A Decision Theoretic Framework for Ranking using Implicit Feedback

Onno Zoeter Michael Taylor Ed Snelson John Guiver Nick Craswell Martin Szummer
Microsoft Research Cambridge
7 J J Thomson Avenue
Cambridge, United Kingdom
{onnoz,mitaylor,esnelson,joguiver,nickcr,szummer}@microsoft.com

ABSTRACT

This paper presents a decision theoretic ranking system that incorporates both explicit and implicit feedback. The system has a model that predicts, given all available data at query time, different interactions a person might have with search results. Possible interactions include relevance labelling and clicking. We define a utility function that takes as input the outputs of the interaction model to provide a real valued score to the user's session. The optimal ranking is the list of documents that, in expectation under the model, maximizes the utility for a user session.

The system presented is based on a simple example utility function that combines both click behavior and labelling. The click prediction model is a Bayesian generalized linear model. Its notable characteristic is that it incorporates both weights for explanatory features and weights for each query-document pair. This allows the model to generalize to unseen queries but makes it at the same time flexible enough to keep in a 'memory' where the model should deviate from its feature based prediction. Such a click-predicting model could be particularly useful in an application such as enterprise search, allowing on-site adaptation to local documents and user behaviour. The example utility function has a parameter that controls the tradeoff between optimizing for clicks and optimizing for labels. Experimental results in the context of enterprise search show that a balance in the tradeoff leads to the best NDCG and good (predicted) clickthrough.

Categories and Subject Descriptors

H.3.3 [**Information Systems Applications**]:

Keywords

clickthrough, learning, ranking, metrics

1. INTRODUCTION

This paper presents a system for learning to rank in a decision-theoretic framework. In such a framework each potential top-k ranking is thought of as an action that could be made by the search engine. Then retrieval is a decision procedure, of choosing an optimal action according to a given utility function.

The decision theoretic view of IR has a long-standing tradition (see e.g. [12, 4, 8] for succesful uses). In this paper we explore the idea of using it to learn a ranker based on multiple streams of feedback. The utility function is then not only based on judge labels, but also on characteristics of a user's session. A model is learned on historic data to predict the user's interaction with a result list. Although many characteristics of the user's session could be incorporated in such a utility function, we will mainly concentrate on one particular and important one, namely clicks.

The reason to consider both labels and clicks in the utility function is that each provides a different sort of relevance information:

- Quantity and cost. Click information is available at zero cost as long as the system has some users, and the quantity depends on the level of user activity. Relevance judges are usually paid, so the quantity of labels depends on budget.
- Explicitness. Judges give explicit relevance labels. With clicks, dwell-time, and abandonment, relevance information must be inferred.
- Real user population. Clicks come from the true user population, so may reflect real relevance. Relevance judges in laboratory conditions may disagree with the real users.
- Deep/negative judgments. Relevance judges can be paid to label a large pool of documents per query, including many bad documents. Clicks tend to happen only on top-ranked documents, and gathering negative click information has a detrimental effect on users, because the bad documents must be retrieved near the top.

The question is how to build a model that works well, incorporating explicit and implicit relevance information. One approach (Figure 1a) is to choose an evaluation measure as the gold standard for relevance, such as the label-based metric DCG [6], and build a model to optimize it such as LambdaRank [2] or SoftRank [13]. The inputs may be features characterizing the quality of the query-document

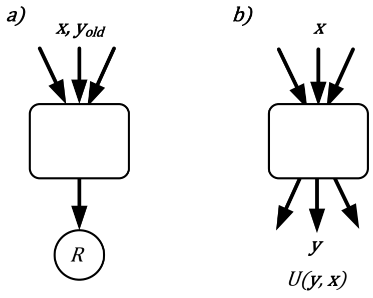

Figure 1: Two different approaches to the incorporation of implicit feedback into a ranker; a) uses historic user behavior as input to predict a single relevance score R. Approach b), proposed in this paper, constructs the best possible model to explain outputs y from inputs x and separately defines a utility function U that puts a preference ordering on possible explicit and implicit behaviors.

match. Historical implicit feedback can be incorporated as additional input features [1]. The output of the model gives a scalar-valued score by which documents are ranked via a sort. Note here that the value of an individual document's score has no practical interpretation.

Our approach (Figure 1b) is different and based on an extension of the decision theoretic framework for IR, as described in [14, 8]. The inputs and outputs of the model are all observable: inputs are query-document features and outputs are implicit/explicit relevance information. The sole task of the model is predicting outputs. We then define a utility U which is a function of these predicted outputs, namely both implicit and explicit judgments and behaviors. Ranking is then a decision procedure, to find the results list with maximum utility.

The specific contributions of this paper are as follows.

- We propose to construct rankers that combine many sources of information using the decision theoretic framework for IR. We discuss what an ideal setup would look like and how it would add diversity to result lists, correctly incorporate real world characteristics such as position bias, balance authoritiveness and popularity, and more.

- As an initial implementation of the approach, we present a Bayesian logistic regression model that predicts both relevance judgments and click rates. The model has one weight per query-document pair that acts as a "memory" of the historic click rate that is not already explained by the other features. We combine it with a crude utility function that is far from the ideal sketched setting, but already introduces many of the potential benefits the combination of two datastreams can bring.

- We evaluate the decision theoretic system in an enterprise search scenario, demonstrating that the click-predicting part of the model can adapt to a new enterprise.

2. RANKING AS A DECISION THEORY PROBLEM

Decision theory is a very well established field which dates back at least to the works of Daniel Bernoulli in the 18th century. The information retrieval problem of presenting a list of results given a specific user query has been interpreted as a decision theory problem in several studies in the past. The probability relevance principle [12] for instance can be motivated from such a view. Interesting and successful applications can also be found in amongst others [8, 4]. In this section we first review the abstract decision problem in its general form, and then move on to describe how it can be applied to incorporate both explicit and implicit feedback in a common framework.

At the basis of the decision theoretic view is a *utility function.* It represents user satisfaction in a single scalar, larger being better. Formally it is a mapping of all relevant quantities of interest (searcher charteristics, query, clicks, dwell time, etc.) to the real line. In the remaining we will make a distinction between two sets of information: *inputs* and *outputs.* Inputs are those quantities that are available *before* a result list needs to be compiled, outputs are those quantities that have become available in the user session *after* the result list is presented to the user. This includes clicks, dwell time, click backs, etc., but also explicit labels if we ask the user to act as a human judge.

The ideal utility function could be very complex incorporating detailed characteristics of a user, intent of the query, etc. It would increase if interesting results were found, decrease as more and more effort is needed to find them. We will discuss some of the potential properties of an ideal utility function in Section 2.1. In real world use we will have to make simplifications, such as is done in Section 2.2.

If we would know ahead of time exactly how a user would interact with a particular search result list it would be easy to select the optimal one. It would simply be *that* result list that maximizes user satisfaction. Since at query time we do not know the user's response, we need to construct a model that predicts user behavior. The optimal decision (the optimal list) is then the list that *in expectation* under the model maximizes the users utility.

In summary and formally we can represent the decision theoretic view of IR as follows. Given a set of inputs (query-document features) $\mathbf{x} \in \mathcal{X}$ the ranker is asked to select an action (result list) $a \in \mathcal{A}$. After performing the action we observe outputs (judgments, user behaviour etc.) $y \in \mathcal{Y}$. A utility function $U : \mathcal{X} \times \mathcal{A} \times \mathcal{Y} \mapsto \mathbb{R}$ assigns a scalar utility to the observed $\mathbf{x}, a, y$-triple[1]. The outputs y in general do not follow deterministically from the inputs $\mathbf{x}$ and action a. A model $p(y|\mathbf{x}, a)$ gives the probability of observing y after selecting a when $\mathbf{x}$ is observed. The optimal action a^* is the action that leads to the maximal expected utility

$$a^* = \underset{a}{\operatorname{argmax}}\, \mathbb{E}_{p(y|\mathbf{x},a)} \left[U\left(\mathbf{x}, a, y\right)\right] . \tag{1}$$

We propose to use the traditional decision theoretic interpretation of IR to combine multiple sources of data in a principled way. We treat the different sources of implicit feedback as extra dimensions in the output vector y.

2.1 Utility functions

The utility function gives a real valued score to a user ses-

[1]Note that alternatively we could include $\mathbf{x}$ and a into the observation y, but this notation emphasizes the flexibility of the approach.

sion that represents his satisfaction. Thinking about what the ideal utility function would look like can easily be dazzling. For a well defined navigational query such as "What is the next train connection between Cambridge and London Kings Cross?" we might argue that finding the answer gives a fixed utility and any work that needs to be done to get to that point (reading snippets, clicking on potential answer pages, clicking back, etc.) will lead to deductions. But what about informational queries? What is the utility for one, two, or three interesting documents in the result list. Do three interesting documents have three times as much utility as a single one, or is there a law of diminishing returns? What is the "cost" of a misleading snippet? Some sources are very authoritive, some have very fresh content. How should these two properties be traded-off? Should that be done in the same way in all contexts? A small time spent thinking about these things leads easily to an extremely complex function.

Even coming up with a procedure of going about constructing a utility function is a difficult problem. Here we discuss briefly two approaches. A first approach would be to conduct lab experiments with users where they are asked to explicitly score their satisfaction with a session. The experiments in [5] form a fascinating approach in that direction for instance. Assume for simplicity that we have a binary satisfaction signal

$$t \in \{\text{thumbs up}, \text{thumbs down}\} \ ,$$

and a simple utility function

$$U(t = \text{thumbs up}) = 1 \quad \text{and} \quad U(t = \text{thumbs down}) = 0 \ .$$

In daily use the explicit satisfaction scores t are not available. To overcome this we could learn, based on $\{t, y\}$-pairs, a special model $p(t|y)$ (not to be confused with the output prediction model) and work with a "learned utility"

$$\tilde{U}(x, y, a) = \mathbb{E}_{p(t|\mathbf{x},y,a)} \left[U\left(t\right) \right] \ . \tag{2}$$

Combined with an output prediction model $p(y|x, a)$ we could then use Equation (1) for ranking.

In a second approach we ask experts to craft a simple utility $U(x, y, a)$ and iteratively improve it. Perhaps in the first version only a few sources of feedback are modelled in y and this is expanded in the next, or we find that certain tradeoffs looked good on paper but used in practice leads to complaints from real users.

In both approaches constructing the model $p(y|x, a)$ is a classical machine learning problem. Using historical $\{x, y, a\}$-triplets we can train and select the appropriate user behavior prediction model. An important benefit in practice is then that the problem of designing a reasonable utility function and constructing a good prediction model can be decoupled. The prediction can be tested on historic data. Adjusting and tuning the utility function can be done incrementally over time without the need of retraining the model with each new attempt.

To summarize: constructing a utility function is a very difficult problem and can leave one with the awkward feeling that a golden standard or ground truth is not available. We would argue that the IR problem simply is this complex. Any choice in a real world system will make some approximation and is likely to require changes and improvements over time.

In the Section 2.1.3 we introduce what arguably is the simplest possible utility function that combines both a signal stream of explicit label feedbacks and a stream of implicit user clicks. It is a simple convex combination of a label based utility and a click based utility intoduced in Sections 2.1.1 and 2.1.2 respectively.

2.1.1 Discounted Cumulative Gain

In some approaches to ranking the aim is to maximize a function of the labels in the result set. It is easy to see that these approaches form a special case of the framework considered here. If we look at the discounted cumulative gain (DCG) [6] for instance we see that it is an example of a utility function that only takes into account the human relevance judgments at every position. It is based on a discount function $d(p)$ over positions $p \in \{1, \ldots, n\}$, and a gain function $g(s)$ over human relevance judgments, e.g. $s \in \{1, \ldots, 5\}$. The position discount function is monotonically decreasing from the top position $p = 1$, to the bottom position $p = n$: $d(1) > d(2) > \cdots d(n)$, and a gain function $g(s)$ that is increasing for better relevance judgments: $g(1) \leq g(2) \leq \cdots \leq g(5)$. If $s[1], \ldots, s[n]$ are the scores received for the documents selected by a, the discounted cumulative gain is given by

$$DCG\left(s[1], \ldots, s[n]\right) = \sum_{p=1}^{n} d(p) g(s[p]) \, . \tag{3}$$

To maximize the DCG we would select and rank such that the expected DCG is highest. The expectation is then with respect to the observation model $p(y|\mathbf{x}, a) = p(s[1], \ldots, s[n]|\mathbf{x}, a)$ which represents the best estimate of the human relevance judgments for the documents selected by a given $\mathbf{x}$

$$a^* = \arg\max_a \mathbb{E}_{p(s[1],\ldots,s[n]|\mathbf{x},a)} \left[\sum_{p=1}^{n} d(p) g(s[p]) \right] \, .$$

Different choices of $g(s)$ lead to different ranking principles (decision rules). If $g(s)$ is convex in s the resulting principle is *risk seeking*: for two documents with the same expected judgment but different variances the document with the larger variance is preferred. This is because a larger than expected judgment leads to a bigger rise in utility than the decrease in utility that results if a lower than expected judgment is encountered. We could say that such a convex gain function leads to a "going for the jackpot" effect. The often used exponential function $g(s) = 2^s - 1$ has this effect. It is important to realize that this is not a conservative ranking principle.

If we have a linear gain $g(s) = s$, the expected utility only involves the expectations of judgments:

$$\begin{aligned} a^* &= \arg\max_a \mathbb{E}_{p(s[1],\ldots,s[n]|\mathbf{x},a)} \left[\sum_{p=1}^{n} d(p) g(s[p]) \right] \\ &= \arg\max_a \sum_{p=1}^{n} d(p) \mathbb{E}_{p(s[1],\ldots,s[n]|\mathbf{x},a)} \left[s[p] \right] \, . \end{aligned}$$

hence we get a ranking principle that simply orders documents according to their expected human relevance judgment:

$$a^* = \arg\max_a \sum_{p=1}^{n} d(p) \mathbb{E}_{p(s[p]|\mathbf{x},a)} \left[s[p] \right] \, .$$

This utility function is an example where the optimal action can be found in $\mathcal{O}(|\mathcal{D}|)$ time, where $|\mathcal{D}|$ is the number of documents in the corpus. This is despite the fact that the space of all possible selections and rankings is $|\mathcal{D}|^n$. This is due to the fact that the judgment probability $p(s[p]|\mathbf{x}, a)$ is not explicitly a function of position (the judge is presented with each document independently). This means that the expected judgment can be computed for each document and the documents simply sorted to obtain the optimal ranking. There are many interesting utility functions that lead to $\mathcal{O}(|\mathcal{D}|)$ ranking principles, but in general approximations might be necessary.

Note that, since there is no element in the utility function that encourages diversity in the results, we need to explicitly add the constraint that links to documents cannot be replicated. Otherwise a^* would be n duplications of the link with the highest expected relevance judgments.

2.1.2 Clicks

An analogous utility function that only takes into account whether or not a user clicked on a document could be given by a "click-DCG" utility

$$U_{\text{clicks}}(c[1], \ldots, c[n]) = \sum_{p=1}^{n} d(p)c[p] \,. \tag{4}$$

If $p(c[p] = 1|\mathbf{x}, a)$ (the probability of a click on the document that was put in position p by a) is modeled based on a link specific and position specific contribution it will in general not simplify to an $\mathcal{O}(|\mathcal{D}|)$ ordering rule. This is because now $p(c[p]|\mathbf{x}, a)$ is explicitly a function of p — any given document will be clicked with a different probability depending on where it is placed. It can be that position and link effects combine in complex non-linear ways. However there are suitable heuristics for ordering in $\mathcal{O}(|\mathcal{D}|)$, e.g. compute the probability a document will be clicked if it were placed in position 1, and order by that.

This click-DCG assigns a positive utility to the act of clicking itself. Philosophically this is not really sound, since the act of clicking is actually a nuisance, and only from the actual reading of an interesting document is utility obtained. So in order to motivate (4) we need to appeal to an argument along the lines of the learned utility in (2): because we have established in the past that the act of clicking on an interesting link leads to an interesting page we can assign an (expected) utility to the act of clicking. However, from a more practical point of view (4) then still has problems. If we motivate the value of a click from an apparent interest in the result page, we assume that all interesting snippets point to interesting landing pages. This will unfortunately not always be the case in practice. To overcome this the utility can be extended by incorporating a minimal dwell time as proxy for an endorsement of the landing page.

To encourage diversity, one simple approach would be to introduce a concave function f of the simple DCG-like sum of clicks:

$$U_{\text{clicks page}}(c[1], \ldots, c[n]) = f\left(\sum_{p=1}^{n} d(p)c[p]\right) \,. \tag{5}$$

This captures the notion that the step from 0 clicks to 1 click on a page is bigger than that from 1 to 2. The transformed utility would penalize systems with click-DCG near zero. For an ambiguous query with several types of result, a ranking optimized to avoid zero click-DCG could potentially present results of each type, hedging its bets by giving a more diverse results list.

To take advantage of this type of diversity-encouraging utility, one must combine it with a model that can capture *correlations* between click events on different documents on a page. For instance, for ambiguous queries, clicks on links to two different interpretations will in general be anti-correlated: someone clicking on a link of one type will be less likely to also click on a link of the other type, presuming they have one interpretation of the query in mind when searching. To do this requires a model for the joint distribution $p(c[1], \ldots, c[n]|\mathbf{x}, a)$, which is in general a difficult modeling task. An independence assumption $p(c[1], \ldots, c[n]|\mathbf{x}, a) = \prod_{p=1}^{n} p(c[p]|\mathbf{x}, a)$ does not capture these correlations, but is a reasonable simplifying modeling assumption if one is using the more straight-forward click-DCG utility of (4).

2.1.3 Combinations of basic utilities

The decision theoretic framework allows for a principled trade-off between desired behavior of the searcher and relevance cues from a selected set of human judges. In general the utility function should depend on both. A straightforward scheme is to take a weighted combination of the basic utility functions presented in Section 2.1.1 and 2.1.2.

2.2 Properties of the basic click-label utility

In the experiments in Section 3 we will use a utility function as sketched in Section 2.1.3:

$$U(y) = (1 - \lambda)U_{\text{DCG}}(y) + \lambda U_{\text{clicks}}(y) \,. \tag{6}$$

The parameter λ is still a design choice in this parametric form.

As argued in Section 2.1.2 the click part in the utility function (6) is only weakly motivated by the guiding principles, but it forms a good starting point since it captures already a few interesting characteristics from the two data streams.

- If there is noise in the labeled set, or if the model makes poor label predictions for a query, a suboptimal ordering can be corrected by clicks.
- If the model correctly predicts labels but there are ties, a top three of only good documents say, users effectively vote with their mouse which one they prefer.
- Since the framework consists of a model that *predicts* clicks based on features, an improvement in the ranking for popular queries also extends to unseen queries. For instance if Excel documents prove to be popular in a particular search context, they can be boosted for all queries in that context.
- Effectively the click based component in the utility will boost results that are predicted to be popular. If judges are instructed to label according to authority, the λ parameter allows us to trade-off popularity and authority. For instance in experiments with web-search data we found for instance that for the query "adobe" the url `www.adobe.com` is predicted to get the highest label, but the acrobat reader download page is the most popular. One could argue that the ideal result list has `www.adobe.com` at the top position and the link to the reader as the second link. This was the list returned in our experiments with $\lambda = 0.5$.

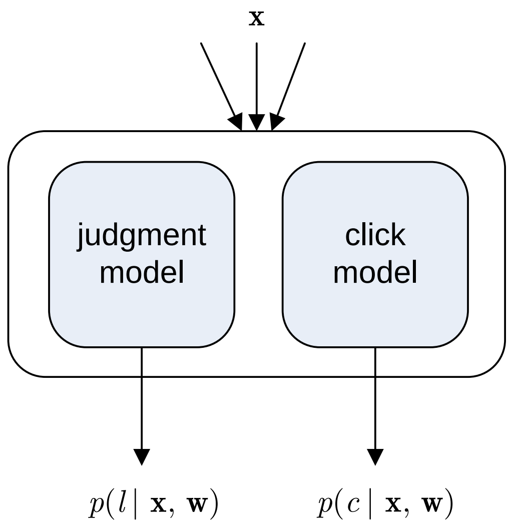
x
judgment model
click model
$p(l \mid \mathbf{x}, \mathbf{w})$
$p(c \mid \mathbf{x}, \mathbf{w})$

Figure 2: The model implemented in this paper sets out to predict two things: namely the probability of a click event $p(y_c|\mathbf{x}, \mathbf{w})$ and the probability of a particular relevance judgment $p(y_j|\mathbf{x}, \mathbf{w})$. The GLM model implies that the two sub-models factorize, and thus can be learned independently.

- By having the position of a document as part of the inputs x and fitting appropriate weights in the model, a position bias is automatically accounted for.
- If x contains characteristics of the user, the ranker automatically gives a personalized result.
- If the model is sophisticated enough such that it captures the interaction between documents, e.g. predicting that the probability of being clicked for near duplicate documents is anti-correlated, the ranker will, with a click-utility component from (5) automatically diversify the result list.

3. ON-SITE ADAPTATION OF INTRANET SEARCH SYSTEMS

An interesting application of the decision theoretic framework is in enterprise search. When a search system is installed out-of-the-box its ranker is based on a generic training set. Since intranets and their user bases can be quite diverse, it makes sense to use implicit feedback to adapt the ranker to the specific site for which it will be used.

It is generally difficult to obtain judged queries complete with clickthrough data from external organizations. Hence for this work, we were obliged to test the adaptation framework using an artificial corpus split created from data obtained from the Intranet of a single large multinational software corporation.

To reflect a significant change from the train set to the adaptation set, we created a split of our queries. For training, we use all queries and documents concerning the general areas of administration and marketing. For the adaptation set, simulating a potentially very different Intranet site, we use queries and documents of a technical nature.

The admin/marketing dataset used to train the out-of-the-box model consists of 546 queries. For each query, about 100 documents from the top of a ranked list from a baseline ranker were judged, and some of them had click information. The click-prediction part of the model is further trained using the adaptation query set, consisting of 201 technical queries. This simulates the on-site adaptation of the system to the user's clicks in the enterprise. In this case, the explicit judgments are not used for adapting the model, but instead used for evaluation only.

The click data we use is noteworthy in the following sense. We record not only the clicked documents, but also the documents that are skipped, or passed over, on the way to a click. In this work, inspired by [7], we assume a sequential scan of the result list, and as a consequence, that any document that is above the last click on the list is *examined.* In this way, we can aggregate the number of clicks and the number of examinations for a given query-document pair: a document which is clicked each time it is examined is intuitively good, and a document that is rarely clicked *having been examined* is probably a poor result. Importantly, we cannot infer much about the relevance of documents that have few examinations. This can happen if a result is either low in the ranking, or near the top yet just below a very good result.

3.1 A Bayesian generalized linear model

In this first illustration we use a generalized linear model (GLM) [9] for $p(y|\mathbf{x}, a)$. A GLM consists of a likelihood model $p(y|\theta)$, a linear combination of inputs $\mathbf{x}$ and model weights $\mathbf{w}$: $\mathbf{x}^\top\mathbf{w}$, and a link function $g(\theta)$ that maps the parameter θ to the real line. In this section we will use building blocks that have a binomial likelihood model and a probit link function. In a generative model interpretation the *inverse* probit link function

$$g^{-1}(s) = \Phi\left(s; 0, \frac{1}{\pi}\right)$$

plays a central role. This inverse link function is the well known cumulative normal function that maps the outcome of the inner product $\mathbf{x}^\top\mathbf{w} \in \mathbb{R}$ to the $[0, 1]$ space of the success parameter θ in the binomial. The inverse precision π can be set to an arbitrary number to fix the scale of the s-space. Here we will put a Gamma prior on π and integrate it out to obtain a robust model. If we have N examples in our training set for which the inputs have value $\mathbf{x}$, and we observe c positive outcomes, the likelihood becomes:

$$p(c|\mathbf{x}, \mathbf{w}) = Bin\left(c; g^{-1}\left(\mathbf{x}^\top\mathbf{w}\right), N\right) . \tag{7}$$

In Figure 2 we show a more detailed version of Figure 1b, where we are more explicit about what we set out to predict with the model. In this initial implementation, the output y in the model describes for each position $p = 1, \ldots, n$ a single implicit feedback: the click event y_c, and a single explicit feedback: the relevance judgment y_j.

Figure 4 shows the ordinal regression submodel $p(l|x, w)$ which is a generalization of the click model. Instead of one of two outcomes it has one of five possible label values it can output. Along with the other weights we therefore also learn four boundary values $b_1, \ldots, b_4$ that mark the edges in s-space of the five categories. Each has a Gaussian prior. The IsBetween factor in the figure represents two stepfunctions that bound the interval for label l. Added to the sum

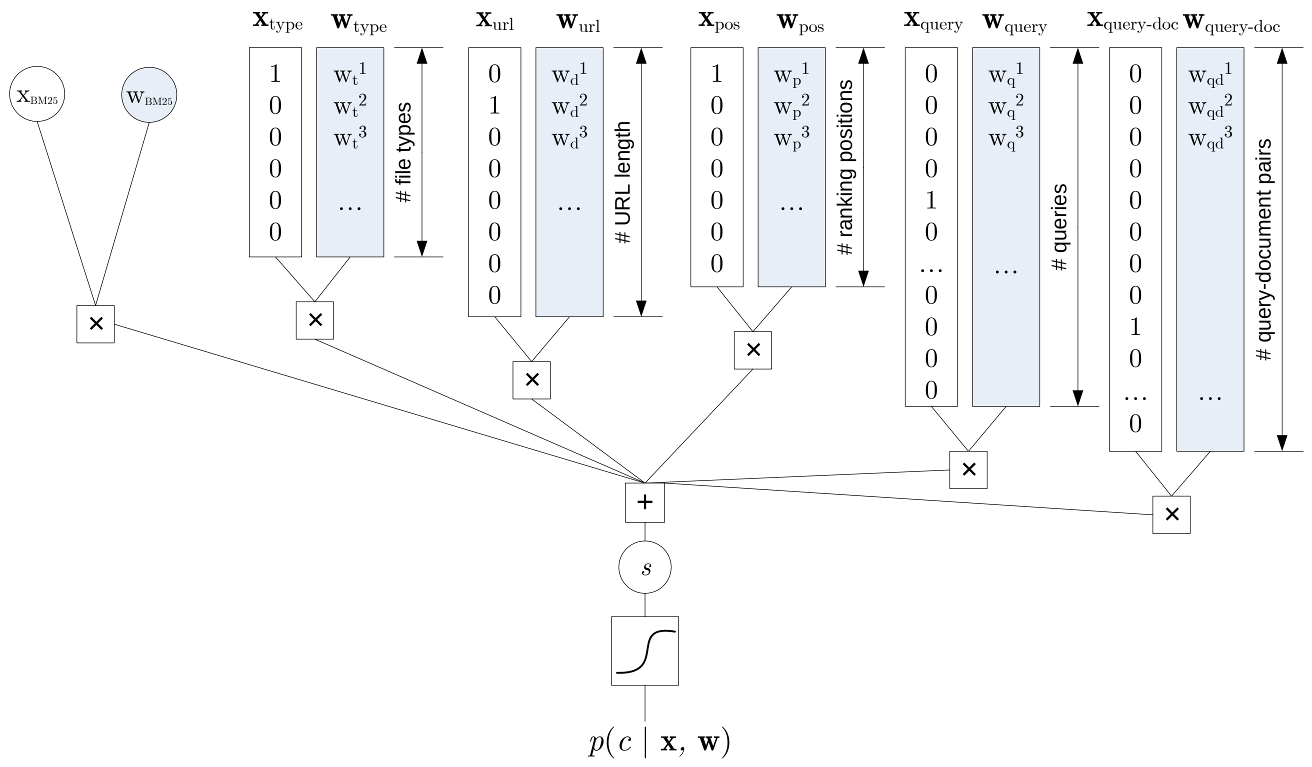
$\mathbf{x}_{\text{type}}$
$\mathbf{w}_{\text{type}}$
$\mathbf{x}_{\text{url}}$
$\mathbf{w}_{\text{url}}$
$\mathbf{x}_{\text{pos}}$
$\mathbf{w}_{\text{pos}}$
$\mathbf{x}_{\text{query}}$
$\mathbf{w}_{\text{query}}$
$\mathbf{x}_{\text{query-doc}}$
$\mathbf{w}_{\text{query-doc}}$
x_{BM25}
w_{BM25}
# file types
# URL length
# ranking positions
# queries
# query-document pairs
s
$p(c \mid \mathbf{x}, \mathbf{w})$

Figure 3: Indicator binary features and the GLM. This is a specific example of the click model shown on the right in Figure 2. Here the inputs x are made explicit as one real-valued feature (BM25) and five bags of binary features. The output is the predicted probability of click.

s is a Gaussian disturbance with inverse precision π. This disturbance can be interpreted as a softening of the step-function such that some noise in the label is supported by the model. It is the direct analog of the choice of probit link function instead of a hard step function in the discrete click prediction case.

3.2 Features

Figure 3 takes a more detailed look at the inputs (features) used for just the click model GLM. The input $\mathbf{x}$ contains parts that are query specific, parts that are document specific, and parts that are derived from the query-document pair. A BM25F score [11] is used as a general input that indicates the match between query and document.

Document specific features include the document file type (e.g. Html, Pdf, Excel etc) and the length of the url. Apart from these basic descriptive features, the vector $\mathbf{x}$ includes binary indicators for the query ID and the query-document ID, and also the rank (position) of the document in the list for which the click event was observed or is to be predicted.

The descriptive features give the model the ability to generalize between queries and documents, and the identifier (ID) weights effectively serve as an instance-specific memory. For frequently seen documents for popular queries the model can store, using the identifier weights, very accurate click predictions, even if they are far from the general trend predicted by the descriptive features. A bias term that is always 1 is included to capture a grand average.

3.3 Training

To learn the distributions of $\mathbf{w}$ we use the approximate Bayesian inference procedure from [15] with a factorized Gaussian prior. The ordinal regression part is treated as in [3] with the difference that here we do not resort to an ML-II approximation of π but integrate it out.

The main benefit of the Bayesian procedure is that with each individual weight in $\mathbf{w}$ a notion of the uncertainty about its optimal value is maintained. This results in a learning algorithm that correctly updates imprecisely determined weights more than precisely determined ones, which is essential for our model. The weights for descriptive features effectively see a lot more data than the query and document specific identifier weights. The Bayesian update rules ensure that each get updated at the right rate — in particular, a small number of examinations will not change the weight distributions nearly as much as a large number. This is something that could not easily be handled in for example maximum likelihood approaches.

3.4 Results

Before any implicit feedback data is available the ranker is based on a model that predicts clicks and labels. The utility we used in the experiments is a simple weighted combination between DCG and click-DCG as given in Equation 6. The specific setting of λ in this utility is a design choice. The dotted line in Figure 5 shows, for the out-of-the-box model, the NDCG@10 on the adaptation set as a function of λ. We see that using the click utility ($\lambda = 1$) actually reduces the NDCG@10 score. This is to be expected, since the NDCG score does not depend on observed clicks. The utility in (6)

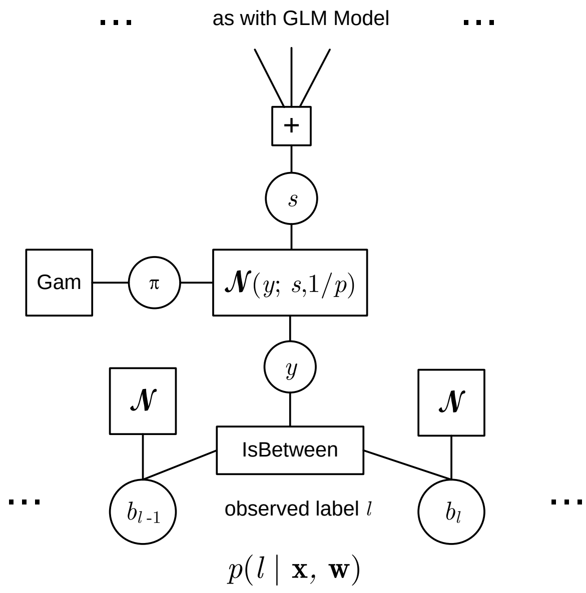

Figure 4: This is a specific example of the label model shown on the left in Figure 2. The inputs are the same as for the click model. However the output is handled differently. First noise is added to the variable s; the result is then constrained to lie between the two threshold variables which correspond to the observed label. Thresholds and noise precision are learnt in addition to the weights.

with $\lambda = 0$ is equivalent to DCG, and setting λ to another value encourages the ranker to optimize a different metric than NDCG@10 shown on the y-axis in Figure 5.

If we use two months of adaptation data, i.e. the site specific click feedback, we get the analogous solid/crossed curve in Figure 5. Here we see that incorporating clicks leads to an improvement of NDCG@10. A value for λ other than 0 and 1 leads effectively to the combination of the two datasets (the train set and the adaptation set). This improves performance, even if we measure the performance of the resulting system with NDCG, an evaluation metric that does not reward clicks.

The lower dashed line in Figure 5 represents a BM25F baseline, with no click data. We note that it is a horizontal line since ranking by BM25F does not involve a λ parameter. We see a 1 point NDCG@10 gain from the features alone ($\lambda = 0$) and an additional 2 points gain if we set $\lambda = 0.5$).

3.4.1 A proxy for a click-metric

The NDCG metric reported in Figure 5 is a well-known metric, but if $\lambda \neq 0$ it is strictly speaking not the metric that the ranker seeks to optimize. If it is decided that (6) with a particular value for λ is the utility that represents end user satisfaction the best, then *that* utility should be the final evaluation metric. Ideally we would like to test a ranker in an on-line setting where it can control the results lists. In such a setting we could monitor the accumulated utility by the ranker. However, since we only have historic

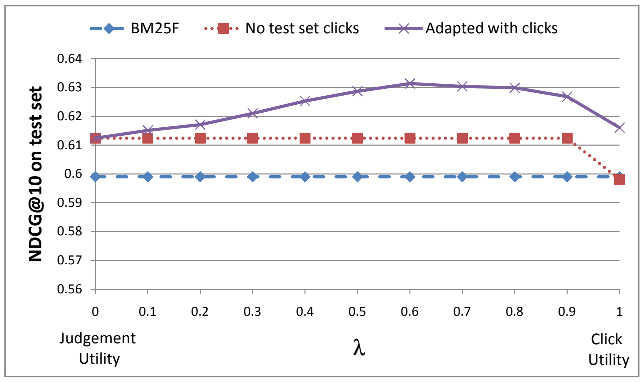

Figure 5: NDCG@10 scores for the different rankers as function of λ, the relative weight given to the click-part in a combined utility function.

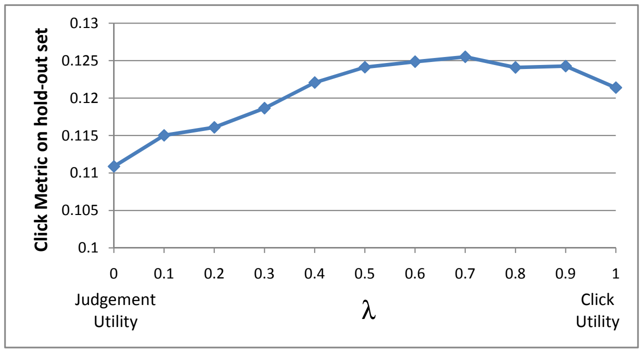

Figure 6: The click based scores from Equation (8) for the different rankers as function of λ.

data available we use the following proxy click metric:

$$S_{clicks} = \frac{\sum_{p=1}^{n} d(p) N_c(p)}{\sum_{p=1}^{n} N_e(p)} \tag{8}$$

where we denote the total number of clicks for the document on position p with $N_c(p)$ and the number of examinations with $N_e(p)$. The numerator in (8) uses the same discount function $d(p)$ as used in (3). Extra in this proxy evaluation metric is the normalization represented by the denominator. This ensures that documents that were not shown to users in the dataset (and hence have 0 clicks and 0 examinations) are properly disregarded. The score above is for a single query, and the total score would be the average over all queries.

Figure 6 shows a plot analogous to Figure 5, but now with the click-based evaluation metric from (8). We note that this new metric gets better with increasing λ. This is to be expected: a ranking formed from a utility based predominantly upon predicted click rate should do better when evaluated with a click-based metric. This provides further orthogonal evidence that combining implicit and explicit feedback improves search results.

To get a feel for the qualitative changes that the different choices of utility function imply it is instructive to look at

Relevance Judgments Utility (DCG)
1.: `http://vsts`
2.: `http://develop/vs2005field`
3.: `http://msdnprod/vstudio`
Click Utility
1.: `http://devdiv`
2.: `http://msdnprod/vstudio`
3.: `http://infoweb/c16/visualstudiodotnet`
Mixed Utility ($\lambda = 0.5$)
1.: `http://msdnprod/vstudio`
2.: `http://vsts`
3.: `http://devdiv`

Table 1: Reorderings of the top-ranked positions for the "Visual Studio" query

a specific example. Table 1 shows the top three results for the query "Visual studio" for $\lambda = 0$ (DCG ranking), $\lambda = 1$ (click ranking) and $\lambda = 0.5$ (balanced ranking). In this example the DCG based top three are all documents that could claim to be a definitive result for searchers interested in using the Visual Studio product. They were all labeled "good" by human assessors. If the ranker is using the click-only utility ($\lambda = 1$) we see that the top three changes. Of the three "good" results in the DCG list, the `msdnprod` link and snippet is apparently the most appealing to the users in the adaptation phase, containing technical information. The other two documents that have entered the top three reflect different interpretations of the query "Visual studio": the devdiv page gives information about the team that creates Visual studio, and the infoweb provides marketing data.

This example demonstrates that there is no unique definition of relevance. If we deem the most popular page to be the most relevant, we should pick the click utility. However, if we want the result list to be more authoritative, a utility based upon explicit judgments might promote pages that are more likely to have been overlooked in a straight snippet-based popularity contest. This advantage of an increased reliability of explicit judgment usually comes with the disadvantage of a single user interpretation of relevance: there is a natural tradeoff between judgment accuracy and result diversity. As Table 1 shows a mixed utility allows us to find a balance between these two extremes.

Including click feedback has had two qualitative effects for the Visual Studio query: (i) a rearrangement of, from the external perspective equally good, documents according to local preferences, and (ii) a promotion of alternative interpretations of the query that are common at the specific intranet. Although we present a single example here, we have seen these effects in many other queries, together with a third major effect: (iii) the correction of erroneous human judgments.

4. SUMMARY

In this paper we have explored the decision theoretic framework for IR and studied how it can be used to combine several sources of feedback into a single ranker. The approach is based on a utility function that describebes the user satisfaction after a search session, and a model that predicts user actions such as clicking and labelling based on known quantities at query time. Constructing a model and formulating a utility function are both difficult problems. But we observe that in the case of label and click stream data the simplest possible utility function and a reasonable prediction model already give many of the potential benefits that the combination of the two streams can bring.

In experiments in an enterprise search setting, we see that the approach leads to increased performance. Qualitatively we see that mislabelled queries get filtered out, result lists for ambiguous queries change to better reflect the most often intended interpretation by users, and tie breaking of identically labelled results is done according to population preference. In terms of NDCG@10 including the click stream in the decision theoretic ranker leads to a two point gain. This is despite the fact that the ranker does not aim to optimize this metric. Given the qualitative results we expect that end user experience improves even more than the two point NDCG gain indicates.

5. REFERENCES

[1] E. Agichtein, E. Brill, and S. Dumais. Improving web search ranking by incorporating user behavior information. In *SIGIR*, 2006.

[2] C. Burges, R. Ragno, and Q. V. L. Le. Learning to rank with nonsmooth cost functions. In *NIPS*, 2006.

[3] W. Chu and Z. Ghahramani. Gaussian processes for ordinal regression. *JMLR*, 6:1019–1041, 2005.

[4] I. J. Cox, M. L. Miller, T. P. Minka, T. Papathomas, and P. N. Yianilos. The Bayesian image retrieval system, pichunter: Theory, implementation and psychophysical experiments. *IEEE Transactions on Image Processing, Special Issue on Image and Video Processing for Digital Libraries*, 9(1):20–37, 2000.

[5] S. Fox, K. Karnawat, M. Mydland, S. T. Dumais, and T. White. Evaluating implicit measures to improve the search experience. *ACM Transactions on Information Systems*, 23:147–168, 2005.

[6] Järvelin and J. Kekäläinen. IR evaluation methods for retrieving highly relevant documents. In *SIGIR*, 2000.

[7] T. Joachims. Optimizing search engines using clickthrough data. In *Proceedings of Knowledge Discovery in Databases*, 2002.

[8] J. Lafferty and C. Zhai. Document language models, query models, and risk minimization for information retrieval. In *SIGIR*, pages 111–119, 2001.

[9] P. McCullagh and J. A. Nelder. *Generalized Linear Models.* CRC Press, 2nd edition, 1990.

[10] S. Robertson, H. Zaragoza, and M. Taylor. A simple BM 25 extension to multiple weighted fields. In *CIKM*, pages 42–29, 2004.

[11] S. E. Robertson. The probability ranking principle in IR. *Journal of Documentation*, 33:294–304, 1977.

[12] M. Taylor, J. Guiver, S. Robertson, and T. Minka. SoftRank: optimizing non-smooth rank metrics. In *WSDM '08*, pages 77–86. ACM, 2008.

[13] S. K. M. Wong, P. Bollmann, and Y. Y. Yao. Information retrieval based on axiomatic decision theory. *General Systems, 19(2)*, 23(2):101–117, 1991.

[14] O. Zoeter. Bayesian generalized linear models in a terabyte world. In *IEEE Conference on Image and Signal Processing and Analysis*, 2007.