UNIVERSITY OF CALIFORNIA
SANTA CRUZ

RETRIEVING STRUCTURED ITEMS VIA UTILITY ESTIMATION

A dissertation submitted in partial satisfaction of the
requirements for the degree of

DOCTOR OF PHILOSOPHY

in

COMPUTER SCIENCE

by

Shawn R. Wolfe

December 2018

The Dissertation of Shawn R. Wolfe
is approved:

Professor Yi Zhang, Chair

Professor Lise Getoor

Richard M. Keller, Ph.D.

Lori Kletzer
Vice Provost and Dean of Graduate Studies

Table of Contents

List of Figures

List of Tables

Abstract

Retrieving Structured Items via Utility Estimation

by

Shawn R. Wolfe

Searching for items by their attribute values or metadata is a commonplace task in e-commerce and science today, for instance, when searching for a product by its technical specifications. Finding a desirable item in such a catalog requires that the user specify desirable properties, specifically desirable attribute values. Current search tools support a retrieval style similar to a database, requiring users to place hard constraints on acceptable attribute values to limit the result set, as in Boolean or faceted search. 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.

In contrast, modern information retrieval systems have largely abandoned constraint-based retrieval models for those that estimate relevance to the latent user need. Such systems can avoid the problems of constraint-based search, such as empty results sets, by instead ranking by estimated relevance. They also shift the user's mental model from how to retrieve desired results, to simply what results are desired. Such information retrieval techniques have been successfully applied to a wide range of retrieval problems, but not to item retrieval, particularly given numeric attribute values.

This dissertation develops a model of relevance for item retrieval based in part from concepts in multi-attribute decision making theory. We cast the problem as that of utility estimation, and in contrast to the Boolean and faceted approaches, our approach does not use constraints. First, we develop a core model based

on multi-attribute utility theory that trades off among conflicting criteria on the user's behalf, and in this way get closer to the underlying query intent. Second, we develop a flexible model of subutility for numeric attributes, using a Bayesian graphical model to learn the specific subutility functions. Finally, we expand our subutility model to handle other types of attributes and to interpret vague natural language queries. We evaluate our model on several item recommender and retrieval datasets, as well as two user studies, and compare its performance to the de facto standard of Boolean retrieval as well as several models proposed in the literature.

To Christina

Acknowledgments

First of all, I would like to thank my advisor, Yi Zhang. Her research and guidance inspired me to come up with this topic and it would not have come to be had it not been for her. Whenever the research seemed to stall and I was grasping for ideas, she was always quick to suggest a completely new direction. This journey took a lot longer than either of us had expected, but she showed limitless patience and endless encouragement.

I would also like to thank my other committee members, Rich Keller, Lise Getoor, as well as Neoklis "Alkis" Polyzotis and Manfred Warmuth. Like my advisor, their suggestions helped me improve the research as they brought ideas from their areas of expertise. I would like to thank Rich for his always steadfast support, bringing a fresh perspective to the research and always encouraging me to step outside my comfort zone. Likewise, I really appreciate Lise's encouragement and suggestions, and flexibility despite challenging circumstances and timeframe. I am also grateful to Alkis for directing me to important database research and answering my implementation questions, and for Manfred teaching me many of the theories and approaches to machine learning.

I am also grateful to have met other students at the Information Retrieval and Knowledge Management (IRKM) Lab, Sarah Tyler, Aaron Michelony, Lanbo Zhang, Jian Wang, Qi Zhao, Yize Li, Jessica Gronski and Jonathan Koren. As a returning and part-time student, it was easy to feel out of place, but they welcomed me and made me feel part of the group. It was easier to face the challenges of graduate school together, and I enjoyed being able to discuss the technical and the not-so-technical with my lab-mates.

I am also indebted to Jason Wright of Penn State for allowing me to set up my search engine to search his exoplanet data. Picking a topic that had no relevant

test collections was a major obstacle, and gathering data was the bulk of the work. The exoplanet web site was a perfect venue to finally test my ideas. I am grateful to Y. Katherina Feng as well, who helped connect me to Jason and encouraged me in my research. I also would like to thank my employer, NASA, who supported me through this journey both financially and professionally.

Even before I embarked on this endeavor, I am indebted to my parents, Jim and Susie, who raised me from a baby. They always believed in me and gave me what I needed to succeed without ever really asking for anything in return. They gave me a firm foundation and encouraged me to be whatever I wanted to be.

Most of all, I have the deepest gratitude to my wife, Christina. Her love and unwavering support gave me the fuel to keep going. The joy of the much-too-few moments we spent together during this time were sparks of light in the darkest moments. The remainder is too sappy for this dissertation, so I conclude by dedicating this dissertation to you, Christina.

Chapter 1

Introduction

Techniques for the retrieval of items (often thought of as records or data) and information (primarily text) have been developed independently over the years, with little overlap. Automated information retrieval was envisioned as early as 1948 by Vannevar Bush [17], with the earliest systems using a Boolean retrieval paradigm. The Boolean paradigm gave way to other classic models, such as the vector space and probabilistic models, as they were able to accommodate partial matches to queries [8]. Today, the Boolean paradigm has largely fallen by the wayside in information retrieval, used in only a few specialized area where recall is essential [50], for instance in systematic medical reviews [69].

In contrast, when searching for items by attribute values, the retrieval systems in use at the time of writing adhere to the Boolean framework. These systems operate by placing conditions on the attributes of the matching items, coming in two varieties: a faceted search/navigation interface, or a database-like querying approach. In either case, the user provides an explicit specification on how to find desired items: defining the results set by including or excluding items by specific attribute values, and/or an explicit sort order. In contrast to information retrieval, this *explicit* form of retrieval is more like that in a database, and is best

suited to the user who can clearly articulate the user need (i.e., what is sought) and provide clear specification of what should be returned.

Why are item and information retrieval treated so differently? Weikum [80] points to historical precedence, where item retrieval is assumed to best match a database-like approach. Databases have been used to retrieve data that is primarily numeric or categorical; information retrieval systems mostly retrieve documents of unstructured text. Database researchers find efficient ways to retrieve data given precise query semantics; information retrieval researchers seek to maximize the relevancy of results given uncertain user needs. As Fuhr pointed out in his Salton award lecture[1], this division is not illogical: the goal of an information retrieval system is to support the user in their task or problem solving, whereas a database is intended for precise logic-based retrieval (e.g. in a computer program), with any ambiguity or uncertainty falling outside the scope of the database, to the application layer [34].

Are user needs best served by the Boolean model when retrieving items that are not unstructured text? Recently researchers have begun to challenge this status quo. Information retrieval has branched out into structured domains, most notably the retrieval of XML documents or fragments [45]. Information retrieval has also been applied to non-textual domains, as in music information retrieval [19], content-based image retrieval [24], and other forms of multimedia retrieval. Database researchers, for their part, have noticed the increasing prevalence of substantial text within their structured holdings and have advocated for leveraging information retrieval techniques [80] and integration into the database query framework [22].

Nonetheless, when it comes to records, i.e., structured items that are defined by

[1]The Salton award is a prestigious lifetime award for information retrieval researchers, named after Gerald Salton, who made seminal contributions in the early development of information retrieval.

their attributes, the presumption of a database-like approach for retrieval remains persistent. Databases are meant to provide speedy access to structured items, and are often used programmatically. Therefore, database researchers tend to think in terms of logical semantics and efficiency of retrieval. In the database paradigm, the caller should have an understanding of the layout of the database, a precise and unambiguous definition of what information is desired, and the sophistication to correctly state this in the query language. This contrasts with information retrieval researchers who tend to think in terms of vague user needs and estimations of relevance to that need.

Human users also often need to search for items by their attribute, though, and their needs are not always satisfied through this rigid form of retrieval. We illustrate some of the difficulties through a extended hypothetical example:

> Muhammad is struggling in his probability and statistics class. His calculus is rusty and wasn't very good in the first place, so he wants a reference that he can brush up on it in the evenings. He would like a book that is cheap, not too long, and has a high rating with a good number of reviews (so he can trust the rating). He searches online at a popular retail site using "calculus" as the sole keyword and gets over 30,000 matches. Overwhelmed, he modifies the query to restrict price < 25, review $= 5$ and gets no results. He relaxes his query restrictions to review ≥ 4 and gets almost 2000 books. The books that come up on the first page are a mishmash of various topics, studying for the Advanced Placement (AP) exam, physics applications of calculus, and books that look far too advanced. He tries sorting this list by price, but the books that are listed first are basic calculus books that don't look to be much help. He continues to tweaking his query, alternating between too many results and none, and repeating this process until frustration sets in. As it turns out, there was a book "Understanding Calculus" that nearly matched his criteria, only five dollars more expensive than his stated limit and with somewhat fewer reviews than he had desired. Muhammad never found that book, and instead ended up buying the cheapest one, which was a translation from a Russian text. That book isn't very helpful and he ends up failing the class.

1.1 Shortcomings of the Boolean Approach

We consider a general situation that at first might appear best suited for the database paradigm: a large collection of structured items with primarily numeric attributes. We assume that the user does understand the structure of these items and is able to at least approximate the need using the query language of the database system. However, we assume that the user has no knowledge of the specific items in the database. Moreover, we assume that the user's need is such that it cannot be precisely specified- either there is some element that cannot be expressed in the language (but can be evaluated through manual inspection), or many items fit the need, but some satisfy it more than others.

For such a user need, we identify several ways a Boolean "all-or-nothing" query approach can fail to give good results.

- **Too few results.** If the user specifies an overly specific specification of the user need, no items might match that query, even in a large dataset. Even if some results are returned, it maybe that the user has listed some constraints but not others, and so the "best" match may violate one of the listed constraints and thus be missing from the result set.

- **Too many results.** On the other hand, if the user specifies a broader version of the need, perhaps after being frustrated by the above case, the number of results returned may be overwhelming. The "best" match may indeed be in the result set, but finding it in the list can amount to finding a needle in a haystack. In the pathological worst case, the broad query returns all results, and the query engine has failed to add any value.

- **Unranked results.** Even if the user has specified the query in the best possible terms, including all desired results in the result set and excluding

as many as possible, the result set remains unranked. For large data sets or user needs that are not specified precisely, this may very well be a large list. Of course, in most database retrieval systems it is possible to order the list by a user-specified function. However, this makes another strong demand of the user– that the user have a precise understanding of how to rank results according to the need and the ability and patience to describe it in a formula.

The issues all stem from the basic design of the retrieval algorithm in a Boolean query system, namely that the user must *explicitly* provide a definition of relevance. In contrast, modern information retrieval systems allow use an *implicit* paradigm, where the user provides some description of the need, and the system infers what is relevant. Of course, when searching for items, there may be some situations when an explicit retrieval model is best suited, just as the Boolean paradigm remains in use for certain specialized information retrieval tasks. The assumption that explicit retrieval methods are appropriate for all or most item retrieval tasks, however, is not warranted.

1.2 The Goal of Utility Estimation for Item Retrieval

In contrast to Boolean retrieval in the example above, the information retrieval community has developed various retrieval models, such as vector space models, language models, and inference networks. These retrieval systems have moved away from a Boolean database paradigm to a non-Boolean direction, which does not assume a user can issue an accurate, explicit query, and thus tries to infer the information needs or intention of the user. As demonstrated in information

retrieval research, these information retrieval systems usually work better than Boolean retrieval systems in typical information retrieval tasks, particularly in text retrieval systems.

This leads to our our main hypothesis, which we will test in this dissertation: *an item retrieval system that estimates relevance to the user's need can offer better performance than one that relies on explicit query semantics.* The overarching goal of this dissertation is to test this hypothesis by developing such an item retrieval framework. The first challenge is develop a basic retrieval model, which defines what form the query may take and how to relate this to items in the corpus. We choose to use utility estimation as our approach, using a query consisting of desired attribute values as reference points of maximal utility. We use principals from multi-criteria decision making to combine these multiple criteria into a single utility estimate per item, and use these estimates to produce a ranking of results. The second challenge is to develop a model of how the item's attribute values translate into the criterion rating, i.e., to develop a method to estimate per-attribute subutility. We choose to do this by developing a parameterized subutility function, derived from expected properties of such a function, and embedded in a Bayesian graphical model to learn its parameter values from training data. The third challenge is extend the model so that it can be applied in complex domains, using natural query paradigms. We explore this by extending our approach to handle the attribute types of a space science dataset, and to interpret imprecise values from natural language queries.

1.3 Contributions of the Dissertation

This dissertation evaluates methods for the retrieval and ranking of items given a user's query. We develop a retrieval framework, inspired by multi-criteria

decision making theory, that estimates relevance by means of a multiple attribute utility model. We develop the model in several steps:

Combine Multiple Criteria Ratings Into a Relevance Estimation. We evaluate how multi-criteria decision making models perform as retrieval models, by applying two representative models from the main branches of multi-criteria decision making, an outranking model and a utility estimation model. We evaluate these in the context of two recommendation tasks, and compare their performance with a Boolean retrieval model. We develop our first retrieval model, translating query and attribute values into criteria ratings, and combine these in a model of utility. We evaluate this first utility model in a online experiment in an airline ticketing domain, along with three explicit retrieval models, including Boolean retrieval. We also use a listwise learning-to-rank algorithm to adjust the parameters of the utility model, and compare its performance to several item retrieval models from the literature.

Estimate Criteria Subutility from Attribute Values. We test our earlier assumption of a linear subutility model by comparing its ranking performance on two recommendation tasks. Based on our findings, we develop a parameterized yet flexible subutility function, guided by several principles relating the item's attribute value to that which the user seeks. We evaluate this with another online experiment, seeking nutritionally-based meals, and compare the performance of our enhanced model with two explicit retrieval models, again including Boolean retrieval. We further extend the model by embedding the utility function in a Bayesian graphical network, learning the model's parameters in pairwise learning-to-rank framework, and compare its performance to several item retrieval models from the literature.

Extend Retrieval Model to New Data Types, Uses and Query Modalities.

We explore how the retrieval model could be used in new domains and query interfaces. We expand the model cover Boolean, enumerated and textual attribute types, as well as to accept numeric ranges. We evaluate the extended model's performance on data gathered from a scientific vertical search site. We also explore an open-ended query model where users describe their user need using natural language. We gather such queries in an online experiment, and identify several categories of imprecise query clauses. We interpret these clauses in our utility-based model, as well as two explicit retrieval models, and evaluate the performance of each, again in the nutrition domain.

1.4 Outline of the Dissertation

The dissertation is structured as follows. Chapter 2 provides an overview of work relevant to item retrieval, in particular multi-criteria decision making theory and algorithms, economic and psychological models of decision making, and the few item retrieval methods that exist in the literature. Chapter 3 describes our basic utility estimation model, its use in item retrieval, and evaluation in a user study. Chapter 4 enhances this model by developing a model of subutility functions and evaluates its performance in another user study. Chapter 5 extends the model further by expanding it to cover the attribute types of a vertical search engine and to accept imprecise natural language queries, evaluating both in retrospective studies. Chapter 6 concludes with a summary of the work and opportunities for further improvements.

Although the chapters build upon one another, each is a self-contained contained unit and can be read in isolation. Readers with a basic familiarity in information retrieval and who are unconcerned with theory and other item retrieval methods can skip Chapter 2. Those uninterested in the development of

the basic framework can pass over Chapter 3 and instead focus on the more advanced versions in Chapters 4 and 5. Those wanting only a basic idea can read Chapter 6, and possibly, the chapter you are currently reading.

Parts of the thesis have been previously published, although at times the coverage in this dissertation is quite different. Chapter 3 includes material published in the ACM Conference on Research and Development in Information Retrieval (SIGIR) in 2009 [82] and 2018 [84]. Chapter 4 includes material published in the International Conference on User Modeling, Adaptation, and Personalization in 2010 [83] as well as the ACM Conference on Research and Development in Information Retrieval (SIGIR) in 2018 [84]. In contrast, chapter 5 contains our most recent work that has yet to be submitted for publication.

Chapter 2

Background

2.1 Multi-criteria Decision Making

Multi-criteria decision making (MCDM), also referred as multiple criteria decision analysis, is a branch of operations research designed to assist in the making of particular types of decisions [67]. These decision problems involve several criteria (hence the name) which presumably are somewhat at odds. In other words, the problem is assumed to be difficult, in the sense that the criteria are at least partially in conflict, meaning that some options are preferred under a particular criterion while disfavored under a different criterion.

The focus of multiple criteria decision making is specifically on combining these criteria to lead the decision maker to a particular *alternative* (i.e., choice). It is a prescriptive rather than a predictive approach, in that it is meant to present options to the decision maker that are more likely to be accepted, and perhaps to persuade such selections, rather than predict what decision would have ultimately been made without any decision making tool.

Information retrieval has similarities to MCDM. Like MCDM, its focus is on modeling the decision making of a user– in this case, the specific decision of

whether or not a given item is relevant to the user's information need. Like MCDM, the operational process is prescriptive rather than predictive. Moreover, several researchers have framed the information retrieval problem in terms of multiple criteria [30, 31, 51, 63, 11, 82]. Indeed, the popular multidimensional "bag of words" representation in information retrieval is similar to the multi-criteria decision matrix, and there is some similarity in the methods used in both.

2.1.1 Principles

MCDM, like information retrieval, can be applied to different classes of problems. We focus on the ranking problem, assuming a finite number of alternatives to rank, where the goal is to rank the alternatives from best to worst, without making any claim to the value of a particular rank. The objective of ranking is to produce a ranking of alternatives such that no *rank inversions* occur. A rank inversion occurs whenever an lower ranked alternative is preferred to a higher ranked alternative; i.e., the lower ranked item should have been ranked higher.

Many multi-criteria decision-making methods assign a numeric score to each alternative, so that a ranking can be derived simply from ordering by the corresponding scores. However, the choice of scoring function and the interdependencies of the criteria affect one another, as described in multi-attribute utility theory (MAUT) [28]. We review these restrictions next. We assume that for each pair of alternatives, there is either a preference for one or the other or an indifference (all pairs are comparable). Since we have restricted ourselves to a finite set of alternatives, it is possible to assign a score to each alternative such that a ranking by these scores is free of ranking inversions.

Consider a subset of criteria. That subset is said to be *preference independent* if and only if the preferences for alternatives, given these criteria, are not affected

by constant (shared) ratings on the other criteria. Consider a restaurant selection problem with three binary criteria: ambience, tastiness, and price. If you prefer restaurants with poor ambience and tasty food to those with good ambience with lousy food whenever the price is cheap (in both alternatives), and this preference does not change when the price is expensive (for both alternatives), then ambience and tastiness are *preference independent* of price.

Preference independence for a subset of criteria indicates that some decomposition of the scoring function is possible, but does not necessarily yield a straightforward formula. On the other hand, *mutual preference independence* does. *Mutual preference independence* means that all subsets of criteria are preference independent. In this case, the overall score of the i^{th} alternative has a multiplicative score, as given in Eq. 2.1:

$$u(A_i) = \frac{\prod_{j=1}^{n} [cw_j u_j(a_{ij}) + 1] - 1}{c} \tag{2.1}$$

where $u_j()$ is a function that maps the ratings of the j^{th} criterion onto a subutility curve, w_j is a weighting factor for the j^{th} criterion, and c is an overall scaling factor.

A stronger notion yet is that of *difference independence*. A subset of criteria is said to be *difference independent* if and only if the difference in utility, given values on these subsets, are not affected by constant (shared) ratings on the other criteria. In our restaurant example above, if ambience and tastiness are difference independent of price, and the difference in the cheap case for the ratings stated above was 0.2 in overall utility, then the difference for the expensive case with the same pairs of ratings would also have to be 0.2 in overall utility.

Like preference independence, *mutual difference independence* is difference independence in all subsets of criteria and yields a powerful new functional form. In

difference independence, gains or losses in utility can be calculated independent of the ratings of the other criteria. This amounts to an additive effect on the overall utility, and so in the case of mutual difference independence, the score is simply a linear combination, as in Eq. 2.2:

$$u(A_i) = \sum_{j=1}^{n} [w_j u_j(a_{ij})] \tag{2.2}$$

with the same definitions as for Eq. 2.1. Since mutual difference independence implies mutual preference independence, Eq. 2.2 must be a special case of Eq. 2.1, and apparently this is the case when the scaling constant c is zero [28].

However, it is worth stressing that for mutual difference independence, the same difference on the *rating* do not necessarily produce the same change in overall utility. Instead, it is the difference in the value of the criterion's subutility function u_j, which is not necessarily linear. Indeed, the presence of these criterion subutility functions make discovering the true underlying functions more challenging. As an example, compare a multiplicative (mutual preference independence) overall utility function with linear criterion subutility functions to that of an additive (mutual difference independence) overall utility function with log criterion subutility functions. Both functions produce the same preferences, but do not have the same utility values.

2.1.2 Ranking Models

In this section, we review the two major approaches to ranking alternatives used in the methods: utility-based and outranking. Utility-based approaches give an intrinsic score to each alternative, and rank according to that score. Outranking methods use information about the preferences to induce a ranking, with less dependence on the magnitudes of the ratings. The most basic outranking notion

is that of *dominance*. An alternative A_i *dominates* alternative A_k iff $\forall j : a_{ij} \geq a_{kj}$ and $\exists j : a_{ij} > a_{kj}$. Unfortunately, dominance is insufficiently prevalent to be of much use in most real world problems, and an utility-based approach that assumes monotonicity will respect dominance anyway. Therefore, outranking techniques attempt to go beyond the notion of dominance. For all methods, we assume a complete set of ratings for all alternatives on each criterion, and a set of nonzero criteria weights that typically are constrained to sum to one.

Weighted Sum

The simplest model, a linear combination of the ratings, is the most commonly used approach. The score is computed as in Eq. 2.3 below:

$$s\left(A_i\right) = \sum_{j=1}^{n} w_j a_{ij} \tag{2.3}$$

The Analytical Hierarchical Process (AHP) [68] uses the weighted sum to calculate scores once the ratings and weights have been determined. Typically, the ratings are normalized per criterion either so the sum of the ratings equals one (referred to as the distributed mode) or so that the greatest rating is one (referred to as the ideal mode). UTASTAR (UTilité Additives) [70] also uses a weighted sum as its basis for combining criteria. Note that in UTASTAR, the criterion weights are implicitly modeled in its learned utility functions approximations, and that the linear combination is on those functions and not the criteria ratings. The TOMASO method [53] does not quite fit the independent linear model, as it remaps the scores by comparing them with others, and has different sets of weights for different permutations, but effectively amounts to a linear combination, something we will also see in other methods.

Weighted Product

On the other hand, the weighted sum has been criticized because it combines items of incomparable units. For instance, in our restaurant example, cost and deliciousness are combined into a single score, despite have presumably different units. (However, this can be easily remedied by consider the weights as units of utility per criterion unit.) A ratio comparison in a weighted product [75], as given in Eq. 2.4, has been proposed as a "dimensionless" measure that avoids this issue:

$$r\left(A_i, A_k\right) = \prod_{j=1}^{n} \left(\frac{a_{ij}}{a_{kj}}\right)^{w_j} \tag{2.4}$$

As given in Eq. 2.4, this yields many more values (in the square of number of alternatives) and is not the same type of scoring function as other methods reviewed so far, though it presents no obstacle to ranking. Nonetheless, it turns out that an equivalent order can be obtained by using the scoring function in Eq. 2.5:

$$s\left(A_i\right) = \prod_{j=1}^{n} \left(a_{ij}\right)^{w_j} \tag{2.5}$$

Moreover, as log is an order preserving function, we can create the same ranking as in Eq. 2.5 using its log, which once again becomes a linear combination. So if we allow for subutility functions (as in UTASTAR) and are concerned with only the ranking and not the absolute values, we may again use the weighted sum.

ELECTRE II

The ELECTRE method [32] is probably the best known outranking approach and certainly one of the most venerable, having multiple versions and leading to new outranking approaches. We describe ELECTRE II, which is intended to

order alternatives, although many of the concepts are common to all ELECTRE models.

ELECTRE is based on a notion of *concordance* (acceptance of a proposition) and *disconcordance* (rejection of a proposition). For a given pair of alternatives A_i and A_k, the proposition to evaluate is that A_i is at least as preferable as A_k. Each criterion is given a weight: concordance holds when the sum of the criteria weights where A_i is at least as good as A_k meets some predefined threshold. Discordance holds when some rating on A_k is greater than that of A_i by some predefined amount. Note that it is possible to have concordance and discordance on the same pair.

From this, ELECTRE II defines both a strong and a weak preference, with the latter using less stringent thresholds, following the description in [32]. For both preferences, A_i is preferred over A_k when there is a concordance without a discordance. Two orders are created from both preferences, a direct ranking and an inverse ranking. In the direct ranking, all alternatives that do not have a more strongly preferred alternative are ranked above all others. Within this group, items are further ordered by a analogous process using the weak preference, with any ties broken randomly. The process is repeated on any unranked alternatives until all alternatives are ranked.

The inverse ranking is created starting with a bottom-most group, consisting of all alternatives that are not preferred over any other alternative, but otherwise in the same manner as the direct ranking. The end results is two complete rankings that may not be identical. A final order can be reached by sorting by the average of ranks from the two orders.

PROMETHEE II

PROMETHEE II [13] (and its predecessor, PROMETHEE I) is an outranking method designed to address some of the difficulties with the ELECTRE family of methods. A difference mapping function is used on each criterion to measure the degree to which an alternative is preferred over another (ranging from zero, indicating no preference, to one, indicating total preference). For each criterion, the decision maker must choose a particular difference mapping function.

The average differences are calculated as positive and negative outranking flows, similar to the notions of concordance and discordance. These are then combined into a single fuzzy measure, with the final order is now determined by this score, as would be done in a utility-based method. Indeed, a little algebraic manipulation reveals that the final score is once again a linear combination of criteria-based quantities, as it is with several utility-based methods. However, as these quantities are computed by comparisons with other alternatives, PROMETHEE II maintains its outranking perspective.

TOPSIS

TOPSIS [76] further simplifies the outranking concept to the point that it is worth questioning if TOPSIS is an outranking method at all. TOPSIS is based on a weighted Euclidean distance measure and comparison to positive and negative ideals (i.e., the best and worst possible alternatives). The final score of each alternative is defined as the ratio of the weighted Euclidean distance to the negative ideal versus the sum of the weighted Euclidean distances to both ideals, measuring how much closer the alternative is to the positive ideal than it is to the negative ideal. As with the other methods with a single score, the final ranking is simply done by this score.

2.2 Behavioral Economics and Mathematical Psychology

Behavioral Economics and Mathematical Psychology are subfields of their respective disciplines that develop models of decision making. Behavioral Economics differs from standard economics in that the goal is to build models of how people actually behave (like psychology), rather than assuming rationality and a quest for optimality. Likewise, researchers in mathematical psychology strive to build models of human behavior for the purposes of prediction, unlike standard psychology which focuses more on understanding the behavior than making testable predictions. The two subfields have substantial overlap and little practical distinction for our purposes.

Experimental findings from these subfields support the exemplar-based paradigm common in multi-criteria decision making, though the goals (predictive versus prescriptive) differ. Kerimi, Montgomery and Zakay [42] found people consumer choice generally matches the weighted-sum approach of MAUT (see Section 2.1.1), but switches to a Euclidean distance to the ideal when MAUT fails to produce a winner. Dieckmann, Dippold and Dietrich [26] compared the a weighted-sum prediction with that of a noncompensatory model (in this case, a lexicographic model, but part of the wider outranking family) in consumer choice and found the weighted sum prediction had better agreement with the actual choices.

Perhaps the most well-known, influential, comprehensive and thoroughly tested model of decision making is *prospect theory* [77, 41]. Daniel Kahneman, the surviving co-author of prospect theory, was awarded a Nobel prize[1] in 2002 in recognition for his development of prospect theory. Prospect theory was developed in

[1]The Nobel prize in economics is separate from the others in that is not one of the five areas designated in the will of Alfred Nobel, the originator of the Nobel prize.

part to explain the so-called Allais paradox, where a the same individual makes both *risk-seeking* (e.g., playing the lottery) and *risk-avoiding* (e.g., buying insurance) choices – an apparent inconsistency. Prospect theory has several aspects, but for this dissertation the most salient its utility function; that people evaluate options based on a reference point, concave in the region of gains (greater than the reference) and convex in the region of losses (below the reference). Originally developed to model choices under uncertainty (e.g., lotteries), prospect theory has been successfully applied to consumer choice [18].

2.3 Multiple Criteria Approaches to Information Retrieval

Most of the research in information retrieval that uses multiple criteria has been in information filtering. Manouselis and Costopoulou categorize 37 recommender systems that implicitly use some multi-criteria aspect in their operation [51]. The majority of these systems' methods can be viewed as the weighted sum method presented in this paper. Of the information filtering systems we are aware of, PENG [63] is the most similar to the one in our experimental study. PENG is a multi-criteria news bulletin filtering system that utilizes several criteria, including content, coverage, reliability, novelty and timeliness. A later evaluation of PENG, using only content and coverage, showed comparable or superior performance to other approaches [11]. Farah and Vanderpooten have explored the use of multiple criteria in the context of search using rank-based methods. In their work [30], the user provides query terms as the only input (and thus criterion) for the search process. From this, additional criteria are formed from elements of the web page, such as text, keywords, anchor text and incoming links. Later work [31] expanded

on this notion by using the rankings produced several high performing algorithms, with each algorithm essentially acting as a criterion or critic. In both cases, their findings were that the use of multiple criteria provided performance comparable to the best single criteria. However, they equate criteria to document features, which differs from our view of criteria as part of user's decision of what makes a document good.

In related fields, multi-criteria methods have been used to combine multiple forms of evidence, in the same spirit of the work of Farah and Vanderpooten, rather than multiple user-specified criteria. In text classification, a lazy classification technique consisted of generating rules for each unclassified document based on classified documents with similar content or linkages [78]. Each rule generated a weighted vote which was combined to a single verdict, resulting in accuracy comparable to other approaches but derived in much less time. Multiple similarity metrics have been combined in a case-based reasoning domain [46]. ELECTRE II, a rank-based MCDM algorithm that is designed to work with conflicting criteria, was used, and we have also used this algorithm in our study as a simple member of the rank-based family of MCDM algorithms. Their results show an improvement in performance over single criterion techniques when the good criteria weights are chosen. Finally, a multi-criteria approach has also been applied to feature selection [27]. It was shown in experimental results that a slight improvement in performance can be gained by combining the top selections of several feature selection methods (e.g., mutual information, chi-square, etc) over using any single method.

2.4 Search Methods for Items

Common item retrieval methods use a Boolean retrieval paradigm, either in database-like query or faceted search [37], with the latter a popular choice with many e-commerce websites [25]. The Boolean model has also been applied to new types of data, for instance object-oriented objects [35], Extensible Markup Language (XML) [86] documents, and Resource Description Format (RDF) [85] data. Database researchers have also expanded the query model while still preserving clear retrieval semantics, notably with top-K approaches [40], which retrieve the k-highest scored items given a scoring formula, and ranking given uncertain data [72]. Skyline queries [39] 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, 43, 44, 23]. 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 we address in this dissertation, namely that of 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. [21] and Su et al. [73] 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 [57] 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. [52]. CQAds [65] 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 [54, 55] 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 [33], 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 experiments.

Package retrieval is complementary task to item retrieval (retrieving a composite set of item instead of an individual item). Prior research has focused on recommending packages that meet the user's constraints and maximizing a provided objective function, thus more aligned with the *explicit* retrieval of database technologies rather than the *implicit* retrieval models of information retrieval. Package recommendation [89] has been explored in a number of areas, such as trip planning [38, 88, 74, 6], student course planning [61, 62, 60], compatible products [9], diversity in restaurants [5] and web page conglomeration [12]. Given the large number of potential packages, recommended packages are typically generated on the fly, typically an NP-complete problem.

2.5 Metrics and Statistical Significance

We calculate a variety of statistics to evaluate the performance of retrieval models. The first of these is mean squared error. Given a vector $\mathbf{y}$ consisting of

n observations, and a corresponding vector $\hat{\mathbf{y}}$ of n predictions, the mean squared error (MSE) is calculated as:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i) \tag{2.6}$$

Mean squared error is a useful metric for evaluating the performance of a regression model.

Although some form of numeric prediction often is a part of a retrieval system, evaluating the accuracy of those predictions is rarely possible as the actual value is unknown. Instead, documents (or items, in our case) are graded on relevance to the user need. When this scale is binary, as it is in this dissertation, it would be possible to calculate classification accuracy, but this is rarely helpful given a large class imbalance (most documents are not relevant). Two fundamental metrics are used to evaluate a result set in information retrieval, recall and precision. Recall is the fraction of relevant results returned, given as:

$$r = \frac{|\mathcal{X} \cap \mathcal{R}|}{|\mathcal{R}|} \tag{2.7}$$

where $\mathcal{X}$ is the result set and $\mathcal{R}$ is the set of relevant documents. Maximal recall can be achieved trivially by returning the entire corpus. Precision is the fraction of the result set that is relevant, calculated as:

$$p = \frac{|\mathcal{X} \cap \mathcal{R}|}{|\mathcal{X}|} \tag{2.8}$$

with $\mathcal{X}$ and $\mathcal{R}$ defined as above. Precision is more difficult to optimize, and is usually of more interest in retrieval systems as it is rarely necessary to review all relevant documents.

Ranking of the documents in a result set is also of interest, and the metrics

typically incorporate precision or recall, explicitly or implicitly. The result set is often quite large and subdivided into a paged display, so users essentially determine the result set size by the number of pages reviewed. Perhaps the simplest rank-based metric is reciprocal rank, which is the precision of the ranked set of results, ending with the first relevant result. Given n queries, mean reciprocal rank (MRR) is:

$$MRR = \frac{1}{n} \sum_{i=1}^{n} \left(\frac{1}{s_{i,1}} \right) \tag{2.9}$$

where $s_{i,1}$ is the rank of the first relevant document returned for the i^{th} query. Expanded the metric to cover all relevant results yields mean average precision (MAP), given by:

$$MAP = \frac{1}{n} \sum_{i=1}^{n} \left[\sum_{j=1}^{|\mathcal{R}_j|} \left(\frac{j}{s_{i,j}} \right) \right] \tag{2.10}$$

where $\mathcal{R}_j$ is the set of relevant documents for the i^{th} query, and $s_{i,j}$ is the rank of the j^{th} relevant result, with ranks strictly increasing with increasing j. The innermost summand is the precision of the result set stopping at the j^{th} relevant result.

Mean reciprocal rank and mean average precision are both metrics that calculate an average over queries. However, these averages can be misleading if there is imbalance of underlying conditions. As an example, consider a taste test between two competing products with 100 test subjects, of which 30 are female and 70 are male. An (unweighted) average would be shifted towards the male perspective, which may not be desirable.

In such cases, it can be helpful to compute a micro-average and a macro-average, which are both weighted averages of averages. Given the a set $\mathcal{X}$ to

average, the micro-average is computed as:

$$\overline{micro} = \sum_{i=1}^{n} \frac{|\mathcal{X}_i|}{|\mathcal{X}|} \left(\sum_{x \in \mathcal{X}_i} x \right) \tag{2.11}$$

where $\mathcal{X}$ is subdivided into n disjoint sets according to the underlying condition of interest. As each inner mean is weighted by the size of the corresponding subset, the micro-average is the same as the (unweighted) average over the entire set. In contrast, the macro-average is defined as:

$$\overline{macro} = \sum_{i=1}^{n} \frac{1}{n} \left(\sum_{x \in \mathcal{X}_i} x \right) \tag{2.12}$$

with the same quantities as the micro-average.

Observed differences between test statistics are sometimes the result of random chance and so do not show a meaningful difference. Thus, it is necessary to calculate statistical significance to decide if an apparent result can be trusted. Many popular statistical tests make assumptions that hold only given a particular underlying distribution or a large enough sample. In contrast, the randomization test is a simulation-based approach that does not make such assumptions [71]. The randomization test evaluates a null hypothesis that two sets of observations come from the same distribution, and so the observed difference is an outcome of chance. It tests this hypothesis by randomly reassigning the observations to each group many times, and counting the number of times the resulting difference's magnitude was at least as large as that observed. This number is commonly referred as the p-value and represents the chance the null hypothesis is correct. Typically, the null hypothesis is rejected in favor of the alternate hypothesis (in this case, that two sets of observations are drawn from different distributions) when the p-value is no greater than 0.05, a standard we adopt throughout this

dissertation.

Chapter 3

Leveraging Multiple Criteria in Item Retrieval

In this section, we explore how multi-criteria decision making principles could be applied to item retrieval problems. In the first part, we compare several representative multi-criteria decision making algorithms [82]. We focus solely on how best to apply ratings from differing criteria, without investigating how to derive criteria rating from a user's query and item attribute values. In the latter part, we introduce the user's query to our solution framework [84]. We formulate a simple query model that is can be readily used to translate item attribute values into estimates of criteria ratings. We evaluate this retrieval model against retrieval paradigms commonly in use at the time of writing, as well as several similar approaches proposed in the literature.

3.1 Leveraging Criteria in Retrieval Problems

The operations research community has extensively studied the use of multiple criteria in multi-criteria decision making (MCDM), also known as multi-criteria

decision analysis, which aids decision makers in making difficult choices evaluated under potentially conflicting criteria. A variety of methods have been developed for MCDM, ranging from straightforward single formula methods to more complex methods that use multiple stages to induce a ranking. Though these techniques are designed for decision analysis, it is worth exploring how can they be adapted to the ranking problem à la information retrieval. As a starting point, we have applied MCDM techniques to two different retrieval applications: air travel booking and information filtering (of news articles).

Airline tickets are potentially well-suited to multi-criteria approaches because it is not content-based and (possibly as a result) does not have a single criterion that is likely to dominate the user rankings of available tickets. The criteria are also a mix of objective criteria (e.g., cost) and subjective criteria (e.g., desirability of destination), with both often at odds. On the other hand, the information filtering task may not be well suited to MCDM techniques. In this domain, one criterion (subject relevance) is consistently the strongest indicator of user interest, and all of the criteria are highly correlated. Therefore, there may be little benefit in considering additional criteria in information filtering.

To evaluate the potential of MCDM in item retrieval, we adapted several MCDM algorithms [75]. The most familiar of these is sorting into a lexicographic order, consistent with *non-compensatory* preferences, which indicates that no criterion rating difference of a lower order can compensate for even the smallest criterion rating difference of a higher order. The other two algorithms support *compensatory* preferences. The simpler of these two algorithms is the weighted sum: the score for each option is linear combination of criteria weights and criteria ratings. The second method, ELECTRE II, is an outranking method which orders the different options directly by combining combining partial orderings with

progressively more relaxed consistency conditions.

3.1.1 Multi-Criteria Based Ranking Methods

Casting the item retrieval problem as a multi-criteria decision making problem allows us to bring to bear the large amount of operation research that has been developed for such problems. To illustrate the idea, we applied three MCDM algorithms for ranking: lexical sort, weighted sum and ELECTRE II. Our selection of algorithms is not meant to be exhaustive, but representative of the varied type of multi-criteria ranking methods, and quite diverse.

Lexical Sort

Of all the multi-criteria models, lexical sort is the most readily available method today, easily accessibly in database queries and e-commerce search, often in a restricted form. The user specifies a sort order (ascending or descending) on criteria in turn, with subsequent criteria used only to break ties. As such, the model is *non-compensatory*, since no difference in later orders can override even the slightest difference in an earlier order. In many current e-commerce sites, this is restricted to a single sort order.

Weighted Sum

The second MCDM algorithm we evaluate has a simple formulation, ranking the items by a score from a linear combination of criteria ratings and weights. By convention, the criteria ratings range from zero to one, with one being the most desirable, and the criteria weights sum to one. For the airline ticketing domain, we have only attributes rather than criteria ratings, so we assume that the criteria correspond to the ticket attributes. We also have the slight complication that

for most attributes (such a price), lower values are more preferable, leading to a somewhat more complicated subutility function. For the j^{th} criteria, we define $g_j()$ to be the normalized subutility function:

$$g_j(d_{ij}) = \begin{cases} \frac{d_{ij} - \perp_j}{\top_j - \perp_j} & \text{prefer high} \\ \frac{\top_j - d_{ij}}{\top_j - \perp_j} & \text{prefer low} \end{cases} \tag{3.1}$$

where d_{ij} is the rating on the j^{th} criterion for candidate $\mathbf{D}_i$, and $\top_j$ and $\perp_j$ are the greatest and least rating on criteria j. These normalized subutilities are then combined in a linear combination:

$$f(\mathbf{D}_i) = \sum_j w_j \times g_j(d_{ij}) \tag{3.2}$$

where w_j is the weight of criteria j, with candidates ranked in descending order of this score. In general, a user could directly specify a real-valued weights for the criteria, something we explore in our next experiment. However, for this study we adopt a simpler approach: given a ranking of criteria (with rank of one most important), we assign weights as follows:

$$w_j = \frac{2 \times (m + 1 - r_j)}{m \times (m + 1)} \tag{3.3}$$

where m is the number of criteria and r_j is the rank of the j^{th} criterion. This weighting scheme results in a constant difference between consecutively ranked criteria, with the weight of the highest ranked criterion m times that of the lowest ranked criterion.

ELECTRE II

Though it uses the same normalized scores and weights as WeightedSum, the outranking ELECTRE II is considerably more complex than weighted sum. We

adapt the ELECTRE II formulation of Lamontagne and Abi-Zeid [46] for item retrieval, and as such, our formulation differs slightly from the original ELECTRE II.

ELECTRE II defines several logical propositions, and uses these to establish that an item should outrank another. In all, ELECTRE II defines five logical propositions on pairs of items, and from these, arrives at two item rankings which are combined into a final ranking. ELECTRE II apparently seeks to avoid items that are particularly poor in some way rather than seeking those that greatly excel on some criteria.

The base proposition is a weighted vote for item $\mathbf{D}_i$ over item $\mathbf{D}_k$:

$$V(\mathbf{D}_i, \mathbf{D}_k) \equiv \sum_j w_j \times sgn\left(g_j(d_{ij}) - g_k(d_{ik})\right) > 0 \tag{3.4}$$

where sgn is the sign function and other parameters are defined as in Eqs. 3.1 and 3.3. In essence, the total weight of criteria that $\mathbf{D}_i$ rates higher than $\mathbf{D}_k$ must exceed the total weight of criteria where $\mathbf{D}_k$ rates higher than $\mathbf{D}_i$ in order to win the vote.

Next, we define two acceptance propositions, the first being:

$$A_\theta(\mathbf{D}_i, \mathbf{D}_k) \equiv \sum_j w_j \times [g_j(d_{ij}) \geq g_j(d_{kj})] > \theta \tag{3.5}$$

where θ is a threshold value, with higher values providing a more stringent test, [ ] is the Iverson bracket, and the other parameters defined as before. In essence, this proposition sets a minimum total weight of criteria that $\mathbf{D}_i$ is at least as good as $\mathbf{D}_k$ to be acceptable.

The second acceptance proposition is:

$$B_\lambda(\mathbf{D}_i, \mathbf{D}_k) \equiv \sum_j \Big[g_j(d_{kj}) - g_j(d_{ij}) > (1 - w_j)^\lambda \Big] = 0 \tag{3.6}$$

where λ is another threshold value, [ ] is the Iverson bracket, and the other parameters are defined as before. This proposition sets a minimum standard for each criteria such that $\mathbf{D}_i$ cannot be accepted over $\mathbf{D}_k$ if $\mathbf{D}_i$, $\mathbf{D}_k$ is that much better rated. The original ELECTRE II used the same threshold for all criteria regardless of the criterion weight, but in our experimentation we found that incorporating the weights lead to better performance, leading us to the formulation above.

From these three propositions, ELECTRE II defines a strong and weak preference. The strong preferences is defined as:

$$\begin{aligned} P_s(\mathbf{D}_i, \mathbf{D}_k) \equiv & V(\mathbf{D}_i, \mathbf{D}_k) \wedge \\ & [(A_{\theta_1}(\mathbf{D}_i, \mathbf{D}_k) \wedge B_{\lambda_1}(\mathbf{D}_i, \mathbf{D}_k)) \vee \\ & (A_{\theta_2}(\mathbf{D}_i, \mathbf{D}_k) \wedge B_{\lambda_2}(\mathbf{D}_i, \mathbf{D}_k))] \end{aligned} \tag{3.7}$$

and the weak preference is defined as:

$$P_w(\mathbf{D}_i, \mathbf{D}_k) \equiv V(\mathbf{D}_i, \mathbf{D}_k) \wedge A_{\theta_3}(\mathbf{D}_i, \mathbf{D}_k) \wedge B_{\lambda_3}(\mathbf{D}_i, \mathbf{D}_k) \tag{3.8}$$

where θ_1, θ_2, θ_3, λ_1, λ_2, λ_3, were somewhat arbitrarily set to 0.87, 0.75, 0.63, 2, 2, and 3, in our experiments respectively, with these values giving better performance than others we had tried.

The strong and weak preference functions are used to create two rankings, a direct ranking and an inverse ranking, which are combined to create the final

ranking. For the direct ranking, the highest rank consists of all items that are not outranked by any item outside the highest rank (i.e., no item outside the highest rank is strongly preferred over an item in the highest rank). Should there be more than one option in the highest rank, the weak preference breaks ties. This process continues with the second-highest rank, which contains options not outranked by any other unranked option, and so on, until all options are ranked in the direct ranking. The inverse ranking works in a similar fashion but in reverse: first the lowest rank is selected, which consists of items that do not strongly outrank items outside the lowest rank, applying the weak preference in an equivalent manner to break ties and iterating as before until the complete inverse rank is created. We use the mean of the direct and inverse ranking as our final rank for each option.

3.1.2 Retrieval Domains and Criteria

For the ticketing experiment, we culled information from several online databases [14, 15] to develop a representative set of ticketing options and expected delay profiles, with just over 5000 tickets used in our study. The following criteria were identified, with the criterion shorthand in parentheses: distance to the desired origin of the flight[1] (ODist); Distance to the desired destination of the flight (DDist); the price of the fare (Cost); the expected flight time (FTime); the number of connections (Legs); the expected delay (FDelay); popularity, defined as the number of tickets sold for this final destination (Desire); stopover popularity (Stop), calculated like the prior criterion, but for the connection airports (presumes sightseeing at the connection is possible).

In addition, we generated five tasks for five fictional persons, loosely inspired by real situations with which we were familiar, as shown in Table 3.1. We chose cases

[1] Nearby airports were also included.

Table 3.1: Scenarios in the ticketing experiment.

Task	From	To	Criteria Order	Scenario
T1	SJC	ROC	Cost ODist DDist	Student returning home
T2	DCA	ANC	ODist DDist FTime Fdelay	Scientist giving talk
T3	GFK	n/a	Cost ODist Legs	Retired man on vacation
T4	GFK	n/a	Desire ODist Fdelay Ftime	Retired woman on vacation
T5	LAX	BOS	ODist DDist Stop Cost	Professor attending conference

that we thought would cover a range of tasks, but does not necessarily match the distribution of actual consumer travel patterns. Three subjects served as judges for our search results, marking tickets as relevant or not relevant, according to the task. The subjects fly regularly but are not heavy travelers (around five trips per year). Each subject was given a description of the fictional person, the trip, and the listed criteria, and then allowed to interpret the appropriateness for each ticket for the stated need. Though real trips would typically also involve return flights, in part to reduce the burden on our test subjects, we treated the trips as if they were one way. Each subject evaluated every ticket as a match (or not) for the given task.

For the news filtering experiment, we used a data collected from a previous study: for more information on the dataset see [91]. In that study, approximately 20 subjects rated news articles on several criteria from a corpus of almost 9000 articles. On this data set, the following criteria are included: novelty, authority, readability, and relevancy to the news article category. This data had been gathered previously and the subjects did not weight the criteria, so we give each criterion equal weight. We treated the highest rated articles as relevant, with the goal of the system to rank all items evaluated by each subject as if they were returned from a single query.

3.1.3 Evaluation of Multi-Criteria Methods

Table 3.2: MCDM Algorithms Results

Dataset	Subjects	Queries	WS	ELECTRE II	1-Sort	All-Sort
Tickets	3	15	**0.59***	0.51	0.15*	0.34*
News	23	23	0.54	0.53	0.45*	**0.58**
Both	26	38	**0.56**§	0.52*	0.34*	0.49

Table 3.2 shows the results of our MCDM-based studies, as evaluated on mean average precision (MAP), with the best performer bolded. In addition to a full lexical sort (All-Sort), we also include for comparison a sort on only the top criterion (1-Sort) as most users start with simpler queries. We use a randomization test [71] to evaluate statistical significance with a million samples per test. The null hypothesis is that a different user need (as scenarios for the ticketing experiment, subjects in the news filtering experiment) would have been equally likely to have flipped the observed difference among methods, so we randomly swap results[2] and observe how often our randomly generated difference is of equal or greater magnitude (regardless of sign), with significance at the standard p-value of 0.05, although these are less common given the sample size. Nonetheless, the weighted sum (WS) method appears to outperform the others, with statistical significance against all but ELECTRE II marked by asterisk (*), and against all but All-Sort marked by the double S (§). All differences versus 1-Sort are also significant. All-Sort outperformed the weighted sum on news filtering, but this difference was not statistically significance. Even though the observed difference between weighted sum and all-sort on the combined results of the experiment were greater than that with ELECTRE II, the result was not significant as the difference stems from fewer user needs (mostly in the ticketing domain).

[2]These random "swaps" either swap results or leave as-is with equally likelihood.

The news filtering experiment used equal weights, but lexical sort requires some prioritization of criteria, so we used the best criteria ordering over all the news data as a whole. This may have given the lexical sort an unrealistic advantage over the uniformly weighted algorithms in the news filtering domain. Despite its complexity, ELECTRE II did not perform as well as the simpler weighted sum algorithm. It may be the the domains chosen were not suited to this algorithm; ELECTRE II is designed to find compromise solutions in the presence of conflicting criteria, which was not particularly problematic in these applications.

3.2 Translating Queries to Criteria Ratings

Multi-criteria decision making models usually assume that the user has provided a rating for each option on every criterion. This assumption is reasonable given a small number of options and a critical decision, but is generally not a good fit for item (or information) retrieval problems. Therefore, a practical retrieval system must induce criteria ratings from its representation of the user's need. In this section, we explore translating a user's query to criteria ratings.

3.2.1 A Unifying Relevance Model

We make several assumptions in order to adapt multi-criteria decision making algorithms to query-based item retrieval. First, as mentioned previously, we assume that the criteria correspond to attributes of the items. Second, we assume that there is some single attribute value for each attribute of interest that yields maximal utility, i.e., a most desirable value, similar to TOPSIS [76]. This suggests the form of the queries, i.e., that users will provide desirable attribute values on a subset of the attributes. Finally, for simplicity, we assume a linear relationship

between the subutility of a criterion and the absolute difference of the attribute and desired value.

According to multi-attribute utility theory (MAUT) [28], 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 preference for a set of values is unaffected by the values of other attributes. As an example, this would mean if red wine is preferred over white wine for two otherwise identical meals, then red wine will always be preferred over white regardless of the meal's remaining components. This assumption entails that the underlying utility function must take a multiplicative form:

$$f(\mathbf{Q}, \mathbf{D}_i) = \frac{\prod_j [1 + cw_j \times g_j(q_j, d_{ij})] - 1}{c} \tag{3.9}$$

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, $\mathbf{D}_i$ 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, and c is a constant greater than -1.

Expanding Eq. 3.9 for a particular instance can enhance understanding. Consider the two attribute case:

$$\begin{aligned} &\frac{[1 + cw_1 \times g_1(q_1, d_{i1})] \times [1 + cw_2 \times g_2(q_2, d_{i2})] - 1}{c} \\ &= w_1 \times g_1(q_1, d_{i1}) + w_2 \times g_2(q_2, d_{i2}) + cw_1w_2 \times g_1(q_1, d_{i1})g_2(q_2, d_{i2}) \end{aligned}$$

Each weighted subutility evaluation appears independently, as well as a combination of the two. The function of c is also more clear; when $c = 0$, the subutility of each criterion contribute independently to the overall utility. In that case, the utility function reduces to a linear combination:

$$f(\mathbf{Q}, \mathbf{D}_i) = \sum_j [cw_j \times g_j(q_j, d_{ij})] \tag{3.10}$$

Interestingly, although Eq. 3.9 is only a linear combination when $c = 0$, there is an closely related linear combination that will produce the same ranking. Consider the following, which we get by multiplying Eq. 3.9 by c, adding 1, and taking the logarithm:

$$\begin{aligned} \log\left(1 + c \times f(\mathbf{Q}, \mathbf{D}_i)\right) &= \log\left(\prod_j [1 + cw_j \times g_j(q_j, d_{ij})]\right) \\ &= \sum_j \log\left(1 + cw_j \times g_j(q_j, d_{ij})\right) \end{aligned} \tag{3.11}$$

Multiplying by a positive constant, adding a constant, and taking the logarithm are all operations which maintain the original order. However, c may be negative or zero, though Eq. 3.9 is already a linear combination when $c = 0$. When $c < 0$, ranking by Eq. 3.11 would invert the correct order. This can be remedied by ranking by the negation of Eq. 3.11 when $c < 0$, or equivalently, multiplying by $\text{sgn}(c)$:

$$\begin{aligned} \text{sgn}(c) \log\left(1 + c \times f(\mathbf{Q}, \mathbf{D}_i)\right) &= \text{sgn}(c) \log\left(\prod_j [1 + cw_j \times g_j(q_j, d_{ij})]\right) \\ &= \text{sgn}(c) \sum_j \log\left(1 + cw_j \times g_j(q_j, d_{ij})\right) \\ &= \sum_j \text{sgn}(c) \log\left(1 + cw_j \times g_j(q_j, d_{ij})\right) \end{aligned} \tag{3.12}$$

With some abuse of the theory, we can define a new subutility function $g_j'(q_j, d_{ij}) = \text{sgn}(c) \log\left(1 + cw_j \times g_j(q_j, d_{ij})\right)$ when $c \neq 0$, or $g_j'(q_j, d_{ij}) = w_j \times g_j(q_j, d_{ij})$ when

$c = 0$, and rank the results by $\sum_j g_j'(q_j, d_{ij})$, keeping in mind that though $g_j'(q_j, d_{ij})$ deviates from the latent subutility function, it nonetheless produces in an equivalent ranking. This also gives more insight into the constant c, with values less than zero producing convex shapes, and those above producing concave. As such, we assume the underlying utility function must be linear combination of ratings, yielding our base utility function:

$$f(\mathbf{Q}, \mathbf{D}_i) = \sum_j w_j \times g_j(q_j, d_{ij}) \tag{3.13}$$

where quantities are the same as defined for Eq. 3.9, with items ranked in order of decreasing utility.

Nonetheless, in terms of the exact form of the subutility function, MAUT offers no further guidance. Subutility evaluations are typically given 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:

$$g(q_j, d_{ij}) = \left(1 - \frac{|q_j - d_{ij}|}{\max\left(|q_j - \perp_j|, |q_j - \top_j|\right)}\right) \tag{3.14}$$

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. 3.13. 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.

3.2.2 Experimental Design

We return to the domain of booking airline tickets for our experiment, as such items are well described by their attribute values, have attribute values that are not highly correlated, and does not require specialized knowledge from the participants. In all, each ticket has nine attributes: price, as well as departure time, arrival time, connections and duration for both outbound and return flights. Our interface largely mimicked that of a popular ticketing website (see Fig. 3.1), with the query mechanism ("Refine Filters" in the right interface) changing per the model, as detailed below.

Live Models

Many of our models used the same query parameters, or subsets of those parameters. This allowed us to limit the number of query interfaces and retrieval models implemented in the live experiment to four, randomly chosen.

SortedBoolean We reviewed the ticketing Web sites Expedia[3], Travelocity[4], Orbitz[5] and Priceline.com[6] to develop our the *SortedBoolean* model. Though

[3]http://www.expedia.com
[4]http://www.travelocity.com
[5]http://www.orbitz.com
[6]http://www.priceline.com

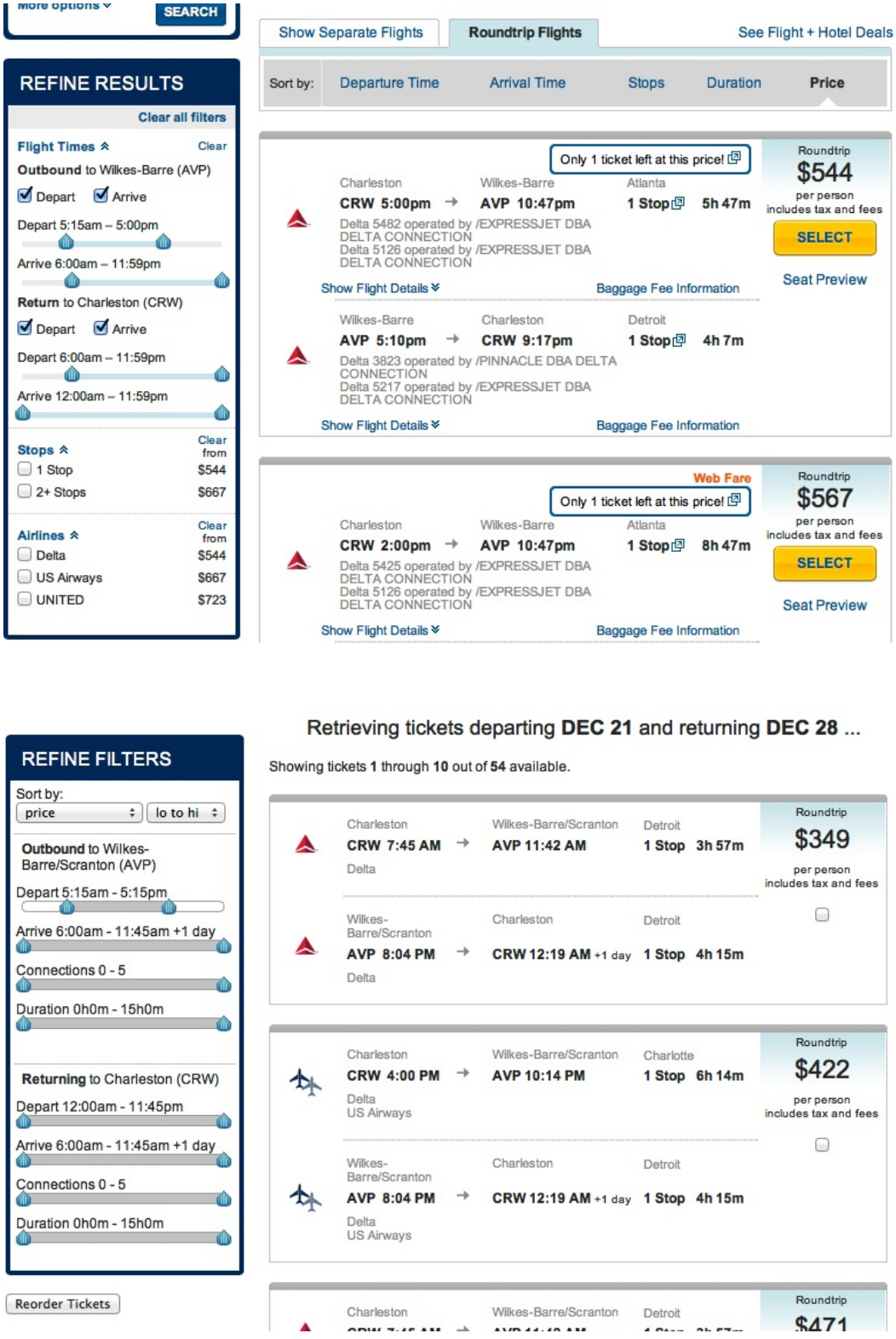

Figure 3.1: Commercial Website (top) versus User Study Interface (bottom)

there were minor differences among the Web sites, at the time we developed our user study their capabilities were largely the same. Most importantly, they allowed users to restrict the result set by limiting the returned tickets to ranges of attributes (with the exception of price), much like a boolean retrieval model. In addition, the tickets could be ranked by by a single attribute, chosen from a limited set of attributes[7]. We developed the emph-SortedBoolean baseline to match both the functionality and look-and-feel of one of the Web sites closely. The query interface is shown in Figure 3.2.

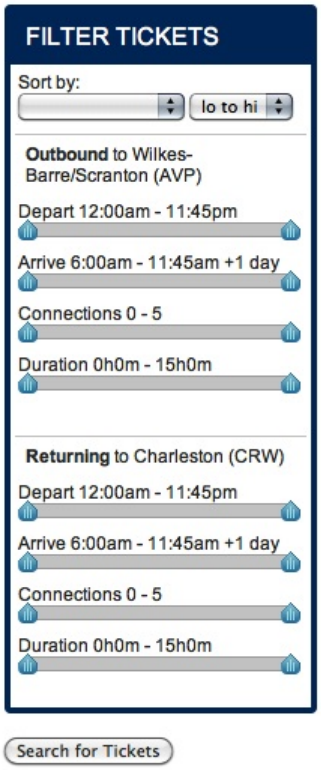

Figure 3.2: Query Interface for SortedBoolean

Lexical As an alternative baseline to the *SortedBoolean* model, we developed a lexical sorting model. Though not in use for booking tickets the time of writing, sorting is a common paradigm for ordering results in a database

[7]Namely, total travel duration, total connections, outbound departure or arrival time.

or spreadsheet. It is also differs from the *SimpleMAUT* approach as it is a *noncompensatory* model, like the models of preference databases, as a more attractive option on a less important attribute cannot compensate for a a less attractive option on a more important attribute, no matter how great or small the relative differences are. In our interface, the test subject can choose up to four attributes to sort the tickets, each in either descending or ascending order. The items are first ordered by the first specified attribute, with ties broken by the second specified attribute, and so on. The query interface is shown in Figure 3.3.

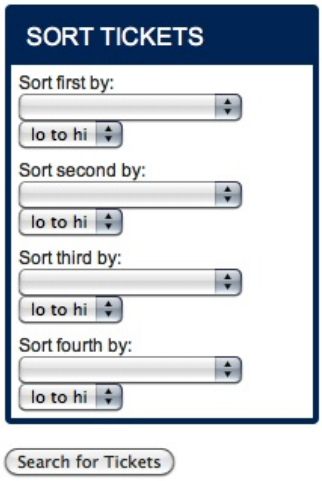

Figure 3.3: Query Interface for Lexical

SimpleMAUT This is our novel approach, described in detail in Section 3.2.1. The query interface is shown in Figure 3.4. The *SimpleMAUT* retrieval model solicits a query consisting of two components: an "ideal" item, which is the item the user would most like to find, and a set of priorities over the attributes of this item. We use the 1-9 priority scale as recommended by Saaty [68] for weights, with the addition of an "ignore" value to indicate that the attribute is not of interest to the test subject (essentially giving the

criterion a 0 priority). These priorities, re-scaled onto a 0-1 scale, become the coefficients of the linear combination in base utility Eq. 3.13. For this model and those to be described later, we define division by zero to be one when the numerator is zero, infinity otherwise.

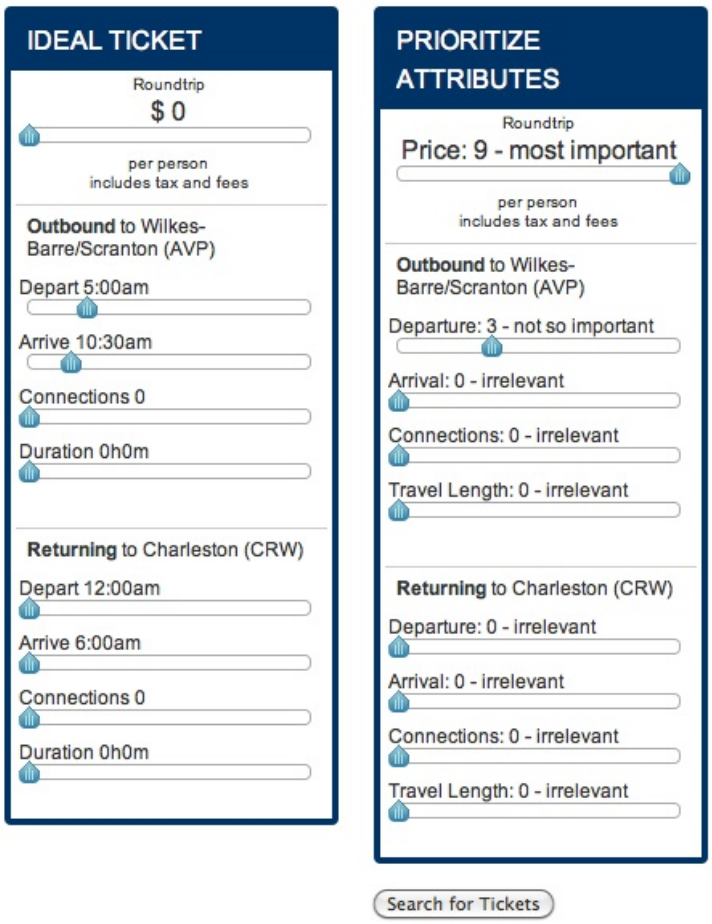

Figure 3.4: Query Interface for SimpleMAUT

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:

$$f(\mathbf{Q}, \mathbf{D}_i) = \sum_j t_j |q_j - d_{ij}| \tag{3.15}$$

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

parameters defined as before. The query interface is shown in Figure 3.5.

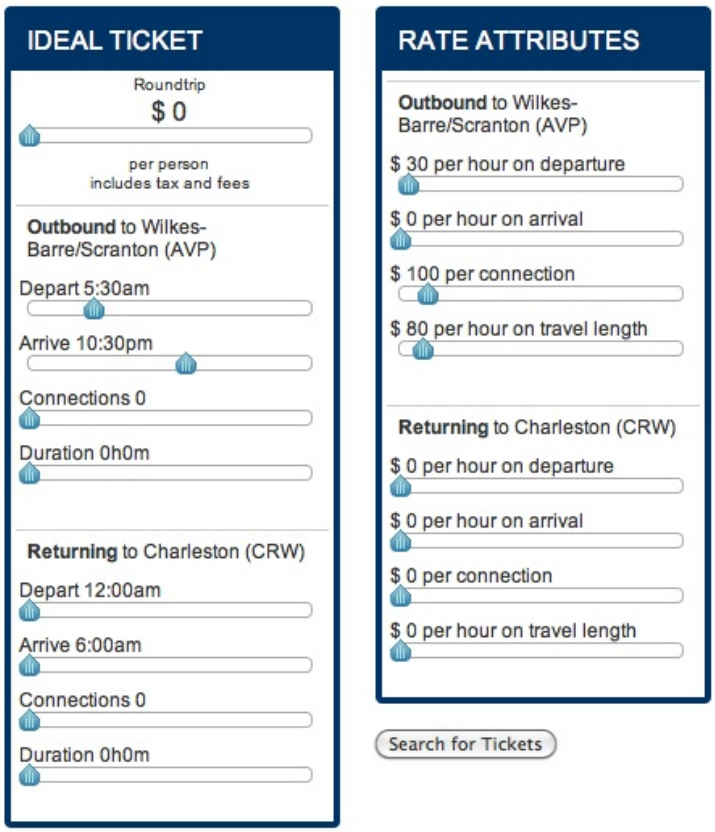

Figure 3.5: Query Interface for Tradeoff

Post Study Models

The remainder of our models use the same queries as the above, so we could apply them to the queries and choices gathered in our user study. Variables are defined as defined for *SimpleMAUT* (see Section 3.2.1) unless otherwise specified.

AIMQ This model estimates relevance using a different normalization and global weights derived from functional dependencies (see [56, 57] for details), rather than user-specified weights. Items are ranked by decreasing score order, where the score of d_i is defined as:

$$f(\mathbf{Q}, \mathbf{D}_i) = \sum_{j \in \mathbf{Q}} w_j \left(1 - \min\left(1, \frac{|q_j - d_{ij}|}{q_j}\right)\right) \tag{3.16}$$

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 [2]. They used an inverse document frequency (IDF) term for weighting, defined below for query element q_j as:

$$w_j = \log\left(\frac{n}{\sum_{k=1}^{n} \exp(-\frac{1}{2}\left(\frac{d_{kj}-q_j}{h_j}\right)^2)}\right) \tag{3.17}$$

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:

$$f(\mathbf{Q}, \mathbf{D}_i) = \sum_{j \in \mathbf{Q}} w_j \exp\left[-\frac{1}{2}\left(\frac{d_{ij} - q_j}{h_j}\right)^2\right] \tag{3.18}$$

with items ranked by decreasing score.

CQAds Like *AIMQ*, *CQAds* [65] estimates relevance in a similar fashion but with a different normalization, and without attribute-specific weights. In our implementation of *CQAds* scoring, items are ranked by decreasing score order, where the score of d_i is

$$f(\mathbf{Q}, \mathbf{D}_i) = \sum_{j \in \mathbf{Q}} \left(1 - \frac{|q_j - d_{ij}|}{R_j}\right) \tag{3.19}$$

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.

LexPref This is an alternative method of lexical sorting, by decreasing absolute difference from the query value, in order, instead of by ascending or descending order. This is our implementation of the cascading ranking in preference databases.

ProspectTheory We adapt prospect theory [41], to use as an alternative subutility function, combining in linear combination like many of the models:

$$f(\mathbf{Q}, \mathbf{D}_i) = \sum_{j \in \mathbf{Q}} w_j \theta_j |q_j - d_{ij}|^{\alpha} \tag{3.20}$$

where α is a constant controlling the curve of the subutility, $\mathbf{\Theta}$ is an additional weighting adjustment, defined as:

$$\theta_j = \begin{cases} 1 & \text{for } \gamma_j d_{ij} \geq \gamma_j q_j \\ -\lambda & \text{for } \gamma_j d_{ij} < \gamma_j q_j \end{cases} \tag{3.21}$$

where λ controls the rate of deprecation for poor outcomes, and $\mathbf{\Gamma}$ controls whether low or high values are preferable for the particular attribute. For $\mathbf{\Gamma}$, only the sign matters, which we define as positive for departures (later departures are preferable, with subsequent arrival times modeled separately) and all others as negative. We use the parameter values originally derived by Tversky and Kahneman, $\alpha = 0.88$ and $\lambda = 2.25$, as reported in [59].

Skyline The skyline operator does not assume a particular scoring function; rather, under the assumption of monotonicity, it returns the Pareto set of all non-dominated results. In this domain, a ticket x dominates y if x's

attributes are at least as good as y's and better in at least one case. We define "better" as a lower absolute difference from the query value when the corresponding priority was nonzero, ignoring the attribute when the priority is zero. We use the skyline operator progressively to rank the results (listing non-dominated tickets first, then listing tickets non-dominated tickets in the remaining set, etc.), with tickets in each equivalence class ranked arbitrarily.

VAGUE The *VAGUE* framework [54] provides a "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:

$$f(\mathbf{Q}, \mathbf{D}_i) = \sqrt{\sum_j \left[w_j \left(\frac{|q_j - d_{ij}|}{\sigma_j} \right) \right]^2} \tag{3.22}$$

with items ranked by increasing score, using user-supplied weights like *SimpleMAUT*.

LearnMAUT Finally, we also explore if it is possible to improve performance by learning for a parameterized version of *SimpleMAUT* model as our exemplar. Items are ranked by descending score order, where the score of d_i is given by:

$$f(\mathbf{Q}, \mathbf{D}_i) = \sum_j w_j \times \left[1 - \frac{\max\left(\lambda_j \left[q_j - d_{ij}\right], \gamma_j \left[d_{ij} - q_j\right]\right)}{\max\left(|q_j - \bot_j|, |q_j - \top_j|\right)} \right] \tag{3.23}$$

where $\lambda_j, \gamma_j \in [0, 1]$ allow for different weights above and below the query point, and with other variables are defined as before. We used a listwise

learning-to-rank approach to learn the parameters of the model. Our approach uses gradient descent on a smoothed version of the mean average precision metric, as suggested by several authors [87, 20, 64]. The core insight is to re-express the rank of an item by the sum of indicator functions of greater score, comparing the item's score to the others in the corpus. These indicator functions are then approximated by a differentiable function, with our formulation following that of Qin, Liu and Hang [64]. We use 20 fold cross-validation to evaluate the learned model, with each fold corresponding to the queries for a particular scenario. The model is trained for each fold by using the data from the other 19 folds. This prevents results from the same scenario from influencing the model, matching a realistic learning situation. However, given the difference in scenarios and the limited number of responses, the learning problem is fairly difficult.

LearnSymmetricMAUT For comparison, this is the symmetric version of *LearnMAUT*, where the subutility function is symmetric on both sides of the query value (i.e., $\forall j : \lambda_j = \gamma_j$). We include this version to isolate the effect of adjusting weights symmetrically versus asymmetrically, per attribute.

Data Used

We wanted to make our scenarios as representative as possible, so we reviewed several relevant sources. We restricted our experiment to U.S. domestic travel, using a ten percent sample of tickets gathered in 2006 [15] to model the distribution of travel patterns. For each scenario, we randomly selected a ticket from this sample and used its origin and destination for the scenario. Unfortunately, other recorded aspects of the ticket were not useful, such as schedule, which was coarsely recorded in a resolution of a quarter year. To compensate, we retrieved tickets from

Expedia.com (using dates of our choosing) with the same origin and destination. These tickets were retrieved during the later months of 2011, with approximately 60 tickets retrieved for each scenario.

We consulted a survey from more than 26,000 U.S. households to capture who travels by air and the reasons why [16]. We created twenty scenarios following their breakdown, with 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. For the business trips, we would also randomly sample time constraints from 9 AM to 4 PM (for meeting times); we also included explicit time constraints one of the pleasure scenarios. Depending on the scenario, these constraints would range from trivial to more restrictive, and we removed constraints that universally satisfied. Conversely, tickets that did not match the constraints were not removed from the corpus. For half of the remaining scenarios, we listed other criteria (such as "get home early"), while leaving the others open-ended. Finally, we also sampled demographic information from the same survey (gender, age, income) to give more context.

Subject's tasks and rewards

We developed a reward structure to motivate test subjects to take the task seriously and put effort into choosing the best tickets. As previously mentioned, Amazon Turk workers were our test subjects [3]. These workers are paid for their efforts, and may receive additional payment at the discretion of the requester. We used this bonus mechanism to entice good work. Work may also be rejected by the requester. In our case, we also restricted the experiment to workers with high

completed work acceptance rates (95% or better).

We developed a game where workers would alternate between two ticket selection roles. In the first role, workers would play the role of a "ticket agent". Their task was to read the scenario and choose three tickets that they thought would be most likely to be selected by the client. The workers playing "ticket agent" would use the retrieval models in our user study. The other role was that of the "client", who would be presented a smaller subset of tickets, choosing only one. In this role, the tickets were simply listed in random order with no search capability. Each worker could complete a scenario in only one of the roles.

Workers would be randomly matched with another worker: ticket agent to client, and client to client. The client matched with a ticket agent would see a subset of tickets that included the three chosen by the ticket agent; if the client chose one of the ticket agent's suggestions, the ticket agent would be paid the bonus. (The listing does not indicate which tickets were suggested by the ticket agent.) Likewise, the client would be matched with another client who reviewed the same subset of tickets; if their selections matched, the client would receive the bonus. In short, we pay the workers to check each other's work. The responses of the client were only used for this mechanism and are not factored into our results.

Regardless of role, the workers were paid a modest base pay and given a bonus twice this amount when their answers match, as described above. The large bonus relative to base pay was designed to motivate the workers to try hard. However, since workers were randomly matched, most likely there were some inequities (i.e., some workers with good answers weren't rewarded, and some workers who didn't try hard got lucky). To eliminate noisy ticket agent responses, we filtered out roughly half, with 554 responses retained. For each scenario, we calculated the median probability of being matched with another ticket agent who

selected at least one ticket in common. We discarded all responses that fell below this median. The two groups (discarded and preserved) showed a statistically significant difference on all our evaluation metrics consistent with more effort according to a randomization test (also described below).

Evaluation Metrics

We compare the performance of the different retrieval models by several standard information retrieval quality metrics. We are focused on ad hoc retrieval instead of interactive retrieval, and so we calculate these metrics only on the first query issued. We use the tickets chosen by the test subject as the indication of relevance for mean average precision (MAP). However, a test subject may be likely to choose a higher ranked item when utility is roughly the same. This is irrelevant when comparing the retrieval models used directly by test subjects ("Live" models), but could bias the results when comparing retrieval models in our post hoc analysis. To compensate in that case, 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, evaluating each query separately on each set of chosen tickets (except from the set chosen by the querier).

In addition, we also calculate several usage and user evaluation statistics for the live models:

Mean Queries This includes the initial query and any subsequent query revisions.

Mean Pages The number of pages viewed by the subject. Each page contains ten tickets, except for the last page which may have fewer tickets.

Mean Time The total time spent on the task after issuing the initial query. We excluded time spent issuing the initial query because we solicited information that was unused by the *SortedBoolean* and *Lexical* models.

Mean Qualitative Each test subject was asked to evaluate the following statement: "If my favorite ticketing Web site supported it, I would use this method to search for tickets," with responses ranging from 1 (strongly disagree) to 5 (strongly agree).

We had an unbalanced distribution of retrieval models among the scenarios after running the experiment and discarding responses with low inter-rater agreement (see above). Some scenarios were more difficult than others, and so models with a disproportionate number of responses from such scenarios could have artificially lower performance measures. We calculate two means for all metrics to compensate. 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 the unbalance in scenario 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 [71] in two ways to calculate statistical significance. The first method is used when comparing responses by different subjects in the live experiment; 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 when comparing different models on the same response: here our null hypothesis is each method was equally likely to have produced the

Table 3.3: MAP of Live Models

Model	micro	macro
SortedBoolean	0.377	0.367
Lexical	0.435	0.418
Tradeoff	0.387	0.380
SimpleMAUT	**0.542**†	**0.526**†

Table 3.4: Community-evaluated MAP

Model	micro	macro
AIMQ	0.373	0.305
Autorank	0.452	0.381
CQAds	0.441	0.396
LexPref	0.441	0.363
Skyline	0.437	0.360
ProspectTheory	0.418	0.361
VAGUE	0.428	0.391
SimpleMAUT	0.483†	0.418†
LearnMAUT	**0.532**†	**0.461**†
LearnSymmetricMAUT	0.500†	0.425*

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.

Table 3.5: Community-evaluated MAP, comparing user weights

	micro		macro	
	User	Binary	User	Binary
Model	Weights	Weights	Weights	Weights
LexPref	**0.441**†	0.393	**0.363**†	0.321
ProspectTheory	0.418	**0.437**†	0.361	**0.370**†
VAGUE	0.428	**0.429**	0.391	**0.397**
SimpleMAUT	0.483	**0.484**	**0.418**	0.417

3.2.3 Results

Table 3.3 shows the results for self-evaluated retrieval quality metrics (i.e., using that test subject's choices) for the various models, with the overall best performance bolded. The *SimpleMAUT* model performed best overall, a statistically statistical difference. On the other hand, the *SortedBoolean* model performed the worst, despite being the model in most common uses today, although the differences were not always statistically significant. For the community-evaluated MAP (table 3.4), the MAUT models performed best overall, with statistically significant difference versus lower scoring models indicated by the dagger (†) and likewise excepting against *SimpleMAUT* indicated by the asterisk (∗). We include *SimpleMAUT* as a reference point, and to show the difference in scoring of the querier's MAP (given in table 3.3) and the community's MAP. The relative success of the learning methods show that a global weighting of criteria can lead to better results. The improvement of *LearnMAUT* over it symmetric cousin leads us to believe that the subutility functions are not symmetric, i.e., exceeding the desired value is not the same as falling short. This is not entirely surprising, particularly when considering attributes such as price. However, it is not necessarily true that more (conversely, less) is better: the only model with monotonically increasing or decreasing subutilities, our adaptation of prospect theory, did not perform particularly well.

Table 3.5 compares MAP when all non-zero user weights are set to one ("binary weights") with the originally provided weights ("user weights") on the community MAP, with the non-learning models that used such weights. In this case, the best performer in each comparison pair (user versus binary weights on the same model) is bolded and statistically significant differences are indicated by the dagger (†). Surprisingly, the querier's criteria weights hurt performance for all but *LexPref*,

Table 3.6: MAP: Constrained Scenarios

Model	micro	macro
SortedBoolean	0.382	0.361
SimpleMAUT	**0.568**†	**0.543**†

Table 3.7: 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%)

though the difference was only statistically significant in the case of our use of prospect theory. Importantly, *LexPref* is not affected by the weight ratios, using them only as ordinals. At the very least, the use of user supplied weights may not be worth the added user input complexity, and may actually hurt performance.

Table 3.6 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, *SortedBoolean* is still outperformed by the unconstrained *SimpleMAUT*. Table 3.7 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.*

Several of the models (*AIMQ*, *CQAds*, *Tradeoff* and the MAUT variants) scored items by a linear combination of absolute differences from the query point. *AutoRank*, *ProspectTheory* and *VAGUE* also used linear combinations with somewhat different attribute scoring functions. A common problem for many of these

Table 3.8: MAP and MRR, Optimized

	MAP		MRR	
Model	micro	macro	micro	macro
SortedBoolean	**0.936**	**0.934**	**0.971**	**0.974**
SimpleMAUT	0.721	0.716	0.835	0.832
Lexical	0.461	0.470	0.598	0.618
Tradeoff	0.695	0.687	0.829	0.829

methods (and often the only thing differentiating them) were the coefficients of these linear combinations, which can be conceptually divided into two parts: a scaling factor and a weight. The scaling factors are needed because the units of the various attributes are not necessarily the same: ideally all would be rescaled to some common unit, though this is not trivial in practice. Even when the scales are identical, weights are needed to capture relative importance, for instance with departure and arrival times which are measured on the same scale. *AIMQ*, *CQAds* and *VAGUE* suffered because of these scaling factors, which assumptions about relative magnitudes of the query and data values, number of outliers, and variability of the data. These may have held in the domains they were originally evaluated in (if they were evaluated at all), but not in our domain. Overall, we found it was not so critical to have a good scaling as to avoid having a bad one, as in many cases the scaling lead to degenerate cases, leading to either no or excessive differences in the evaluation of different items. *Tradeoff* had poor weights, and we found users generally could not provide good weights. The subutility functions of *Autorank* and *ProspectTheory* did not perform well. For *AutoRank*, the subutility often dropped off too rapidly. For *ProspectTheory*, changing the subutility so that it always decreased with absolute distance from the query point brought the performance in line with the leader, despite its lack of scaling.

We wanted to understand why the poorly performing explicit models (*SortedBoolean*, *Lexical*, and *Tradeoff*) did not perform better. Did the test subjects have

Table 3.9: Queries and Pages Viewed

Model	Mean Queries micro	Mean Queries macro	Mean Pages micro	Mean Pages macro
SortedBoolean	2.112	2.104	4.030	4.170
SimpleMAUT	**1.420**	**1.415**	**3.647**	**3.638**
Lexical	1.513	1.574	3.875	4.062
Tradeoff	1.517	1.483	4.356	4.148

Table 3.10: Review Time and Qualitative Ratings

Model	Mean Time micro	Mean Time macro	Mean Rating micro	Mean Rating macro
SortedBoolean	161.418	171.671	**3.664**	**3.710**
SimpleMAUT	**136.653**	**138.104**	3.533	3.511
Lexical	161.355	161.352	3.500	3.525
Tradeoff	165.576	159.720	3.169	3.132

difficulty effectively querying, or was the model not powerful enough? We used a greedy optimization approach on each individual query to estimate an upper performance bound for each model. Table 3.8 shows the results of this greedy optimization. The results for Lexical is the lowest of all four models, and not greatly superior to the original queries. This indicates that the test subjects were fairly adept at using the Lexical model, but the model itself is not powerful enough. In contrast, the optimized *SortedBoolean* results are greatly improved, the best of all models and nearly perfect. Apparently the *SortedBoolean* model can produce excellent results, but the test subjects had difficulty creating optimal queries. Finally, the greedily optimized version of *SimpleMAUT* and *Tradeoff* have virtually the same evaluations, unsurprising as it is possible to translate queries among those two models.

Table 3.9 shows the number of queries issued and pages of tickets reviewed for the various models, with *SimpleMAUT* having the fewest queries and pages

reviewed, with the query differences statistically significant at p=0.05. This indicates that test subjects with *SimpleMAUT* found their chosen tickets with less effort than the other models. The mean time in table 3.10 also shows *SimpleMAUT* with a lower time spent, but this difference is not statistically significant. The *SortedBoolean* has a higher mean qualitative rating than *SimpleMAUT* or the other models, but in fact this is a statistical tie.

3.2.4 Summary and Contributions

In this chapter, we explored how concepts from multi-criteria decision making theory could be applied to improve retrieval. In the first part, we applied representative multi-criteria decision making methods to two significantly different retrieval problems: airline ticket search and news filtering. We also compared the results of these methods to the standard Boolean paradigm. Both multi-criteria decision making methods performed consistently as well or better than the Boolean paradigm, with the weighted sum method slightly outperforming the more complicated outranking method, ELECTRE II.

In the second part, we developed an item retrieval model based on multi-attribute utility theory, dubbed *SimpleMAUT*. We adopted an implicit query model, akin to those in information retrieval, where the user provides a partial description of what is desired, rather than an explicit specification of the result set, contrasting to current *de facto* item retrieval methods. We established a technique to translate the user's query and the item's attribute value into subutility estimations, which we combined into a single rating used to rank results.

We conducted a user study with Amazon Mechanical Turk to test this retrieval model in an airline ticket search domain. For this crowd-sourced study, we developed a game-like experiment structure which used competition and incentives to

promote good work and reduce experimenter workload. We also included several explicit retrieval models for comparison, including the common Boolean model, a lexical (sorting) model, and an explicit tradeoff model, which uses the same underlying formula as *SimpleMAUT*, but with explicitly specified model parameters.

We analyzed the performance of the explicit retrieval models to understand why they did not perform better. Simply sorting results by (possibly multiple) attributes is not powerful enough to accurately capture the user's need. Eliminating items from the result set based on a user's range restrictions is hazard-prone; the sharp divisions created by the restrictions may not match the user's own evaluations. Moreover, users are not effective at expressing a good relevance function, as demonstrated by the *Tradeoff* model.

We also performed a post-hoc study with raw data gathered from our ticketing experiment. We used a listwise learning-to-rank algorithm to learn the model parameters for our retrieval model, and compared the results to those from baselines in the literature, primarily from database researchers, as well as an adaptation of the utility model of prospect theory. Our learned model outperformed these models as well, although the formula were often similar. We found the other models often made normalization choices that may have performed well in their initial application, but not in a new domain.

To summarize, we have made the following contributions in this chapter:

- We evaluated the performance of representative algorithms from the two main branches of multi-criteria decision making theory on two recommendation tasks.

- We derived a novel method for retrieving items with numeric attributes, *SimpleMAUT*, by applying concepts from multi-criteria decision making, informed by our evaluation above. We later optimized the parameters of

this model using a listwise learning-to-rank model, *LearnedMAUT*.

- We showed that a linear combination over transformed criteria ratings can produce the correct ranking when the mutual utility independence assumption holds.

- We conducted a user study to evaluate item retrieval methods, using Amazon Turk. We compared the results of our model against standard methods as well as those in the literature, and identified causes for the differences in performance.

- We developed game structure for the crowd-sourced experiment to motivate good work and to reduce the need for quality control from the experimenter, reducing the overhead and a potential source of bias.

Chapter 4

Learning Subutility Functions

In this section, we examine what form the subutility functions take in a retrieval setting. In the first part, we return to a more standard multicriteria problem, where the criteria are user-supplied ratings rather than attribute values. We evaluate estimating a user's overall rating using linear subutility functions, as we have used in the Chapter 3, against nonlinear models, on two recommendation tasks [83]. In the latter part, we incorporate our observations on the subutility to a multi-criteria query framework [84]. As before, we evaluate this retrieval model against retrieval paradigms commonly in use at the time of writing, as well as several similar approaches proposed in the literature.

4.1 Criteria Subutility in Recommendation

In Chapter 3, we explored how multicriteria decision-making models might be used to combine ratings from several criteria or attributes, without investigating the relationship between the criteria and overall rating. Is the relationship linear or nonlinear? If it is nonlinear, can we get some insight on form of the underlying subutility function? Is there one basic subutility function, or does it vary across

criteria? To answer these questions, we perform our study within the context of two very different recommendation tasks: news article recommendations — a representative task for information filtering; and product recommendations (for flat panel televisions) — a representative task for collaborative filtering. On the two tasks, we test for the presence of nonlinear subutilities by comparing the mean squared error (MSE) of learned linear and nonlinear user models for predicting the overall item rating or recommendation.

4.1.1 Datasets

We used two recommendation datasets for our research. Each dataset had four criteria and one overall rating defined. The range of these ratings are different for different criteria, as the data were originally collected for other research. For consistency, we have rescaled all ratings to have minimum and maximum values of 0 and 1, respectively. After this rescaling, the ratings were either binary (0 or 1) or five-valued (0.0, 0.2, 0.4, 0.6, 0.8 or 1.0). For both data sets, we restrict ourselves to user-item pairs with complete ratings (i.e., any items with missing ratings were excluded from our study).

News recommendation

Our news recommendation data were provided by the University of California, Santa Cruz and Carnegie Mellon University [91]. The data were previously collected in a user study performed on the Yow-now news filtering system. Yow-now was an information filtering systems that delivered news articles to users from various RSS feeds. Approximately twenty-five users used the Yow-now system for about a month, reading news for at least one hour each day, rating approximately 9000 articles in all, with an average of 383 articles rated per user (with a standard

deviation of 252.8). This allowed us to explore creating personalized user models with the Yow-now dataset.

The users rated each article according to the following four criteria:

Authoritative : how authoritative the article appeared (binary).

Novel : the novelty of the article (five-valued).

Readable : the ease of reading the article (binary).

Relevant : the degree to which the article was relevant to the general subject category of the article (five-valued).

The overall user rating of the article was given on a five-point scale.

Product Recommendation

Our product recommendation data came from a crawl of the Epinions.com review site. Our dataset is restricted to flat panel television reviews. Approximately 1100 users reviewed 1200 items, with an average of 1 review per user (with a standard deviation of 0.29). With such a small number of reviews per user, it was clearly not possible to build personalized user models with this dataset.

The users rated each product according to the following four criteria:

Sound : The sound quality of the television (five-valued).

Ease of Use : Ease of use of the various features and menus (five-valued).

Picture Quality : All visual aspects of the television's picture (five-valued).

Durability : Durability of the television set (five-valued).

The overall user rating of the article was given on a five-point scale.

4.1.2 Approach

To test for nonlinear subutilities in the final decision/rating process, we compared the performance of two sets of models on a rating prediction task. The first model is a linear combination of ratings on the criteria, so the subutility functions themselves are linear, as we had assumed in Chapter 3. The second model assumes that subutility is nondecreasing as the criterion rating increases, but does not assume linearity. Both models take the user's item rating on each criterion as input, and output a prediction of the item's overall rating.

In this experiment, we first used machine learning to estimate the model parameters from training data. We then compared the prediction accuracy of the two sets of models on testing data. If the nonlinear model performed better, then we would have expected similar results in practice under conditions comparable to our study. On the other hand, if the latent subutility functions are linear, the nonlinear model should have performed no better than the linear model on the test set, and possibly worse due to overfitting. As mentioned earlier, we used MSE as our evaluation measure, as is often done for similar recommendation problems.

4.1.3 Linear Model

The linear model is simply a linear combination over the ratings for each criterion; the independent variables are the ratings on the criteria, plus a bias term, and the dependent variable is the overall rating. If it was possible to select the best nonlinear model in every case, the MSE of the linear model would serve as a upper bound on MSE, as the linear model is a special case of nonlinear models described below. However, due to overfitting, it is possible to select a nonlinear model that is suboptimal and worse than the linear model. The MSE achieved by the linear model is our baseline and a failure to improve upon it would indicate a

lack of evidence nonlinear subutilities. The linear model is simply:

$$f_L(\mathbf{x}) = \sum_{i=1}^{m} w_i x_i + b \tag{4.1}$$

where m is the number of criteria, $\mathbf{x}$ are the criteria ratings for an item, with b as a bias term and $\mathbf{w}$ as an m-length vector of coefficients to be learned. We restrict the weights to be non-negative.

4.1.4 Nonlinear Model

We modeled nonlinear subutility functions by creating derived binary features that correspond to specific ratings on criteria in a linear combination:

$$f_{NL}(\mathbf{x}) = \sum_{i=1}^{m} \sum_{j=1}^{|V_j|} w_{ij} [x_i \geq v_{ij}] + b \tag{4.2}$$

where V_j is the set of distinct ratings possible for the j^{th} criterion (so $|V_j|$ is the number of distinct possible ratings), [ ] is the Iverson bracket, m is the number of criteria, $\mathbf{x}$ are the criteria ratings for an item, with b as a bias term, and $\mathbf{w}_i$ as an vector of coefficients to be for the corresponding criterion (with a separate coefficient per possible rating) to be learned. As with the linear model, we restrict the weights to be non-negative.

Our representation exploits the fact that there is a small number of possible ratings for each criterion: for the news recommendation dataset, there are a total of 2+5+2+5=14 weights to be learned (compared to 4 in the linear model); for the product recommendation dataset, there are a total of 5+5+5+5=25 weights to be learned (again, compared to 4 in the linear model). A larger set of possible values would limit opportunities for generalization, and continuous valued ratings would require infinitely many weights.

Sine the learned weights are non-negative, and the weights accumulate as the ratings increase, the learned subutility functions are non-decreasing. For example, consider a rating of 0.75 for sound on the product rating dataset. Let the learned weights for sound ratings of 0.0, 0.25, 0.5, 0.75, and 1.0 be $w_{1,1}$, $w_{1,2}$, $w_{1,3}$, $w_{1,4}$, $w_{1,5}$, respectively. Since 0.75 is greater or equal to 0.0, 0.25, 0.5, and 0.75, the subutility evaluation of a 0.75 rating would be $w_{1,1} + w_{1,2} + w_{1,3} + w_{1,4}$. As an aside, since each criterion rating is at least as much as the lowest possible rating, we drop these from the model, absorbing them into the single bias term b, making the formulation slightly different, though equivalent, to Eq. 4.2.

Regularization

Since both sets of models take a linear form (as we have represented the non-linear form as a linear model on a new feature space, described above), we use a non-negative least squares solver to find model parameters that minimize MSE on the training data. However, our goal is to minimize MSE on the unseen testing data, not the training data, and given the small training set size, some form of regularization is needed to avoid overfitting. This is particularly important for the more complex nonlinear model, as the increased complexity can lead to an overly specific model that fits more of the noise in the data. We use Tikhonov regularization, a special case of L_2-norm regularization or ridge regression. The analytical solution to the minimize MSE with regularization is:

$$\mathbf{W} = (\lambda \mathbf{I} + \mathbf{X}^T \mathbf{X})^{-1} (\lambda \mathbf{W}_0 + \mathbf{X}^T \mathbf{Y}) \tag{4.3}$$

where an exponent of T indicates matrix transposition, λ controls the amount of regularization, $\mathbf{I}$ is the identify matrix, $\mathbf{X}$ is the instance matrix, $\mathbf{Y}$ is the vector of target values, $\mathbf{W}_0$ is the regularization vector we specify and $\mathbf{W}$ is the vector

of coefficients we seek. Larger values of λ causes the solution to be closer to $\mathbf{W}_0$.

For the linear model, we biased towards the following regularization vector:

$$\mathbf{W}_0 = \begin{bmatrix} 0.25 & 0.25 & 0.25 & 0.25 & 0.0 \end{bmatrix} \tag{4.4}$$

where the last position is the constant bias term and the other terms are the coefficients for the four criteria. We chose W_o such that all criteria would be weighted evenly, and the minimum (maximum) overall rating would be predicted when the minimum (maximum) rating was given on each criterion.

For the nonlinear models, we biased the model so that it would prefer more linear subutilities and evenly weight all criteria as with the linear model. The sum of all the weights for a criterion's ratings sum to 0.25, so we have four sets of repeated terms (one set per criterion), followed by the constant bias term:

$$\mathbf{W}_0 = \begin{bmatrix} \frac{0.25}{|V_1|} & .... & \frac{0.25}{|V_2|} & .... & \frac{0.25}{|V_3|} & .... & \frac{0.25}{|V_4|} & .... & 0.0 \end{bmatrix} \tag{4.5}$$

where V_j is the same as in Eq. 4.2. Since the number of unique criteria ratings varies, the size of $\mathbf{W}_0$ also varies.

Tuning and Model Selection

The λ term in equation 4.3 controls the tradeoff between coefficients that minimize MSE on the training set, and coefficients that are closer to the regularization vector ($\mathbf{W}_0$) described above. Higher values of λ moves the solution closer to the regularization vector, while allowing for higher MSE; lower values of λ do the opposite. We choose λ by further dividing each training set into a 10% validation set and another (sub-)training set from the rest of the data, which we use to train the model with a given version of λ. The validation set is used to evaluate the MSE for the corresponding value of λ. We repeat this for several values of λ,

Table 4.1: Mean Squared Error for Global Model

	micro		macro	
Domain	linear	nonlinear	linear	nonlinear
News	0.5633	**0.5629**	0.5392	**0.5385**
Product	0.6506	**0.6307**†	0.6618	**0.6408**†

Table 4.2: Mean Squared Error for Personal Model

	micro		macro	
Domain	linear	nonlinear	linear	nonlinear
News	0.4314	**0.4241**†	0.4668	**0.4542**

and having observed a peak value, we train a new model on the entire training dataset using this value for λ. This process is repeated for each fold and used independently for both the linear and nonlinear models.

4.1.5 Results

We evaluate the mean squared error by both a mean of means over users, a *micro-average* and a *macro-average*. The microaverage is calculated by calculating a mean squared error for the ratings from each user; each mean is weighted by the number of responses for the corresponding user, and so the end result is the same as the overall MSE without regard to the individual users. By contrast, the macro-average weights each user equally, without regard to the number of responses, calculating a mean of these means. In addition, we train a global model for both news and product recommendation and a personalized model for each news recommendation test subject. (The product recommendation dataset average less than two reviews per user, so personalized models in this framework was not feasible.)

Each model was trained as described in Section 4.1.2, with the personalized models trained on each user's data individually. Thus, the global models were

trained to minimize the microaverage of squared error, and the overall fit was dominated by more prolific users. To test statistical significant with p=0.05, we use a randomization test [71]. In this test, the null hypothesis is that the observed difference between each model on a given response was equally likely to favor the other model, and so the observed difference is due only to random chance. Thus, we randomly reassign the observed difference on each response (flipping or not) for all responses, calculate the observed difference, and repeat this process a million times, recording the fraction of randomized responses that produce a difference at least as large as that observed.

Table 4.1 shows the results for a universal model for all users, with the best performer (comparing linear and nonlinear) bolded and statistical difference signified by the dagger (†). In all cases, the nonlinear model outperforms the linear model, although the reduction in MSE is slight. For the news recommendation domain, this difference is not statistically significant, but for the product recommendation domain, both the resultant micro- and macro- average is significant. For the news recommendation domain, we also train a personal model for each test subject, with the results shown in Table 4.2. In this case, the microaverage results are statistically significant, but the macroaverage results are not. Given that the users with less data presumably have corresponding models with a less accurate fit, this is not surprising.

Our results show that underlying subutility functions of criteria-based information retrieval models are nonlinear, at least in some cases, as measured by an observed reduction in MSE when fitting to nonlinear models. We observed this reduction in both non-personalized and personalized models. However, the amount of MSE reduced by exploiting non-linearity was slight in the datasets we used. Moreover, there was not a consistent shape of subutility function we could

observe, nor an underlying structure. Most estimated subutility functions roughly fit one of three shapes. Moving from right to left, in some cases the subutility dropped rapidly and then leveled off; in others, it descended slowly at first and then rapidly; and in yet others, it combined the two, at first decreasing slowly, then dropping quickly, then slowly again, much like a Gaussian function. In no case did it match a step function or exactly linear function.

Despite our use of regularization, overfitting remained a problem, as evidenced by the occasional *increase* in MSE over the linear model. This could be expanded to a Bayesian framework, using prior probabilities to avoid selecting less probable models when there is not sufficient support in the data. Even without these improvements, in our experiments we were successful in reducing the overall mean MSE by exploiting nonlinearity in the underlying subutility functions.

4.2 Incorporating Nonlinear Subutility into Query-based Retrieval

In this section, we expand on the retrieval framework we developed in section 3.2. Given our observation that underlying subutility functions can be better modeled with nonlinear functions, we develop a parameterized form that can represent the shapes of subutility functions we have observed. We also conduct a user experiment using a new, more complex domain, that of daily meal plans. In addition to providing a differing domain to test our ideas, meal plans also provide important differences from airline tickets, leading to a more thorough evaluation of our concept. First, meal plans naturally suggest specific and varying nutritional targets, in contrast to booking airline tickets, which would mostly seek to minimize values (such as price and duration). Second, meal plans are comprised of several

meals, each consisting of dishes, a thus are really a set of items, also referred to as a *package*.

Package recommendation has received a fair amount of attention in the literature [38, 88, 74, 6, 61, 62, 60, 9, 5, 12]. In this approach, recommended packages are generated on the fly subject to some hard constraints, typically an NP-complete problem. Our package retrieval approach contrasts with these by selecting from a fixed (though large) corpus of packages, eschewing constraints and explicitly stated objective functions for estimations of utility. We compare our utility-based approach to using hard constraints to find appropriate packages (meal plans). We also investigate how to effectively aggregate multiple values for a given attribute of a package.

4.2.1 Towards a model of subutility

In Chapter 3, we developed a simple retrieval model inspired by multi-attribute utility theory (MAUT) [28], unimaginatively dubbed *SimpleMAUT*. To briefly review, MAUT assumes *mutual utility independence*, which means that the utility function takes either an additive or multiplicative form: we showed that a linear combination exists that produces the same ranking, and so *SimpleMAUT*'s utility function is a linear combination of ratings:

$$f(\mathbf{Q}, \mathbf{D}_i) = \sum_j w_j \times g_j(q_j, d_{ij}) \tag{4.6}$$

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. For *SimpleMAUT*, we assume a linear subutility function $g_j(q_j, d_{ij})$:

$$g(q_j, d_{ij}) = \left(1 - \frac{|q_j - d_{ij}|}{\max\left(|q_j - \perp_j|, |q_j - \top_j|\right)}\right) \tag{4.7}$$

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. 4.6.

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 as well as an underlying additive form. This has nonintuitive consequences, for instance, it implies that reducing the price by \$5 is just a compelling when for a \$1000 item as it would be for a \$10 item, given the same desired price. 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 the meal plans in our 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, given

the benefit of such that we observed in Chapter 3.

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

$$\begin{aligned} g_j(q_j, d_{ij}) = [q_j \geq d_{ij}] \exp\left(-\left(\frac{|d_{ij} - q_j|}{\phi_j^{\geq}\sigma_j}\right)^{\rho_j^{\geq}}\right) \\ + [q_j < d_{ij}] \exp\left(-\left(\frac{|d_{ij} - q_j|}{\phi_j^{<}\sigma_j}\right)^{\rho_j^{<}}\right) \end{aligned} \tag{4.8}$$

where σ_j is the standard deviation of j^{th} attribute, [ ] is the Iverson bracket, $\boldsymbol{\rho}^{\geq}$, $\boldsymbol{\rho}^{<}$, $\boldsymbol{\phi}^{\geq}$ and $\boldsymbol{\phi}^{<}$ are model parameters, and others are defined as above.

Finally, we chose to aggregate multiply valued subutilities 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 $x_1, ..., x_n$:

$$M(x_1, ..., x_n) = \left[\frac{1}{n}\sum_{i=1}^{n} x_i^{\psi}\right]^{\frac{1}{\psi}} \tag{4.9}$$

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.

4.2.2 Learning

The *AdaptiveMAUT* model (Figure 4.1) 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. 4.6, using the subutility function $g()$ in Eq. 4.8, and the general mean (for dish ratings only) in Eq. 4.9, the likelihood function $L()$ is:

$$L(\boldsymbol{\rho}^{\geq}, \boldsymbol{\rho}^{<}, \boldsymbol{\phi}^{\geq}, \boldsymbol{\phi}^{<}, \mathbf{w}, \psi; \mathcal{Q}, \mathbf{D}, \mathcal{R}, \mathcal{U}) = \prod_{\mathbf{Q} \in \mathcal{Q}} \prod_{r \in \mathcal{R}} \prod_{u \in \mathcal{U}} \left(\frac{b}{2} + \frac{1-b}{1 + \exp(-c(f(\mathbf{Q}, \mathbf{D}_r) - f(\mathbf{Q}, \mathbf{D}_u)))} \right) \tag{4.10}$$

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 (unrelated to the c in Eq. 3.9, and set to 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.

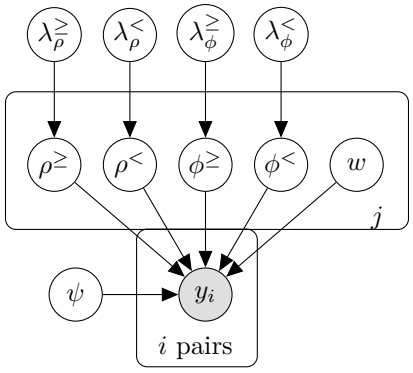

Figure 4.1: Hierarchical Bayesian model of *AdaptiveMAUT*

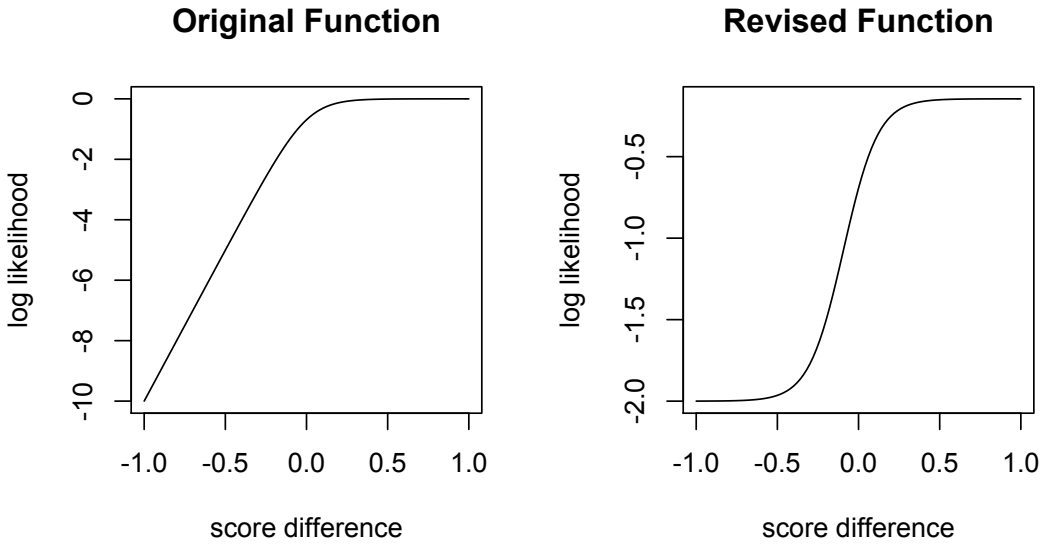

Figure 4.2: Log Likelihood

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, in order to avoid a lower overall probability where most pairs were likely but offset but a few very unlikely pairs. Unfortunately, this meant that model was also misclassifying most 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 4.2 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 4.2.4 using the mode of the resulting posterior

distributions.

4.2.3 Experiment

We follow the same general experimental framework we used in Section 3.2.2 to compare several models' ranking of results and to compare the quality of packages found using the different retrieval models as well as to hand-built packages. However, we make some improvements to the experimental protocol, adding a head-to-head comparison, dropping the poorly performing *Lexical* and *Tradeoff* baseline models while adding a *Faceted* model, and improving our model as described in Section 4.2.1. Though *SortedBoolean* also performed poorly in our previous study, we include it as the *de facto* retrieval model and to compare its results with the also popular *Faceted* model.

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.

SortedBoolean Like the earlier ticketing user study, this interface allowed users to restrict the result set by attribute ranges and to provide a sort order. In this experiment, we allowed restriction on any attribute and up to four sort orders. An example of the query interface is shown in Figure 4.3.

Faceted We created a basic faceted search model, inspired by those in use in popular e-commerce sites today, to compare with our other models. All attributes were split into a small number of equally sized facets, with seven

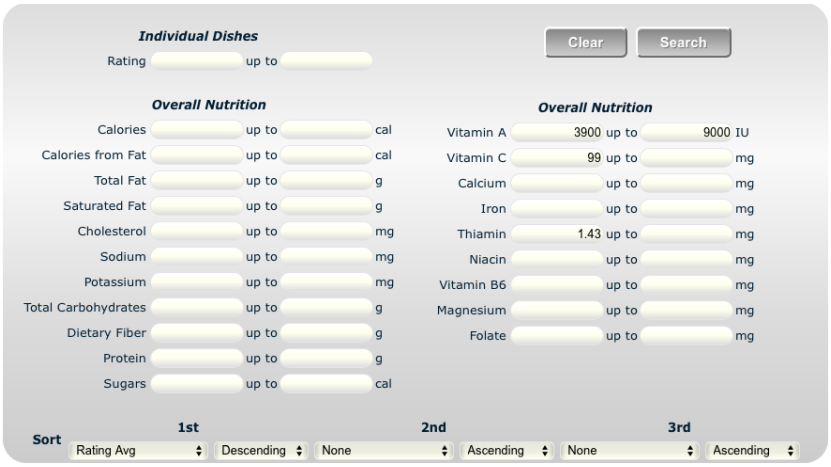

Figure 4.3: SortedBoolean search interface.

to twelve facets per attribute. Up to four sort orders could also be chosen. An example of the query interface is shown in Figure 4.4.

Point The point-based user interface allows a user to specify single values for each attribute, allowing the user to give specific attribute values of interest. Partial specifications (attributes can be left blank) are acceptable, as with other interfaces. This interface was used initially for *EnhancedMAUT* in the user study, with the collected data re-used post-hoc for *AdaptiveMAUT* and the baselines from the literature. The point-based interface is shown in Figure 4.5

Figure 4.4: Faceted search interface.

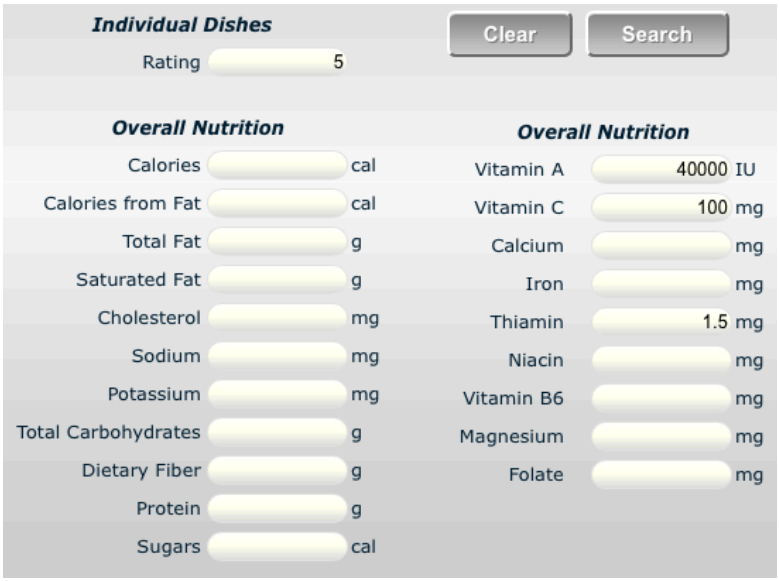

Figure 4.5: Point-based search interface.

Retrieval Models

We used our proposed models, the *de facto* item retrieval methods of Boolean and faceted search, we use the same models from the literature as our previous study in Section 3.2.2, which we include again here. Variables are defined as in Section 4.2.1 unless otherwise specified.

AIMQ *AIMQ* [56] estimates relevance using a different normalization and global weights derived from functional dependencies. Items are ranked by decreasing score order, where the score of d_i is defined as:

$$f(\mathbf{Q}, \mathbf{D}_i) = \sum_{j \in \mathbf{Q}} w_j \left(1 - \min\left(1, \frac{|q_j - d_{ij}|}{q_j}\right)\right) \tag{4.11}$$

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 [2]. They used an inverse document frequency (IDF) term for weighting, defined below for query element q_j as:

$$w_j = \log\left(\frac{n}{\sum_{k=1}^{n} \exp(-\frac{1}{2}\left(\frac{d_{kj} - q_j}{h_j}\right)^2)}\right) \tag{4.12}$$

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:

$$f(\mathbf{Q}, \mathbf{D}_i) = \sum_{j \in \mathbf{Q}} w_j \exp\left[-\frac{1}{2}\left(\frac{d_{ij} - q_j}{h_j}\right)^2\right] \tag{4.13}$$

with items ranked by decreasing score.

CQAds *CQAds* [65] 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

$$f(\mathbf{Q}, \mathbf{D}_i) = \sum_{j \in \mathbf{Q}} \left(1 - \frac{|q_j - d_{ij}|}{R_j}\right) \tag{4.14}$$

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/SortedBoolean *Faceted* and *SortedBoolean* 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 *EnhancedMAUT* and *AdaptiveMAUT* were described in Sections 4.2.1 and 4.2.2, respectively. All model parameters for *EnhancedMAUT* were chosen to be one for the user experiment, simplifying the subutility function:

$$g_j(q_j, d_{ij}) = \frac{1}{\exp\left(\frac{|d_{ij} - q_j|}{\sigma_j}\right)} \tag{4.15}$$

where parameters are the same as in Eq. 4.8. Not coincidentally, these chosen values are the mode for each prior distribution used by *AdaptiveMAUT*. As the final form of *EnhancedMAUT* given here had not been developed at the time of the user study, an earlier precursor was used that had a slightly different subutility formulation: $\frac{1}{1+\frac{|d_{ij} - q_j|}{\sigma_j}}$. After the user study had completed, *AdaptiveMAUT* was trained and evaluated post-hoc as described in Section 4.2.2.

VAGUE The *VAGUE* framework [54] provides a "similar-to" operator that calculates a weighted Euclidean distance from the query point and the item. This operator can use subutility functions, but none are prescribed, so we choose the absolute difference divided by the standard deviation:

$$f(\mathbf{Q}, \mathbf{D}_i) = \sqrt{\sum_j \left[w_j \left(\frac{|q_j - d_{ij}|}{\sigma_j} \right) \right]^2} \tag{4.16}$$

with items ranked by increasing score. As with the MAUTs above, uniform weights in the meal plan study as we had dropped user-supplied weights based on our earlier results.

Only *EnhancedMAUT* and *AdaptiveMAUT* 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*, and *AdaptiveMAUT*), and *VAGUE* all accept the same query input, differing only in how they rank results. Thus, only *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, *AdaptiveMAUT* was trained with data after user study completion instead of on-line.

Data Used

As with the ticketing study, we wanted test subjects to perform realistic tasks, using appropriate real world data. Similarly, we developed twenty short scenarios based on the literature. We consulted a popular nutritional resource [66] 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 [81] 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 4.6 gives an example of one of the meal plan scenarios.

```
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 4.6: Example Scenario for Meal Plan Study

We used the meal plan components (individual dishes) to create the corpus, as large open 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 4.5) which

could be simply summed. The other attribute was allrecipe.com individual dish ratings, which were preserved for each meal plan.

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 [3], restricting to workers within the United States and with high completed work acceptance rates (95% or better). 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.

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 the two meal plans entered by the two searchers, along with two randomly selected meal plans, in a random order. When the judge chooses a searcher's entered meal plan, that worker is given a bonus. A second judge would be given the same set to evaluate, and should the second judge made the same selection as the first, both get a reward. The use of a direct contest between two searchers using different search engines allowed for results from different search engines could be directly compared (*head-to-head*).

In addition, we asked each test subject (whether searcher or judge) to provide a justification for their selection. Work was rejected when justifications were inadequate and eliminated from our study, eliminating about 10% of the responses. From the accepted work, 205 test subjects completed 321 tasks. As each task had exactly one initial query and one item chosen, 321 initial queries and 321 final selections were collected.

Evaluation Metrics

As mentioned in Sec. 4.2.3, we use only the first query from each session to calculate mean average precision (MAP). 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. Given the large number of meal plans, 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. Finally, we evaluate over the subset of the corpus actually viewed by the test subject, sometimes referred to as *induced MAP*, which has been shown to better correspond to the true MAP value given incomplete judgements [90]. We assume that the user views meal plans starting from the top until either the end of the result set is reached, or a

meal plan is selected.

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 [71] 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 is the fraction of times this redistribution produced a difference for the metric that was at least as great as the actual observation.

4.2.4 Results

We provide various results of our experiment below, with the leader bolded and statistically significant difference (at p=0.05 or better) against all others indicated by the dagger (†), and with the asterisk (∗) indicating a statistically significant

difference against all other models except *VAGUE*.

Table 4.3: Community-evaluated Induced 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
AdaptiveMAUT	**0.393**†	**0.382***	0.**407**†	**0.396***

Table 4.3 gives the community-evaluated induced 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 *Adaptive-MAUT* 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 4.3) yielded better results, mostly by a statistically significant differences; even so, the differences with the *AdaptiveMAUT* result remained statistically significant.

Table 4.4: Searcher Success by Search Engine

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

Another way to evaluate search result quality is to see how often a model was used to find the winning meal plan. Table 4.4 shows searchers using the

Table 4.5: Head-to-Head

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

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 4.5, 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 4.2.3), 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. In contests between a hand-built meal plan and one found by any search engine, those found via search were preferred 18 times to 9, even though it typically took five times longer to build a meal plan than to find one.

Table 4.6: Turker Behavior

Paradigm	Queries	Time	Parameters	Orders
EnhancedMAUT	**1.80**†	2:43	4.50	**0.0**†
Faceted	3.48	**2:34**	**2.00**†	0.12
SortedBoolean	2.57	2:44	3.81	0.28

Table 4.6 provides various microaverages (average without respecting scenario

difficulty effects), with the leader bolded and statistically significant differences indicated by the dagger (†). We assume smaller values are preferable throughout. The meaning of the various averages follow:

1. **Queries**. Average number of queries issued per task. We did not try to detect duplicate queries, i.e., re-issuing a query that had been issued before.

2. **Time**. The total time spent searching and reviewing the results. This does not include time waiting for the search engine, which was minimal in any case.

3. **Parameters**. The total number of meal plan attributes, per query, that had constraints or criteria specified.

4. **Orders**. The total number of sort orders, per query. The *EnhancedMAUT* paradigm does not allow for sort orders.

As can be seen from the table, searchers were able to find a desired item issuing fewer queries when using the *EnhancedMAUT* paradigm, though the comparison with *Faceted* is less meaningful as its queries are built progressively. Yet, there was little difference in the average time taken to make a selection. Sorting was a rarely used feature.

4.2.5 Summary and Contributions

In this chapter, we explored the subutility functions for our MAUT-based model. In the first part, we tested our prior assumption the these functions were linear on two recommendation tasks: product recommendation (specifically, televisions) and news article filtering. As each task had a finite and small rating domain, we were able to build a nonlinear subutility model capable of fitting any

nondecreasing subutility function. We evaluated the performance of this nonlinear subutility model against a linear one on mean squared error, and found the nonlinear model had superior performance, although the difference was only statistically significant for the product recommendation task. We fit separate subutility models for each criterion, and found little commonality upon inspection, implying that subutility may vary by criterion rather than taking a single, universal form.

Based on these observations, we built upon our earlier basic multi-criteria decision making theory-inspired model, *SimpleMAUT*, to a more advanced model, *EnhancedMAUT*. We extending the theory, incorporating a nonlinear subutility function, and then developed a Bayesian graphical model around the core utility model so we could tune the model parameters in a pairwise learning-to-rank framework, dubbed *AdaptiveMAUT*. We evaluated the model in a nutritionally-based user study conducted with Amazon Mechanical Turk. As before, we compared our MAUT models with 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.

Explicitly constraining the result set hurt the performance of *SortedBoolean* and *Faceted*. 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 *SortedBoolean* and *Faceted* are quite different, the underlying query semantics are identical, and we observed nearly identical retrieval performance. These models that required users to give an explicit ranking, performing worse than *EnhancedMAUT*, which implicitly ranks by attempting to glean query intent. As before, test subjects were most successful using the *implicit* MAUT query model.

In the post-hoc analysis, the models that accept single attribute values instead of ranges (*AIMQ*, *AutoRank*, *CQAds*, MAUT variants, and *VAGUE*) vary widely in their performance, despite their similarities. *AIMQ* and *AutoRank* suffered because of their estimated attribute weights; replacing these with uniform weights improved performance. As with the study Chapter 3, the research baseline models did not perform well in a new domain, with their parameters based from an analysis of the corpus or from assumptions. In contrast, *AdaptiveMAUT* was significantly better than other models in every category.

One of our user studies involved package retrieval, where a set of meals would be returned as a daily meal plan. Although most of the attributes of the component dishes could be summed up in the package, ratings could not. We found that our learning model, *AdaptiveMAUT*, tended to prefer component attribute aggregations closer to the geometric mean rather than the often assumed arithmetic mean. Using the geometric mean in our baselines also improved retrieval performance. We also found that searching a large corpus of packages was more effective than creating a meal plan by hand. This may be an alternative to the often explored approach of generating a package on the fly under constraints, particularly when the problem does not have true hard constraints.

To summarize, we have made the following contributions in this chapter:

- We evaluated the performance of linear and nonlinear subutility functions on two recommendation tasks.

- Continuing beyond the guidance of multiattribute utility theory, we developed a parameterized yet flexible subutility function from several principles, yielding *EnhancedMAUT*. We later developed a graphical Bayesian model, *AdaptiveMAUT*, using *EnhancedMAUT* as its core and learned the parameter values in a pairwise learning-to-rank model.

- We conducted another user study to evaluate item retrieval methods in a new domain, again using Amazon Turk. We compared the results of our enhanced model against standard methods as well as those in the literature, and identified causes for the differences in performance.
- We explored how to combine multiple ratings given a group of items, and compared the efficacy of searching for a package against generating one manually.

Chapter 5

Extending the Retrieval Model

In this section, we explore how to extend the model to handle a wider range of queries, including those of an actual item search site and those stemming from natural language queries.

5.1 Extending to an Active Search Engine

Up to this point, we have evaluated our retrieval model under somewhat controlled conditions. Our experiments have simulated realistic conditions, but out of necessity, have had a limited number of user needs and ideal data conditions. In this section, we take our evaluation out of the laboratory and explore applying it to an existing vertical form-based search engine, the Exoplanets Orbit Database (EOD) [36]. We extend our retrieval model handle the additional attribute types of the EOD and evaluate retrieval performance against queries submitted to the EOD.

5.1.1 The Exoplanet Orbital Database

The Exoplanet Orbit Database is produced and maintained by Professor Jason Wright at Penn State University. Exoplanets are planets that occur outside of our solar system. There may be a variety of reasons to study exoplanets, but the most often cited (at least in non-scientific circles) is to find other planets that could support life. Indeed, the popular interest in exoplanets has grown so much that new discoveries are sometimes reported in the national news, as was the case for TRAPPIST-1 and its bevy of temperate terrestrial planets in 2017. Nevertheless, although the EOD is available to the public it is nonetheless quite technical, and it is doubtful that anyone without an appropriate scientific background would be able to query effectively or make much use of the results.

We adapted an existing database search engine to provide a form-based search capability to aid in our research, with complementary set of strengths and weaknesses. In terms of strengths, the EOD is a completely natural search environment, meaning users come to the website with their own needs and interests, so there is no need to create scenarios, tasks, or other activities to induce searching. In addition, the domain is scientific and quite technical, but appears to be used by a community that has the appropriate background, so there is no need to develop training materials or simplify the domain.

On the other hand, difficulties arise from any analysis of a search engine "in the wild". Users may have their own search needs when searching, but these are not revealed directly to the experimenter, and the technical domain makes it challenging to discern. Likewise, given this latent user need, what is relevant to a query is hard to surmise. Also, nearly all data is available directly from the search results (in tabular form), presumably decreasing clickthroughs that might otherwise provide insight.

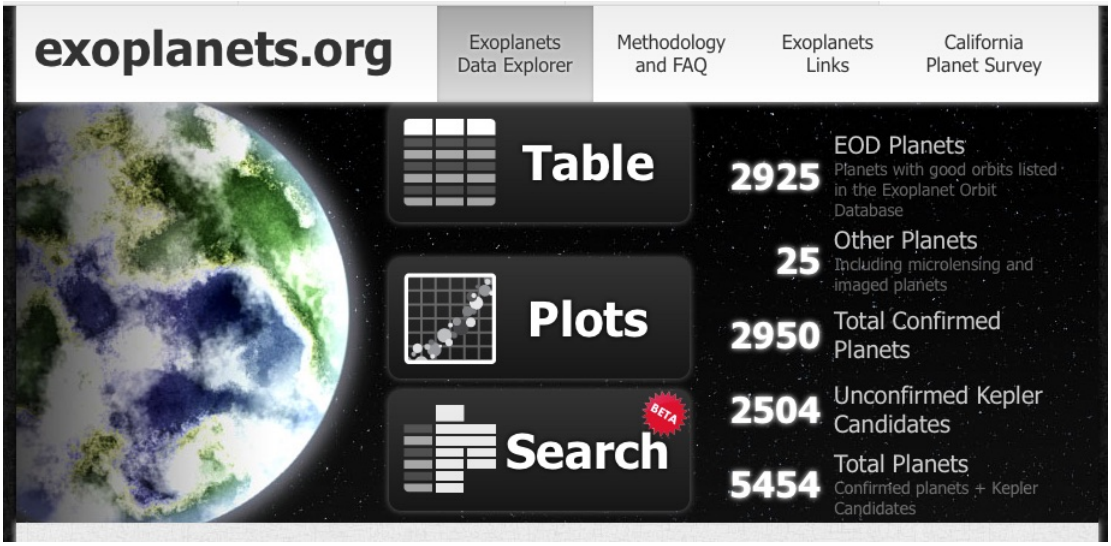

Figure 5.1: exoplanets.org Front Page Excerpt.

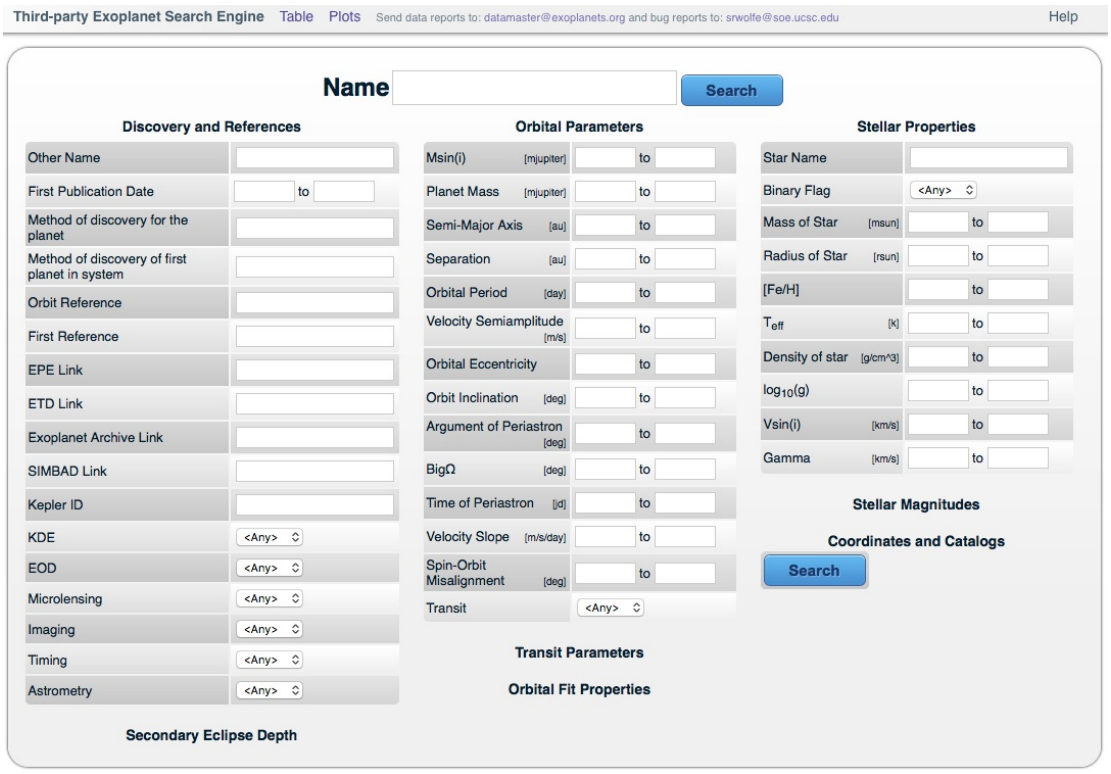

Figure 5.2: Form-based EOD Search Engine.

The EOD is not terribly large, in fact it is possible to download the entire dataset in a comma-separated file, something the website makes easy. Given this, it is perhaps surprising that search is used at all. The EOD also has its own search and plot capability, the Exoplanet Data Explorer (EDE), developed by Dr. Onsi Fakhouri. The EDE is quite powerful and supports complex queries with its own query language, but does require some reading of the documentation. We do not have access to any usage data of the EDE. Figure 5.1 shows the top part of the EOD front page, with the EDE accessible from "Table" and "Plot" buttons. Our form-based search engine is available via the "Search" button, with the search form shown in Figure 5.2. The search form initially hides some of the attributes as to not overwhelm new users; the other headers (such as Secondary Eclipse Depth) can be clicked on to reveal corresponding form fields, and existing headers (such as Discovery and Reference) can be closed to save screen space.

The Form-based Search Engine

The form-based search engine is straightforward. Its most important feature, from our point of view, is the extensive logging of use. All user interactions are logged, including queries, results, and any clickthroughs, along with additional metadata such as request time, response time and IP address. The search supports constraints on three type of fields: numeric, text and enumerated. For numeric fields, an lower and upper bound (both optional) can be given; only exoplanets with values falling with the (possibly one-sided range) will be returned (thus, those with missing values are excluded). No formulae are allowed, and any nonnumeric query value in a numeric field results in an error message to the user. Any value is allowed in the text and enumerated input fields, with the search engine returning only records that contain the input. Likewise, Booleans are con-

trolled by a pulldown menu, and can either remain unspecified (the default) or a Boolean value.

5.1.2 Characterizing the EOD and its Use

At the time of writing, the EOD contains 5307 total planets. The EOD is a dynamic resource, so both the total number of planets and some of their attribute values change over time. The EOD does not change rapidly, however, and in our analysis we treat it as static. Table 5.1 gives the breakdown of the attribute types the in EOD schema, which are either textual (either names or identifiers), numeric, Boolean or enumerated fields. Values are unknown for many exoplanets ("missing" values). Approximately half of the attribute values are unknown, and varies widely by attribute type, with six attributes known for all exoplanets to a low of only three known values each for two attributes. Error bounds are also given for numeric attributes, but are not available as query parameters in our interface.

Table 5.1: Attribute Types

Attribute Type	Count
Numeric	60
Textual	19
Boolean	11
Enumeration	2

We collected approximately 3 years of search and usage logs, from 2015-08-10 to 2018-08-29. Two months of this data was unusable due to a bug in logging. During this time, 29233 total queries were captured, averaging about 25 queries per day. This total excludes invalid queries (e.g., queries that had non-numeric values in numeric fields) and our own testing queries. We have not attempted to

eliminate queries from crawlers or other bots; we do assume, however, that such automated programs will not fill out the query page.

We use clickthrough data as an approximate indication of relevance. Our form-based search returns search results with attribute values in tabular form, providing nearly all available information in the results page. This probably depresses the clickthrough data, a problem that also occurs in popular search engines [48], sometimes referred to as *good abandonment*. In response, we only analyze queries with clickthroughs in their session, as we detail below, limiting our dataset to 11350 queries.

Queries are not necessarily isolated events but can be part of a series in a longer information seeking activity. Therefore, we also group queries into sessions. For simplicity, we define a session as all queries from the same IP address, ending the session with any period of inactivity spanning and hour or more. In addition, we terminate a session with any query that results in clickthroughs, presuming that the information need has been met. Subsequent queries are then treated as a new session. Sessions without clickthroughs are excluded from our analysis. Thus, in this study a session consists of zero or more queries without subsequent clicks, terminated by a single query where the user clicks on one or more results. 5472 such sessions were identified in our data set, with approximately 2 queries per session on average.

Table 5.2 shows the top ten attributes queried. By far, the most common field queried is the NAME field, with just over two thirds of all queries using this field and only this field. The majority of these queries are presumably to find data on a known exoplanet, rather than to discover new exoplanets of interest. Consistent with findings in web search engines, queries tend to be short: 77% of queries only constrained one field, with 2% not constraining any field. Of the rest,

Table 5.2: Top 10 Queried Attributes

Column	Description	Type	Usage count
NAME	Name	string	8270
MASS	Planet mass	number	741
PER	Orbital period	number	633
STAR	Star name	string	589
MSTAR	Mass of Star	number	530
TRANSIT	Transit across star	Boolean	431
A	Orbit semi-major axis	number	420
OTHERNAME	Other name	string	355
TEFF	Effective temperature of star	number	281
BINARY	Multiple star system	Boolean	255

8% constrained exactly two fields, with 13% constraining more then two fields. Almost one third of queries were "failing" queries, i.e., queries that produce an empty result set.

5.1.3 Expanding to Other Attribute Types

We expand on our previous MAUT-based formulation from Chapter 4 to handle non-numeric attributes. As before, MAUT allows us to model the overall utility function be a linear combination of the subutility evaluations:

$$f(\mathbf{Q}, \mathbf{D}_i) = \sum_j w_j \cdot g_j(q_j, d_{ij}) \tag{5.1}$$

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. For non-numeric attributes, only one desired value is allowed per attribute; for numeric attributes, we expand the model to allow for a (continuous) range of attributes.

The subutility function differs according to the attribute type. For numeric

attributes, the subutility function is expanded from that in Eq. 4.8, replacing the single attribute query value with a range from a low of $\underline{q_j}$ to a high of $\overline{q_j}$:

$$\begin{aligned} g_j(\underline{q_j}, \overline{q_j}, d_{ij}) = &\left[\underline{q_j} > d_{ij}\right] \exp\left(-\left(\frac{|d_{ij} - \underline{q_j}|}{\phi_j^{>}\sigma_j}\right)^{\rho_j^{>}}\right) \\ &+ \left[d_{ij} \in [\underline{q_j}, \overline{q_j}]\right] \\ &+ [\overline{q_j} < d_{ij}] \exp\left(-\left(\frac{|d_{ij} - \overline{q_j}|}{\phi_j^{<}\sigma_j}\right)^{\rho_j^{<}}\right) \end{aligned} \tag{5.2}$$

where σ_j is the standard deviation of j^{th} attribute, [ ] is the Iverson bracket when containing a test, $[\underline{q_j}, \overline{q_j}]$ is the closed interval with limits $\underline{q_j}$ and $\overline{q_j}$, $\boldsymbol{\rho}^{>}$, $\boldsymbol{\rho}^{<}$, $\boldsymbol{\phi}^{>}$, and $\boldsymbol{\phi}^{<}$ are model parameters, and others are defined as above. If only one side of the interval $[\underline{q_j}, \overline{q_j}]$ is given, the unspecified endpoint is taken as $-\infty$, ∞ for the low and high limits, respectively. This is the fully parameterized form; however for this experiment we use 1 for the model parameter values $\boldsymbol{\rho}^{>}$, $\boldsymbol{\rho}^{<}$, $\boldsymbol{\phi}^{>}$, and $\boldsymbol{\phi}^{<}$.

For textual attributes, a great number of possible text retrieval models could easily be used, for example cosine similarity with TF-IDF, BM25, Divergence from Randomness or a learning-to-rank approach. However, the text fields in the EOD are short character strings typically representing a name of some type (e.g., *Kepler-107 d*). Based on input from our EOD collaborators, the form-based search engine uses a case-insensitive substring match, which we also use in this experiment. We acknowledge this is an area that could use further improvement.

The subutility function for Boolean attributes is necessarily straightforward. Since Boolean attributes may only have one or two values, each item (without a missing value) will either have the desired value or the opposite value. Thus, the subutility function for Boolean attributes is simply:

$$g_j(q_j, d_{ij}) = [q_j = d_{ij}] \tag{5.3}$$

where quantities are as defined above.

Finally, the subutility for enumerated attributes can be viewed as an extension of that for Boolean attributes, as they too have a fixed set of possible values but may have more than two possibilities. This makes the problem more complicated, however. Given our assumption that subutility is maximal when an item has the query value, what remains is to estimate, for each pair of attribute values, what the subutility is when an item has an attribute value different than the desired value. Indeed, this is the general subutility estimation problem (estimating subutility given different attribute values), but given the finite number of possible attribute values, only a finite number of subutility evaluations need to be estimated, leading to:

$$g_j(q_j, d_{ij}) = \sum_{v_q \in \mathcal{V}_j} \sum_{v_d \in \mathcal{V}_j} [v_q = q_j, v_d = d_{ij}]\, \omega_{v_q, v_d} \tag{5.4}$$

where $\mathcal{V}_j$ is the set of possible values of the enumeration, ω_{v_q, v_d} is the estimated subutility for v_d when v_q was desired, with [ ] again as the Iverson bracket and other quantities defined as before. Since exactly one of the tests will be true, the subutility will equal one of the $\boldsymbol{\omega}$ estimations rather than the sum of several. We can see that this reduces to Equations 5.3 for two attribute values, assuming symmetry among the substitutions and absorbing minimal subutility into the attribute weight in 5.1. The various values of could be developed from a domain theory or learned from data; for this experiment, we estimate the subutility as 1

when $v_q = v_d$ and 0 otherwise.

Missing values are one final issue that must be dealt with for all subutility functions, a frequent issue with the EOD data. Depending on the domain, missing values could be treated as a separate value with a separate subutility estimation, or use a variety of value imputation methods to infer a subutility. In this experiment, we take the former approach, assuming a missing value is the worst type of outcome for an attribute of interest and should have a subutility of zero.

5.1.4 Results

We evaluate the ranking performance of several retrieval models on the query and clickthrough data described in Section 5.1.2, once again using mean average precision (MAP) as our metric.

Boolean In this model, the user's query is interpreted as hard constraints: only exoplanets that exactly match the user's query is returned. This is identical to the *ScoredBoolean* model in Chapters 3 and 4, except no user-specified sort order is provided. Instead, returned items are ranked by default order, in this case an internal identifier. This default order is also used to break ties for the models below. The *Boolean* model was the only model used by the users of the website, and so its results are identical to those returned to the user during the search recording phase, with the exception of any relevant changes to the corpus in the interim.

SoftBoolean The *SoftBoolean* model divides the corpus into to disjoint subsets: those that match the constraints fully (exactly the results of the *Boolean* model above), and those that do not. The entire corpus is returned, with the fully matching subset ranked before the remainder, which each group ordered by the default order.

ScoredBoolean The *ScoredBoolean* further orders the violating subset by scoring all items in the corpus. The score for each item is the number of query attributes matched, with the results ranked by descending order. As stated above, ties are broken by the default order.

AIMQ As before, *AIMQ*'s [56] score uses global weights derived from functional dependencies in a linear combination of per attribute similarity estimations, ranking in decreasing order as:

$$f(\mathbf{Q}, \mathbf{D}_i) = \sum_{j \in \mathbf{Q}} w_j \times AIMQ\left(q_j, d_{ij}\right) \tag{5.5}$$

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. *AIMQ* does not appear to handle ranges, so we alter the numeric similarity formula in a similar way as *ExpandedMAUT*:

$$AIMQ\left(\underline{q_j}, \overline{q_j}, d_{ij}\right) = 1 - \min\left(1, \frac{\max\left(0, d_{ij} - \underline{q_j}, \overline{q_j} - d_{ij}\right)}{q_j}\right) \tag{5.6}$$

Note that at most one of the terms in the max function in Eq. 5.6 will be positive. *AIMQ* handles nonnumeric attributes by comparing associated "supertuples". The supertuple for an attribute value v is defined as the bag of values from all other attributes for items in the corpus whose attribute value is v. From this, the Jaccard similarity coefficient (with bag semantics, rather than set semantics) is used:

$$AIMQ(q_j, d_{ij}) = \frac{|S_{q_j} \cap S_{d_{ij}}|}{|S_{q_j} \cup S_{d_{ij}}|} \tag{5.7}$$

Missing values are treated as with *ExpandedMAUT*, and are excluded from the supertuples.

AutoRank This is the (unnamed) model of Agrawal et al [2]., motivated by the classic TF·IDF weighting scheme. They conclude that term frequency is not useful in this context, and so they rank items by the combination of inverse document frequency (IDF) estimations:

$$f(\mathbf{Q}, \mathbf{D}_i) = \sum_{j \in \mathbf{Q}} AR\left(d_{ij}, q_j\right) \tag{5.8}$$

in decreasing order. For numeric ranges, the formula used in Chapters 3 and 4 is expanded to the maximum over the range:

$$AR\left(\underline{q_j}, \overline{q_j}, d_{ij}\right) = w_j \max_{v \in [\underline{q_j}, \overline{q_j}]} \left(\exp\left[-\frac{1}{2} \left(\frac{d_{ij} - v}{h_j} \right)^2 \right] \right) \tag{5.9}$$

where the weight is

$$w_j = \log\left(\frac{n}{\sum_{k=1}^{n} \exp(-\frac{1}{2}\left(\frac{d_{kj} - q_j}{h_j}\right)^2)} \right) \tag{5.10}$$

and n is the number of items and h_j is a "bandwidth" parameter, chosen by Agrawal as $h_j = 1.06\sigma_j n^{-\frac{1}{5}}$. For nonnumeric attributes, the estimation is a more familiar version of inverse document frequency:

$$AR\left(d_{ij}, q_j\right) = \log\left(\frac{n}{\sum_{k=1}^{n} \left[d_{kj} = q_j\right]} \right) \left[d_{ij} = \ q_j\right] \tag{5.11}$$

where the brackets are the Iverson bracket. Missing values are treated as with *ExpandedMAUT*.

CQAds– *CQAds–* is similar to *AIMQ*, albeit without attribute specific weights and a different normalization. Items are ranked in decreasing score order, calculated by the linear combination of attribute scores:

$$f(\mathbf{Q}, \mathbf{D}_i) = \sum_{j \in \mathbf{Q}} CQ\left(q_j, d_{ij}\right) \tag{5.12}$$

For numeric attributes, the attribute score is calculated as:

$$CQ\left(\underline{q_j}, \overline{q_j}, d_{ij}\right) = 1 - \frac{\max\left(0, d_{ij} - \underline{q_j}, \overline{q_j} - d_{ij}\right)}{R_j} \tag{5.13}$$

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. *CQAds* [65] mined a general knowledge repository (Wikipedia) to develop a term similarity measure and included an autocorrect feature. Given the very domain-specific lexicon of the EOD, a general term similarity function is unlikely to improve performance and could hurt it; on the other hand, users often misspelt names, so a domain-sensitive spell correction feature could boost performance. In any case, we did not implement either feature in our experiment, and so refer to this model as *CQAds–*. Instead, we treat nonnumeric attributes the same as *ExpandedMAUT*, which amounts to *CQAds* with a binary string similarity function (1 given a substring match, 0 elsewise) and original spelling. Missing values are treated as with *ExpandedMAUT*.

VAGUE The *VAGUE* framework [54] uses the notion of metrics to develop a "similar-to" operator. As a framework, many approaches are supported, but none are prescribed, leaving choices to the implementor. Attribute dissimilarity scores are combined in a Euclidean distance formula:

$$f(\mathbf{Q}, \mathbf{D}_i) = \sqrt{\sum_j \left[w_j V\left(q_j, d_{ij}\right)\right]^2} \tag{5.14}$$

where w_j is a user-supplied attribute weight (uniform in our experiment) and $V()$ is the attribute dissimilarity function. For numeric attributes, we used the minimum distance to the range divided by the standard deviation:

$$V\left(\underline{q_j}, \overline{q_j}, d_{ij}\right) = \frac{\max\left(0, d_{ij} - \underline{q_j}, \overline{q_j} - d_{ij}\right)}{\sigma_j} \tag{5.15}$$

treating missing values as infinitely dissimilar. *VAGUE* presumes that other attributes are mapped to numbers, but does not give a method for doing so. Thus, we use the same approach for other attribute values as we did for *ExpandedMAUT*, scoring matches as defined by *ExpandedMAUT* with zero and mismatches (including missing values) as one standard deviation. Missing values are treated as with *ExpandedMAUT*.

ExpandedMAUT This is the expanded model described in Section 5.1.3. As with the other models, any tie is broken by the default order.

Its worth noting that all models will first return all exoplanets that exactly match the user's query. For *Boolean*, no other exoplanets are returned. Since the MAP metric never penalizes for including more items at the end of the result set, *Boolean* cannot outperform any of the other models. This does not entail that any differences will be statistically significant, though, or even that there is a difference at all. Additionally, due to the assumptions we made in this experiment, the *ScoredBoolean* and *ExpandedMAUT* models differ only in how they handle numeric attributes: thus, they will have the same performance on all queries that omit such attributes.

As with our other studies, we evaluate ranking results using mean average precision (MAP), broken down by micro-average and macro-average. The micro-average is merely the average over all responses, whereas the macro-average is an average of averages, in this case the MAP evaluation for all queries of a session. Thus, responses from each session are weighted evenly regardless of session length. We use p=0.05 as the minimum value for statistical significance, using a randomization test [71] to evaluate. In this test, the null hypothesis is that observed difference in ranking performance between models on a given response was equally likely to favor the other model, and the observed difference is only random chance. Thus, we randomly reassign the observed difference on each response (flipping or not) for all responses, calculate the randomly observed difference, and repeat this process a million times and record the fraction of randomized responses produce a difference at least as large as that observed.

There are a few quirks about the data that are worth mentioning before we discuss the results. First, as noted earlier the majority of queries are simply short "look-up" queries on the name of the exoplanet. Given the choices we made, all models will return the exactly the same results for these queries. Secondly, the data was gathered with a Boolean search engine, so in the end, users were not able to click on anything outside a Boolean search result. Since all the search engines we evaluate return this Boolean set first in the ranking, they will have the same MAP performance on terminal queries. Finally, most sessions are short, with almost a third consisting of just one query, but there is a long tail that pushes up the average, with just over three-fourths of the sessions falling below the mean length of just above six queries. As a result, differences in ranking performance are greatly diluted by the large number of queries, and the microaverage can be quite different from the macroaverage.

Table 5.3: All Queries: MAP

Model	Micro	Macro
Boolean	0.4412	0.6659
SoftBoolean	0.4506	0.6701
ScoredBoolean	0.4605	0.6743
AIMQ	0.4621	0.6737
AutoRank	0.4627	0.6755
CQAds–	0.4626	0.6755
VAGUE	0.4627	0.6755
ExpandedMAUT	**0.4644**‡	**0.6762**‡

Table 5.4: All Differing Queries: MAP

	micro		macro	
Model	Compared Baseline	Expanded MAUT	Compared Baseline	Expanded MAUT
Boolean	0.006217	**0.05967**‡	0.3196	**0.3499**‡
SoftBoolean	0.04542	**0.1919**‡	0.2496	**0.3444**‡
ScoredBoolean	0.08774	**0.1424**‡	0.2849	**0.3264**‡
AIMQ	0.09484	**0.1328**‡	0.2543	**0.3114**‡
AutoRank	0.09076	**0.1268**‡	0.2750	**0.2980**‡
CQAds–	0.06816	**0.1181**‡	0.2727	**0.2970**‡
VAGUE	0.06140	**0.1030**‡	0.2643	**0.2908**‡

Table 5.3 gives the micro- and macro- averaged MAP of the various models, with the leader bolded and a statistically significant difference against all other models indicated by the dagger (†). The *ExpandedMAUT* model outperforms the others by a statistically significant difference, though the magnitude of this difference is quite small due to identically performing queries, as noted earlier. Therefore, we present the MAP evaluation only using queries that yield different average precision evaluations over each pair of models in Table 5.4. Here, the MAP differences are much more apparent, with the better performance again bolded and statistical differences among the paired performance indicated by the

double dagger (‡). In this table, and those following that only use the differing queries, entries in the same row can be meaningfully compared but not those in the same column, as the differing queries are derived by model pair and may differ across rows. That is why the *ExpandedMAUT* evaluation changes from column to column – because the underlying set of differing queries is changes for each compared baseline.

Table 5.5: Single Attribute Queries: MAP

Model	Micro	Macro
Boolean	0.4751	0.6800
SoftBoolean	0.4833	0.6845
ScoredBoolean	0.4833	0.6871
AIMQ	0.4849	0.6883
AutoRank	0.4842	0.6881
CQAds–	0.4842	0.6881
VAGUE	0.4850	0.6882
ExpandedMAUT	**0.4851***	**0.6886***

Table 5.6: Differing Single Attribute Queries: MAP

	micro		macro	
Model	Compared Baseline	Expanded MAUT	Compared Baseline	Expanded MAUT
Boolean	0.001558	**0.02522**‡	0.3233	**0.3480**‡
SoftBoolean	0.02371	**0.1012**‡	0.2313	**0.3256**‡
ScoredBoolean	0.02371	**0.1012**‡	0.2419	**0.2949**‡
AIMQ	0.01683	**0.03009**‡	0.2254	**0.2404**
AutoRank	0.02321	**0.2012**‡	0.2477	**0.2738**‡
CQAds–	0.06584	**0.1983**‡	0.2389	**0.2704**‡
VAGUE	0.01763	**0.0513**	0.2293	**0.2573**‡

We analyze several subsets of the queries to be understand what factors might be contributing to the differences in ranking performance. Table 5.5 shows the

performance for queries on exactly one attribute, with best performance indicated as before, and the asterisk (∗) indicating a statistically significant difference versus all others save *VAGUE* for microaverage and *AIMQ* for macroaverage. Given that there is only one attribute, combining multiple criteria is not of concern and *SoftBoolean* and *ScoredBoolean* should have the same performance, which is true. The soft constraints of *SoftBoolean* and *ScoredBoolean* are apparently advantageous over the hard constraints of *Boolean*, and though not indicated in the table, the difference is statistically significant. *ExpandedMAUT* on this subset differs from *SoftBoolean* and *ScoredBoolean* only in how it handles numeric attributes, and this is enough for a statistically significant albeit small difference. Table 5.6 zooms in on the subset of queries that have differing ranking evaluations, which explains the difference: the Boolean models have rather poor performance on this subset, whereas the *ExpandedMAUT* model has more reasonable performance. The other models performed better than the Boolean models as well, though not as well as *ExpandedMAUT*. In this table, statistically significant difference against all other models are indicated by the dagger (†).

Table 5.7: Multiple Attribute Queries: MAP

Model	Micro	Macro
Boolean	0.2624	0.3850
SoftBoolean	0.2787	0.3959
ScoredBoolean	0.3401	0.4294
AIMQ	0.3422	0.4195
AutoRank	0.3493	0.4347
CQAds–	0.3484	0.4353
VAGUE	0.3453	0.4325
ExpandedMAUT	**0.3556**‡	**0.4385**‡

Table 5.7 analyzes the complementary set of queries, those queries that use multiple attributes, noting the leader and statistical significance as in Table 5.3.

Table 5.8: Differing Multiple Attribute Queries: MAP

	micro		macro	
Model	Compared Baseline	Expanded MAUT	Compared Baseline	Expanded MAUT
Boolean	0.02462	**0.1958**‡	0.2244	**0.3195**‡
SoftBoolean	0.05067	**0.2138**‡	0.2475	**0.3420**‡
ScoredBoolean	0.1098	**0.1566**‡	0.2891	**0.3191**‡
AIMQ	0.1141	**0.1581**‡	0.2596	**0.3210**‡
AutoRank	0.09674	**0.1202**‡	0.2773	**0.2925**‡
CQAds–	0.07481	**0.1094**‡	0.2739	**0.2889**‡
VAGUE	0.06179	**0.1034**‡	0.2637	**0.2902**‡

As expected, performance on these queries are generally lower, and *ScoredBoolean*'s ranking stratification by the number of satisfied constraints improves performance over *SoftBoolean* by a statistically significant difference. *ExpandedMAUT* again has the best performance, however, and this difference is magnified by evaluating the queries with differing ranking performance in Table 5.8.

Table 5.9: Queries using Numeric Attributes: MAP

Model	Micro	Macro
Boolean	0.2957	0.4234
SoftBoolean	0.3038	0.4297
ScoredBoolean	0.3348	0.4460
AIMQ	0.3426	0.4418
AutoRank	0.3465	0.4543
CQAds–	0.3456	0.4543
VAGUE	0.3467	0.4540
ExpandedMAUT	**0.3554**‡	**0.4592**‡

Finally, Tables 5.9 and 5.11 show the performance on queries using and not using numeric attributes, respectively, with the results on the differing queries given in Tables 5.10 and 5.12. The asterisk (∗) indicates a statistically significant differ-

Table 5.10: Differing Queries using Numeric Attributes: MAP

	micro		macro	
Model	Compared Baseline	Expanded MAUT	Compared Baseline	Expanded MAUT
Boolean	0.02806	**0.1646**‡	0.2276	**0.3132**‡
SoftBoolean	0.04687	**0.1666**‡	0.2483	**0.3303**‡
ScoredBoolean	0.08774	**0.1424**‡	0.2849	**0.3264**‡
AIMQ	0.09464	**0.1351**‡	0.2543	**0.3130**‡
AutoRank	0.08955	**0.1261**‡	0.2720	**0.2940**‡
CQAds–	0.06816	**0.1181**‡	0.2727	**0.2970**‡
VAGUE	0.06140	**0.1030**‡	0.2643	**0.2908**‡

Table 5.11: Queries not using Numeric Attributes: MAP

Model	Micro	Macro
Boolean	0.4751	0.6905
SoftBoolean	0.4848	0.6949
ScoredBoolean	**0.4898**	**0.6981**
ExpandedMAUT	**0.4898***	**0.6981***

ence versus *Boolean* and *SoftBoolean*, but not *ScoredBoolean*, with other markings as before. The difference between *ScoredBoolean* and *ExpandedMAUT* shows the advantage of using Eq. 5.2 over a Boolean approach for numeric attributes. In contrast, given our choices *ScoredBoolean* and *ExpandedMAUT* are identical in the absence of numeric attributes, which is reflected in Tables 5.11 and 5.12. The performance of the non-Boolean baselines are likewise nearly identical to *Scored-Boolean* and *ExpandedMAUT* and are without statistically significant differences, and so are not included in these tables.

Table 5.12: Differing Queries not using Numeric Attributes: MAP

	micro		macro	
Model	Compared Baseline	Expanded MAUT	Compared Baseline	Expanded MAUT
Boolean	0.001081	**0.035**‡	0.3266	**0.3485**‡
SoftBoolean	0.0361	**0.3543**‡	0.2254	**0.3223**‡
ScoredBoolean	–	–	–	–

5.2 Moving Beyond Structured Search Interfaces

In this section, we go beyond the search interface to evaluate our retrieval concept with a more natural representation of the user need. Structured search interfaces force the user to adhere to the accepted input (e.g., constraints and sort orders for sorted boolean search). Thus, the evaluation of such search engines is measuring both the efficacy of the retrieval model given the query, and the impact of translating the user need into the query format. Can the cost of this translation be eliminated?

NASA has long relied on database-like query languages or form-based search (faceted or user-specified Boolean constraints) to allow researchers to search through its vast data holdings (see the PDS Image Search[1] as typical example). So far, our search interfaces have followed the same design, even our MAUT-based search, as the user can only search by adhering to the structure given in the presented form-based interface.

Although this is not an uncommon approach for searching through data, less specialized search interfaces supporting other tasks are more free-form: popular search engines like Google[2] and Bing[3] use an open-ended text entry box; question-

[1] https://pds-imaging.jpl.nasa.gov/search/
[2] https://www.goooogle.com
[3] https://www.bing.com

answering has been an active research topic for years [58]; and recent advances in natural language processing (NLP) has lead to intelligent assistants such as Apple's Siri [7] and Amazon's Alexa [4], not to mention various chatbots and conversional recommenders available on the World Wide Web. In addition to ease-of-use, one advantage of the NLP approach for search is that it does not impose a rigid query structure on the user, so that queries may presumably be stated in a manner that more accurately reflects the latent user need. Would item search be improved by leveraging these new developments?

5.2.1 Search Interface and Data Used

To answer these questions, we set up a new search engine that accepts natural language queries entered into a text box. On the back end, we used SEMPRE (Semantic Parsing and Execution) [10, 49], an open source toolkit that translates natural language into logical forms, to convert the user queries into something executable in our retrieval framework. A previous application of SEMPRE used allrecipes.com data [79], as we also did in our experiment in Chapter 4, so we used that same experiment structure with the hope of reusing the publically available grammar developed for that previous application of SEMPRE. We briefly review our prior experiment below, with more detail available in Section 4.2.3.

We developed twenty short nutritionally-based core scenarios from the literature, describing someone (often with a medical condition) and their nutritional needs for one day. Recommended levels of each relevant nutrient was given, as well as a range consisting of the Estimated Average Requirement and Tolerable Upper Limit [81] when such are defined. The test subjects' task was to chose an appropriate meal plan (one day's worth of meals) for the person described in the scenario. We developed the corpus by using recipes from allrecipes.com as the

meal components. We wrote a meal plan generator that created approximately a quarter million meal plans, describing each by total nutritional information (e.g., calories, vitamin A, etc.), as well as component dish names, ratings, and photos when available.

In the end, the publicly available SEMPRE grammar that we had hoped to use was not adequate for our experiment, so we developed our own grammar. SEMPRE grammars are developed by associating phrases with logical forms or executable code. Typically, these phrases are short, but can refer to other phrases, and through this composition complex phrases may be successfully parsed. Nonetheless, the designer of the grammar must anticipate what sorts of statements will be encountered. SEMPRE has some useful natural language processing but cannot parse arbitrary statements into the correct translation.

As before, we used Amazon Turk to run the experiment [3]. We accepted the results of 69 test subjects before terminating the experiment. We had assumed that since the domain and data was the same, the natural language queries would resemble the sort of queries that had been given in our form-based interface of the prior experiment (Section 4.2.3). This assumption was quite incorrect, and as a result, the search engine did a poor job of satisfying the test subject's intent, with less than a third of the queries parsing correctly. We had assumed that the test subjects would provide exact numeric values, but in fact the majority of queries were open to interpretation. We classify the query constraints encountered into five types below:

- *Range*. This is the type of constraint we had built our grammar to handle, where the specific numeric values are given. Ranges could be bounded both low and high, or just on one side (e.g., *less than 2000 calories*, with no lower limit given).

- *Equal*. The quantity is given as a specific value (e.g., *2500 calories*). Although this appears to be a precise value, that may not match the user intent well. After all, few if any items will have exactly the specified value.

- *Low* and *High* (e.g., *high calcium*). The intent behind such constraints seems relatively clear, though exactly how it should be expressed numerically is not (what quantity is high)? In addition, extreme values may not be what the user desires, particularly since excessive nutrition may have negative side effects (for example, high calcium may contribute to kidney stones).

- *More* and *Less* (e.g., *less calories*). The intent here is less clear, as the other element of the comparison is not stated (less than what?). One possibility is that this is reference to earlier results, but these constraints often arise in the first query of the session, when there is no prior search results. Instead, it may be that less is analogous to low, and more to high.

- *Unspecified* (just a nutrient, e.g., *vitamin A*). The intent of such queries is quite uncertain. It could indicate a desire for items with significant amounts of the nutrient, or simply that it is greater than zero.

Table 5.13 gives the co-occurence of these constraints types on the test subjects' first queries, as well as the overall frequency on the diagonal. For the most part, only one type of constraint was used per query– indeed, most queries only had a single constraint.

5.2.2 Handling Imprecise Constraints

When we devised the grammar, we started with the query forms we knew how to handle, and developed natural language phrasings that would correspond with those queries. However, even given the focus of the scenarios, the test subjects

Table 5.13: Constraint Category Co-occurence on Initial Queries

	range	highlow	moreless	equal	unspecified
range	21	0	0	1	0
highlow	0	24	0	3	0
moreless	0	0	4	0	0
equal	1	3	0	11	0
unspecified	0	0	0	0	11

favored different query forms whose intent is not clear, as detailed in the prior section. For an effective natural language query capability, we must solve the inverse problem, taking the users' queries and translating that into something executable. So how should these imprecise constraints be interpreted?

The problem decomposes on two dimensions. When an exact value is given, as was the case with "range" and "equal" queries, a fairly precise user need is described relating to that value. The question is whether it is a necessary requirement, or something the user might compromise on given other considerations. When no value is given, the question has an additional dimension. Is there an unspecified value that should be used, for example, an acceptable minimum value? If so, we still must ask if this a hard constraint or not. If not so, then it would imply a strict ordering, i.e., for *high calcium*, the higher calcium choice would be preferred among any pair, with the highest calcium option preferred overall.

Interestingly, these possible interpretations easily lead us back to now familiar retrieval models, as we detail below:

Boolean Like the Boolean model presented earlier in this chapter, all constraints are treated as necessary conditions and without any sort order. The "range" and "equal" constraints are taken as is, with only items meeting those constraints returned. For "highlow", "moreless", and "unspecified" constraints,

we arbitrarily choose some specific value to function as a range constraint. We used the 10^{th} percentile for a given attribute for "low", and similarly the 90^{th} percentile for "high". These are not necessarily optimal values for all attributes, but we note other percentiles we tried did not offer better performance. Furthermore, we interpreted "less" the same as "low" (i.e., 10^{th} percentile or less), and "more" and "unspecified" the same as "high" (i.e., 90^{th} percentile or higher).

SortedBoolean Analogous to the *SortedBoolean* models utilized in Chapters 3 and 4, this model combines hard constraints with a sort orders. Like *Boolean* above, the "range" and "equal" constraints are taken as is, with only items meeting those constraints returned. However, for "highlow", "moreless", and "unspecified" constraints, no query value is assumed. Instead, these are interpreted as sort orders, with "high", "more", and "unspecified" interpreted as a sort from high to low, and "low" and "less" interpreted as a sort from low to high. Multiple sort orders are applied in an arbitrary order, but in fact this has little effect on the results as very few queries had more than one such constraint.

ExpandedMAUT Finally, the *ExpandedMAUT* uses the same range interpretations as *Boolean* above (using percentiles as appropriate), but interprets these constraints as soft constraints. Specifically, the query results are ranked according to Equations 5.1 and 5.2, as described in Section 5.1.3. In addition, for "highlow", "moreless", and "unspecified" constraints, we hedge our bets by combining Eq. 5.2 with a sigmoid function below:

$$g_j^{lo}(\underline{q_j}, \overline{q_j}, d_{ij}) = \frac{g_j(\underline{q_j}, \overline{q_j}, d_{ij})}{1 + \exp(\frac{d_{ij} - \underline{q_j}}{\sigma_j})} \tag{5.16}$$

$$g_j^{hi}(\underline{q_j}, \overline{q_j}, d_{ij}) = \frac{g_j(\underline{q_j}, \overline{q_j}, d_{ij})}{1 + \exp(\frac{\overline{q_j} - d_{ij}}{\sigma_j})} \tag{5.17}$$

where $g_j^{lo}()$ is applied when lower values are desired, $g_j^{hi}()$ otherwise, with quantities defined as before. These functions establish a preference for lower and higher values, respectively, and will give the same ranking as *Sorted-Boolean* for single constraint queries. The ranking for multiple constraint queries will likely differ, however, as *SortedBoolean* is a noncompensatory model and *ExpandedMAUT* is not.

5.2.3 Results

We evaluate the performance of the models above retrospectively, using the micro- and macro- average of MAP and evaluating statistical significance using a randomization test, as we have done earlier in this chapter and throughout. We again turn to the *community induced* MAP of our earlier chapters. The test subject's selections are influenced by which results they see, particularly in a large corpus where only a tiny fraction of the corpus will be seen. Therefore, we use the selections of *other* test subjects on the same scenario – in other words, evaluating the same user need – to avoid bias. However, in this case the test subject's selections were likely compromised by search engine's failure to properly interpret the natural language query, as noted in Section 5.2.1, so we discard the

selections from the NLP user study entirely and instead evaluate each NLP query with the selections from each test subject in the analogous experiment in Chapter 4. Given that each query may be evaluated on several times, the macro-average is over queries rather than scenarios.

Table 5.14: Community MAP

	Boolean		SortedBoolean		Expanded MAUT	
Model	micro	macro	micro	macro	micro	macro
All	0.085	0.081	0.152	0.156	**0.217**†	**0.209**†
Range	0.103	0.113	0.103	0.113	**0.232**†	**0.230**†
Equal	0.001	0.001	0.005	0.011	**0.279**†	**0.242**†
Highlow	0.097	0.090	0.206	0.197	**0.250**†	**0.238***
Moreless	0.078	0.057	0.163	0.174	**0.165**	**0.176**
Unspecified	0.006	0.009	**0.178**	**0.202**	0.161*	0.173*

Table 5.14 gives the results of our experiment, with the leader in each row bolded and statistically significant differences versus the other two models indicated with the dagger (†), and a statistically significant difference only against *Boolean* indicated by the asterisk (∗), evaluated only for *ExpandedMAUT*. The first row gives the performance over the entire dataset, with *ExpandedMAUT* having the best performance and by a statistically significant difference. Overall, the interpretation of the queries by *ExpandedMAUT* is superior to the others, but the performance varies by category of query, so we also give the results separated by query category in the latter rows. Sometimes multiple categories occurred in the same query as noted in Table 5.13, which adds some noise to these results.

The *Boolean* and *SortedBoolean* models interpreted precise values identically, as hard constraints, and have very similar performance on both categories, with differences due to the inclusion of other categories in the query. Consistent with our earlier experiments, this performance is inferior to that of *ExpandedMAUT*.

The performance of queries with "equal" constraints for *Boolean* and *Sorted-Boolean* is quite poor, giving strong support for the position that users are not seeking exact values in such cases, but something around the query value. This, too, is consistent with our earlier results. Interestingly, *ExpandedMAUT* has its best performance with these type of queries.

The *Boolean* and *ExpandedMAUT* models treated "highlow" and "moreless" also as "range" queries, using the 10^{th} percentile or 90^{th} percentile values, as appropriate, as the implied precise range endpoints. In contrast, *SortedBoolean* treated these as sort orders instead of ranges. *ExpandedMAUT* again mostly had the best performance, but the difference was statistically significant against both models only for the "highlow" microaverage. In contrast, the *SortedBoolean* model actually had the best performance overall; however, the difference versus *ExpandedMAUT* is not significant.

5.2.4 Summary and Contributions

In this chapter, we extended the model in two ways. In the first part, we showed how our retrieval method could be expanded to handle additional attribute types, namely Boolean, enumerated, and textual attributes. In addition, we expanded the numeric framework to handle ranges instead of just point values. We compared this expanded version to Boolean search, also including a couple of soft Boolean variants to isolate what effects our various assumptions have on performance, in addition to our usual baselines. In contrast to our previous studies, we evaluated our model with the queries of actual users of a space science vertical search website rather than in a controlled user study.

Nonetheless, our results largely confirm what we have observed in our prior studies. The Boolean retrieval model has the worst performance overall, as eval-

uated on MAP. A "soft" Boolean model that ranks non-matching results after matching results performs better; a model that further ranks these by the number of matching constraints performs better still. However, all are bested by the utility-based *ExpandedMAUT* model. Though this is in line with our previous findings, it is nonetheless surprising to see in a scientific domain. Our previous studies have been conducted in consumer domains, where it is reasonable to assume that the users is willing to compromise on some aspects and maximize utility. Science, however, is presumably much more objective, with crisp delineations and not subject to the whims of humans. Nonetheless, we find that hard constraints are still a poor model, and that more flexible models perform better. Perhaps the Boolean model still is the best model for some uses, but where?

In the second part, we examined how to support natural language queries so that we could get a more accurate description of the user need without imposing a particular query structure, again using the meal plan domain from Chapter 4. We had hoped to have test subjects use a natural language interface to query directly, so we developed a grammar to handle natural language rephrasings of the queries we captured earlier using the form-based interface. However, once free of the structure imposed by the search form, most users chose to query in unanticipated ways that our parser could not handle.

We analyzed the types of constraints the test subjects submitted and found that they fell into five categories, of which only two had been anticipated. The intent of most of these clauses was not clear, and so we speculated on possible interpretations. Interestingly, these different interpretations readily led to retrieval models we had experimented with before: *Boolean*, *SortedBoolean* and *ExpandedMAUT*. We retrospectively evaluated the queries with these models and found *ExpandedMAUT* performed the best, with a statistically significant difference overall

and on most categories.

One additional observation of note is that the retrieval performance was markedly lower than with the form-based interface on the exact same user needs. This may be the fault of the search engine, particularly if there are problems with the interpretation as described above. The other possibility lies with the users. It may simply be due to a lack of experience, as users have far more experience with form based queries. Or it may be that the structure of the form forces the user to think about the problem in a beneficial way. Or, conversely, it may be that a natural language interface encourages bad habits. The queries tended to be less detailed (fewer clauses), perhaps because of the additional typing needed, and often introduced imprecision, which may not have been beneficial. We leave exploring these considerations to future work.

To summarize, we have made the following contributions in this chapter:

- Expanded our retrieval model to handle non-numeric attribute types, as well as numeric ranges, dubbed *ExpandedMAUT*.
- We evaluated the performance *ExpandedMAUT* using the queries and click-through data gather from an active vertical space science engine. We compared this performance to the Boolean model used in search engine, along with two enhancements of the Boolean model.
- We collected natural language queries for the nutrition domain introduced in Chapter 4, identified several unexpected categories of imprecise clauses.
- We formulated possible interpretations of these imprecise clauses that naturally lead to the *Boolean*, *SortedBoolean* and *ExpandedMAUT* models we had used earlier. We evaluated the performance of these interpretations retrospectively on data gathered from the user study.

Chapter 6

Conclusion and Future Work

In this chapter, we summarize the main contributions of the dissertation, and identify several opportunities for future work.

6.1 Contributions

We identify contributions to the main focus of this dissertation, supporting item retrieval, as well as secondary contributions, in the following section.

6.1.1 Contributions to Items Retrieval

The main contribution of this dissertation is a method to rank items in response to a user's query, thus making the item retrieval problem more akin to information retrieval than database retrieval, as is the norm at the time of writing. We can break this down into several supporting contributions.

The first contribution is to develop the general framework for the item ranking problem, which we did in Chapter 3. We chose to do this by casting the problem as utility estimation, estimating how well an item satisfies the user's query. Accordingly, we defined the query to be a (potentially partial) specification of

the desired item's attribute values, rather than a set of constraints. Given this, we used a straightforward mapping to criteria ratings, or subutilities, to generate ratings on each attribute query value. Finally, we adapted principles of multi-attribute utility theory to combine multiple attribute subutilities on each item into a single utility estimation.

The second contribution is to develop a model of per attribute subutility, which we did for numeric attributes in Chapter 4. After identifying several desirable properties of a subutility function, we designed a flexible subutility function that met these properties and could assume a wide variety of forms with only a few parameters. We used this improved subutility function in our utility estimation framework, embedding the model in a Bayesian hierarchical model so we could infer parameter values from training data.

The third contribution is to expand the model so it could be applied to a wide variety of item retrieval settings, which we did in Chapter 5. Having focused only on the difficult challenge of numeric attributes, we extended the model to handle other attribute types. We also extended the model to handle imprecise query values, which we observed from users of a natural language query facility.

6.1.2 Contributions Beyond Item Retrieval

In addition to our main goal of supporting item retrieval, portions of this dissertation may also benefit other areas.

Contribution to Information Retrieval We developed a method to estimate the utility of an item based on a query consisting of desired attribute values. However, the framework in general and could be applied to resolve other multiple objective problems. For instance, many of the techniques could be used to incorporate additional considerations when retrieving information,

for instance recency, novelty, etc. Indeed, in many ways this work was inspired by earlier research into addressing such concerns.

Contribution to Multi-criteria Decision Making Theory Our utility estimation method is rooted in multi-criteria decision making theory (MCDM) and could be considered a MCDM algorithm in its own right. MCDM techniques take as input ratings over multiple criteria for a set of options to consider. Manually rating each option on each criterion is a difficult and time-consuming process, which raises cost of using such methods and limits the problems on which they can be applied. Since we estimate the ratings (subutility) directly from the attribute values, our techniques could be used to apply MCDM approaches to a wider range of problems.

6.2 Opportunities for Future Research

Finally, we identify several opportunities for future work, involving possible improvements to our core utility framework, as well as new aspects that could improve performance.

6.2.1 Improving the Core Framework

Throughout this work, we assumed mutual utility independence, an assumption that was never tested. Although intuitive and appealing, it is difficult to imagine situations where it might not apply, for instance given "deal-breaking" attribute values or attributes that interact in some manner. Standard methods exist for establishing or refuting mutual utility independence [28], but they assume consistency and a testable decision maker. Instead, one could ask how often the problem is correctly solved under an mutual utility independence assumption.

An inspection of the meal plan results in Chapter 4 show that the model solved the problem just over a quarter of the time (i.e., correctly ranked the desired item first), but perfection is generally rare in retrieval problems. Another possible measure is how often does this assumption produce no worse than the current standard of Boolean retrieval, either within a domain or across domains. Or one could simply test the assumption directly, for instance by comparing performance to a model that does not make this assumption, similar to our approach to testing subutility linearity in Chapter 4. In any case, should mutual utility independence be abandoned, a new model would need to be chosen. Such would diverge from MAUT, so a more complex MCDM model could be adopted, such as the weighted quadratic approach [29]. However, at this point we may be best served by beyond the confines of MCDM, considering other function estimation or learning-to-rank models while keep the general MCDM principals (if not approaches) in mind.

We also assumed that numeric subutility was invariant with regard to value location, and used a parameterized model of subutility. The first assumption is most likely incorrect: for example, a $5 discount on a $10 pizza is exciting, but a $5 discount on a $1,000,000 house is not, even though the discount is the same in both cases. Though our parameterized subutility formula is flexible, it nonetheless cannot exactly fit any function, and may miss subtle features of the underlying subutility. A universal function approximator could capture such subtle features, though such would presumably require greater amounts of training data. Finally, we had assumed specific interpretations for the imprecise query values we observed in natural language queries. Similarly, more training data could allow enable learning of global or attribute-specific substitute values, or other possible interpretations.

6.2.2 Extending the Framework

There are also opportunities to incorporate new elements into the utility estimation model. One appealing avenue is personalization. It's easy to imagine that different people will value particular attributes differently, for instance, in product search, one person might put a greater emphasis on price, while another might put more emphasis on ease of use. Personalization could easily be incorporated into a Bayesian graphical model by incorporating latent variables for each person, which are then influenced by the global latent variables currently in the model.

In addition to personalization, global preferences are likely to improve performance. Upon examination of the queries in our user studies, it was evident that many of the queries were underspecified. For instance, in an e-commerce domain, it would be reasonable to assume that lower prices are preferable, even when no price was provided in the query. This, too, could easily be incorporated into our Bayesian framework, with an additional latent variable to represent the unexpressed query value along with the other model parameters. Nor would this global preference need to be truly global; as above, personalized values could be learned.

There are also ample opportunities to use domain knowledge to improve results. Although we are staunch believers in letting the data guide us, this does not preclude us from also using domain knowledge as a guide. Indeed, domain knowledge led us to explore asymmetric subutility functions in the airline booking experiment in Chapter 3, for instance reasoning that fares under the price point were not as bad as those above it. Domain knowledge could lead to new subutility functions, for instance, those with periodicity given diurnal schedules, or relationships among criteria, for instance nutrients that are typically needed in concert. Knowledge bases or ontologies may be used to complement intuition

or establish it in unfamiliar domains. The exoplanet domain explored in Chapter 5 has a particularly compelling possible area of exploration. We observed during search log analysis that users sometimes entered references to other objects in attribute fields, for instance Earth, in fields requiring other input, such as exoplanet mass. A domain knowledge-enhanced search engine that could recognize and resolve such entity references would likely be a great benefit, even more so if natural language queries were accepted.

The impact of the user interface warrants more study. Throughout this dissertation, we found that the MAUT-based model consistently provided better retrieval performance than the standard Boolean (or faceted) model. Despite this, when users were asked to evaluate the interfaces, the MAUT-based models did not rate significantly better than the Boolean approach. Is this due to familiarity, or is it because the Boolean result set is well defined and therefore easier to understand? We also found that users employed vague constructs when presented with an open-ended natural language interface in Chapter 5. Was this a better expression of the underlying user need? Is it more compatible with the way people think? Or is it because it fits colloquial parlance, which tends to eschew numeric constructs? The natural language queries tended to perform worse than the form-based queries, which could have several possible explanations. It could be from a loss of specification, as the queries tended to be less detailed, perhaps to avoid a lot of typing. The introduction of vagueness into the query may have hurt performance. Our interpretation of the vague queries may need further improvement. Or it may be that imposing a certain structure onto the query, for instance with a form, may suggest more effective query strategies, even if this is a less natural expression of the user need. These remain open questions.

Finally, we restricted the query attribute values to single values, or for numeric

attributes in our final extension of the model, a continuous range of values. In some cases, it may be better to accept multiple values, or to allow for multi-modal utility functions. As an example, some attributes such as time will have a periodic quality, which may more accurately modelled with a multi-modal utility function. Indeed, the form of the query input remains an open question, and users find it natural to use expressions we have not anticipated, as we observed in our natural language experiment.

Bibliography

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

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

[3] O. Alonso and S. Mizzaro. Can we get rid of trec assessors? using mechanical turk for relevance assessment. In *Proceedings of the SIGIR 2009 Workshop on the Future of IR Evaluation*, 2009.

[4] Amazon.com, Inc. https://developer.amazon.com/alexa. https://developer.amazon.com/alexa, accessed on March 13, 2018.

[5] S. Amer-Yahia, F. Bonchi, C. Castillo, E. Feuerstein, I. Mendez-Diaz, and P. Zabala. Composite retrieval of diverse and complementary bundles. *Knowledge and Data Engineering, IEEE Transactions on*, 26(11):2662–2675, Nov 2014.

[6] Albert Angel, Surajit Chaudhuri, Gautam Das, and Nick Koudas. 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, pages 910–921, New York, NY, USA, 2009. ACM.

[7] Apple Inc. https://www.apple.com/ios/siri/. https://www.apple.com/ios/siri/, accessed on March 13, 2018.

[8] Ricardo A. Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1999.

[9] Senjuti Basu Roy, Sihem Amer-Yahia, Ashish Chawla, Gautam Das, and Cong Yu. Constructing and exploring composite items. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data*, SIGMOD '10, pages 843–854, New York, NY, USA, 2010. ACM.

[10] Jonathon Berant and Percy Liang. Semantic parsing via paraphrasing. In *Association for Computational Linguistics (ACL)*, 2014.

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

[12] Horatiu Bota, Ke Zhou, Joemon M. Jose, and Mounia Lalmas. Composite retrieval of heterogeneous web search. In *Proceedings of the 23rd International Conference on World Wide Web*, WWW '14, pages 119–130, New York, NY, USA, 2014. ACM.

[13] Jean-Pierre Brans and Bertrand Mareschal. PROMETHEE methods. In José Figueria, Salvatore Greco, and Matthias Ehrgott, editors, *Multicriteria Decision Analysis: State of the Art Surveys*, pages 163–195. Springer, 2005.

[14] Bureau of Transportation Statistics. Airline on-time performance data. http://www.transtats.bts.gov/Tables.asp?DB_ID=120, accessed on April 15, 2007.

[15] Bureau of Transportation Statistics. Airline origin and destination survey. http://www.transtats.bts.gov/Tables.asp?DB_ID=125, accessed on April 15, 2007.

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

[17] Vannevar Bush. As we may think. *The Atlantic*, July 1945.

[18] Kurt A. Carlson and Lisa Klein Pearo. Limiting predecisional distortion by prior valuation of attribute components. *Organizational Behavior and Human Decision Processes*, 94(1):48–59, May 2004.

[19] Michael A. Casey, Remco Veltkamp, Masataka Goto, Marc Leman, Christophe Rhodes, and Malcolm Slaney. Content-based music information retrieval: Current directions and future challenges. *Proceedings of the IEEE*, 96:668–696, April 2008.

[20] Olivier Chapelle and Mingrui Wu. Gradient descent optimization of smoothed information retrieval metrics. *Information Retrieval*, 13(3):216–235, June 2010.

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

[22] Surajit Chaudhuri, Raghu Ramakrishnan, and Gerhard Weikum. Integrating DB and IR technologies: What is the sound of one hand clapping. In *Proceedings of the Second Biennial Conference on Innovative Data Systems Research (CIDR 2005)*, pages 1–12, 2005.

[23] Jan Chomicki. Logical foundations of preference queries. *IEEE Data Eng. Bull.*, 34(2):3–10, 2011.

[24] Ritendra Datta, Dhiraj Joshi, Jia Li, and James Z. Wang. Image retrieval: Ideas, influences, and trends of the new age. *ACM Computing Surveys*, 40:1–60, 2008.

[25] Paul Demery. Most new e-commerce platforms designed with 'faceted search', Venda says. http://www.internetretailer.com/2007/05/16/most-new-e-commerce-platforms-designed-with-faceted-search, May 2007. Accessed on March 11, 2013.

[26] Anja Dieckmann, Katrin Dippold, and Holger Dietrich. Compensatory versus noncompensatory models for predicting consumer preferences. *Judgment and Decision Making*, 4(3):200–213, April 2009.

[27] S. Doan and S. Horiguchi. An efficient feature selection using multi-criteria in text categorization. In *Fourth IEEE International Conference on Hybrid Intelligent Systems*, pages 86–91, 2004.

[28] James S. Dyer. MAUT – multiattribute utility theory. In José Figueria, Salvatore Greco, and Matthias Ehrgott, editors, *Multicriteria Decision Analysis: State of the Art Surveys*, pages 266–295. Springer, 2005.

[29] Matthias Ehrgott and Margaret M. Wiecek. Multiobjective programming. In José Figueria, Salvatore Greco, and Matthias Ehrgott, editors, *Multicriteria Decision Analysis: State of the Art Surveys*, page 678. Springer, 2005.

[30] M. Farah and D. Vanderpooten. A multicriteria paradigm of relevance for the web information retrieval problem. In *IEEE Sciences of Electronic, Technologies of Information and Telecommunications*, 2005.

[31] M. Farah and D. Vanderpooten. An outranking approach for rank aggregation in information retrieval. In *30th annual international ACM SIGIR conference on research and development in information retrieval*, pages 591–598, 2007.

[32] José Figueria, Vincent Mousseau, and Bernard Roy. ELECTRE methods. In José Figueria, Salvatore Greco, and Matthias Ehrgott, editors, *Multicriteria Decision Analysis: State of the Art Surveys*, pages 133–162. Springer, 2005.

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

[34] Norbert Fuhr. Salton award lecture information retrieval as engineering science. *SIGIR Forum*, 46(2):19–28, December 2012.

[35] Michael Grossniklaus and Moira Norrie. ETH Zürich, lecture notes: Object-oriented databases (version 2010), 2010. URL: http://www.odbms.org/2010/01/object-oriented-databases-version-2010 Last visited on 2018/11/15.

[36] E. Han, S. X. Wang, J. T. Wright, Y. K. Feng, M. Zhao, O. Fakhouri, J. I. Brown, and C. Hancock. Exoplanet Orbit Database. II. Updates to Exoplanets.org. *Publications of the Astronomical Society of Pacific*, 126:827, September 2014.

[37] Marti A. Hearst. Next generation web search: Setting our sites. *IEEE Data Engineering Bulletin*, 23, 2000.

[38] Daniel Herzog and Wolfgang Wörndl. 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.*, pages 42–48, 2014.

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

[40] Ihab F. Ilyas, George Beskales, and Mohamed A. Soliman. A survey of top-k query processing techniques in relational database systems. *ACM Comput. Surv.*, 40(4):11:1–11:58, October 2008.

[41] Daniel Kahneman. Maps of bounded rationality: Psychology for behavioral economics. *American Economic Review*, 93(5):1449–1475, December 2003.

[42] Neda Kerimi, Henry Montgomery, and Dan Zakay. Coming close to the ideal alternative: The concordant-ranks strategy. *Judgment and Decision Making*, 6(3):196–210, 2011.

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

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

[45] Mounia Lalmas and Anastasios Tombros. Evaluating XML retrieval effectiveness at INEX. *SIGIR Forum*, 41:40–57, June 2007.

[46] L. Lamontagne and I. Abi-Zeid. Combining multiple similarity metrics using a multicriteria approach. In *Eighth European Conference on Case-Based Reasoning*, 2006.

[47] Michael S. Lew, Nicu Sebe, Chabane Djeraba, and Ramesh Jain. Content-based multimedia information retrieval: State of the art and challenges. *ACM Trans. Multimedia Comput. Commun. Appl.*, 2(1):1–19, February 2006.

[48] Jane Li, Scott Huffman, and Akihito Tokuda. Good abandonment in mobile and pc internet search. In *Proceedings of the 32Nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '09, pages 43–50, New York, NY, USA, 2009. ACM.

[49] Percy Liang. https://nlp.stanford.edu/software/sempre/. https://nlp.stanford.edu/software/sempre/, accessed on March 13, 2018.

[50] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. Boolean retrieval. In *Intoduction to Information Retrieval*, pages 1–18. Cambridge University Press, 2008.

[51] N. Manouselis and C. Costopoulou. Analysis and classification of multi-criteria recommender systems. *World Wide Web*, 10(4):415–441, 2007.

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

[53] Patrick Meyer and Marc Roubens. Choice, ranking and sorting in fuzzy multiple criteria decision aid. In José Figueria, Salvatore Greco, and Matthias Ehrgott, editors, *Multicriteria Decision Analysis: State of the Art Surveys*, pages 471–506. Springer, 2005.

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

[55] Amihai Motro. A trio of database user interfaces for handling vague retrieval requests. *IEEE Data Engineering Bulletin*, 12:54–63, 1989.

[56] Ullas Nambiar. *Answering Imprecise Queries Over Autonomous Databases*. PhD thesis, Arizona State University, December 2005.

[57] Ullas Nambiar and Subbarao Kambhampati. Answering imprecise queries over web databases. In *Proceedings of the 31st International Conference on Very Large Data Bases*, VLDB '05, pages 1350–1353. VLDB Endowment, 2005.

[58] National Institute of Standards and Technology (NIST). http://trec.nist.gov/data/qamain.html. http://trec.nist.gov/data/qamain.html, accessed on March 13, 2018.

[59] William Neilson and Jill Stowe. A further examination of cumulative prospect theory parameterizations. *Risk and Uncertainty*, 24:31–46, January 2002.

[60] Aditya Parameswaran, Petros Venetis, and Hector Garcia-Molina. Recommendation systems with complex constraints: A course recommendation perspective. *ACM Trans. Inf. Syst.*, 29(4):20:1–20:33, December 2011.

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

[62] Aditya G. Parameswaran, Hector Garcia-Molina, and Jeffrey D. Ullman. Evaluating, combining and generalizing recommendations with prerequisites. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, CIKM '10, pages 919–928, New York, NY, USA, 2010. ACM.

[63] Gabriella Pasi, Gloria Bordogna, and Robert Villa. A multi-criteria content-based filtering system. In Wessel Kraaij, Arjen P. de Vries, Charles L. A. Clarke, Norbert Fuhr, and Noriko Kando, editors, *SIGIR '07: Proceedings of the 30th annual international ACM conference on research and development in information retrieval*, pages 775–776, New York, NY, USA, 2007. ACM.

[64] Tao Qin, Tie-Yan Liu, and Hang Li. A general approximation framework for direct optimization of information retrieval measures. *Information Retrieval*, 13(4):375–397, August 2010.

[65] Rani Qumsiyeh, Maria S. Pera, and Yiu-Kai Ng. Generating exact- and ranked partially-matched answers to questions in advertisements. *Proc. VLDB Endow.*, 5(3):217–228, November 2011.

[66] Carol Ann Rinzler. *Nutrition for Dummies*. For Dummies, second edition, 1999.

[67] Bernard S. Roy. Paradigms and challenges. In José Figueria, Salvatore Greco, and Matthias Ehrgott, editors, *Multicriteria Decision Analysis: State of the Art Surveys*, pages 3–24. Springer, 2005.

[68] Thomas L. Saaty. The analytical hierarchy and analytic network processes for the measurement of intangible criteria and for decision-making. In José Figueria, Salvatore Greco, and Matthias Ehrgott, editors, *Multicriteria Decision Analysis: State of the Art Surveys*, pages 345–407. Springer, 2005.

[69] Harrisen Scells and Guido Zuccon. Generating better queries for systematic reviews. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, SIGIR '18, pages 475–484, New York, NY, USA, 2018. ACM.

[70] Yannis Siskos, Evangelos Grigoroudis, and Nilolaos F. Matsatsinis. UTA methods. In José Figueria, Salvatore Greco, and Matthias Ehrgott, editors, *Multicriteria Decision Analysis: State of the Art Surveys*, pages 297–343. Springer, 2005.

[71] Mark D. Smucker, James Allan, and Ben Carterette. 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, pages 623–632, New York, NY, USA, 2007. ACM.

[72] M.A. Soliman, I.F. Ilyas, and K. Chen-Chuan Chang. Top-k query processing in uncertain databases. In *Data Engineering, 2007. ICDE 2007. IEEE 23rd International Conference on*, pages 896–905, April 2007.

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

[74] Chang Tan, Qi Liu, Enhong Chen, Hui Xiong, and Xiang Wu. Object-oriented travel package recommendation. *ACM Trans. Intell. Syst. Technol.*, 5(3):43:1–43:26, September 2014.

[75] Evangelos Triantaphyllou. Multi-criteria decision making methods. In *Multi-Criteria Decision Making Methods: A Comparative Study*, pages 8–9. Kluwer Academic Publishers, 2000.

[76] Evangelos Triantaphyllou. Multi-criteria decision making methods. In *Multi-Criteria Decision Making Methods: A Comparative Study*, pages 18–21. Kluwer Academic Publishers, 2000.

[77] Amos Tversky and Daniel Kahneman. Advances in prospect theory: Cumulative representation of uncertainty. *Risk and Uncertainty*, 5:297–323, October 1992.

[78] A. Veloso, J. Wagner Meira, M. Cristo, M. Gonçalves, and M. Zaki. Multi-evidence, multi-criteria, lazy associative document classification. In *15th ACM international conference on Information and knowledge management*, pages 218–227, 2006.

[79] Yushi Wang, Jonathan Berant, and Percy Liang. Building a semantic parser overnight. In *Association for Computational Linguistics (ACL)*, 2015.

[80] Gerhard Weikum. DB&IR: both sides now. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, SIGMOD '07, pages 25–30. ACM, New York, NY, USA, 2007.

[81] WikiMedia Foundation. Dietary reference intake, 2015.

[82] Shawn R. Wolfe and Yi Zhang. User-centric multi-criteria information retrieval. In James Allan, Javed Aslam, Mark Sanderson, ChengXiang Zhai, and Justin Zobel, editors, *SIGIR '09: Proceedings of the 32nd international ACM conference on research and development in information retrieval*, pages 818–819, New York, NY, USA, 2009. ACM.

[83] Shawn R. Wolfe and Yi Zhang. Interaction and personalization of criteria in recommender systems. In Paul De Bra, Alfred Kobsa, and David Chin, editors, *User Modeling, Adaptation, and Personalization*, pages 183–194, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[84] Shawn R. Wolfe and Yi Zhang. Item retrieval as utility estimation. In *The 41st International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '18, pages 795–804, New York, NY, USA, 2018. ACM.

[85] World Wide Web Consortium (W3C). SPARQL 1.1 Query Language. https://www.w3.org/TR/sparql11-query/, Last visited on 2018/11/15.

[86] World Wide Web Consortium (W3C). XQuery 3.1: An XML Query Language. https://www.w3.org/TR/xquery-31/, Last visited on 2018/11/15.

[87] Mingrui Wu, Yi Chang, Zhaohui Zheng, and Hongyuan Zha. Smoothing DCG for learning to rank: a novel approach using smoothed hinge functions. In *Proceedings of the 18th ACM conference on Information and Knowledge Management*, CIKM '09, pages 1923–1926, New York, NY, USA, 2009. ACM.

[88] Min Xie, Laks V. S. Lakshmanan, and Peter T. Wood. Comprec-trip: A composite recommendation system for travel planning. In *Proceedings of the 2011 IEEE 27th International Conference on Data Engineering*, ICDE '11, pages 1352–1355, Washington, DC, USA, 2011. IEEE Computer Society.

[89] Min Xie, Laks V.S. Lakshmanan, and Peter T. Wood. Breaking out of the box of recommendations: From items to packages. In *Proceedings of the Fourth ACM Conference on Recommender Systems*, RecSys '10, pages 151–158, New York, NY, USA, 2010. ACM.

[90] Emine Yilmaz and Javed A. Aslam. Estimating average precision with incomplete and imperfect judgments. In Philip S. Yu, Vassilis Tsotras, Edward Fox, and Bing Liu, editors, *Proceedings of the Fifteenth ACM International Conference on Information and Knowledge Management*, pages 102–111. ACM Press, November 2006.

[91] Yi Zhang. Yow user study data: Implicit and explicit feedback for news recommendation. http://www.soe.ucsc.edu/~yiz/papers/data/YOWStudy.