Relevance Ranking for Vertical Search Engines

Web Searching

- Current challenge: finding relevant results for targeted and specific queries
- Searches that are focused on few specific areas:
  - For example, if you're planning a trip, you may want results about airplane itineraries, baggage checking policies, traffic leading to airports, etc..
- General search engines don't have any way to narrow in on domain-specific information
- **Vertical search engines**, which focus on one "vertical slice" of the internet, can be useful in gathering more in-depth information for a given domain
- Also allows advertisers to provide more targeted ads for a user

Vertical Search Engines

- Vertical search engines work by leveraging domain knowledge, as well as focusing on specific user tasks
- One core component is **relevance ranking**, which is sorting results in the order that is most likely relevant to the query
- There are also two classes of vertical search engines: single domain ranking and multidomain ranking
- Single domain ranking is focused on one specific vertical, such as news or medical domains
- Multidomain ranking involves multiple verticals to get aggregated vertical ranking, multiaspect ranking, and cross-vertical ranking

Learning-to-rank approach

- Learning-to-rank(LTR) algorithms have been successful in optimizing loss functions based off editorial annotations
- Typically the process goes like this:
  - Collect URL-query pairs
  - Ask editors to score the pairs with a relevance grade (perfect, excellent, good, fair, bad)
  - Apply a LTR algorithm to train on data
- To evaluate, we use discounted cumulated gain(DCG)

$$\mathrm{NDCG}_n = Z_n \sum_{i=1}^{n} \frac{G_i}{\log_2(i+1)}$$

where n = number of documents,
G_i is the relevance grade for that document,
Z_n is some normalization factor

- This penalizes documents that appear later, but not by too much

Combining Relevance and Freshness

- Aside from just relevance, we also want to introduce a freshness grade to our URL-query pairs, especially for news searches
- Similar to relevance, we have different grades of freshness: very fresh(+1), fresh(0), a bit outdated(-1), and totally outdated(-2)
- The idea is that using the freshness grade, we can either promote or demote the relevancy grade
- We also introduce an evaluation metric for freshness based off of DCG,

$$\mathrm{DCF}_n = \sum_{i=1}^{n} \frac{F_i}{\log_2(i+1)}$$

- However this requires human editors to keep track of news and provide the actual relevance and freshness judgements

Joint Relevance and Freshness Learning(JRFL)

- We want to create a model that combines the relevance and freshness for a given query and the actual clicked news article, making use of clickthroughs
- We assume that the user's "score", Y_{ni} ,for this URL-query pair can be estimated by the linear combination of the relevance and freshness scores

$$Y_{ni} = \alpha_n^Q \times S_{ni}^F + \left(1 - \alpha_n^Q\right) \times S_{ni}^R$$

- Let :
  - N different queries
  - M different URL-query pairs, such that ($U_{ni} \prec U_{nj}$), in which U_{ni} is clicked but U_{nj} is not
  - X^R_{ni} and X^F_{ni} as the relevance and freshness features for U_{ni} under query Q_n
  - S^R_{ni} and S^F_{ni} are the corresponding relevance and freshness scores for this URL given by the relevance model $g^R(X^R_{ni})$ and freshness model $g^F(X^F_{ni})$
  - α^Q_n as the relative emphasis on freshness aspect estimated by the query model $f^Q(X^Q_n)$, so $\alpha^Q_n = f^Q(X^Q_n)$. To make things easier, we enforce $0 \le \alpha^Q_n \le 1$.

The optimization problem

- For a given set of click logs, we want to determine the models $g^R(X^R_{ni})$, $g^F(X^F_{ni})$, $f^Q(X^Q_n)$ which explain the most pairwise preferences
- We can put this in the form of a constrained optimization problem
- *C* is some tradeoff parameter between model complexity and training error. Set to 5 by the authors.
- ξ_{nij} are nonnegative slack variables that are introduced to account for noise

$$\min_{f_Q, g_R, g_F, \xi} \frac{1}{2}(\|f_Q\| + \|g_R\| + \|g_F\|) + \frac{C}{N}\sum_{n=1}^{N}\sum_{i,j}\xi_{nij}$$

$$\text{s.t. } \forall(n, i, j), \quad URL_{ni} \succ URL_{nj}$$

$$Y_{ni} - Y_{nj} > 1 - \xi_{nij}$$

$$0 \leq f_Q(X_n^Q) \leq 1$$

$$\xi_{nij} \geq 0,$$

Relevance, freshness, and query models

- In order to work with the optimization problem, we also need to define the models used for the relevance, freshness, and query
- The book chooses to use linear models:

$$g_R\left(X_{ni}^R\right) = w_R^\top X_{ni}^R$$

$$g_F\left(X_{ni}^F\right) = w_F^\top X_{ni}^F$$

$$f_Q\left(X_n^Q\right) = w_Q^\top X_n^Q$$

- We can plug this back into our previous equation to get our final JRFL model

Final JRLF model

$$\min_{w_R, w_F, w_Q, \xi} \quad \frac{1}{2}(\|w_Q\|^2 + \|w_R\|^2 + \|w_F\|^2) + \frac{C}{N}\sum_{n=1}^{N}\sum_{i,j}\xi_{nij} \qquad (2.11)$$

$$\text{s.t.} \quad \forall(n, i, j), \quad U_{ni} \succ U_{nj}$$

$$w_Q^{\mathsf{T}} X_n^Q \times w_F^{\mathsf{T}}\left(X_{ni}^F - X_{nj}^F\right) + \left(1 - w_Q^{\mathsf{T}} X_n^Q\right) \times w_R^{\mathsf{T}}\left(X_{ni}^R - X_{nj}^R\right) > 1 - \xi_{nij}$$

$$0 \leq w_Q^{\mathsf{T}} X_n^Q \leq 1$$

$$\xi_{nij} \geq 0.$$

- Due to the associative property of linear functions, we can actually divide the problem into two separate subproblems: the freshness/relevance model estimation and the query model estimation
- Additionally we can use coordinate descent to solve both of them

Algorithm: Coordinate Descent for JRFL

Input: A collection of click preferences $L = \Big\{\big[X_n^Q, \big((X_{ni}^R, X_{ni}^F) \succ (X_{nj}^R, X_{nj}^F)\big), \ldots, \big((X_{nk}^R, X_{nk}^F) \succ (X_{nl}^R, X_{nl}^F)\big)\big]\Big\}$;
Input: Tradeoff parameter C,
Input: Maximum iteration step S,
Input: Relative convergency bound ϵ,
Output: Learned model parameters of (w_R, w_F, w_Q),
Step 0: Randomly initialize (w_R, w_F, w_Q) and set $i = 0$,
Step 1: Update relevancy/ freshness models:

$$(w_R^{(i+1)}, w_F^{(i+1)}) \leftarrow \underset{\mathrm{r}}{\mathrm{argmin}}_{w_R, w_F, \xi} \frac{1}{2}(\|w_R\|^2 + \|w_F\|^2) + \frac{C}{N}\sum_{n=1}^{N}\sum_{i,j}\xi_{nij}$$

with respect to the constraints listed in Eq. 2.11 by fixing the query model o $w_Q^{(i)}$,

Step 2: Update query model.

$$w_Q^{(i+1)} \leftarrow \mathrm{argmin}_{w_Q, \xi} \frac{1}{2}\|w_Q\|^2 + \frac{C}{N}\sum_{n=1}^{N}\sum_{i,j}\xi_{nij}$$

with respect to the constraints listed in Eq. 2.11 by fixing the relevancy/ freshness models to $(w_R^{(i+1)}, w_F^{(i+1)})$,
Step 3: Compute object function value defined in Eq. 2.11 $\rightarrow obj$ and increase $i = i + 1$,
Step 4: if the relative change in obj is greater than ϵ and i is smaller than S, go to ***Step 1***; else return $(w_R^{(i)}, w_F^{(i)}, w_Q^{(i)})$

Temporal freshness features (URL part)

- Aside from the usual text matching features which are used for relevance, we also need temporal features for the freshness of the URL and query models
- For the URL freshness, we have:
  - Publication age – the publication timestamp of the document
  - Story age – using regex to extract dates from the document and using the one with the smallest gap to the query date
  - Story coverage – represents the amount of new content that has not been mentioned previously
  - Relative age – the relative age of the document within the list of returned results

Temporal freshness features (query part)

- For query freshness, we have these features:
  - Query/user frequency – how often a query is made within a time slot, compared with amount of unique users making this query
  - Frequency ratio – the relative frequency ratio of a query within two consecutive time slots
  - Distribution entropy – the distribution of when queries are made; generally we expect a lot of queries right after some breaking news
  - Average CTR – the average clickthrough rate of a URL over all other URLs within a time slot prior to when a query was made
  - URL recency – statistics related to the frequency URL-query pair within a fixed time period. If the URLs associated to one particular query are fresh, then the query is likely to be a breaking news query

Experimentation and Testing

- The book tests the JFRL model on data from Yahoo! News search engine over a 2 month period
- A time slot from the previous slide is defined to be 24 hours
- Each of the those features are also linearly scaled within the range [-1, 1] for normalization
- Compared against RankSVM and GBRank algorithms, neither of which explicitly model relevance or freshness
- To quantitatively compare the retrieval performance, Precision, Mean at Precision, and Mean Reciprocal Precision
  - In order to convert document scores to be “relevant” or “not relevant”, we consider anything with a grade of “good” or above to be “relevant”

Analysis of JRFL

- The first thing tested was to see if the coordinate descent in the JRFL model even converges
- Even with different initial states, the model converges , although randomizing seems to converge the fastest
- The weight of the temporal features also suggest the following:
  - For URL freshness features, the smaller the publication age, story coverage, and relative age, the more recent the news article is
  - For query freshness features, the bigger the query frequency and URL recency, and the smaller the distribution entropy, the more users and news reporters are focusing on this event

Table 2.3 Feature weights learned by JRFL.

Feature	Type	Top Three Features
URL freshness	Neg	**age**$_{\text{pubdate}}$(URL|Query) **LM@5**(URL|Query, *t*) **t-dist**(URL|Query)
	Pos	**q_ratio**(Query|t) **pub_frq**(Query|t) **q_prob**(Query|t)
Query model	Neg	**Ent**(Query|t) **pub_dev**(Query|d) **pub_mean**(Query|d)