Learning to Rank Places

Shuaib Yunus

Disclaimer

This thesis was carried out while under the employ of REACH NOW. The dataset used is collected from apps owned by REACHNOW

Problem Statement

Solve the ranking problem for geospatial search using Learning to Rank's (LTR) machine learning and deep learning based approaches.

Current Location

haupt

DEPARTURE now

STATIONS

Hbf
834 m – Düsseldorf, Deutschland

Hauptbahnhof
7.3 km – Neuss, Deutschland

Hauptfriedhof
9 km – Neuss, Deutschland

Hauptbahnhof
17 km – Solingen, Deutschland

Hbf
19 km – Krefeld, Deutschland

PLACES

Edeka Haupt
7.7 km – Sternstraße 38, 41460 Neuss, Deutschland

Haupt
39 km – Hüttenstraße 47a, 45888 Gelsenkirchen, Deutschland

Pelias - Existing Search Engine

Based around Elasticsearch:

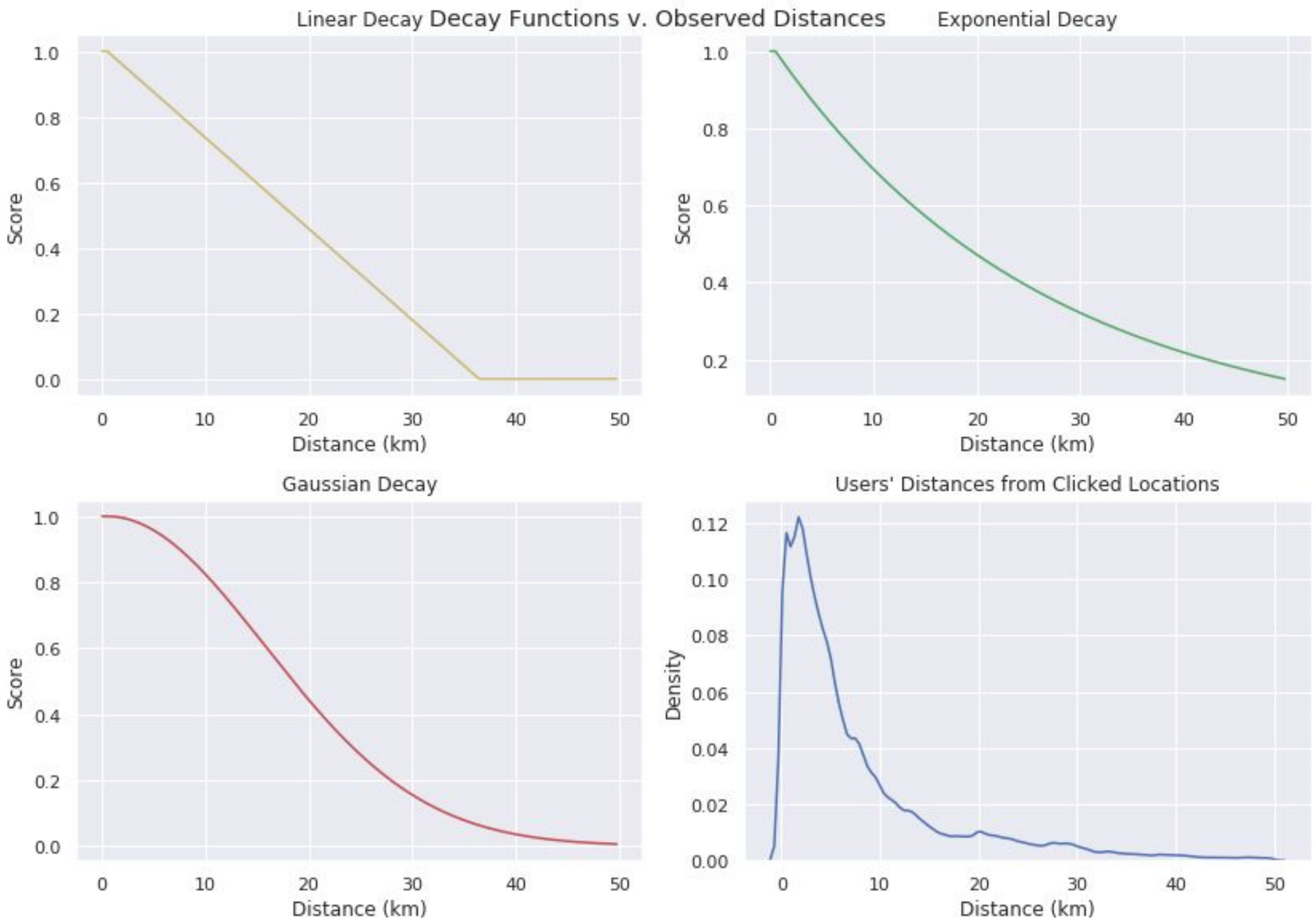
- Performant indexing pipeline for millions of places.
- Sophisticated handwritten analysis chains.
- Intricate Elasticsearch queries to match query texts.

Limitations of Pelias

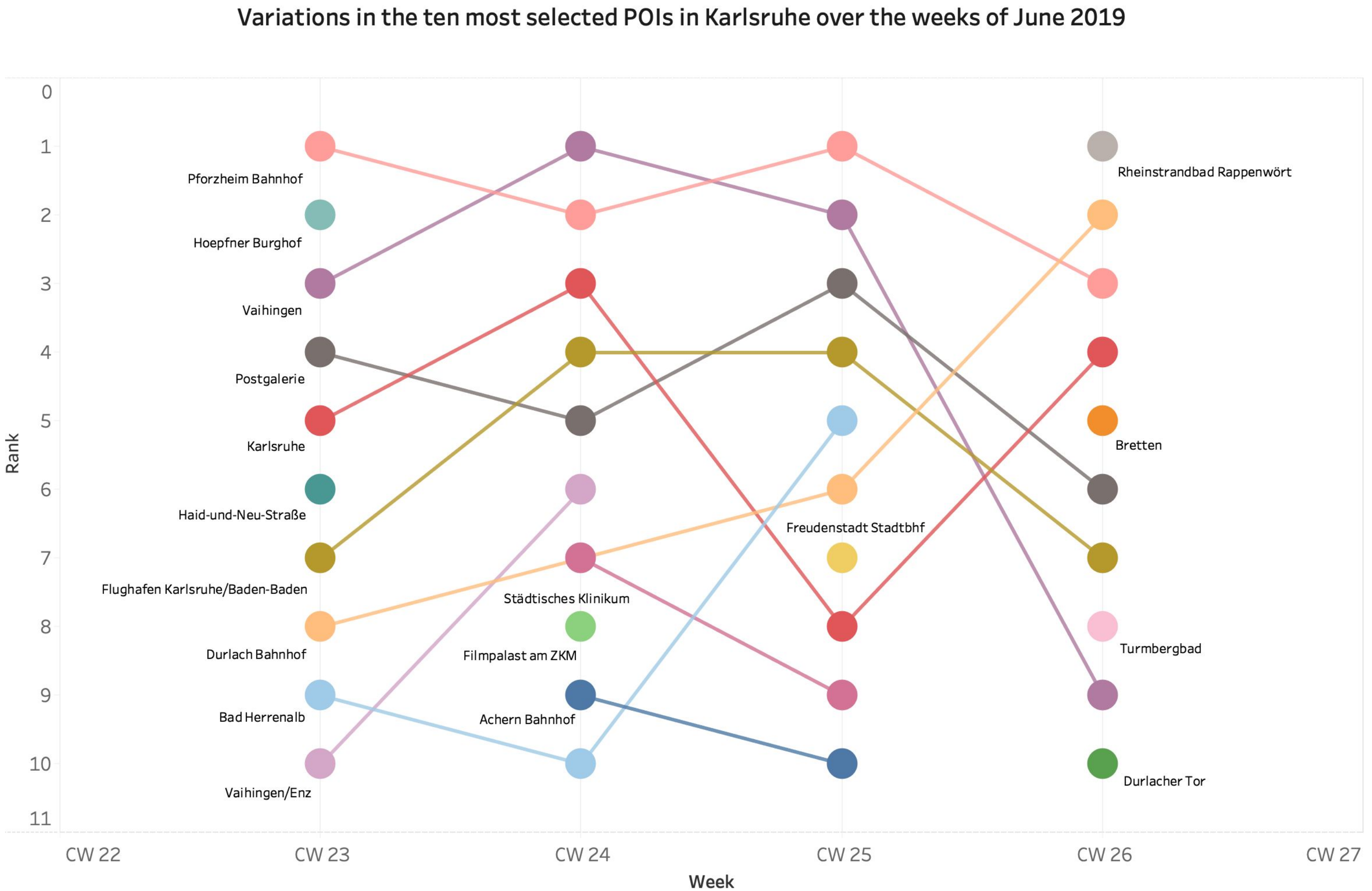
Limitations of Pelias are based around Elasticsearch as well.

- Learning from Historical Data
- Location Biasing
- Temporal Relevance
- Query Parsing
- Location Sharing

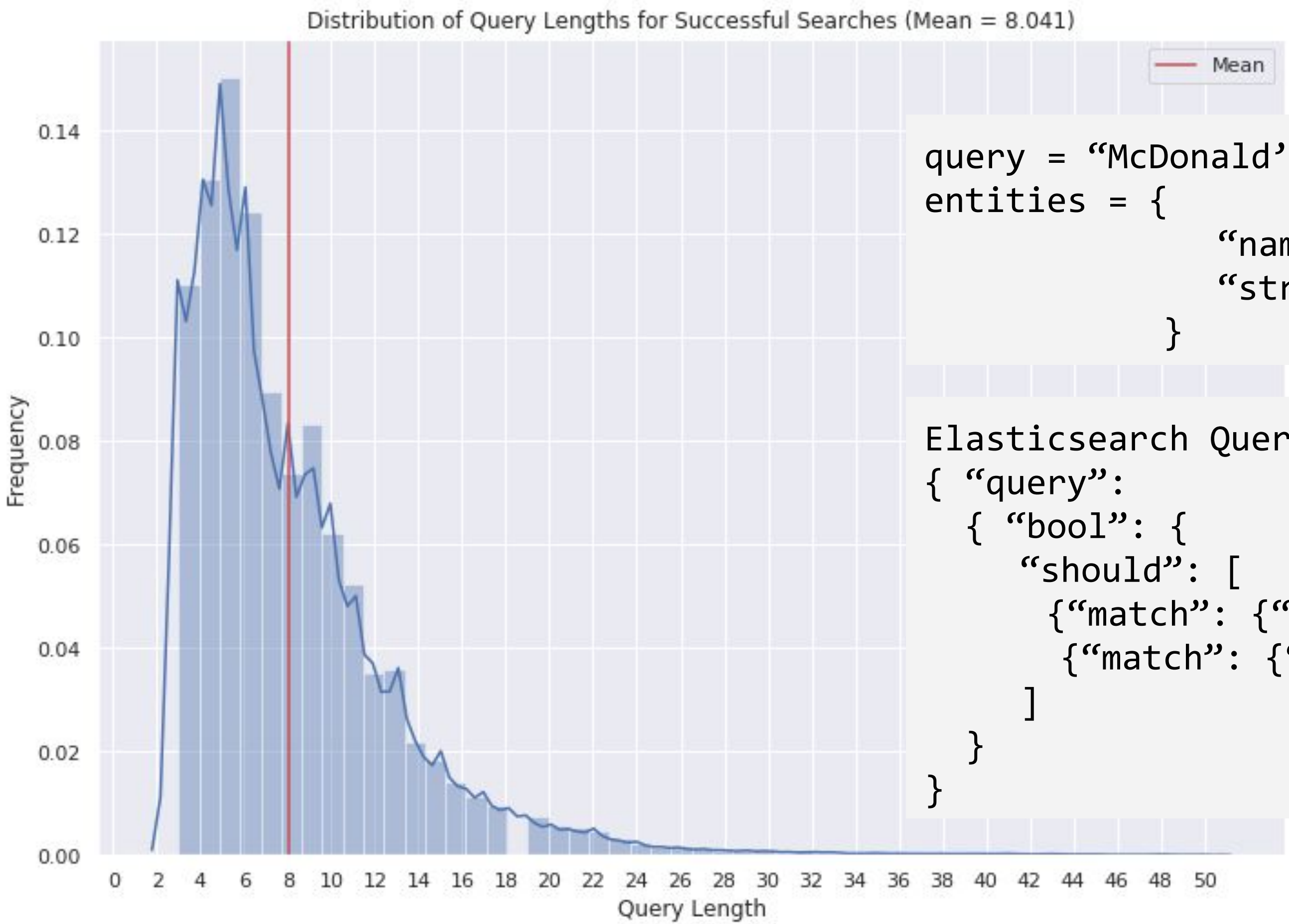
Location Biasing



Temporal Relevance



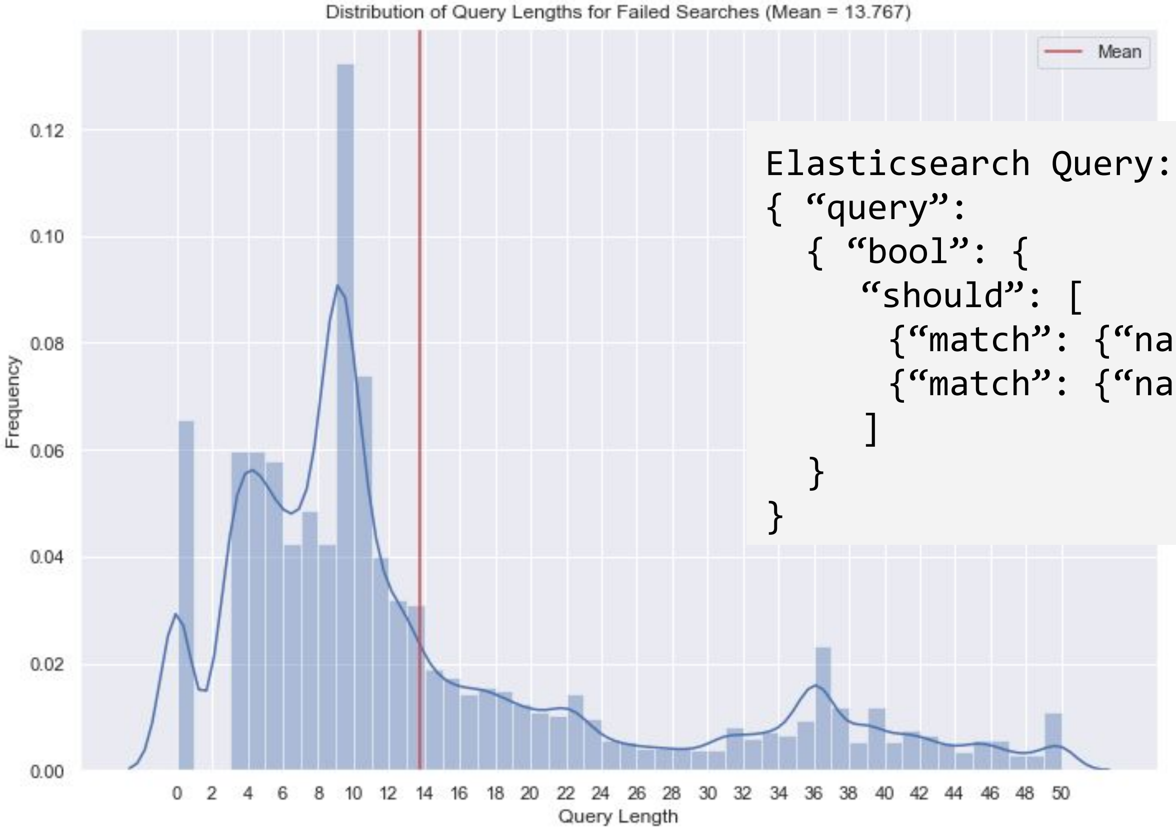
Query Parsing - Successful Queries are short




```
query = “McDonald’s Friedrichs”
entities = {
              “name”: “mcdonalds”,
              “street”: “friedrichs”
           }
```

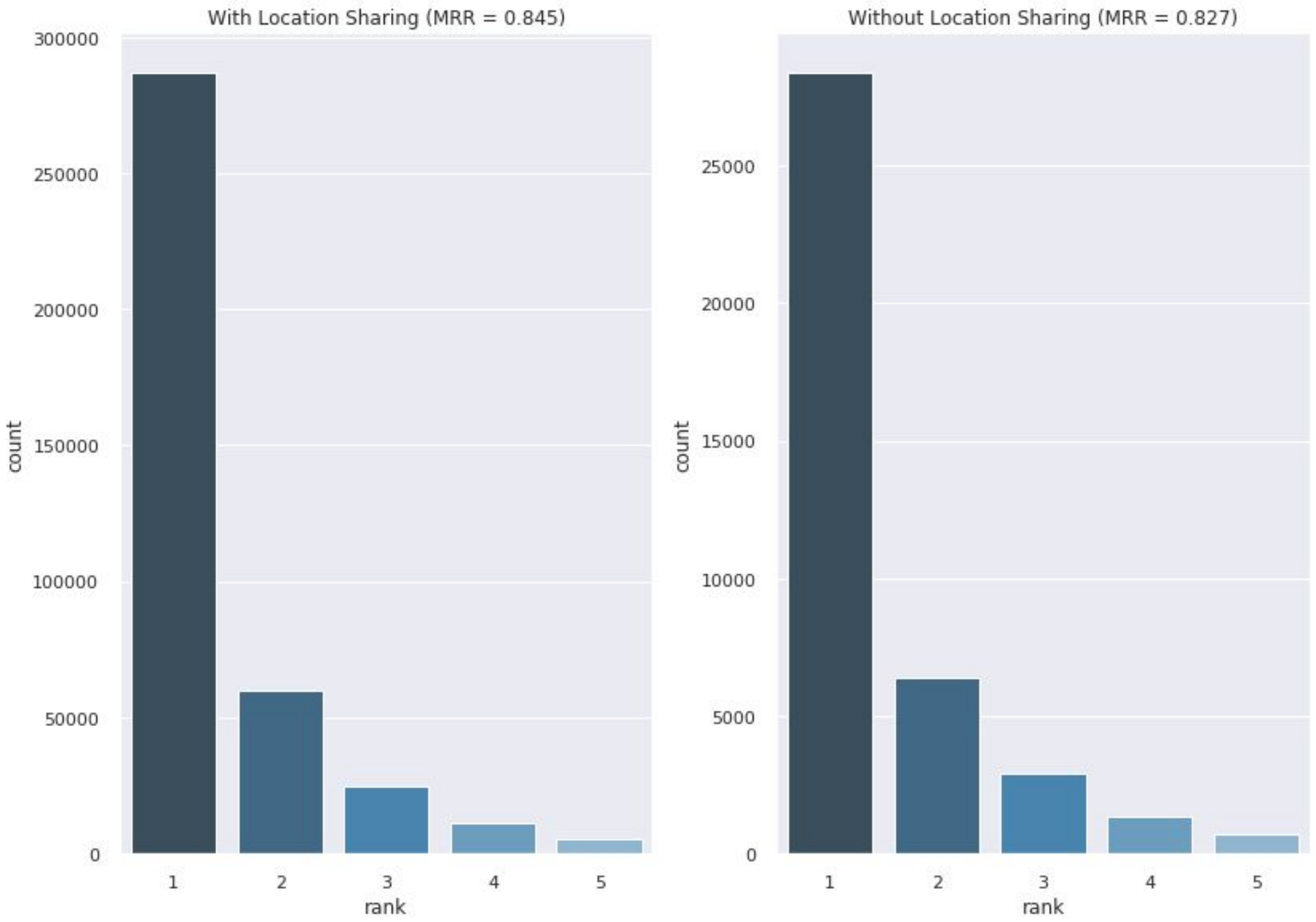
```
Elasticsearch Query:
{ “query”:
  { “bool”: {
     “should”: [
       {“match”: {“name”: “mcdonalds”}},
        {“match”: {“street”: “friedrichs”}}
     ]
  }
}
```

Query Parsing - Failed Searches are lengthy




```
Elasticsearch Query:
{ “query”:
  { “bool”: {
     “should”: [
      {“match”: {“name”: “mcdonalds”},
      {“match”: {“name”: “friedrichs”}}
     ]
  }
}
```

Location Sharing



Learning to Rank (LTR)

LTR as an Empirical Risk Minimization Problem

The goal is to learn a scoring function $\boldsymbol{h} : \mathbb{R}^n \to \mathbb{R}$, which minimizes:

$$\hat{R}(h) = \frac{1}{n} \sum_{q=1}^{n} L\left(\pi\left(h, X_q\right), y_q\right)$$

An LTR algorithm chooses the scoring function f that minimizes the empirical risk $\hat{R}(h)$:

$$f = \arg\min_{h \epsilon \mathcal{H}} \hat{R}(h)$$

The rankings obtained from $\pi(f, X_q)$ should output the best ordering based on the relevance judgements in the form:

$$y_i^q > y_j^q \Leftrightarrow f\left(d_i^q\right) > f\left(d_j^q\right)$$

Dataset

Metadata

app = KVV.mobil

period = 3 months

search results = 3 mil

no. of selected = 500 k

Label

Context Features

Per-Item Features

Dimension	Description
freq	Number of times the place was selected
Query text	Query text
focus point latitude	Latitude of the user
focus point longitude	Longitude of the user
timestamp	Timestamp of the search
target	Whether the search is for an origin or destination
name	Name of the place
locality	Locality of the place
neighbourhood	Neighbourhood of the presented place
borough	Borough of the place
place latitude	Latitude of the place
place longitude	Longitude of the place
type	Type of the place i.e. poi, address, or station

Evaluation Metrics

Mean Reciprocal Rank (MRR)*

$$MRR = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{rank_i}$$

Mean Average Precision at k (MAP@k)

$$\text{MAP@}k = \frac{\sum_{i=1}^{U} AP_i@k}{n}$$, where $AP_i@k = \frac{\sum_{j=1}^{\min\{k,\rho_i\}} rel_{ij} P_i@j}{\sum_{j=1}^{\min\{k,\rho_i\}} rel_{ij}}$ & $P_i@k = \frac{TP_i}{TP_i + FP_i} = \frac{\sum_{j=1}^{\min\{k,\rho_i\}} rel_{ij}}{k}$

Normalized Discounted Cumulative Gain at k (NDCG@k)

$$NDCG_i@k = \frac{DCG_i@k}{IDCG_i@k}$$ Where, $IDCG_i@k = \sum_{j=1}^{|REL_k|}\frac{2^{rel_j}-1}{\log_2(j+1)}$ & $DCG_i@k = \sum_{j=1}^{k}\frac{2^{rel_j}-1}{\ln(j+1)}$

LTR Approaches

<u>Pointwise</u> à la McRank, Ordinal Regression

$$L\left(\pi\left(f, X_q\right), y_q\right) = \frac{1}{n}\sum_{i=1}^{n}\left(f\left(d_i^q\right) - y_i^q\right)^2$$

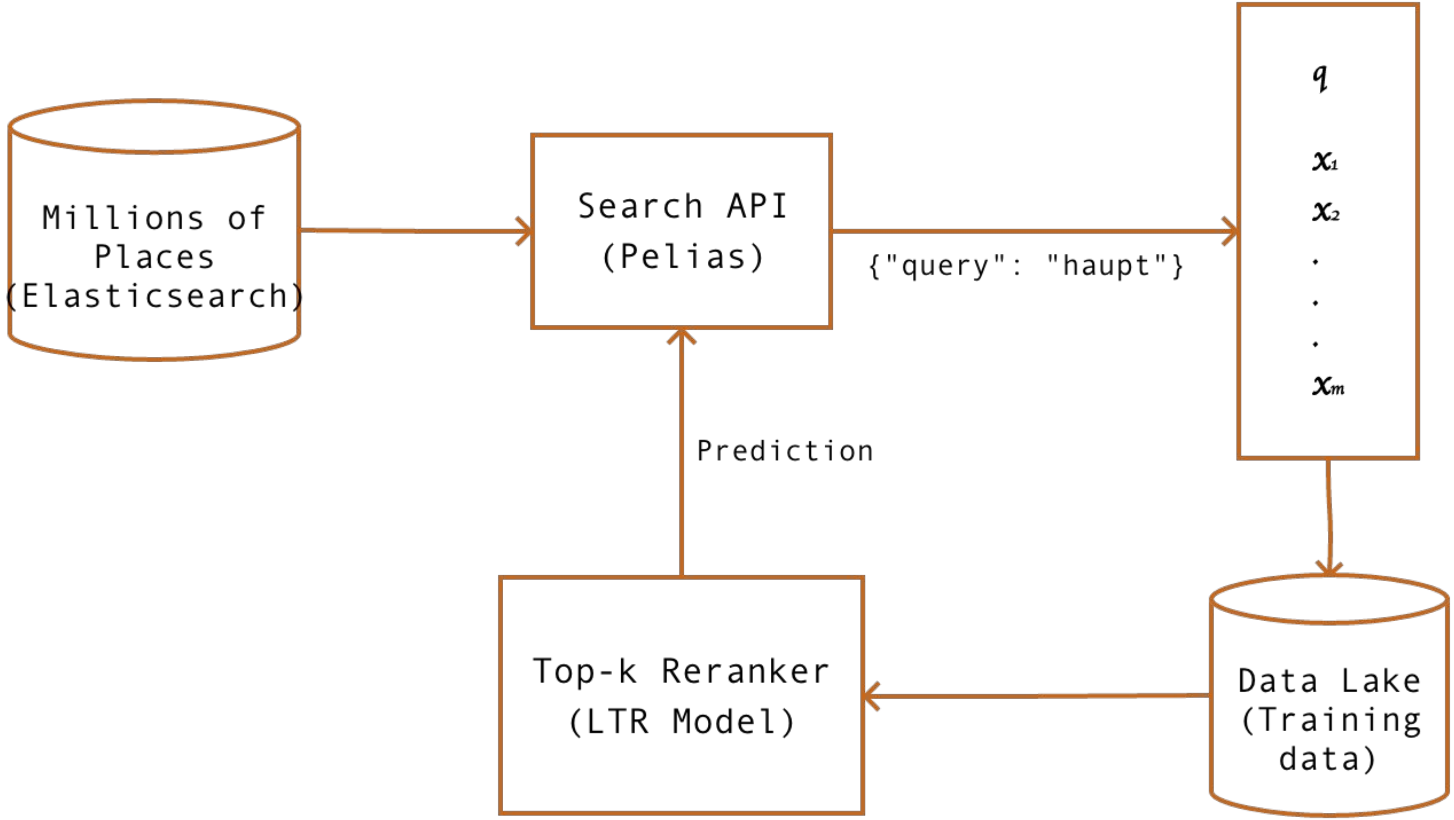
<u>Pairwise</u> à la RankingSVM, RankNet

$$L\left(\pi\left(f, X_q\right), y_q\right) = \sum_{(i,j):\, y_i^q < y_i^q} \log\left(1 + \exp\left(f\left(d_i^q\right)\right) - \left(f\left(d_j^q\right)\right)\right)$$

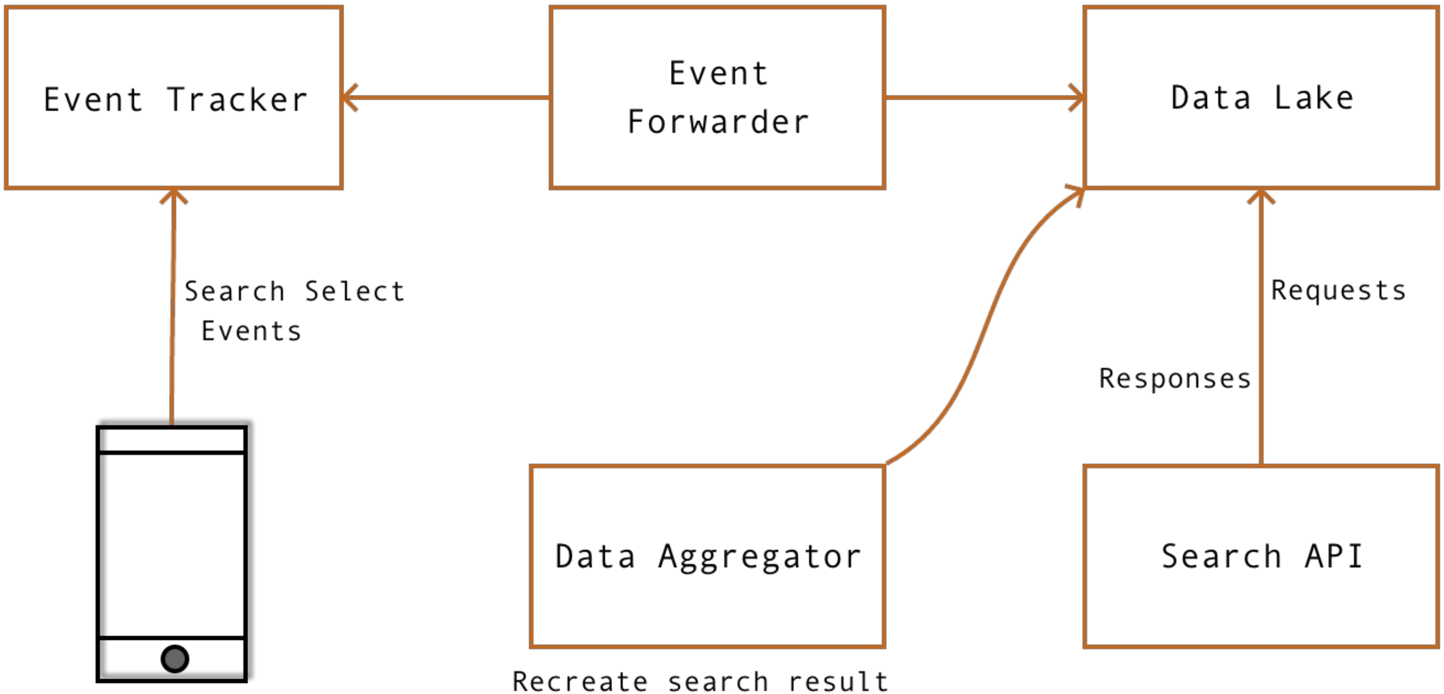
<u>Listwise</u> à la LambdaRank, LambdaMART $\lambda_{uv} = \frac{-1}{1 + e^{f(x_u) - f(x_v)}}$

LambdaMART and LambdaRank uses pairwise errors, but with weighted gradients based on misranked positions.

Learning to Rank Framework Used



Data Engineering



Recreate search result lists by correlating select events with responses from the search API.

Feature Engineering

Temporal Features i.e. hour, day, month, year.

$$x_{\sin} = \sin\left(\frac{2*\pi*x}{\max(x)}\right) \qquad x_{\cos} = \cos\left(\frac{2*\pi*x}{\max(x)}\right)$$

Spatial Features i.e. user’s location, place’s location.

“lon,lat” => [lon, lat]

Textual Features i.e. query text, place name, city etc.

Hashing vectorizer on character ngrams of range [2,5].

Categorical Features i.e. language, place type (stop, address, poi) etc.

One-hot encoder

Train-Test-Validation Split

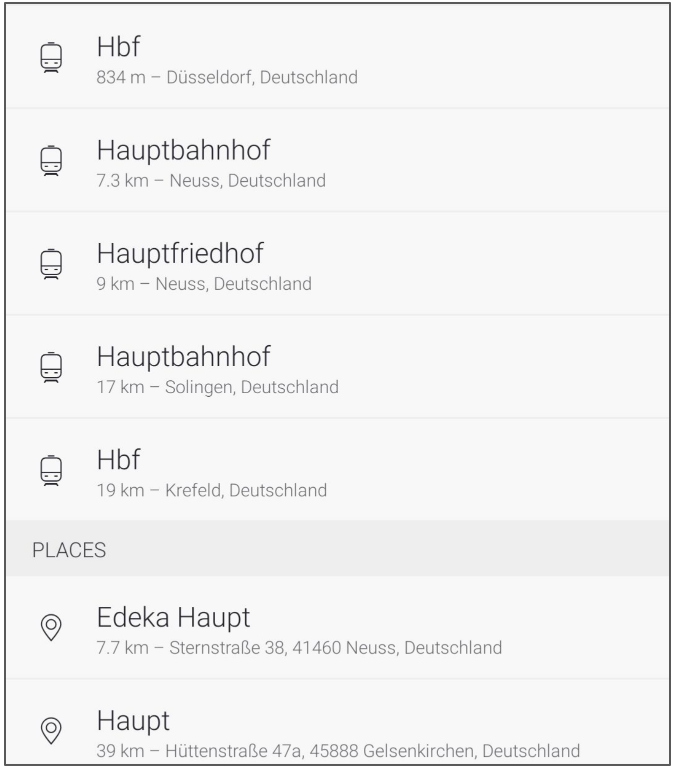
Test = 20%

Validation = 20% of (100% - Test) = 16%

Train = 100% - (Test + Validation) = 64%

Splits treat query groups as a whole, as opposed to the normal split on each observation.

1 query group



Tree-Based Ranker

ML Algorithm: Gradient Boosted Trees

LTR Algorithms: LambdaRank, LambdaMART

Libraries: XGBoost, LightGBM

Hyperparameters:

- Objective function i.e. MAP, NDCG, Pairwise
- Scaling of positive weights i.e. selected results
- Number of rounds i.e. number of trees.
- Minimum child weight
- Maximum tree depth

Neural Ranker

ML Algorithm: Neural Network

LTR Algorithms: LambdaRank, LambdaMART

Libraries: TensorFlow Ranking

Optimizations: Multi-Item Scoring (Groupwise Scoring Functions), Ranking Metric Optimization (LambdaLoss)

Hyperparameters:

- NN Architecture i.e. activation fn, hidden layers, etc.
- LambdaLoss Metric i.e. MRR, NDCG, Mean Squared, etc.
- Group size for Groupwise Scoring Functions

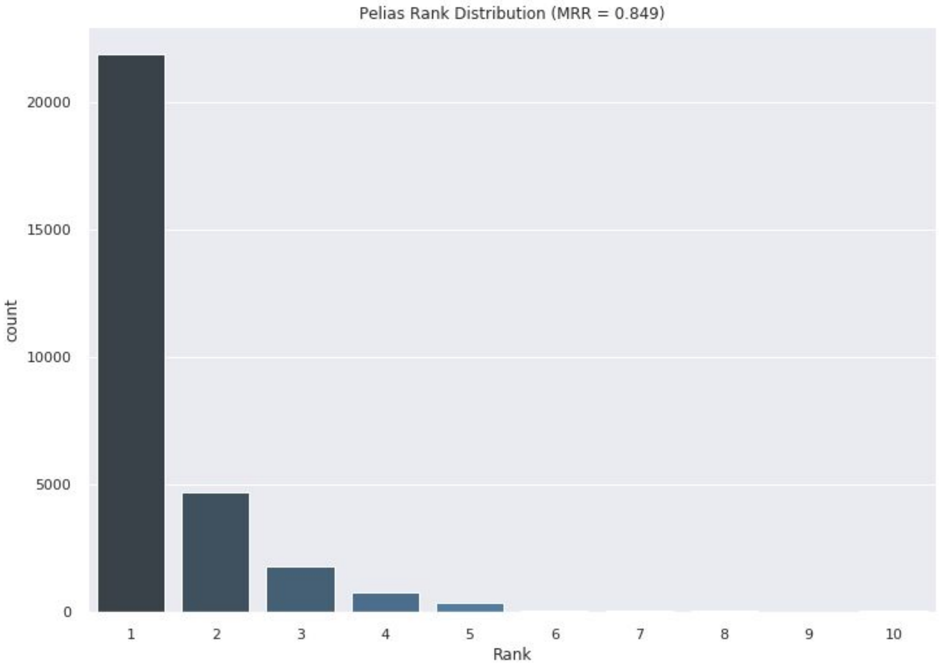

BARI
TARANTO
Potenza
Matera
Barletta
Trani
Andria
Bisceglie
Molfetta
Monopoli
Fasano
Altamura
Gravina in Puglia
Santeramo in Colle
Gioia d. Colle
MESSINA
Golfo di Policastro
Golfo di Taranto
Golfo di Salerno

Results

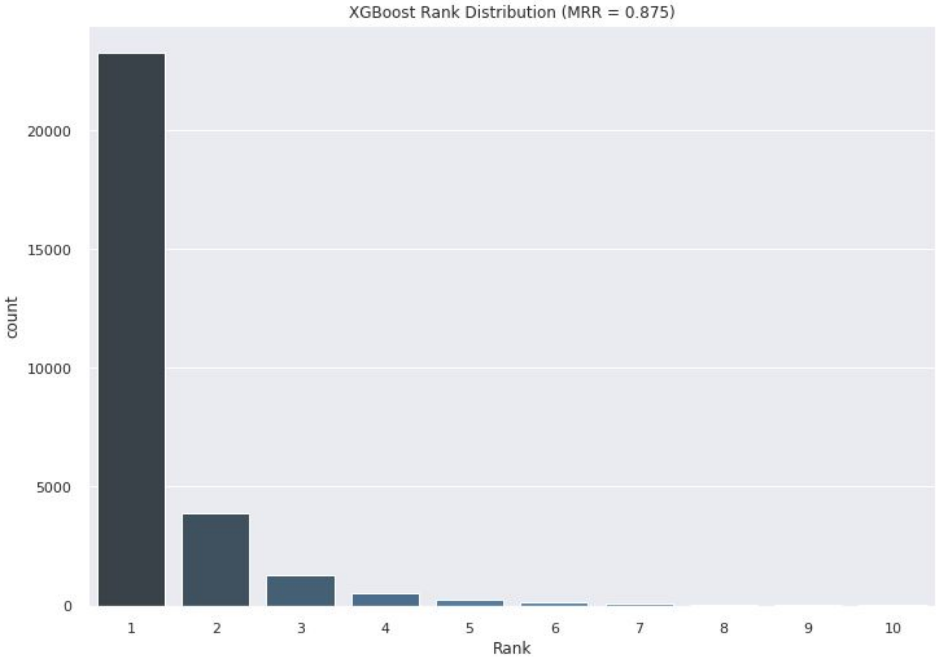
Model Comparisons

Model	MRR	
Pelias	*0.8493*	-
SVM^{rank}	0.7415	-13%
XGBoost	0.8754	+3.1%
LightGBM	**0.8922**	**+5.1%**
TF-Ranking (Linear)	0.8675	+2.2%
TF-Ranking (Deep)	0.8559	+0.78%

Rank Distributions

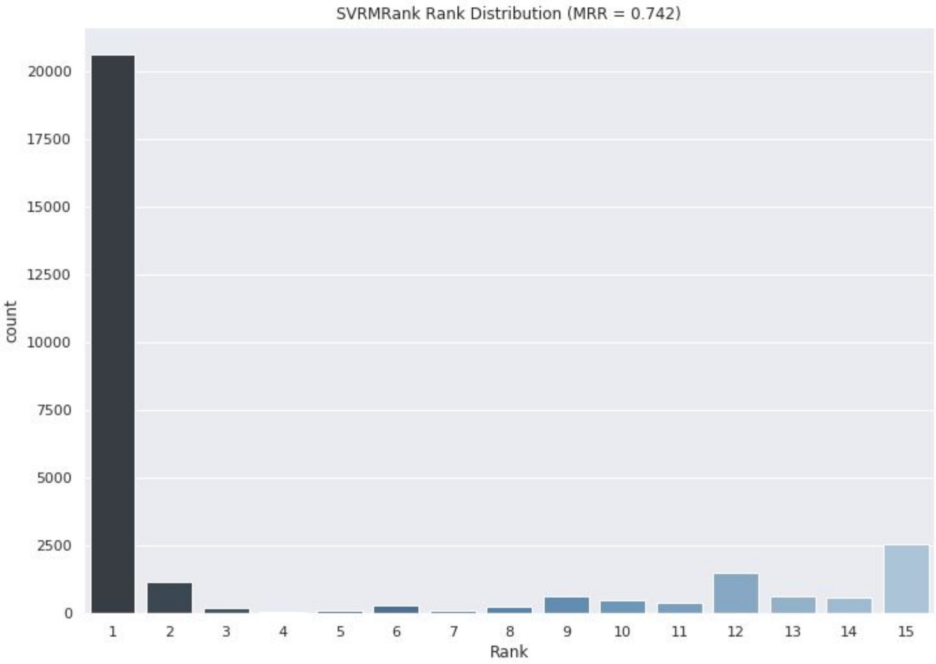


Pelias

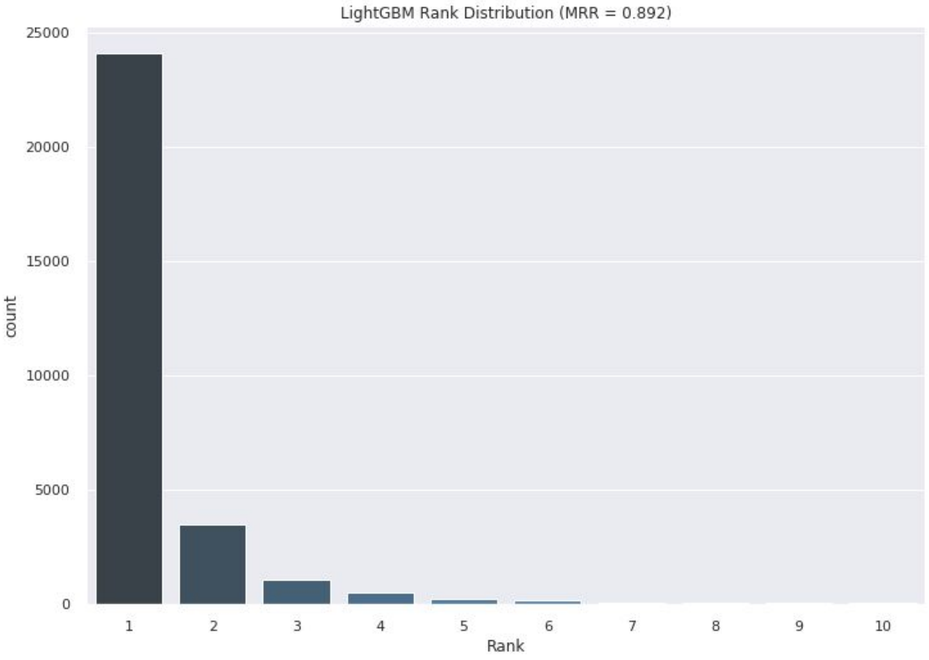


XGBoost

Rank Distributions (cont.)



SVM^{rank}



LightGBM

Tree-Based Ranker - Hyperparameter Tuning

Model	MRR
XGBoost (Default)	0.8531
XGBoost (HP Tuned)	**0.8754**
LightGBM (Default)	0.8826
LightGBM (HP Tuned)	**0.8922**

Tree-Based Feature Importances

Top 20 non-textual features

Weight	Feature
0.1574	x0_stop
0.0036	x0_address
0.0010	x0_poi
0.0005	place_lon
0.0005	place_lat
0.0003	focus_lat
0.0002	focus_lon
0.0000	month_sin
0.0000	hr_cos
0.0000	hr_sin
0.0000	day_sin
0.0000	is_weekend
0.0000	day_cos
0	month_cos

Top 20 features

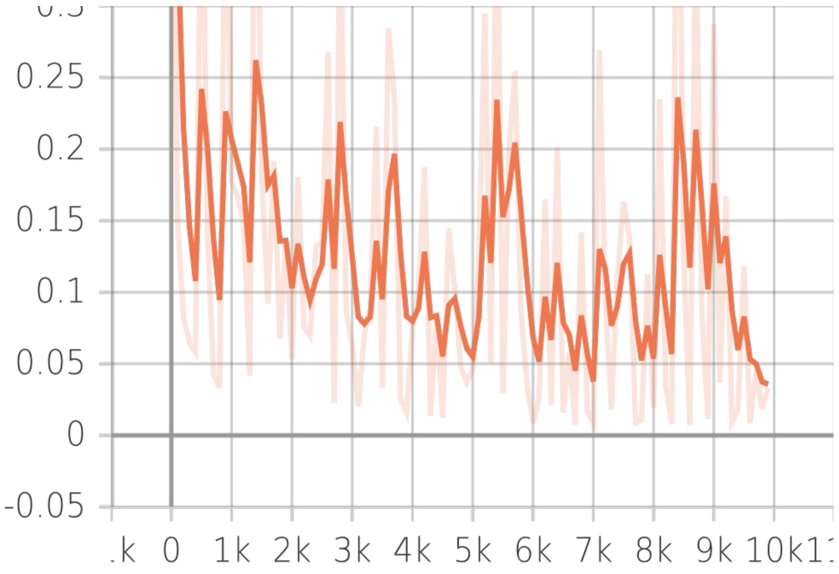
Weight	Feature
0.1574	x0_stop
0.0173	he
0.0173	he
0.0173	he
0.0138	dur
0.0138	dur
0.0138	dur
0.0130	ruh
0.0130	ruh
0.0130	ruh
0.0111	(sp
0.0111	(sp
0.0079	markt
0.0079	markt
0.0053	baden
0.0053	baden
0.0053	baden
0.0050	mark
0.0050	mark
0.0048	ruchs

Neural Ranker Architectures

#	Model	MRR
1	**Linear (sigmoid) (0 hidden layers)**	**0.8675**
2	**FC-64-32-16: 80% dropout**	**0.8559**
3	FC-32 (shallow network, 80% dropout)	0.8523
4	FC-512-256-32 (groupwise with size=5)	0.8247
5	FC-512-256-32	0.8070
6	FC-512-256-32 (tanh)	0.8039
7	FC-64-32-16	0.8038
8	FC-512-256-32 (approximate MRR loss)	0.7790
9	Non-Linear (ReLU) (0 hidden layers)	0.7732

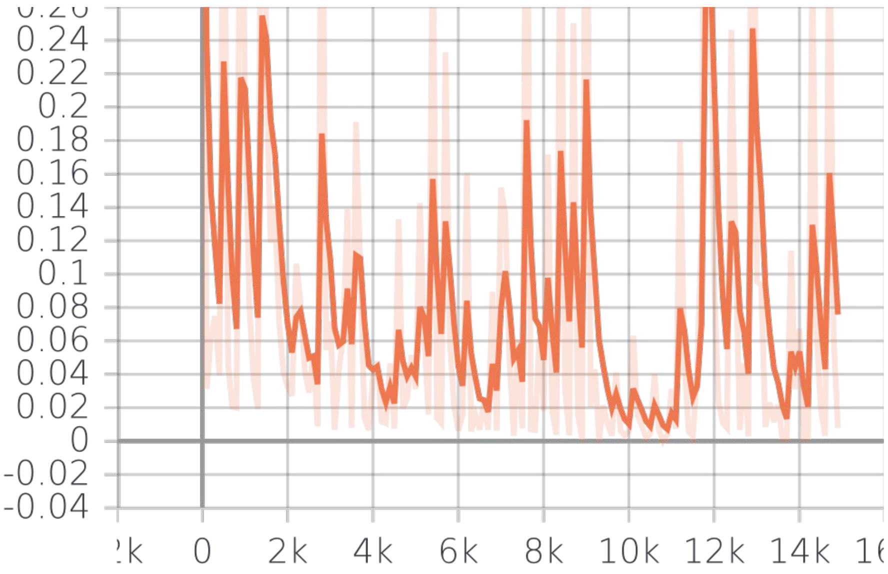
Neural Ranker - Training Loss

Linear (Sigmoid)



MRR = 0.8675

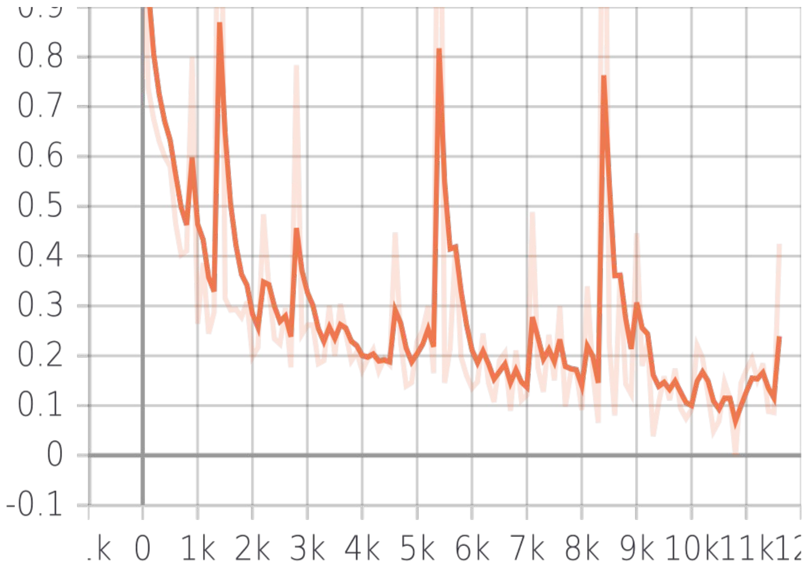
FC-512-256-32



MRR = 0.8070

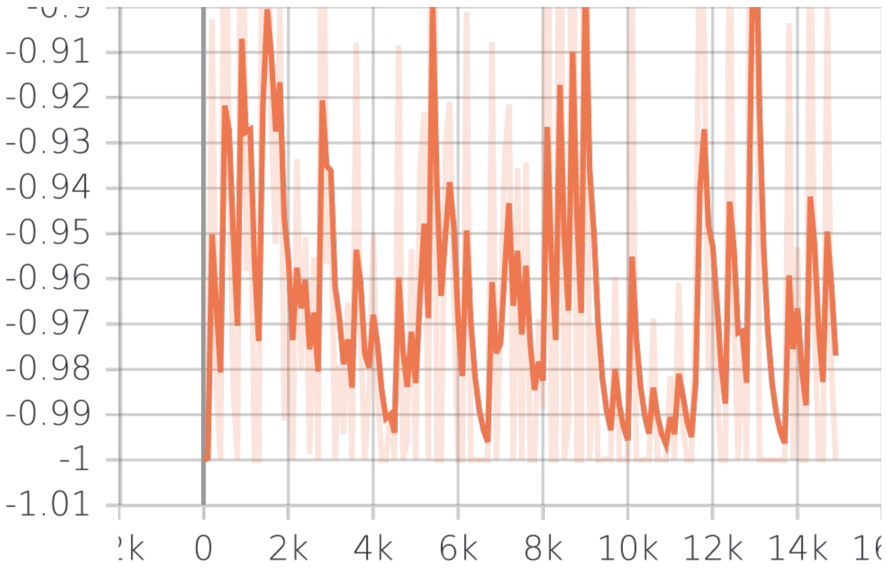
Neural Ranker - Training Loss

FC-64-32-16 (80% Dropout)



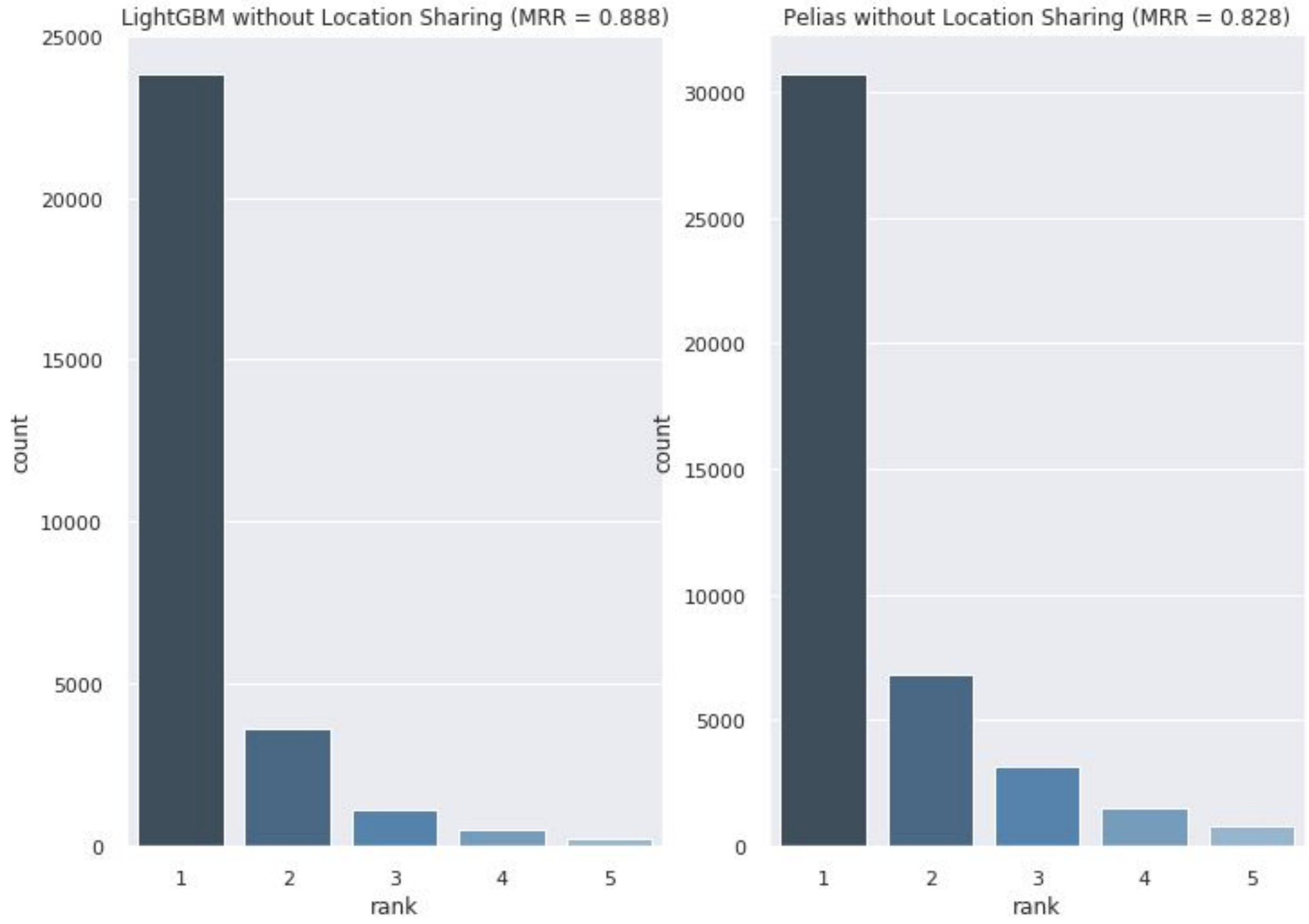
MRR = 0.8559

FC-512-256-32 (MRR Loss)



MRR = 0.7790

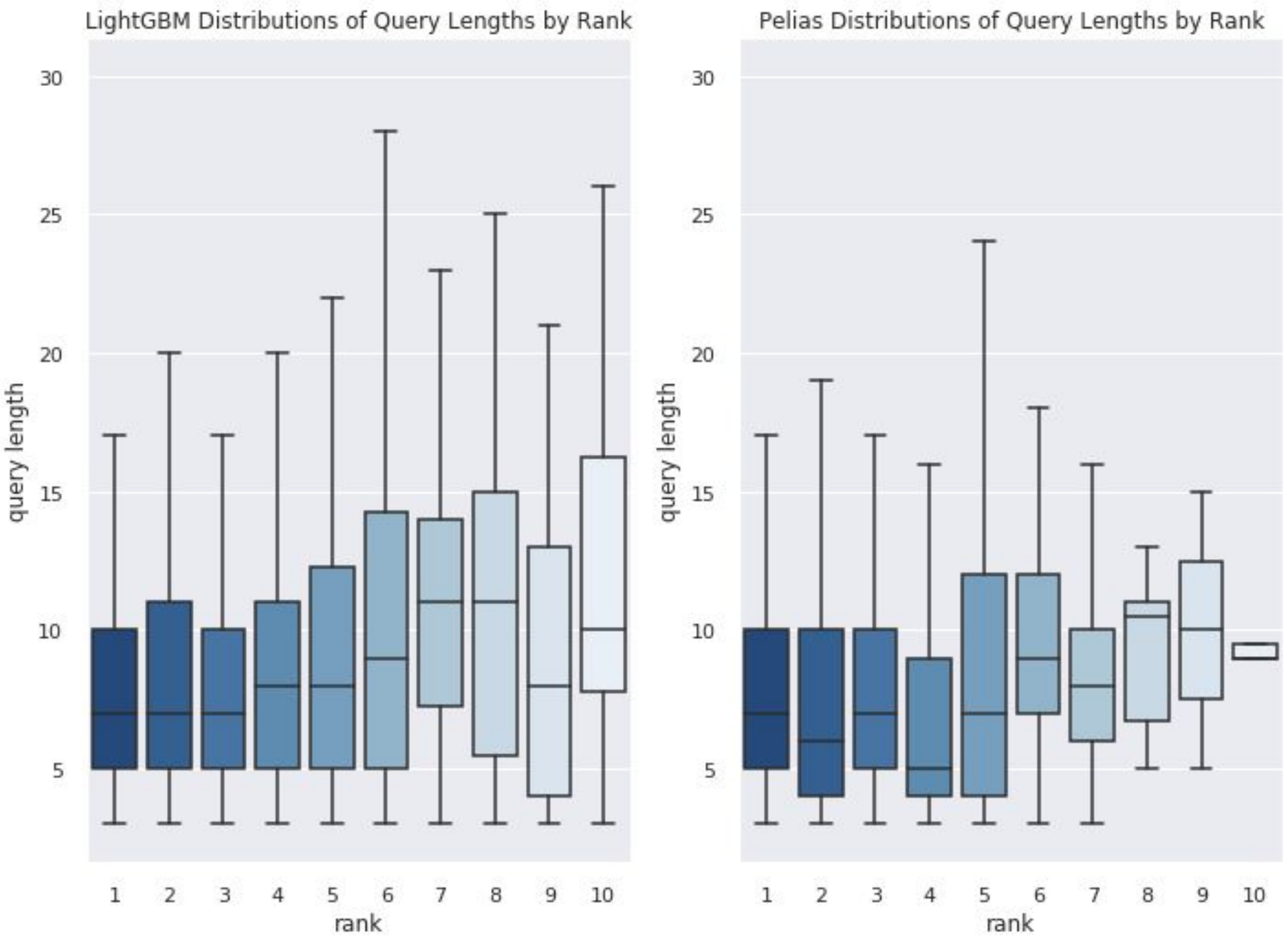
Model vs Pelias - Location Sharing



Model vs Pelias - Query Parsing

Query Character Length Range	LightGBM MRR	Pelias MRR
[3, 10]	0.9003	0.8481
[11, 20]	0.8721	0.8531
[21, 30]	0.8044	0.8612
[31, 40]	0.5647	0.8167

Model vs Pelias - Query Parsing (cont.)



Tree-Based Rankers vs Neural Rankers

	Tree-Based	Neural
Training Time		
Hyperparameter Tuning		
Ease of implementation		
Cool factor		

Conclusion

- LTR models show improvement as top-k rerankers on top of Pelias.
- Tree-based rankers produced higher MRR scores than neural rankers.
- Both tree-based and neural rankers reveal potential for significant improvements on modest tuning.

Future Research

- Interaction between Pelias and the LTR rankers to optimize for different respective retrieval phases.
- Explore applicability of unbiased learning to rank methods.
- Incorporating unsuccessful queries during training.
- Rankers for worldwide search using city-based datasets.

Thank you!


Bornholmer Str.
Pankstr.
Voltastr.
Gesundbrunnen
Bernauer Str.
Humboldthain
Nordbahnhof
Oranienburger Str.
Rosenthaler Platz
Weinmeisterstr.
Hackescher Markt
Reinickendorfer Str.
Schwartzkopffstr.
Naturkundemuseum
Oranienburger Tor
Hauptbahnhof
U55
Bundestag
Friedrichstr.
RB21
RB22
RE1-RE2-RE7-RB1
Bellevue
Brandenburger Tor
U55
TXL
Französische Str.
Mohrenstr.
Potsdamer Platz
Hansaplatz
Stadtmitte
Hausvogteiplatz
Mendelssohn-Bartholdy-Park
Anhalter Bahnhof
Kochstr.
Checkpoint Cha
Wittenbergplatz
Kurfürstenstr.
Gleisdreieck
Nollendorfplatz
Bülow-
Möckernbrücke
Hallesche Tor
U4
Mehringe
Yorckstr.
Großgörschenstr.
Kleistpark
Yorckstr.
Gneise