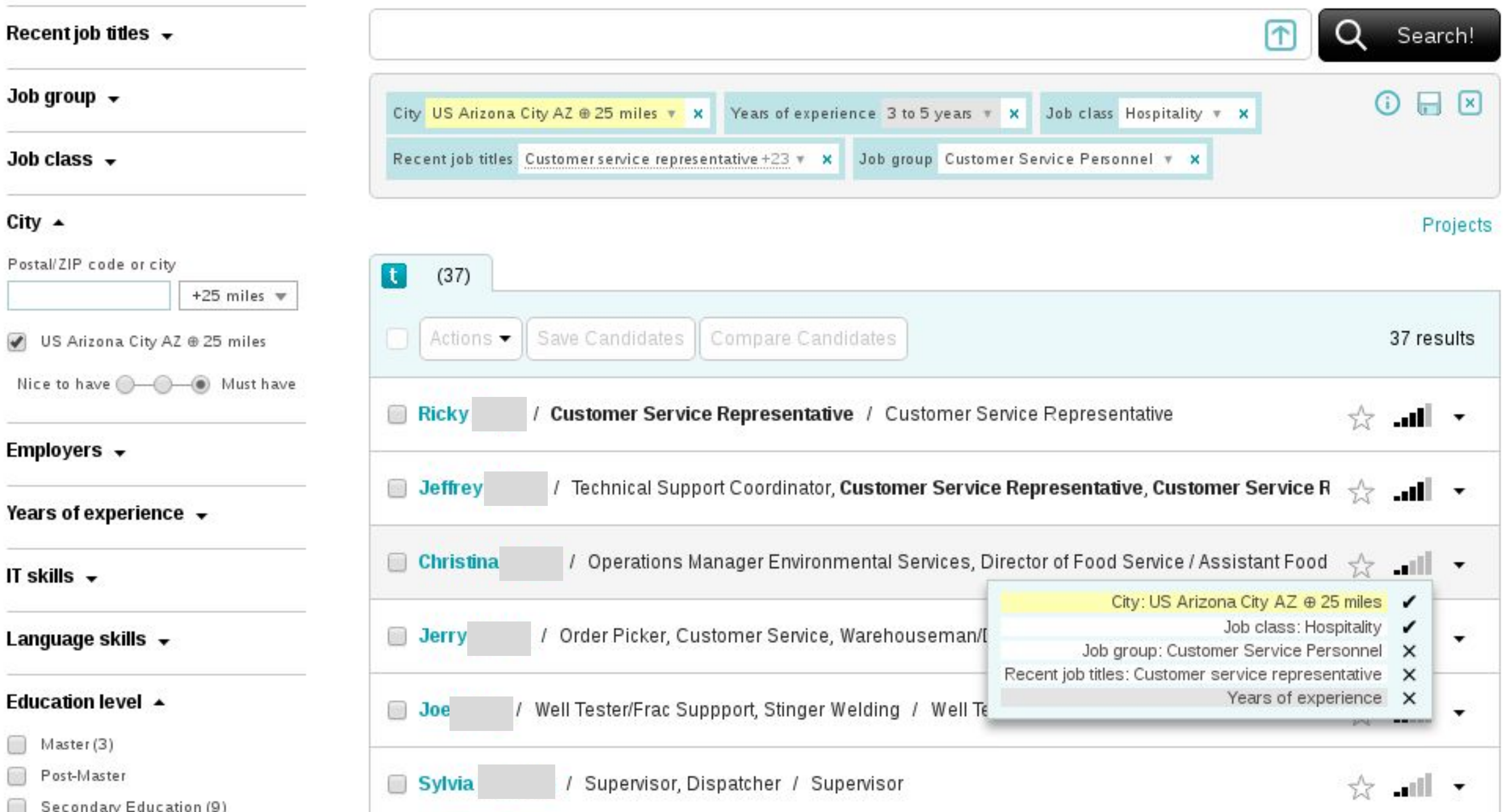
TK search product customer

usg people

- Dutch-based recruitment and human resources company
- In worldwide top 10 of global staffing firms (revenue)
- Few hundred thousand candidates in the Netherlands
- Their recruiters use our system to find candidates

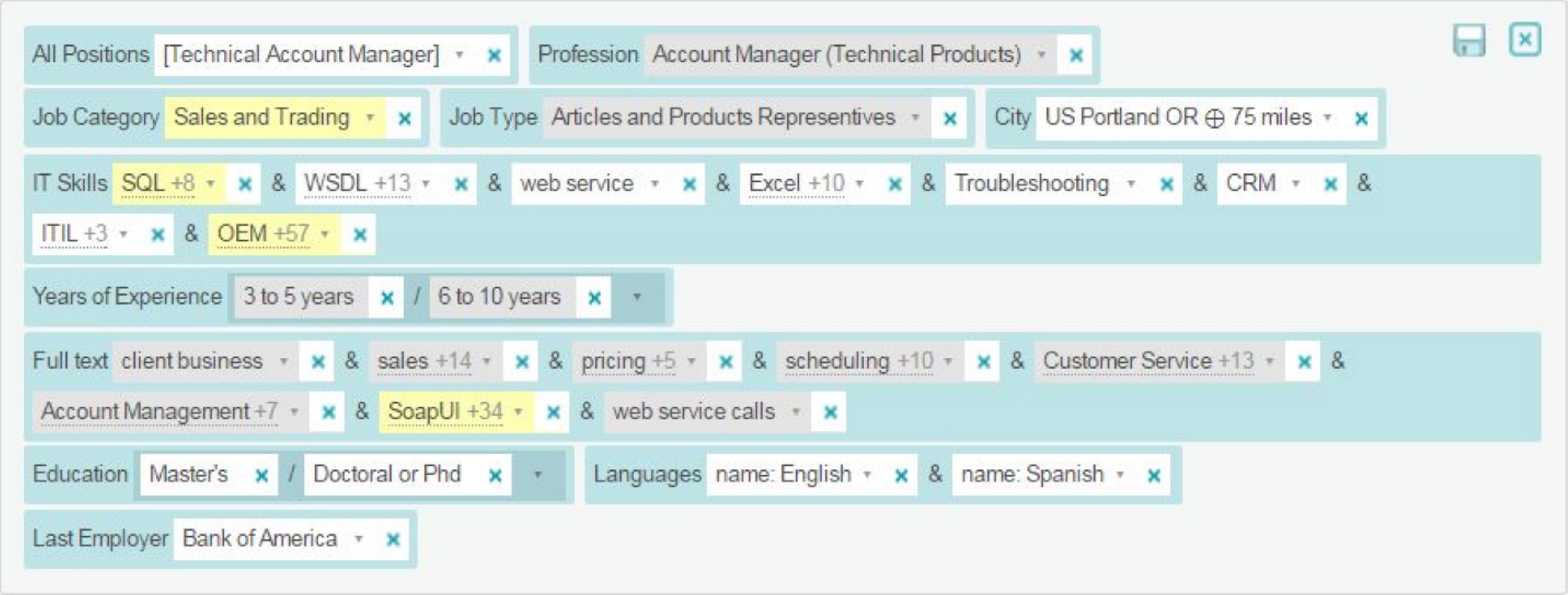
powered by textkernel

Vacancy-to-CV search system



powered by textkernel

Auto-generated query from vacancy



User feedback

- Explicit user feedback given in interface
    - Thumb up for a good result, thumb down for a bad one
- Guidelines:
    - Assess vacancies where they noticed
        - **at least one relevant candidate and one irrelevant candidate**
    - Assess ~ first page of results
    - Assess 1 or 2 vacancies per week

Original Methodology

1. Collect explicit feedback given in interface
2. Generate features for these queries and result-documents
3. Learn reranker model

Two representativeness assumptions

- Query is fully representative of true information need
  - all the recruiter's main needs are in the query
- Explicit assessment is representative of true judgement
  - a positive result means they used a thumb up
  - a negative result means they used a thumb down
    - they won't just see a negative result and do nothing

powered by textkernel

Query is underspecified

Many single-field queries, like:

- *city:Utrecht+25km*
- *fulltext:"civil affairs"*

Criterium	**# queries**	**# assessments**
All	229 (100%)	1514
Matching multiple field criterium	169 (74%)	1092

Assessments are underspecified

For about 75% assessed queries:

- 70% only had thumb up
- 30% only had thumb down

Criterium	# queries	# assessments
All	229 (100%)	1514
Matching multiple assessments criterium	59 (25%)	378

powered by textkernel

Query & assessment underspecification

Criterium	# queries	# assessments
All	229 (100%)	1514
Matching multiple assessments **and** multiple fields criterium	38 (17%)	255

Solving query underspecification

- Remove queries without multiple fields
    - No queries with e.g. only a location field

Solving assessment underspecification

- Many times users assessed, they **skipped** documents
- Assume explicit-assessment skips indicate *implicit* feedback

Original Pos	Relevance
1	N/A
2	1
3	1
4	N/A
5	1
6	1
7	1
8	N/A

irrelevant?

irrelevant?

Solving assessment underspecification

1. Collect explicit feedback given in interface
2. Generate features for these queries and result-documents
3. Also get all un-assessed documents from the logs, and assume these are (semi-)irrelevant
4. Learn reranker

Implicit feedback heuristics

Explicit-assessment skip documents labeling heuristic	Additional query set filtering	NDCG change
None	Without implicit judgements, >=1 explicit assessment	1%
Marked irrelevant	>=1 positive and >=1 negative assessment	4%
Marked irrelevant	>=1 positive and >=1 negative assessment, plus >=3 total assessments	6%
Above the last user assessment: marked irrelevant, below: slightly irrelevant	>=1 positive and >=1 negative assessment, plus >=3 total assessments	6%
Above the last user assessment: marked irrelevant, below: dropped	>=1 positive and >=1 negative assessment, plus >=3 total assessments	6%

Solving assessment underspecification

- Before: **17%** suitable
- After: **31%** suitable **(+14%)** (71 queries)

Reranker algorithm

- LambdaMART
  - state-of-the art LTR algorithm[1]
  - list-wise optimization
  - gradient boosted regression trees
- Least-squares linear regression
  - baseline comparison approach
  - point-wise optimization

1) Tax, N., Bockting, S., Hiemstra, D.: A cross-benchmark comparison of 87 learning to rank methods. Information processing & management 51(6), 757-772 (2015)

Reranker features

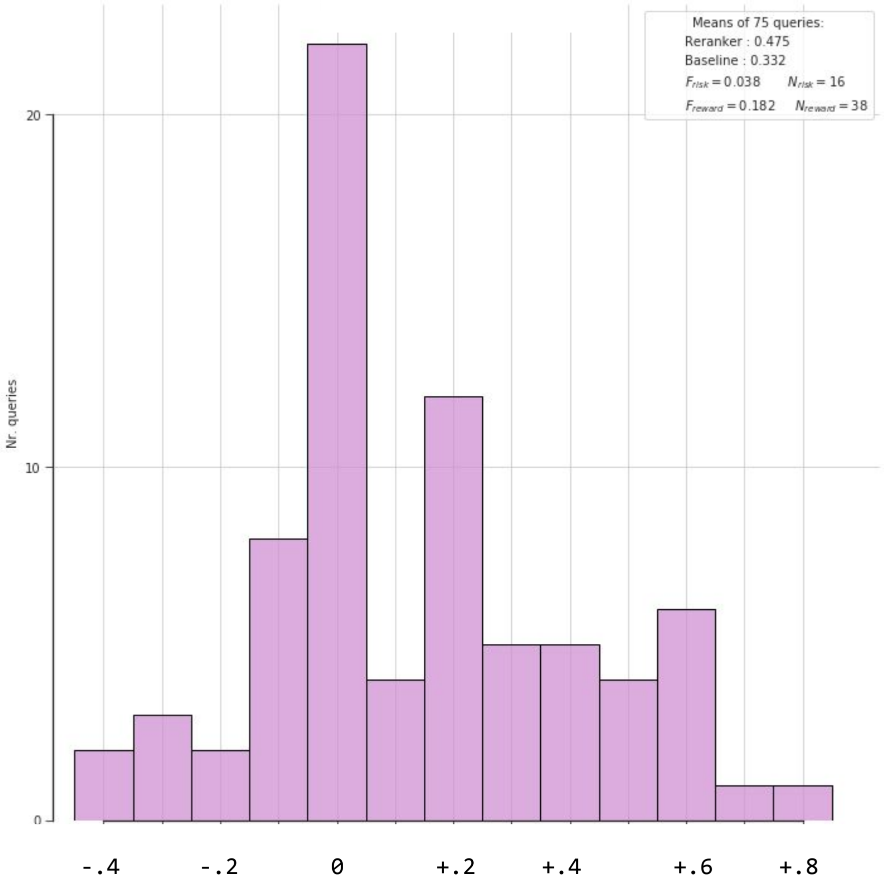
- Vacancy features
  - e.g. desired years of experience or job class
- Candidate features
  - e.g. years of experience, job class, number skills
- Matching features
  - e.g. search engine matching score for jobtitle field

Best learned reranker

	LambdaMART		Linear	
	Baseline	**Model**	**Baseline**	**Model**
NDCG@10	0.33	.47 (+42%)	0.35	0.41 (+18%)
Precision@10	0.23	.32 (+39%)	0.18	0.20 (+7%)
Average number of thumbs up docs in top 10	2.3	3.2 (+0.9)	1.8	2.0 (+0.2)

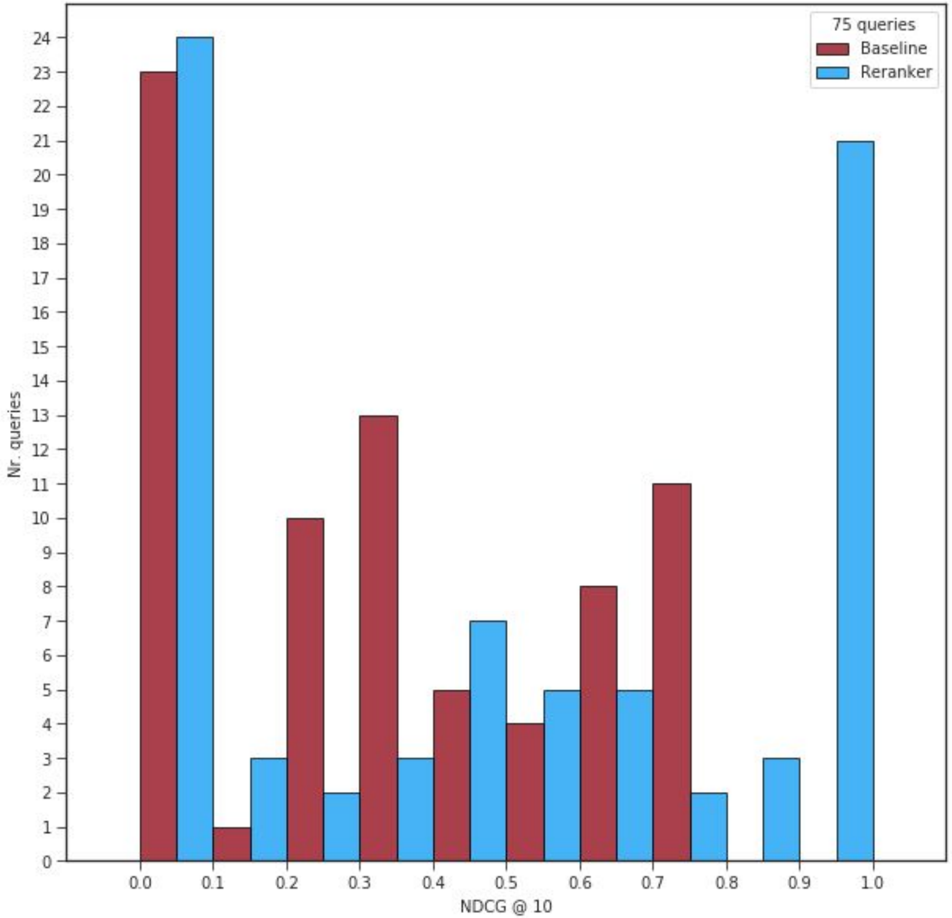
Note that actual search performance is much higher because not explicitly assessed documents are considered irrelevant

Reranker minus baseline score difference plot (NDCG top 10)



Reranker vs baseline score distribution plot (NDCG top 10)

powered by textkernel



Deeper look

- Query underspecification problem seems not solved
  - The learned models are mostly based on document-related features, not so much on query-related ones
  - Qualitative look revealed queries lack requirements

powered by textkernel

Examples

"burgerzaken"

(civil affairs)

Thumb-up documents:

- 9/11 are in Rotterdam, 2/11 in Amsterdam

N/A documents:

- 3/4 are from small towns (non-Randstad)
- 1 is from Amsterdam, but still studying, and her experience is in a small town

Original		Reranked	
Original Pos	Relevance	Original Pos	Relevance
0	1	0	1
1	1	17	1
2	1	1	1
3	N/A	6	1
4	1	5	1
5	1	16	1
6	1	13	1
7	N/A	2	1
8	N/A	7	N/A
9	1	12	N/A
Precision = 0.7		Precision = 0.8	
NDCG@10 = 0.77		NDCG@10 = 0.87	

Lessons learnt explicit feedback

- Two types of underspecification problems:
  - Explicit assessments underspecify order preference
    - Can be solved
      - almost doubled usable data using implicit signals
  - Query underspecifies vacancy
    - Harder to solve with small dataset
    - Serious problem in HR field (discrimination)

powered by textkernel

CareerBuilder Resume Search

- 125 million candidate profiles
- Two search indexes:
  - CB Internal Resume Database
  - Social profiles
- Semantic search

Search Jobs Sourced Help

machine learning engineer java python

New York, NY

30 Miles

Search

Include related keywords

machine learning +8 | engineer +3 | java +7 | python +3

Search History

Candidate Results (51 Total Search Results)

Save Search

Manage Saved Search

My Candidates (51)

All Filters

Clear

Default Filters

- Experience
- Job Titles
- Skills
- Current Employer
- Industries
- Education Level
- Schools
- State
- Country
- Most Recent Source
- All Sources
- Document Type
- Candidate Status in ATS
- Requisition IDs

Talent Gather Filters

Email Campaign Filters

Actions | Sort by: Relevancy | Exclude (None) | Freshness (Last Year)

1-10 ‹ 1 2 3 › »

Nitesh *Last Active On: 07/30/18*

New York, NY US | 9 Years Total Experience | Bachelor's Degree

Current: **Data Scientist (1 year, 7 months)** CGI Group Inc.

Previous: **Data Scientist (1 year, 4 months)** Walt Disney

Skills: Data Mapping | Data Modeling | Extract Transform Load (Etl) | Data Mining | Data Cleansing

Favorite | Forward | + Add to List | Add Note

Your Coworker: Viewed on 8/31/18 You: Viewed on 9/4/18

Vijay Gopal *Last Active On: 05/03/18*

In Your ATS

Harrison, NJ US | 1 Years Total Experience | Master's Degree - New Jersey Institute of Technology - New Jersey, USA

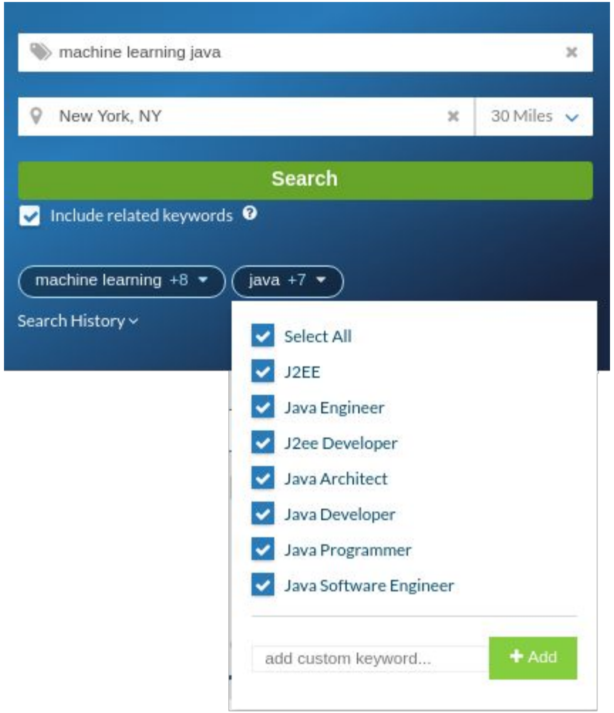
Current: **(0 months)**

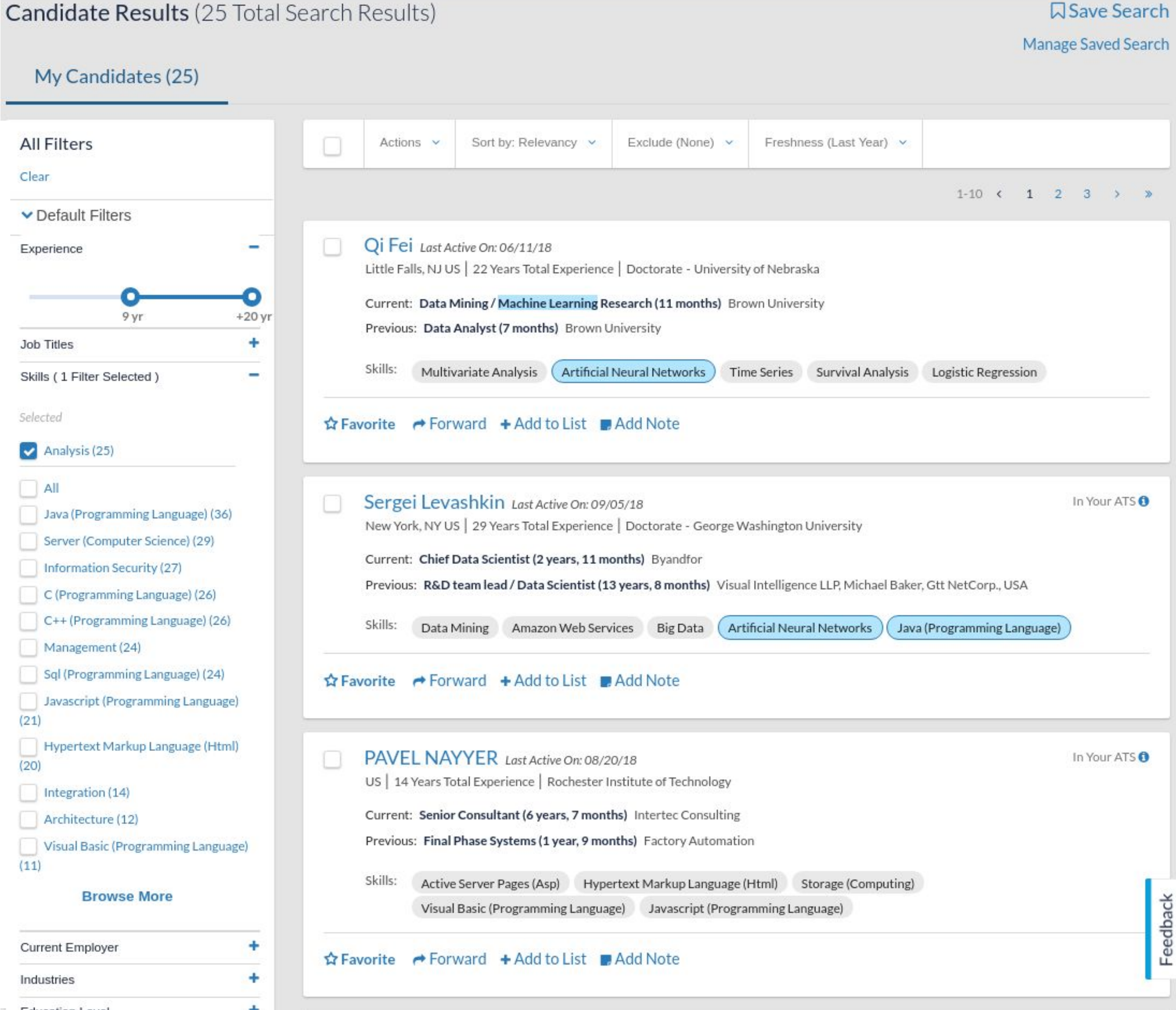
Previous: **Intern (3 months)** GreenLabs

Skills: Amazon Web Services | Java (Programming Language) | Advanced Encryption Standard (Aes) | Extensible Markup Language (Xml) | Hypertext Markup Language (Html)

Favorite | Forward | + Add to List | Add Note

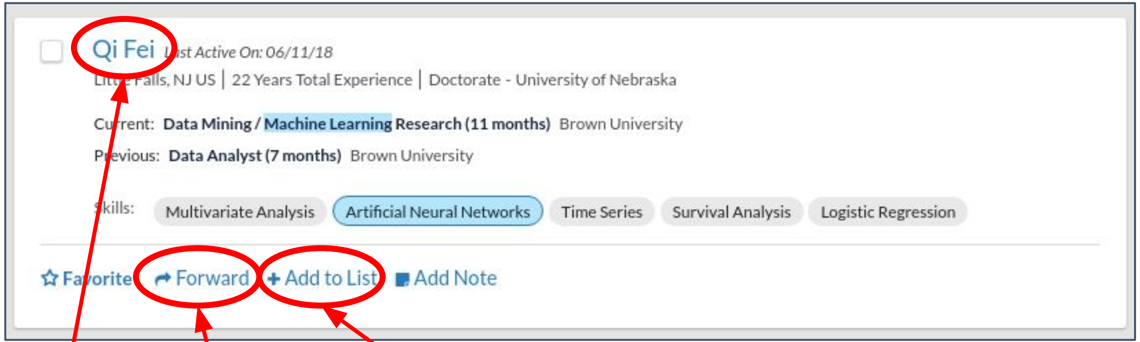
Semantic Search





powered by textkernel

Four Actions

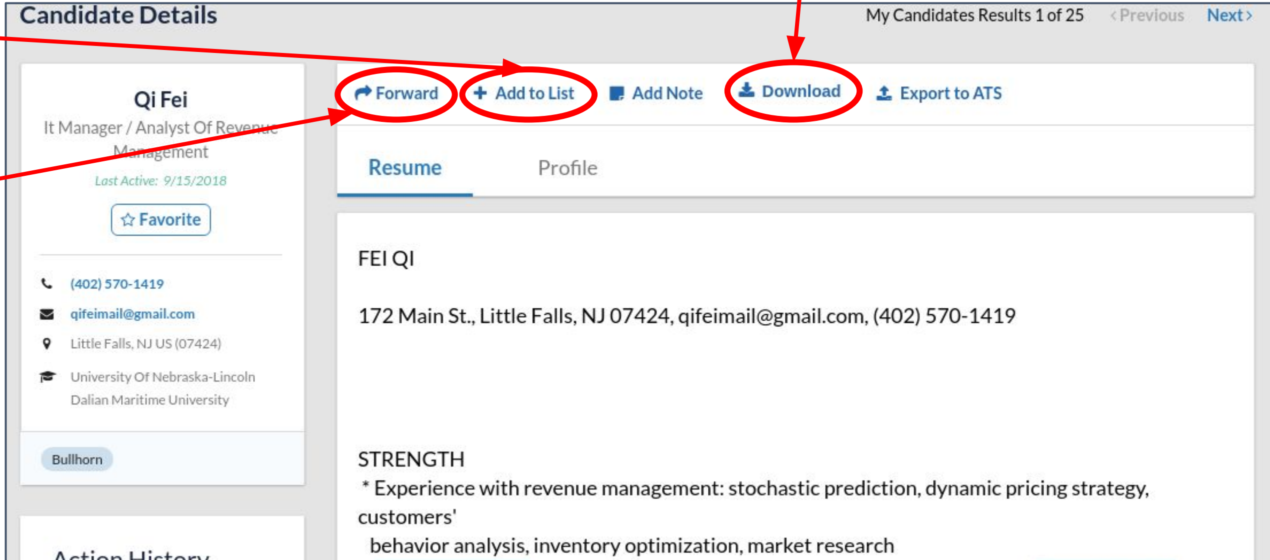


Download

Save

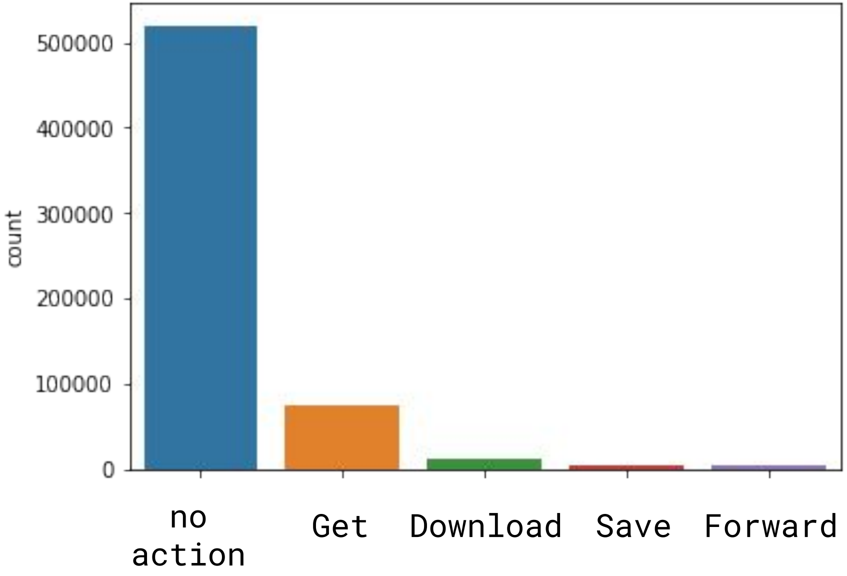
Get

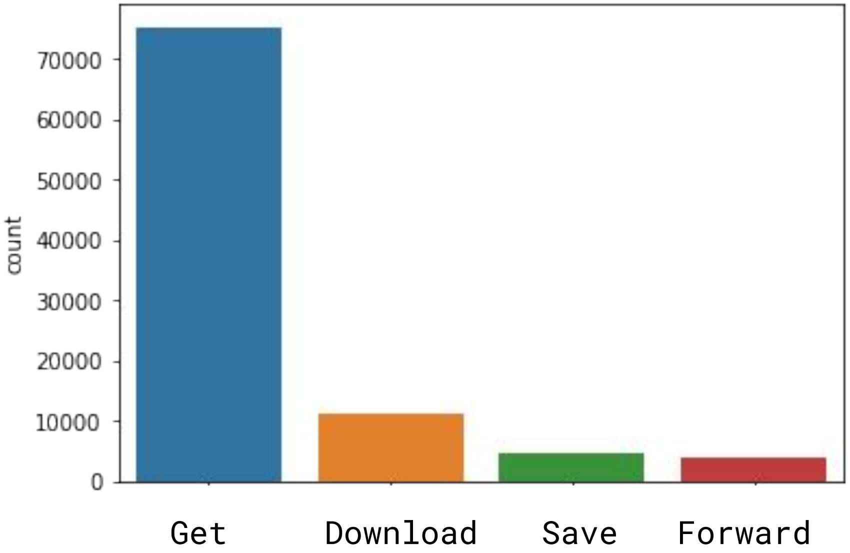
Forward



Action analysis: frequency

- Most users don't interact much with the system
- Most just "click" ("Get") to view a candidate's details





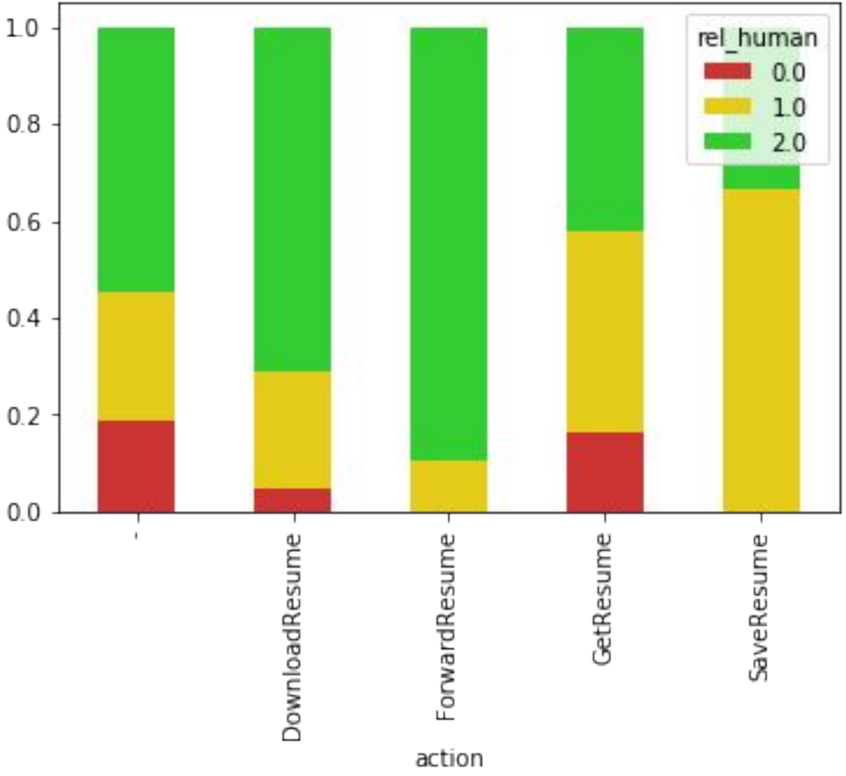
How to interpret actions?

- Check calibration with human-annotated set
  - 200 queries
    - Each query 10 documents
- Relevance scale used by annotators:
  - 0 (bad),
  - 1 (ok),
  - 2 (good)

Learned reranker on human labeled set

- Improvement using 5-fold cross-validation:
    - 5-10% NDCG@10

Action correlation with human labels



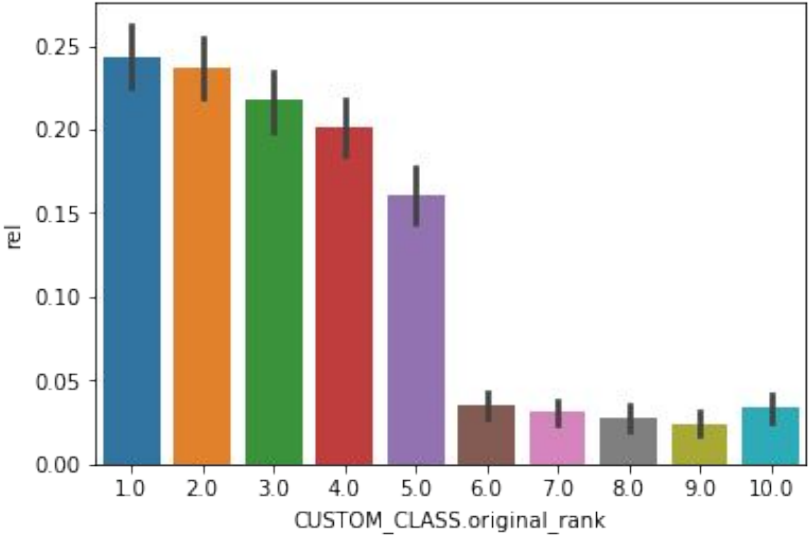
- “Get”: **many** irrelevant results
- “Save”: unclear relation
- “Download/Forward”: reliable

How to interpret actions?

- “Get”: **many** irrelevant results
    - Two subgroups of users:
        - users that take a closer look on “odd” results
        - users that click on good results
- “Save”: unclear relation
    - You can save results as relevant for a *different* query
- “Download/Forward”: reliable
    - “Forward” is an email, can be to yourself

Action usage

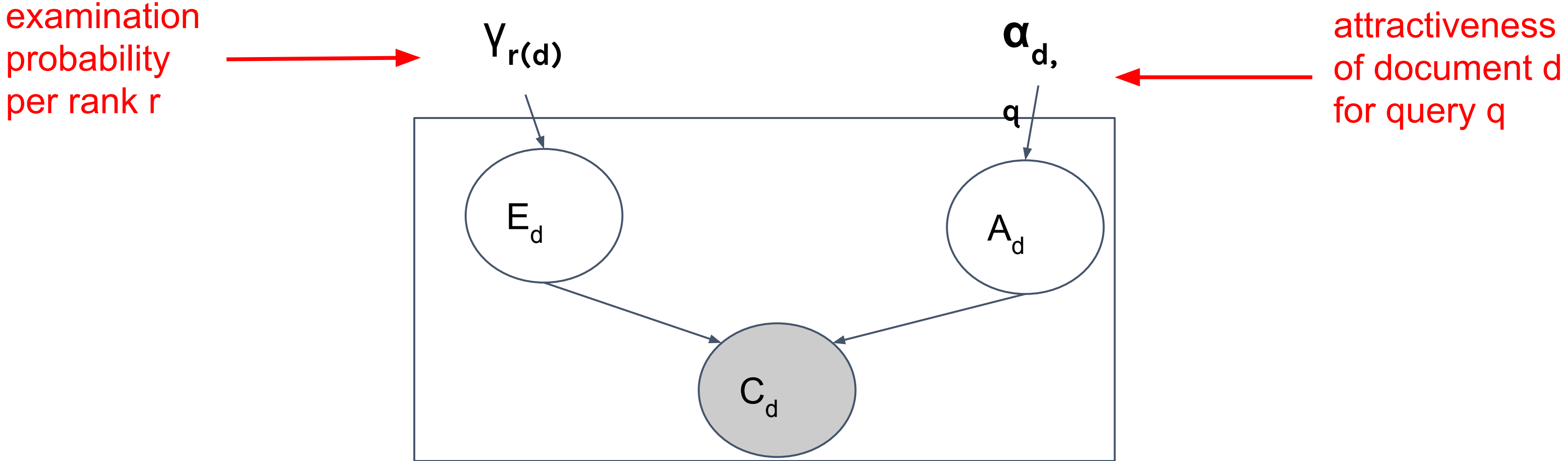
- How to deal with position bias?
- What's the last document to attach relevancy to?



Rank	Clicked	Examined
1	x	y
2		y
3	x	y
4		y
5	x	y
6		?

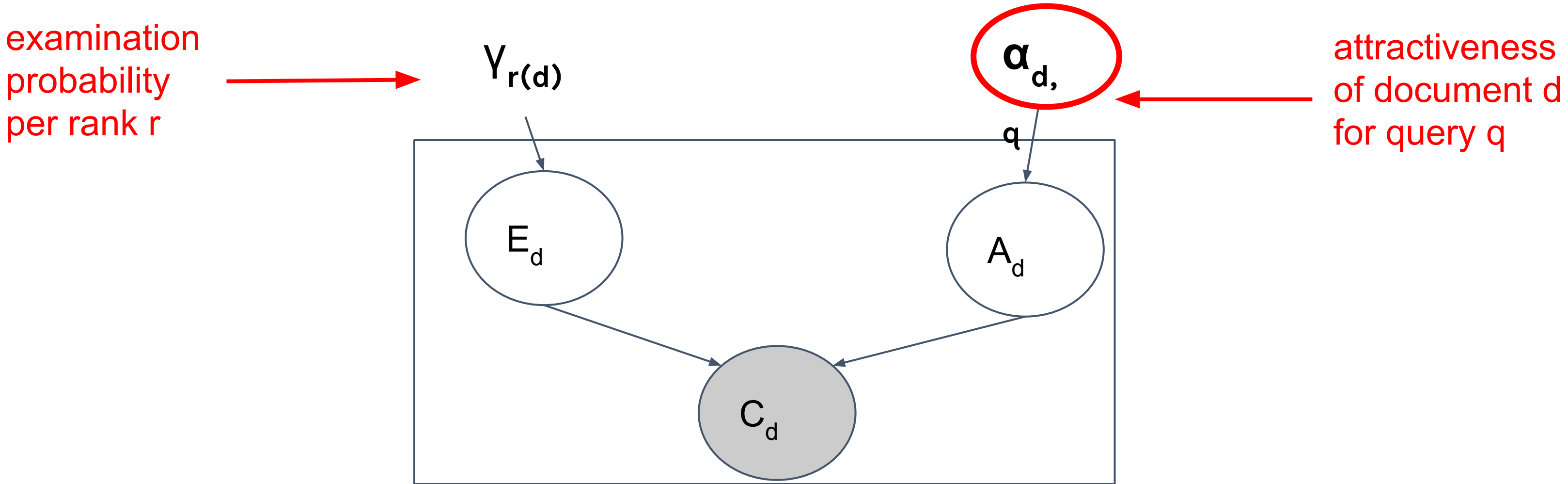
Position bias: click models

- Model probability of examination and attractiveness based on users search behavior.
- Factor out position
- **Position-Based Model:**



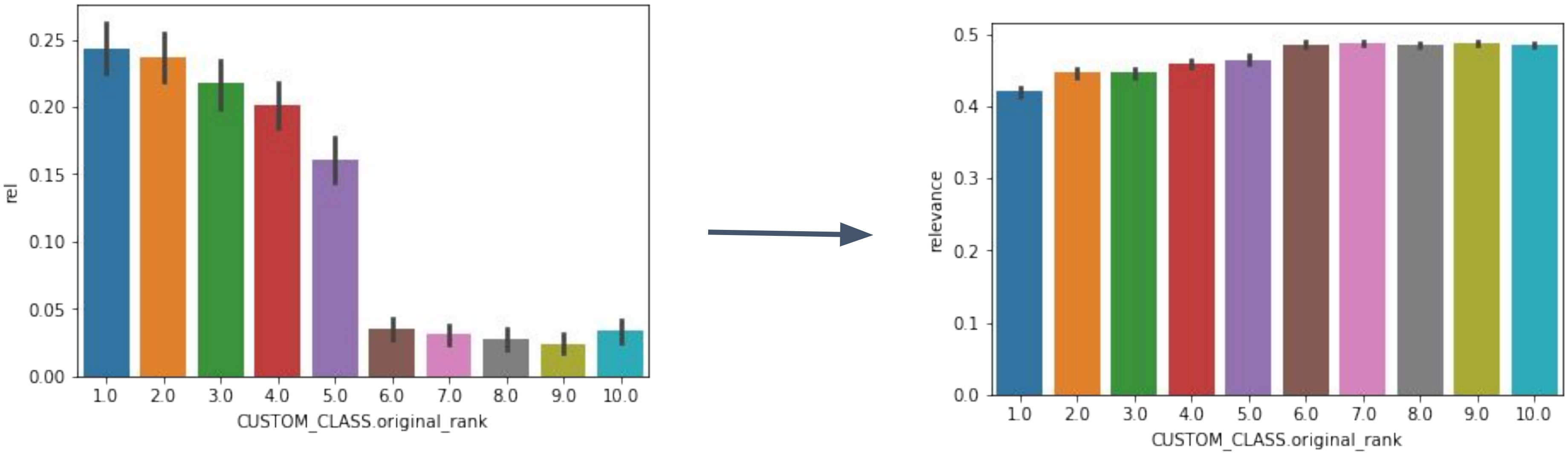
Position bias: click models

- Model probability of examination and attractiveness based on users search behavior.
- Factor out position
- **Position-Based Model:**



Position bias: click models

- Click model (PBM) succeeded in removing position bias



Position bias: click models

- Click model (PBM) however did not boost score
- Possible causes:
    - Few repeated queries
    - Sparse clicks

Last document to attach relevancy to

- Cut-off after last click
  - Makes bottom document always relevant
  - Results in reranker “learning” to put bottom documents at top
- Top-N results
  - Choose top 20
  - (Avg. position last click: 17)

Rank	Clicked	Examined
1	x	y
2		y
3	x	y
4		y
5	x	y
6		?

Query filtering

- Using only queries with at least ‘fulltext’ and ‘location’
  - Queries without that are underspecified and the clicks will be noisy
  - Or the user will probably refine
  - These wo fields turned out to be most important
- Using queries that were executed multiple times
  - If multiple people issued a query, it is likely of higher quality
  - Aggregate the signal so they become more reliable

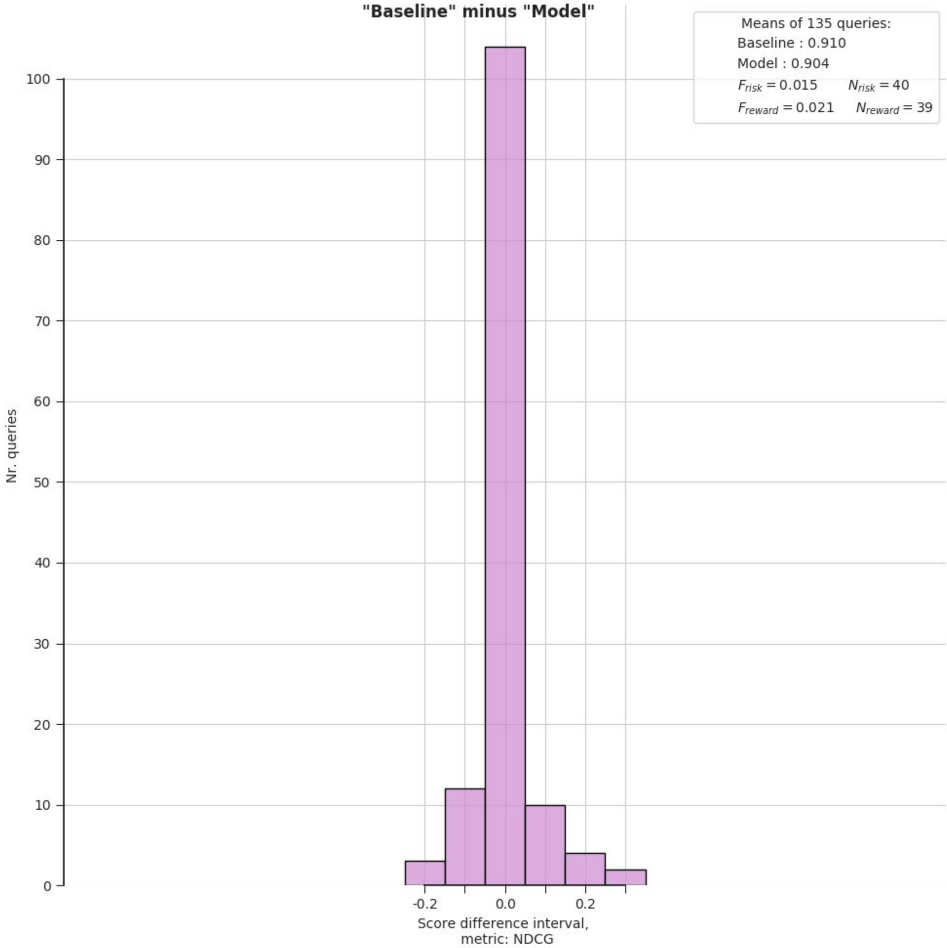
Query/action filtering

- Original data:
    - 1 month
    - 2.1M query-doc pairs
- Filter on queries with > 1 occurrence:
    - 2.3K unique queries
- Filter on queries with
    - ‘fulltext’ and ‘location’
    - >=3 Download/Forward actions
        - 500-600 queries

Results

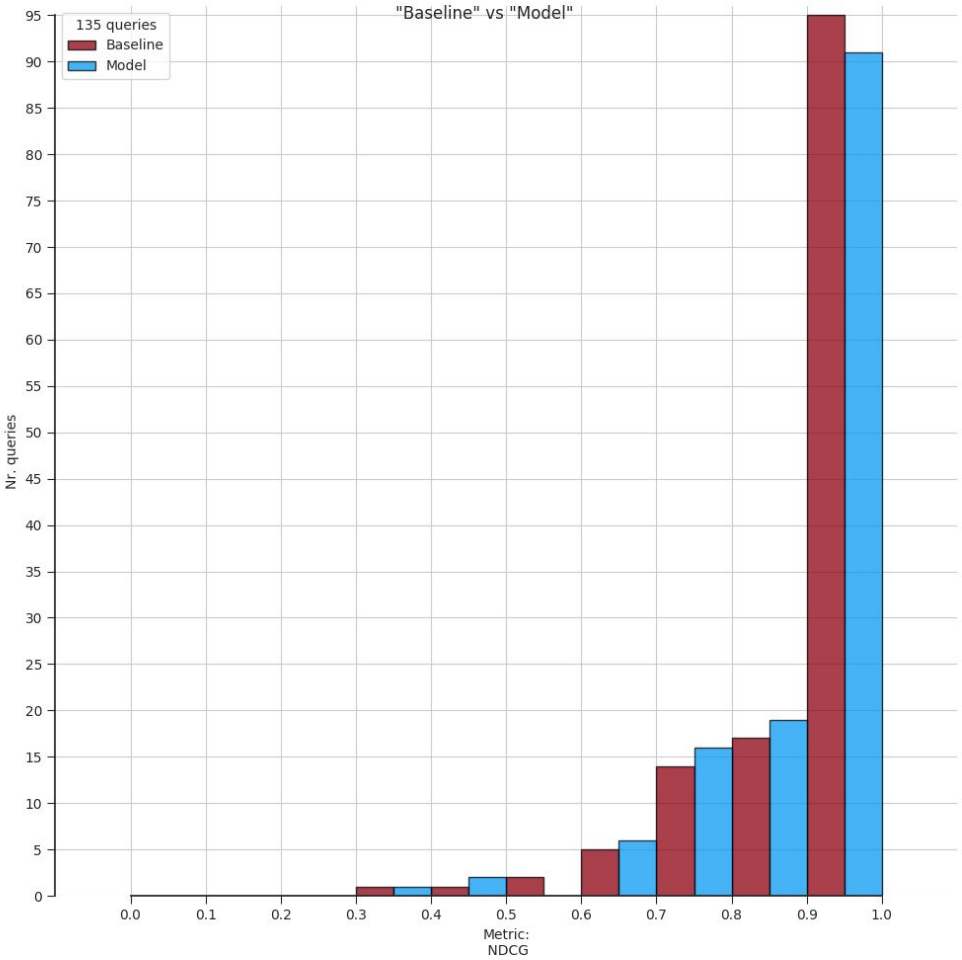
- About 3% improvement on that data set
  - using 5-fold cross-validation
- About 2% deterioration on human assessed set

Results



"Baseline" minus "Model"
Means of 135 queries:
Baseline : 0.910
Model : 0.904
$F_{risk} = 0.015$ $N_{risk} = 40$
$F_{reward} = 0.021$ $N_{reward} = 39$
Nr. queries
0
10
20
30
40
50
60
70
80
90
100
-0.2
0.0
0.2
Score difference interval,
metric: NDCG

Results



"Baseline" vs "Model"
135 queries
Baseline
Model
Nr. queries
95
90
85
80
75
70
65
60
55
50
45
40
35
30
25
20
15
10
5
0
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0
Metric:
NDCG

Summary implicit feedback

- Query underspecification can be solved by filtering
  - Because there are still enough usable queries left
- Assessment ‘underspecification’ becomes ‘ambiguity’
  - Problems with:
    - different subgroups of user behaviour
      - click on odd ***or*** relevant results
    - ambiguity of how people use UI
    - position bias (?)

Summary / conclusion

- Explicit feedback
  - Few data
  - Good improvements
  - Too small set to deploy
- Implicit feedback
  - Much data
  - Small improvements
  - Safe to deploy


Thanks!
Any questions?
contact: vanbelle@textkernel.nl
join us: textkernel.careers