Item Retrieval as Utility Estimation

Shawn R. Wolfe
NASA Ames Research Center
Moffett Field, California
Shawn.R.Wolfe@nasa.gov

Yi Zhang
University of California, Santa Cruz
Santa Cruz, California
yiz@soe.ucsc.edu

ABSTRACT

Retrieval systems have greatly improved over the last half century, estimating relevance to a latent user need in a wide variety of areas. One common task in e-commerce and science that has not enjoyed such advancements is searching through a catalog of items. Finding a desirable item in such a catalog requires that the user specify desirable item properties, specifically desirable attribute values. Existing item retrieval systems assume the user can formulate a good Boolean or SQL-style query to retrieve items, as one would do with a database, but this is often challenging, particularly given multiple numeric attributes. Such systems avoid inferring query intent, instead requiring the user to precisely specify what matches the query. A contrasting approach would be to estimate how well items match the user's latent desires and return items ranked by this estimation. Towards this end, we present a retrieval model inspired by multi-criteria decision making theory, concentrating on numeric attributes. In two user studies (choosing airline tickets and meal plans) using Amazon Mechanical Turk, we evaluate our novel approach against the *de facto* standard of Boolean retrieval and several models proposed in the literature. We use a novel competitive game to motivate test subjects and compare methods based on the results of the subjects' initial query and their success in the game. In our experiments, our new method significantly outperformed the others, whereas the Boolean approaches had the worst performance.

CCS CONCEPTS

• **Information systems** → *Retrieval models and ranking*; Similarity measures; Learning to rank;

KEYWORDS

structured data, item retrieval, vertical search, multi-criteria decision making, utility function

ACM Reference Format:
Shawn R. Wolfe and Yi Zhang. 2018. Item Retrieval as Utility Estimation. In *SIGIR '18: The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, July 8–12, 2018, Ann Arbor, MI, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3209978.3210053

SIGIR '18, July 8–12, 2018, Ann Arbor, MI, USA

ACM ISBN 978-1-4503-5657-2/18/07…$15.00
https://doi.org/10.1145/3209978.3210053

1 INTRODUCTION

Searching for items by their attribute values or metadata is a commonplace task today. For example, consider searching for a particular research paper you recall as published in 2013, or computer monitors under $200. Current search tools support a retrieval style more akin to a database than a modern information retrieval system, most often with faceted or Boolean search. In that model, users place hard constraints on acceptable attribute values to limit the result set.

Unfortunately, this rigid style of retrieval often creates difficulties for the user. In the examples above, what if the sought research paper was published in 2012, not 2013 as remembered? There are thousands of computer monitors under $200, far too many to examine, but constraining all desired attributes might yield no results. And what if a monitor that best fits the user's desire is just outside the stated range, listing at $213? Boolean retrieval often yields no results or too many. Faceted search usually avoids empty result sets, but the facets are often pre-computed and may not match the user's intent well.

The problem with such retrieval systems is that they do not try to understand what the user is seeking. Instead, they give the user a tool to *explicitly* manage the result set by stating what to return and how to order it. In this paper, we explore an alternative approach that is more aligned with current information retrieval approaches – *implicitly* managing result sets by estimating relevance to a description of the sought item. We model the user's utility function, and in the process, allow the system to trade off among conflicting criteria on the user's behalf, and in this way get closer to the underlying query intent. In contrast to the Boolean and faceted approaches in use today, our approach does not use constraints.

We evaluate this new approach against the *de facto* standard of Boolean retrieval and several models proposed in the literature in two user studies using Amazon Mechanical Turk[1]. The domains and tasks in these user studies are diverse, with one involving searching for airline tickets and in the other for healthy (daily) meal plans. In both studies, test subjects read a short scenario and used a randomly chosen retrieval model to find an appropriate item (ticket or meal plan). We ask the following questions:

RQ1 Which approach and specific retrieval algorithm is most effective?
RQ2 Are constraints beneficial or harmful?
RQ3 How should results be ordered?

In this work, we make the following contributions:

(1) We cast the item retrieval problem as finding the item with the most desirable combination of attribute values, and use utility theory to develop a basic model.

[1] http://www.mturk.com

(2) We expand upon the basic retrieval model by developing subutility models that estimate the utility of every possible value of each attribute.
(3) We develop a Bayesian hierarchical model around our item utility model so the models' parameters can be fit to data of users' selections.
(4) We evaluate our novel item retrieval model against the two common approaches, Boolean and faceted retrieval, along with several models from the literature, in two user studies in very different domains.

The rest of this paper is organized as follows. In section 2, we review the related research and introduce the previously published methods included in our study. In Section 3, we introduce an initial retrieval method based on multi-criteria decision-making theory and a subsequent enhancement to that method. The learning framework we later use to tune this model's parameters is described in Section 4. In Section 5, we detail the design of our two user studies, including the baseline retrieval methods that we include for comparison. Finally, we present the results of these user studies in Section 6 and present our conclusions in Section 7.

2 RELATED WORK

Common item retrieval methods use a Boolean retrieval paradigm, either in database-like query or faceted search, with the latter a popular choice with many e-commerce websites. Database researchers have expanded on these approaches while preserving clear retrieval semantics, notably with top-K approaches [16], which retrieve the k-highest scored items given a scoring formula, and ranking given uncertain data [29]. Skyline queries [15] do not use a specific scoring formula, instead returning the Pareto set given desired characteristics. Finally, several researchers have explored incorporating preferences into database queries [1, 11, 17, 18]. The focus of that work has been the semantics of the operators and on efficient execution, and not inferring latent preferences. Overall, the important body of work referenced above is focused on a different problem than ours, namely efficiency and defining explicit retrieval semantics, not query intent. In contrast, we do not assume a scoring function or explicit retrieval paradigm, and instead attempt to maximize user satisfaction by estimating item relevance.

The few item retrieval methods that do rank results according to estimated relevance tend to use methods suited for categorical data on all attributes, even numeric ones, perhaps because of the similarity to the bag-of-words model of information retrieval. Chauduri et al. [10] and Su et al. [30] adapted the binary independence model, discretizing numeric attribute values, similar to faceted search. Agrawal et al. [2] adapted TF*IDF to search database records, but abandoned the term frequency term. AIMQ [22] further advanced the numerical relevance concept through a "like" operator that calculated the bounded absolute percentage difference between query and data attributes, combining them in a linear combination. Agrawal et al.'s method and AIMQ were combined and slightly modified by Meng et al. [20]. CQAds use a normalized absolute difference to compare numerical query and data attributes, combined in a simple summation, to find advertisements (or more precisely, search through "for sale" listings). Finally, the appropriately named VAGUE system [21] was an early retrieval framework that incorporated a "similar-to" operator that would retrieve records close to the desired attribute values, using the system designer's chosen metric function. Vague queries were later incorporated into a probabilistic framework [13], although how to estimate these probabilities was left as a difficult open question. We include Agrawal et al.'s, model, AIMQ, CQAds and VAGUE as baselines in our experiment and give their mathematical formulations in Sec. 5.2.

The healthy meal plan user study can be seen as a package retrieval task (retrieving a composite item instead of individual items), though this is not a central aspect of that study. Prior research has focused on recommending packages that meet the user's constraints and maximizing a provided objective function. Package recommendation has been explored in a number of areas, such as trip planning [4, 14, 31, 33], student course planning [23–25], compatible products [5], diversity in restaurants [3] and web page conglomeration [7]. Given the large number of potential packages, recommended packages are typically generated on the fly, typically an NP-complete problem. Our meal plan retrieval study contrasts with these by selecting from a fixed (though large) corpus of packages and eschewing constraints and objective functions for estimated utility.

We may be the first to apply multi-criteria decision concepts specifically to item retrieval, but others have adapted it to general information retrieval, primarily in information filtering. Manouselis and Costopoulou categorize 37 recommender systems that implicitly use some multi-criteria aspect in their operation [19]. PENG [6, 26] is a multi-criteria news bulletin filtering system that utilizes several criteria, including content, coverage, reliability, novelty and timeliness.

3 TOWARDS A MODEL OF UTILITY

In contrast with the retrieval methods we surveyed in the literature, we develop a retrieval method with clear theoretical justification, building on multi-criteria decision making. In multi-criteria decision making, a decision maker must choose among several candidates, with each candidate evaluated by the decision maker on the same set of criteria. A simple example is choosing a hotel, factoring in price, location, amenities, etc. The decision making process is made difficult by *conflicts* among the criteria, that is, when individual criteria rank options differently, and in particular, when no candidate is rated highest by all the criteria. Multi-criteria decision making approaches typically rank candidates given a rating on each attribute value; our problem is even harder as we must also estimate subutility functions to rate the attribute values. The ratings on each criterion are typically scaled to lie on a 0-1 scale (with higher ratings preferable). Ratings on different criteria are not assumed to be of equal importance, so a weight for each criterion is usually assigned to capture relative importance, with the sum of weights equal to 1.

According to multi-attribute utility theory (MAUT) [12], certain assumptions on the properties of preferences entails that the underlying utility function follow a particular form. We assume *mutual utility independence*, which means for any subset of attributes, the strength of preference for a set of values is unaffected by the values

of other attributes. As an example, this would mean that the difference in utility between two 20 inch computer monitors, priced \$150 and \$200 but otherwise identical, is the same as the difference in utility between two 25 inch computer monitors, priced \$150 and \$200 but otherwise identical. This assumption entails that the underlying utility function must be linear combination of ratings, yielding our base utility function:

$$f(\mathbf{Q}, \mathbf{D}_i) = \sum_j w_j \cdot g_j(q_j, d_{ij}) \tag{1}$$

where j is the index of the j^{th} attribute, w_j is the priority (weight) given to the attribute, $\mathbf{Q}$ are the desired attribute values, and $\mathbf{D}_i$ is the i^{th} item in corpus $\mathbf{D}$, with q_j and d_{ij} the values of the j^{th} attribute of $\mathbf{Q}$ and $\mathbf{D}_i$, respectively. $g_j(q_j, d_{ij})$ is the subutility function which evaluates the utility of attribute value d_{ij} when the user desired q_j. Items are ranked in order of decreasing utility.

At this point, MAUT offers no further guidance. Subutility evaluations are taken as input to the MAUT problem, but we need to estimate subutilities. Estimation is trivial for Boolean attributes, as there are only two possible attribute values, so the subutility is one when $q_j = d_{ij}$, zero otherwise. Categorical attributes (e.g., color) are more challenging. An extreme solution would be to use the same approach as Boolean attributes, estimating zero subutility except when $q_j = d_{ij}$. A more nuanced approach could be derived from domain theory or user choice training data when either is available.

Numeric attributes, on the other hand, have mathematical relationships among their values which suggest other avenues for subutility estimation. A simple yet intuitive method is to relate subutility to the absolute difference from the desired value, which we chose as follows:

$$g(q_j, d_{ij}) = \left(1 - \frac{|q_j - d_{ij}|}{\max\left(|q_j - \perp_j|, |q_j - \top_j|\right)}\right) \tag{2}$$

where $\perp_j$ and $\top_j$ are the least and greatest values of the j^{th} attribute in the corpus, and other variables are as defined in Eq. 1. This formulation gives us our initial retrieval model, *SimpleMAUT*. *SimpleMAUT* accepts as input a query consisting of desired attribute values (0 or 1 per attribute) and attribute priorities (0 when the corresponding attribute is not of interest) and returns items ranked by their estimated utility. *SimpleMAUT* (and the forthcoming MAUT models) could also be extended to support ranges or multiple desired values by giving such maximum utility.

However, the subutility estimation of numeric attributes in *SimpleMAUT* has several limitations. First, the attribute ratings are normalized by the extreme attribute values of the corpus, and so can be radically affected by corpus changes. Second, it assumes a linear relationship between the attribute subutility and the attribute value, implying a constant rate of subutility change. This has nonintuitive consequences, for instance, it implies that a \$5 discount is just a compelling when applied to \$1000 item as it is for a \$10 item. Finally, as is, *SimpleMAUT* does not have way to incorporate the subutilities of a multiply-valued attribute, which we needed for the ratings of multiple dishes in our meal plan user study.

We made several changes in an enhanced version of our model, normalizing numeric attribute subutilities with the standard deviation and including a scaling factor for each subutility. We also developed a more flexible subutility function based on several principles. First, the desired value should have maximal subutility. Second, subutility should never increase as the absolute difference to the desired value increases. Finally, the subutility function should be as flexible as possible with a minimum number of parameters. Accordingly, we used an exponential function, raised to a positive exponent, as our subutility function. It can capture a variety of functions, from a point-like subutility, to gradually diminishing losses, to a bell-shape curve, and even to a boxcar function in the limit. The enhanced model has separate subutility function parameter values above and below the desired value, so that asymmetric subutilities can be modeled.

We can now present the revised numeric subutility function used by our enhanced retrieval algorithm, *EnhancedMAUT*:

$$\begin{aligned} g_j(q_j, d_{ij}) = \left[q_j \geq d_{ij}\right] \exp\left(-\left(\frac{|d_{ij} - q_j|}{\phi_j^{\geq}\sigma_j}\right)^{\rho_j^{\geq}}\right) \\ + \left[d_{ij} < q_j\right] \exp\left(-\left(\frac{|d_{ij} - q_j|}{\phi_j^{<}\sigma_j}\right)^{\rho_j^{<}}\right) + \left[d_{ij} = q_j\right] \end{aligned} \tag{3}$$

where σ_j is the standard deviation of j^{th} attribute, [ ] is the Iverson bracket, $\boldsymbol{\rho}^{\geq}$, $\boldsymbol{\rho}^{<}$, $\boldsymbol{\phi}^{\geq}$, $\boldsymbol{\phi}^{<}$, and $\mathbf{w}$ are model parameters (all 1 in our experiment), and others are defined as above.

Finally, we chose to aggregate multiply valued subutilites with a generalized mean, which only applied to the rating of the component dishes in the meal plan user study. The generalized mean takes a single parameter ψ and its argument, a series of numbers $x_1, ..., x_n$:

$$M(x_1, ..., x_n) = \left[\frac{1}{n}\sum_{i=1}^{n} x_i^{\psi}\right]^{\frac{1}{\psi}} \tag{4}$$

The generalized mean's appeal comes from its flexibility, as particular values of ψ will produce the arithmetic, geometric, and harmonic means, as well as minimum and maximum. Thus, this one function allows us to model several reasonable ways a user might evaluate a set of items. In our case, each x_i is the estimated subutility of the rating of a dish in a meal plan.

4 LEARNING

The *LearnedMAUT* model (Figure 1) has the same formulation as *EnhancedMAUT*, but uses tuned model parameter values for the attribute weights and shapes of the subutility functions, as described below. These are learned in a pairwise learning to rank framework with Bayesian logistic regression, by placing a logistic function in a hierarchical model. Given the utility function $f()$ in Eq. 1, using the subutility function $g()$ in Eq. 3, and the general mean (for dish ratings only) in Eq. 4, the likelihood function $L()$ is:

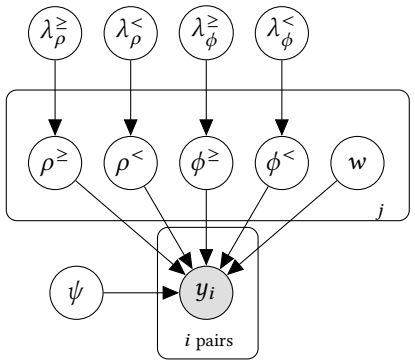

Figure 1: Hierarchical Bayesian model of *LearnedMAUT*

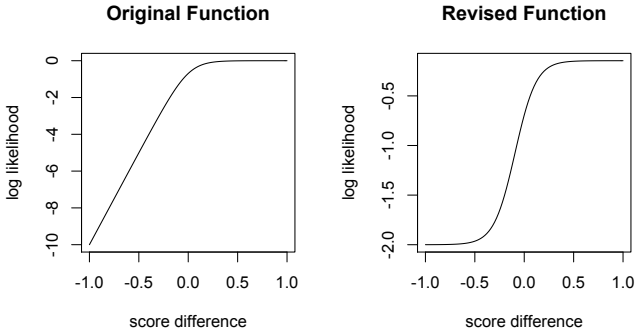

Figure 2: Log Likelihood

$$L(\boldsymbol{\rho}^\geq, \boldsymbol{\rho}^<, \boldsymbol{\phi}^\geq, \boldsymbol{\phi}^<, \mathbf{w}, \psi; \mathcal{Q}, \mathbf{D}, \mathcal{R}, \mathcal{U}) = \prod_{\mathbf{Q}\in\mathcal{Q}} \prod_{r\in\mathcal{R}} \prod_{u\in\mathcal{U}} \left(\frac{b}{2} + \frac{1-b}{1+\exp(-c(f(\mathbf{Q},\mathbf{D}_r) - f(\mathbf{Q},\mathbf{D}_u)))} \right) \quad (5)$$

where $\mathcal{Q}$ are the set of queries, $\mathcal{R}$ are the item indices chosen for query $\mathbf{Q}$, $\mathcal{U}$ are the indices of items not chosen for query $\mathbf{Q}$, b and c are tuning parameters, with others defined above. Parameter b (arbitrarily set to e^{-2} in our experiment, and discussed below) limits the maximum loss from any pair, and c (10 in our experiment) affects gradient smoothness, with results insensitive to small changes in either parameter.

The model parameters $\boldsymbol{\rho}^\geq$, $\boldsymbol{\rho}^<$, $\boldsymbol{\phi}^\geq$, $\boldsymbol{\phi}^<$, $\mathbf{w}$ and ψ are given prior distributions, with ψ modeled as a standard normal distribution and the rest modeled with gamma distributions. The hyperpriors $\lambda_\rho^\geq$, $\lambda_\rho^<$, $\lambda_\phi^\geq$, and $\lambda_\phi^<$ are used to control the modes of $\boldsymbol{\rho}^\geq$, $\boldsymbol{\rho}^<$, $\boldsymbol{\phi}^\geq$, $\boldsymbol{\phi}^<$, and are modeled as a modified gamma distribution that corrects for a drift towards more compact distributions with smaller modes. These hyperpriors and $\mathbf{w}$ were given a mode of 1. The gamma distributions' parameters were calculated to fit the mode and give good regularization.

The tuning parameter b was included to limit model sensitivity to highly unlikely pairs. Initially this parameter was not included (equivalently given a value of zero), yielding a more conventional logistic function, but we found the probabilistic model would gravitate towards fits where most pairs were slightly unlikely yet resulting in a better overall probability than results with high classification accuracy but with a few very unlikely pairs. Our solution was to have our utility function only describe part of the data, modeling the data as a mixture of two processes, the other being a random selection model. This also admits uncertainty into the model; at times, a user may select a different item due to factors that are not captured by the model. Figure 2 compares how the log likelihood changes for a single pairwise comparison as their score difference changes, in the original and revised formulation; note also the difference in scale. Since the overall log likelihood is the sum of each pair's likelihood, it is easy to see that the revised likelihood corresponds much better with the overall classification accuracy. For our experiments, we arbitrarily set the mixing parameter b to e^{-2} (≈ 0.14), noting that results were insensitive to small changes in this parameter.

We used the Metropolis-Hastings algorithm to generate samples from the posterior distribution, using the observed modes as the model parameter values. After the user study, the initial queries and final selections from that study were separated into 20 folds (for cross-validation), training a separate model for each fold, using the other 19 folds for training data and the fold's data for testing. We partitioned the data into folds by scenario, to prevent selections from the test scenario biasing the model. However, given the difference in scenarios, queries and limited number of selections, the learning problem is fairly difficult. We evaluate the learned model in Section 6 using the mode of the resulting posterior distributions.

5 EXPERIMENT

We conducted two user studies using different domains and several retrieval models to compare the models' ranking of results. After the first user study, for our second study (with meal plans) we improved the experimental protocol (adding a head-to-head comparison), the baseline models (replacing *Tradeoff* with *Faceted*), and our retrieval model (using *EnhancedMAUT* instead of *SimpleMAUT*).

5.1 User Interfaces

We developed different user interfaces to support models with different query input (e.g., some supported ranges, some accepted sort orders, etc.). Some retrieval models had identical query input and differed only in the subsequent ranking.

Sorted Boolean For the ticketing user study, we allowed users to restrict the result set by attribute ranges (with the exception of price) and give a single sort order, as with popular travel sites at the time of development. In the meal plan user study, we allowed restriction on any attribute and up to four sort orders. An example of the query interface in the ticketing user study is shown in Figure 3.

Faceted For the meal plan user study, we created a basic faceted search model. All attributes were split into a small number of equally sized facets, with seven to twelve facets per attribute. Up to four sort orders could also be chosen. An example of the query interface is shown in Figure 4.

Point The point-based user interface allows a user to specify single values for each attribute, allowing the user to give

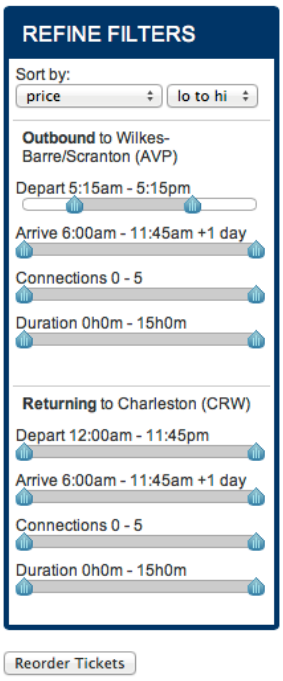

Figure 3: Sorted Boolean search interface, ticket study.

Figure 4: Faceted search interface, menu plan study.

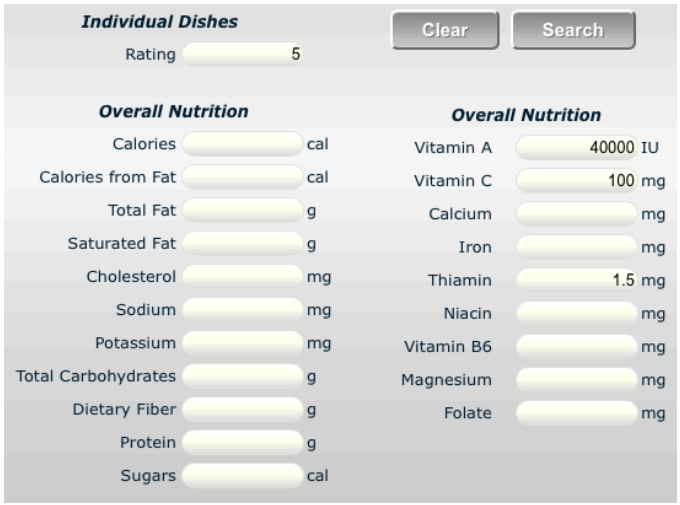

Figure 5: Point-based search interface, menu plan study.

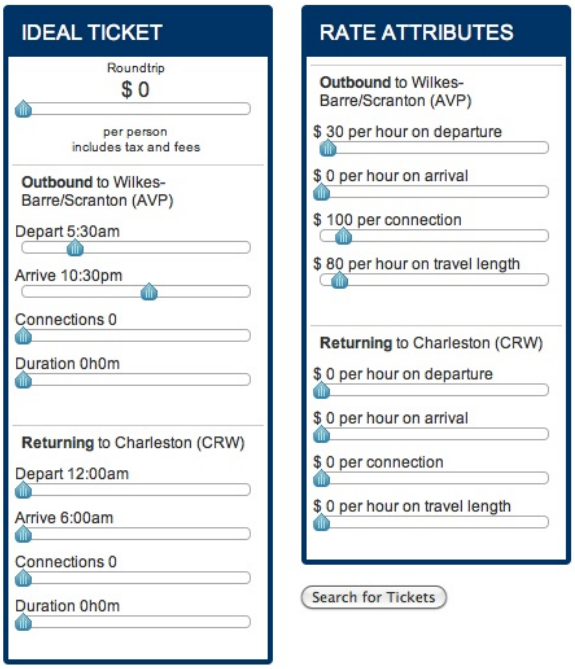

Figure 6: Utility function search interface, ticket study.

specific attribute values of interest. Partial specifications (attributes can be left blank) are acceptable, as with other interfaces. This interface was used for the MAUT-based models as well as baselines from the literature.

Tradeoff As our retrieval model was based on an *implicit* utility function, we created an alternative where users could provide a simple utility function, as shown in Figure 6. As with the *Point* interface above, users could specify desired attribute values, but also provide exchange rates for each unit change, for instance in Figure 6, each connection added to a flight is acceptable only if it saves $100 (or more). The interface was only used by the *Tradeoff* model in the ticketing user study, which is described in the following section.

5.2 Retrieval Models

We used our proposed models, the *de facto* item retrieval methods of Boolean and faceted search, as well as several models from the literature (see Sec. 2), as listed below. Variables are defined as in Section 3 unless otherwise specified.

AIMQ AIMQ estimates relevance using a different normalization and global weights derived from functional dependencies (see [22] for details). Items are ranked by decreasing score order, where the score of d_i is defined as:

$$f(\mathbf{Q}, \mathbf{D}_i) = \sum_{j \in \mathbf{Q}} w_j \left(1 - \min\left(1, \frac{|q_j - d_{ij}|}{q_j}\right)\right) \tag{6}$$

where $j \in \mathbf{Q}$ indicates the j^{th} attribute was given a desired value (not left blank), and w_j is the global weight for the j^{th} attribute.

AutoRank This is the (unnamed) model of Agrawal et al. They used an inverse document frequency (IDF) term for weighting, defined below for query element q_j as:

$$w_j = \log\left(\frac{n}{\sum_{k=1}^{n} \exp(-\frac{1}{2}\left(\frac{d_{kj} - q_j}{h_j}\right)^2)}\right) \tag{7}$$

where n is the number of items, and h_j is a "bandwidth" parameter, chosen by Agrawal as $h_j = 1.06\sigma_j n^{-\frac{1}{5}}$. This is combined in their overall scoring function:

$$f(\mathbf{Q}, \mathbf{D}_i) = \sum_{j \in \mathbf{Q}} w_j \exp\left[-\frac{1}{2}\left(\frac{d_{ij} - q_j}{h_j}\right)^2\right] \tag{8}$$

with items ranked by decreasing score.

CQAds CQAds estimates relevance much like AIMQ, but with a different normalization, and without attribute-specific weights. In our adaption of CQAds scoring, items are ranked by decreasing score order, where the score of d_i is

$$f(\mathbf{Q}, \mathbf{D}_i) = \sum_{j \in \mathbf{Q}} \left(1 - \frac{|q_j - d_{ij}|}{R_j}\right) \tag{9}$$

where R_j is an estimation of the range of the j^{th} attribute, defined as the mean of the ten greatest values minus the mean of the ten least values.

Faceted/Sorted Boolean Faceted search and Sorted Boolean search have the same retrieval semantics with different user interfaces. Items that meet the constraints given by the user are returned and ordered by any provided sort orders.

MAUTs *SimpleMAUT* and *EnhancedMAUT* were described in Section 3, whereas *LearnedMAUT* was described in Section 4. *SimpleMAUT* was used only in the ticketing user study and included user provided attribute priorities, whereas *EnhancedMAUT* was used in the meal plan study with uniform attribute weights. *LearnedMAUT* was trained and evaluated post-hoc on data from both user studies.

Tradeoff Contrasting with *SimpleMAUT*, the Tradeoff model allowed test subject to directly provide a utility function by giving an explicit tradeoff rate (in terms of dollars) they would be willing to spend to get closer to their desired attribute values. Items are ranked by increasing score order, where the score of $\mathbf{D}_i$ is defined as:

$$f(\mathbf{Q}, \mathbf{D}_i) = \sum_j t_j |q_j - d_{ij}| \tag{10}$$

where t_j is the tradeoff rate (in dollars) for the j^{th} attribute.

VAGUE The VAGUE framework provides "similar-to" operator that calculates a weighted Euclidean distance from the query point and the item. The operator can use subutility functions, but none are prescribed, so we choose the absolute difference divided by the standard deviation:

$$f(\mathbf{Q}, \mathbf{D}_i) = \sqrt{\sum_j \left[w_j \left(\frac{|q_j - d_{ij}|}{\sigma_j}\right)\right]^2} \tag{11}$$

with items ranked by increasing score. As with the MAUTs above, user provided attribute priorities were used in the ticketing study and replaced with uniform weights in the meal plan study.

Only *EnhancedMAUT* and *LearnedMAUT* were developed to support multiply-valued attributes (our meal plans have a separate rating for each included dish), so we use the (arithmetic) mean to aggregate such multiple values in the experiment, except where otherwise noted.

AIMQ, *CQAds*, the MAUT-based models (*EnhancedMAUT*, *LearnedMAUT*, *SimpleMAUT*), and *VAGUE* all accept the same query input, differing only in how they rank results. Thus, only *SimpleMAUT* and *EnhancedMAUT* were used during the user study, with the others evaluated post hoc using only the first query from each session, as subsequent queries are influenced by the search engine actually used. Additionally, *LearnedMAUT* was trained with data after user study completion instead of on-line.

5.3 Data Used

For both user studies, we wanted test subjects to perform realistic tasks, using appropriate real world data. We developed twenty short scenarios for each study based on the literature.

```
We're meeting with a potential new distributor for
four days in Omaha, starting Monday, January 9, and
ending Thursday, January 12. I need to leave Sunday,
January 8 to get there on time, and have to leave no
later than 2 PM on January 12. I'm giving a presentation
about the meeting to the directors in Burbank at 8 AM
Friday, January 13. It's business travel, so I won't be
paying for the ticket.
```

Figure 7: Example Scenario for Ticketing Study

For the ticketing study, we consulted a survey from more than 26,000 U.S. households to capture who travels by air and the reasons why [9], using this breakdown to develop 10 scenarios for pleasure, 8 for business, and 2 for personal business. To make the scenarios slightly more compelling, we created somewhat vague reasons for the trip (i.e., "attend a meeting", "visit relatives", "take a vacation"). We chose arbitrary dates to match the scenarios, with personal trips somewhat longer in duration and with random time constraints (from 9 AM to 4 PM) for business trips, ranging from nearly trivial to quite restrictive. For half of the remaining scenarios, we listed other criteria (such as "get home early"), while leaving the others open-ended. Figure 7 shows a scenario with constraints for business

travel. Finally, we also sampled demographic information from the same survey (gender, age, income) to give more context. To build the ticket corpus, we randomly selected origin/destinations pairs from a sample of U.S. domestic travel in 2006 [8] and retrieved tickets from Expedia (using dates of our choosing) with the same origin and destination, which at that time yielded approximately 60 round-trip tickets for each scenario, yielding a separate corpus for each scenario. Each ticket was described by nine attributes (e.g., price, outbound departure, etc., as in the ideal ticket in Figure 6). Dates and origin/destination were treated as unalterable hard constraints.

```
You are choosing meals for Emma, a 30 year old female.
Emma is concerned about her fat intake. She has read that
at most 30% of calories should come from fat, with at most
10% coming from saturated fat. Emma wants a daily meal
plan that follows the nutritional recommendations, with
an emphasis on delicious food, calories from fat, total
fat, and saturated fat.
```

Figure 8: Example Scenario for Meal Plan Study

For the meal plan study, we consulted a popular nutritional resource [27] which tabulated nutritional needs by age and gender, as well as modifications needed for various diseases and lifestyles. In addition to these specific recommendations, we also included a desired nutritional range in the form of Estimated Average Requirement and Tolerable Upper Limit [32] when such are defined. We developed twenty core scenarios, choosing a variety of conditions, genders, and ages. In addition, four meal plan attributes (tastiness and three randomly selected nutrients, typically overlapping with any nutritional modifications) were emphasized to focus the test subject. In all, 119 scenarios were generated during the user study. Figure 8 gives an example of one of the meal plan scenarios.

We used the meal plan components (individual dishes) to create the corpus, as large collections of daily meal plans are not common. We downloaded roughly fifty thousand recipes from the recipe-sharing website allrecipes.com to serve as the building blocks of our meal plan corpus. Allrecipes.com recipes include a variety of metadata (such as type of dish, meal, and cuisine) and nutritional info which made it ideal for building daily meal plans. From this, we used a meal plan generator that selects appropriate main dishes for breakfast, lunch and dinner, adding additional meal components (side dishes, drinks, appetizers and desserts) with decreasing probability as the daily calorie count increases, creating approximately a quarter million meal plans. Twenty of the attributes were nutritional information (e.g., calories, vitamin A, etc, as in Figure 5) which could be simply summed. The other attribute was allrecipe.com individual dish ratings, which were preserved for each meal plan.

5.4 Subject's tasks and rewards

We developed a game with rewards to motivate test subjects to take the task seriously and put effort into choosing the best items. We used Amazon Turk workers as our test subjects, restricting to workers within the United States and with high completed work acceptance rates (95% or better). The game was slightly different in each user study but followed the same basic structure. Several workers would be given the same scenario and were asked to choose the selection(s) that would be most likely to please the person described in the scenario. There were two roles, the searcher and judge, as described below:

Searcher This role was used to generate queries and relevance judgments. The searcher used a randomly selected search engine to search the corpus and select items. These selections were entered into a "contest" and assigned a judge, with the searcher receiving a bonus if their selection won the contest, as described below. For the ticketing user study, three tickets were selected; only one meal plan was selected in the meal plan user study.

Judge This role was used to validate work and provide bonuses. The judge selects items from a randomly ordered list without the benefit of a search engine. The judge would see a small subset of items that included the selection(s) of the searcher; if the judge chose the worker's selection, the worker would get a bonus. A second judge would be given the same set to judge, and if they made the same selection, they would both get a reward. study was altered to include only selections from two searchers using different search engines and two randomly selected meal plans, so the results from different search engines could be directly compared (head-to-head).

In addition, the meal plan user study asked each test subject to provide a justification for their selection. Work was rejected when justifications were inadequate and eliminated from our study.

Table 1: User Study Data

User Study	Test Subjects	Tasks Completed	Initial Queries	Items Chosen
Tickets	366	553	553	1659
Meal Plans	205	321	321	321

We excluded roughly half of the searcher responses (completed tasks) from the ticketing user study to eliminate noisy data as follows. For each scenario, we calculated the median probability of being matched with another searcher who selected at least one common ticket, discarding all responses that fell below this median. The two groups (discarded and preserved) showed a statistically significant difference on time spent according to a randomization test (also described below) at p=0.05. In the meal plan study, we rejected responses with inadequate justifications, eliminating about 10% of the responses. Table 1 summarizes the data for each user study; only the initial query is used in our analysis, thus its count is equal to the number of responses. Likewise, subjects were instructed to pick exactly three tickets and only one meal plan in the corresponding study.

5.5 Evaluation Metrics

As mentioned in Sec. 5.2, we use only the first query from each session to calculate mean average precision (MAP), precision at k (P@k), and mean reciprocal rank (MRR). However, a test subject may be likely to choose a higher ranked item when utility is

roughly the same. Moreover, given a large number of items, the item ultimately chosen is affected by the retrieval model's ranking as not all items will be viewed. This is irrelevant when comparing the retrieval models used directly by test subjects, but could bias the results when comparing retrieval models in our post hoc analysis. To compensate, we break the bond between the test subject's query and ultimate selections by using the selections from the other test subjects on the same scenario, which we refer to as the *community* evaluation. For the ticketing user study, we combined these choices into a single MAP evaluation since all users were given all the tickets, given the small corpus. For the meal plan user study, the combined set of search results from any test subject's session was such a small fraction of the corpus that there was little to no overlap among sessions. Therefore, each query was evaluated separately on each result set. We further only used result sets from queries that won at least the median number of contests.

As differences in response and acceptance rates per scenario gave us varying amounts of data, we average our results in two ways. The first is the *micro-average*, which is the mean over all responses without respect to the scenario. The second is the *macro-average*, which calculates an overall average from the mean of each scenario individually. The macro-average compensates for an unbalanced distribution but may have higher variability, as scenarios with fewer responses are weighted the same as those with more responses.

We use a randomization test[28] in two ways to calculate statistical significance. The first method is used when comparing responses by different subjects in the user study, using in-study models; no subject was allowed to respond to the same scenario more than once. Our null hypothesis is that each test subject would have had the same performance on either of the compared models, and so the observed difference is merely a chance event stemming from random assignment of test subjects. The second method is used in our post-study model evaluation. Here our null hypothesis is each method was equally likely to have produced the observed difference. To test the null hypothesis, we randomly redistribute the responses or the differences, respectively, within the scenarios among the two models one million times. The p-value is the fraction of times this redistribution produced a difference for the metric that was at least as great as the actual observation. We used the same simulation run to calculate the p-value for all metrics jointly (e.g., instead of using Bonferroni's correction), and found the p-values reported below hold for each family as well.

6 RESULTS

We provide various results of our experiment below, with the leader bolded and statistically significant difference (at p=0.05 or better) indicated by the dagger (†), with the overall best performance bolded. Table 2 shows the results of the ticketing user study with the searcher's own selections. The *SimpleMAUT* model outperformed all others by a statistically significant difference, and the *Sorted Boolean* model, despite its widespread use, performed the worst. Table 3 shows the results for community-judged MAP, calculated from the other test subject's ticket selections (see Section 5.5), with the maximum a posteriori estimate of model parameters used for the *LearnedMAUT* model. All models except *Autorank* performed relatively well, with *LearnedMAUT* performing the best, mostly by statistically significant differences.

Table 4 shows results on scenarios with explicit constraints (35% of the ticketing scenarios), again with the searcher's own selections. Surprisingly, though restricting results is more effective on these scenarios, *Sorted Boolean* is still outperformed by the unconstrained *SimpleMAUT*. Table 5 shows why; approximately a third of the final selections had been eliminated by the test subjects' initial constraints. Though restricting the result set was more effective eliminating unwanted choices from the constrained scenarios, as expected, it also eliminated final selections at almost the exact same rate for both constrained and unconstrained scenarios. This further demonstrates the the hazard of using hard constraints to approximate soft preferences, *even when the user need also has hard constraints*.

Table 6 gives the community-evaluated MAP scores for models in the meal plan user study. Qualitatively, the results are similar to the ticketing user study despite differences in the domain and corpus size, with the *LearnedMAUT* model outperforming the others. A unique feature of the meal plan domain was the multivalued dish rating attribute, which we aggregate with a generalized mean. The value of ψ in our experiment was close to the geometric mean (averaging around -0.25 and varying by fold, where a value of 0 yields the geometric mean). Changing the baselines to use the geometric mean (instead of arithmetic as shown in Table 6) yielded better results, mostly by a statistically significant differences; even so, the differences with the *LearnedMAUT* result remained statistically significant.

Another way to evaluate search result quality is to see how often a model was used to find the winning meal plan. Table 7 shows searchers using the *EnhancedMAUT* search engine were very successful, beating the competition (i.e., searchers using a different search engine) nearly two thirds of the time. Moreover, if we use the search engine to rank the contest entries (given as "Judge MRR"), the advantage of the *EnhancedMAUT* model is even clearer. A direct comparison is given in the "head-to-head" performance in Table 8, with each row in the table listing the "victories" in matches between the pair of search engines in the columns. For example, the *EnhancedMAUT* and *Faceted* search engines have competed 53 times (i.e., entered into the same contest, as described in Section 5.4), with the *EnhancedMAUT* paradigm winning 35 contests and losing 18. As with the other comparisons, the difference between *EnhancedMAUT* and the others is statistically significant.

Explicitly constraining the result set hurt the performance of *Sorted Boolean* and *Faceted* (**RQ2**). Hard constraints are not well-suited to expressing preferences, and we found that test subjects often ultimately selected items that were eliminated by their initial restrictions. Though the user interfaces for *Sorted Boolean* and *Faceted* are quite different, the underlying query semantics are identical, and we observed nearly identical retrieval performance. On the other hand, the *Tradeoff* model had a very similar ranking function to *SimpleMAUT*, but with *explicit* utility function parameters, and this lead to poor performance. Indeed, models that allowed users to give an explicit ranking (*Sorted Boolean*, *Faceted* and *Tradeoff*) performed worse than the models that implicitly ranked by attempting to glean query intent (**RQ3**). Overall, test subjects were most successful using the *implicit* MAUT query models (**RQ1**).

Table 2: In-Study Model Results on Ticketing User Study

	MAP		MRR		P@1		P@5		P@10		P@20	
Model	micro	macro	micro	macro	micro	macro	micro	macro	micro	macro	micro	macro
Sorted Boolean	0.378†	0.368†	0.481†	0.476†	0.396†	0.390†	0.249†	0.244†	0.149†	0.145†	0.085†	0.081†
Tradeoff	0.387†	0.380†	0.447†	0.459†	0.314†	0.332†	0.254†	0.241†	0.168†	0.167†	0.105†	0.104†
SimpleMAUT	**0.542**	**0.526**	**0.693**	**0.678**	**0.587**	**0.566**	**0.336**	**0.323**	**0.237**	**0.234**	**0.138**	**0.138**

Table 3: Tickets: Community-evaluated MAP

	MAP@10		MAP@25	
Model	micro	macro	micro	macro
AIMQ	0.394†	0.377†	0.385†	0.370†
Autorank	0.246†	0.269†	0.244†	0.267†
CQads	0.453†	0.453†	0.445†	0.447†
VAGUE	0.457	0.430†	0.451	0.427†
LearnedMAUT	**0.502**	**0.492**	**0.490**	**0.482**

Table 4: Tickets: Constrained Scenarios

	MAP		MRR	
Model	micro	macro	micro	macro
Sorted Boolean	0.382†	0.361†	0.515†	0.509†
SimpleMAUT	**0.568**	**0.543**	**0.703**	**0.684**

Table 5: Tickets Eliminated by Subject's Restrictions

Candidate Ticket	All Scenarios	Constrained Scenarios	Unconstrained Scenarios
Chosen	134 (34%)	49 (34%)	85 (33%)
Not Chosen	3503 (50%)	1627 (61%)	1876 (44%)

Table 6: Meal Plans: Community-evaluated MAP

	MAP@10		MAP@25	
Model	micro	macro	micro	macro
AIMQ	0.313†	0.287†	0.332†	0.306†
Autorank	0.218†	0.193†	0.252†	0.230†
CQads	0.337†	0.314†	0.355†	0.332†
VAGUE	0.323†	0.324	0.343†	0.343
LearnedMAUT	**0.393**	**0.382**	0.**407**	**0.396**

The models that accept single attribute values instead of ranges (*AIMQ*, *AutoRank*, *CQAds*, MAUT variants, *Tradeoff* and *VAGUE*) vary widely in their performance, despite their similarities. *AIMQ*, *AutoRank* and *CQAds* did not use the user attribute prioritizations, whereas *VAGUE* and *SimpleMAUT* did. However, further experimentation showed that the user attribute prioritizations (versus uniform prioritizations) convey only a slight advantage that was not statistically significant; thus we dropped user attribute prioritizations from

Table 7: Meal Plans: Searcher Success by Search Engine

Paradigm	Win Rate	Judge MRR
EnhancedMAUT	**0.61**†	**0.65**†
Faceted	0.44	0.17
Sorted Boolean	0.45	0.11

Table 8: Meal Plans: Head-to-Head

Enhanced MAUT	Faceted	Sorted Boolean
54†		29
35†	18	
	23	**25**

the *EnhancedMAUT*/*LearnedMAUT* model. *AIMQ* and *AutoRank* suffered because of their estimated attribute weights; replacing these with uniform weights improved performance. In contrast, *LearnedMAUT* was significantly better than other models in every category (**RQ1**). Overall, the research baseline models suffered as they did not have a way to adjust to new domains of application. They did not learn from usage, either adjusting their parameters based from an analysis of the corpus or from assumptions. This may have worked in domains used in their development, but not elsewhere. In contrast, we have trained and tested *LearnedMAUT* in two disparate domains, with it performing well in both.

7 CONCLUSIONS

In this paper, we explored the retrieval of items solely by their attribute values in two user studies conducted with Amazon Mechanical Turk. We proposed two models derived from multi-criteria decision making theory, a basic model *SimpleMAUT* and an advanced version, *EnhancedMAUT*, whose model parameters were tuned in a learning-to-rank framework, *LearnedMAUT*. We compared these to the *de facto explicit* retrieval models, where the user explicitly describes what to return and how to order it. We also compared our methods to several *implicit* retrieval models found in the literature, where retrieval and ranking is *implied* by the user's description of what is desired.

Our models outperformed these widely adopted *explicit* retrieval models. We analyzed the performance of the explicit retrieval models to understand why they did not perform better. Applying constraints to limit the result set is hazard-prone; the mismatch between constraints and preferences often eliminates the desired

items. Overall, users were more successful when providing an *implicit* ranking rather than *explicitly* describing the result set (either with constraints and sorting or by providing a utility function, as in the *Tradeoff* model). This largely matches the findings of decades of research in unstructured text retrieval, so retrieving items by attribute may not be as different as presumed. Nonetheless, not all *implicit* retrieval models are equivalent, in particular the MAUT-based models outperformed these baselines in our experiments. The baseline *implicit* models made different assumptions, notably in normalization and weighting, that did not always hold in practice.

We were able to learn a better retrieval model using a pairwise learning-to-rank approach, yet numerous possibilities remain. Analysis of query performance showed that underspecification was often the cause of poor retrieval performance, so enhancing retrieval with universal preferences could improve results. Also, we learned a global retrieval model, but personalized models and interactive retrieval are also promising avenues. Finally, we assumed mutual utility independence, but interaction among attributes should also be explored.

We developed a multi-attribute utility model to solve a particular problem, namely retrieving items by their attribute value. This is a common task itself, but the framework we developed could be applied to other retrieval problems with multiple dimensions, such as incorporating diversity or recency into document retrieval, matching subgraphs in semantic search, and so on. Casting retrieval as utility estimation provides a new way to think about the problem; multi-criteria decision making offers a technique for combining multiple evaluations; and our subutility functions translate raw features into utility estimates, fitting a variety of possible preferences.

REFERENCES

[1] Rakesh Agrawal and Edward L. Wimmers. 2000. A framework for expressing and combining preferences. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data (SIGMOD '00)*. ACM, New York, NY, USA, 297–306.

[2] Sanjay Agrawal, Surajit Chaudhuri, Gautam Das, and Aristides Gionis. 2003. Automated ranking of database query results. In *Proceedings of the First Biennial Conference on Innovative Data Systems Research (CIDR 2003)*. 888–899.

[3] S. Amer-Yahia, F. Bonchi, C. Castillo, E. Feuerstein, I. Mendez-Diaz, and P. Zabala. 2014. Composite Retrieval of Diverse and Complementary Bundles. *Knowledge and Data Engineering, IEEE Transactions on* 26, 11 (Nov 2014), 2662–2675.

[4] Albert Angel, Surajit Chaudhuri, Gautam Das, and Nick Koudas. 2009. Ranking Objects Based on Relationships and Fixed Associations. In *Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology (EDBT '09)*. ACM, New York, NY, USA, 910–921.

[5] Senjuti Basu Roy, Sihem Amer-Yahia, Ashish Chawla, Gautam Das, and Cong Yu. 2010. Constructing and Exploring Composite Items. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data (SIGMOD '10)*. ACM, New York, NY, USA, 843–854.

[6] G. Bordogna and G. Pasi. 2008. A multi criteria news filtering model. In *Annual Meeting of the North American Fuzzy Information Processing Society*. 1–6.

[7] Horatiu Bota, Ke Zhou, Joemon M. Jose, and Mounia Lalmas. 2014. Composite Retrieval of Heterogeneous Web Search. In *Proceedings of the 23rd International Conference on World Wide Web (WWW '14)*. ACM, New York, NY, USA, 119–130.

[8] Bureau of Transportation Statistics. 2006. Airline Origin and Destination Survey. (2006). Retrieved April 15, 2007 from http://www.transtats.bts.gov/Tables.asp?DB_ID=125

[9] Bureau of Transportation Statistics. 2006. *America on the Go: Findings from the National Household Travel Survey*. U.S. Department of Transportation. 6 pages.

[10] Surajit Chaudhuri, Gautam Das, Vagelis Hristidis, and Gerhard Weikum. 2004. Probabilistic ranking of database query results. In *Proceedings of the Thirtieth international conference on Very large data bases - Volume 30 (VLDB '04)*. VLDB Endowment, 888–899.

[11] Jan Chomicki. 2011. Logical Foundations of Preference Queries. *IEEE Data Eng. Bull.* 34, 2 (2011), 3–10.

[12] James S. Dyer. 2005. MAUT – Multiattribute Utility Theory. In *Multicriteria Decision Analysis: State of the Art Surveys*, Jos'e Figueria, Salvatore Greco, and Matthias Ehrgott (Eds.). Springer, 266–295.

[13] Norbert Fuhr. 1990. A probabilistic framework for vague queries and imprecise information in databases. In *Proceedings of the 16th International Conference on Very Large Databases*. Morgan, 696–707.

[14] Daniel Herzog and Wolfgang Wörndl. 2014. A Travel Recommender System for Combining Multiple Travel Regions to a Composite Trip. In *Proceedings of the 1st Workshop on New Trends in Content-based Recommender Systems co-located with the 8th ACM Conference on Recommender Systems, CBRecSys@RecSys 2014, Foster City, Silicon Valley, California, USA, October 6, 2014*. 42–48. http://ceur-ws.org/Vol-1245/cbrecsys2014-paper07.pdf

[15] Katja Hose and Akrivi Vlachou. 2012. A survey of skyline processing in highly distributed environments. *The VLDB Journal* 21, 3 (June 2012), 359–384.

[16] Ihab F. Ilyas, George Beskales, and Mohamed A. Soliman. 2008. A survey of top-k query processing techniques in relational database systems. *ACM Comput. Surv.* 40, 4, Article 11 (Oct. 2008), 58 pages.

[17] Werner Kießling. 2002. Foundations of preferences in database systems. In *Proceedings of the 28th international conference on Very Large Data Bases (VLDB '02)*. VLDB Endowment, 311–322.

[18] Werner Kießling and Gerhard Köstler. 2002. Preference SQL: design, implementation, experiences. In *Proceedings of the 28th international conference on Very Large Data Bases (VLDB '02)*. VLDB Endowment, 990–1001.

[19] N. Manouselis and C. Costopoulou. 2007. Analysis and Classification of Multi-Criteria Recommender Systems. In *World Wide Web*, Vol. 10. 415–441.

[20] Xiangfu Meng, Z. M. Ma, and Li Yan. 2009. Answering approximate queries over autonomous web databases. In *Proceedings of the 18th international conference on World wide web (WWW '09)*. ACM, New York, NY, USA, 1021–1030.

[21] Amihai Motro. 1988. Vague: a user interface to relational databases that permits vague queries. *ACM Transactions on Office Information Systems* 6 (1988), 187–214.

[22] Ullas Nambiar and Subbarao Kambhampati. 2005. Answering Imprecise Queries over Web Databases. In *Proceedings of the 31st International Conference on Very Large Data Bases (VLDB '05)*. VLDB Endowment, 1350–1353.

[23] Aditya Parameswaran, Petros Venetis, and Hector Garcia-Molina. 2011. Recommendation Systems with Complex Constraints: A Course Recommendation Perspective. *ACM Trans. Inf. Syst.* 29, 4, Article 20 (Dec. 2011), 33 pages.

[24] Aditya G. Parameswaran and Hector Garcia-Molina. 2009. Recommendations with Prerequisites. In *Proceedings of the Third ACM Conference on Recommender Systems (RecSys '09)*. ACM, New York, NY, USA, 353–356.

[25] Aditya G. Parameswaran, Hector Garcia-Molina, and Jeffrey D. Ullman. 2010. Evaluating, Combining and Generalizing Recommendations with Prerequisites. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management (CIKM '10)*. ACM, New York, NY, USA, 919–928.

[26] G. Pasi, G. Bordogna, and R. Villa. 2007. A multi-criteria content-based filtering system. In *30th annual international ACM SIGIR conference on Research and development in information retrieval*. 775–776.

[27] Carol Ann Rinzler. 1999. *Nutrition for Dummies* (second ed.). For Dummies.

[28] Mark D. Smucker, James Allan, and Ben Carterette. 2007. A comparison of statistical significance tests for information retrieval evaluation. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management (CIKM '07)*. ACM, New York, NY, USA, 623–632.

[29] M.A. Soliman, I.F. Ilyas, and K. Chen-Chuan Chang. 2007. Top-k Query Processing in Uncertain Databases. In *Data Engineering, 2007. ICDE 2007. IEEE 23rd International Conference on*. 896–905.

[30] Weifeng Su, Jiying Wang, Qiong Huang, and Fred Lochovsky. 2006. Query result ranking over e-commerce web databases. In *Proceedings of the 15th ACM international conference on Information and knowledge management (CIKM '06)*. ACM, New York, NY, USA, 575–584.

[31] Chang Tan, Qi Liu, Enhong Chen, Hui Xiong, and Xiang Wu. 2014. Object-Oriented Travel Package Recommendation. *ACM Trans. Intell. Syst. Technol.* 5, 3, Article 43 (Sept. 2014), 26 pages.

[32] WikiMedia Foundation. 2015. Dietary Reference Intake. (2015). Retrieved April 7, 2015 from http://en.wikipedia.org/wiki/Dietary_Reference_Intake,accessedonJune7,2015

[33] Min Xie, Laks V. S. Lakshmanan, and Peter T. Wood. 2011. CompRec-Trip: A Composite Recommendation System for Travel Planning. In *Proceedings of the 2011 IEEE 27th International Conference on Data Engineering (ICDE '11)*. IEEE Computer Society, Washington, DC, USA, 1352–1355.