Preference Dominance Reasoning for Conversational Recommender Systems: A Comparison Between a Comparative Preferences and a Sum of Weights Approach

WALID TRABELSI
Cork Constraint Computation Centre, Department of Computer Science, University College Cork Ireland
w.trabelsi@4c.ucc.ie

NIC WILSON
Cork Constraint Computation Centre, Department of Computer Science, University College Cork Ireland
n.wilson@4c.ucc.ie

DEREK BRIDGE
Department of Computer Science, University College Cork Ireland
d.bridge@cs.ucc.ie

FRANCESCO RICCI
Faculty of Computer Science, Free University of Bozen-Bolzano Italy
fricci@unibz.it

Received (Day Month Year)
Revised (Day Month Year)
Accepted (Day Month Year)

A conversational recommender system iteratively shows a small set of options for its user to choose between. In order to select these options, the system may analyze the queries tried by the user to derive whether one option is dominated by others with respect to the user's preferences. The system can then suggest that the user try one of the undominated options, as they represent the best options in the light of the user preferences elicited so far. This paper describes a framework for preference dominance. Two instances of the framework are developed for query suggestion in a conversational recommender system. The first instance of the framework is based on a basic quantitative preferences formalism, where options are compared using sums of weights of their features. The second is a qualitative preference formalism, using a language that generalises CP-nets, where models are a kind of generalised lexicographic order. A key feature of both methods is that deductions of preference dominance can be made efficiently, since

this procedure needs to be applied for many pairs of options. We show that, by allowing the recommender to focus on undominated options, which are ones that the user is likely to be contemplating, both approaches can dramatically reduce the amount of advice the recommender needs to give to a user compared to what would be given by systems without this kind of reasoning.

1. Introduction

In an era of overwhelming choice, *recommender systems* are a new source of assistance, helping their users to decide which goods, services or information to purchase or consume.[1,2,3,4] These systems infer user preferences from data gathered either explicitly, e.g., in the form of product ratings, or implicitly by observing user behaviour. They have been successfully used to recommend travel services, books, CDs, financial services, insurance plans and news. From a technical point of view, recommender systems have made use of various forms of supervised learning from data sets of numerical ratings on products (e.g. from 1 = bad to 5 = excellent), expressed by a collection of users on a catalogue of products, to make predictions for products not yet rated by the user. The prediction algorithms used within recommender systems include collaborative filtering, content-based filtering and case-based reasoning, and various hybrid approaches that combine these.[1,2,3] However, these classical approaches have typically supported only a simple 'single-shot' form of human-computer interaction, where a user who has previously supplied a set of ratings, identifies herself to the system and is then given a set of product recommendations.

No matter how good recommender systems become, they are unlikely ever to be sufficiently prescient that their first set of recommendations always satisfies the user. Indeed, users are *rarely* satisfied with the first set of recommendations; they usually want to see more options and they exploit the initial recommendations to refine their preferences and articulate new requests. *Conversational recommender systems* allow for this, and recognise that their users may be willing and able to reveal more of their constraints and preferences, over a short dialogue, thereby moving away from 'single-shot' interaction. This is also an opportunity for the recommender system to guide the user by asking questions, giving advice, displaying candidate products, and giving explanations.[5,6,3,7,8,9,10]

Conversational recommender systems typically involve iteratively showing the user a small set of options (e.g., products) for them to choose between. To select an appropriate set to display at each stage, from a much larger collection of options, the recommender needs information regarding which options are likely to be preferred to others by the user, based on previous responses the user has given in the dialogue; if one assumes that the user has some kind of preference relation over products, this amounts to determining if certain products are dominated according to this preference relation.

In 2007, Bridge & Ricci introduced a new kind of conversational recommendation, which they call *Information Recommendation*.[11] Information Recommendation

has a role to play in systems in which the user repeatedly edits and resubmits a query until she finds a product that she wants.[7,6] The user is unlikely to know much about which queries are satisfiable by products in the product catalogue and which are not. The recommender system can advise the user about which of the edits to her current query will be satisfiable and which will not, to dissuade the user from fruitlessly trying queries that are not satisfiable. However, in general the list of all satisfiable (or unsatisfiable) next possible queries will be too long to be of much help to the user. The recommender must put in place methods to minimise the quantity of advice, while maximising its usefulness. The recommender system, described in more detail in Section 2, does this by inferring constraints on the user's preference relation from her previous contribution to the dialogue, i.e. her earlier queries. For instance, the system might infer that the features mentioned in the user's query are more important, for the user, than features not yet mentioned in the query. The recommender reasons with the constraints to determine whether certain queries dominate others, to rank the next possible queries, and to suggest to the user those that are compatible with her preferences and compatible with the available products.

Information Recommendation hence requires a framework for this kind of reasoning, where a set of models of the user is assumed, each with an associated preference ordering, along with a satisfaction relation between models and statements of constraints on preferences expressed in an appropriate language. Within this framework, given a set of constraints, we infer that one product is preferred to another if this preference holds for all models satisfying the constraints. As the dialogue proceeds, the user's new actions reveal more constraints on her preference ordering. This can narrow the set of models of the user. We describe this framework in more detail in Section 3.

Two instances of this framework are developed and presented in this paper. The first, described in Section 4, is based on a simple quantitative preferences formalism, involving a sum of weights (one for each feature of the recommended products), with an associated language of linear inequalities. This is a very commonly used model for preference representation, specifically, in Multi-Attribute Utility Theory (MAUT).[12] It is widely used in recommender systems, e.g. (Ref. 9). And it is the approach to preference dominance taken in Bridge & Ricci's original Information Recommendation paper.

The second instance of the framework (Section 5) is a qualitative preference formalism, where models are a kind of generalised lexicographic order, and constraints are expressed as comparative preference statements in a language generalising CP-nets.[13] This has not been used in Information Recommendation or any other recommender system before.

Section 6 explains how the two instances of the framework can be used within Information Recommendation. Then, Section 7 describes how the implementations of the approaches have been experimentally tested.

Sections 8 and 9 look at other preference formalisms that could be used, and

draw conclusions about our work so far.

The contributions of this paper include the following. It shows how a comparative preferences approach can be used in a conversational recommender. Approaches of this kind, as we mentioned above, have not been used in recommender systems before. Using them is significant because of their expressiveness. This manifests in two ways. The first, as we show in Section 6, is that there are nuances about user preferences that the recommender can capture in the constraints when using comparative preferences that cannot be captured when using the sum of weights approach. The second, as we discuss in Section 9, is that the comparative preferences approach allows a recommender to express a wider range of constraints than those envisaged by the original Information Recommendation work, including, e.g., statements about conditional preferences. Furthermore, through our experiments, we reveal the viability of the comparative preferences approach. It is sufficiently efficient in a practical (rather than theoretical) sense and it allows inferences to be drawn that are strong enough on the one hand to give useful advice but not so strong on the other hand to incorrectly eliminate options that the user prefers. Indeed, its inferences can be stronger than those drawn in the sum of weights approach, resulting in giving the user shorter advice but still without compromising success in directing the user to the best product. We find that these experimental results are robust in the sense that they pertain irrespective of how the user's true preferences are represented.

2. Information Recommendation

Information Recommendation is concerned with helping a user to find a product to purchase or consume. Throughout this paper, we use hotels as the example product. The user repeatedly edits and resubmits a query until she finds a product that she wants. For example, she might submit a query that asks for a hotel that has air-con and golf. If she is advised that hotels that satisfy her query do exist, she might be encouraged to edit her query and see whether there are hotels that additionally are in the city centre. Perhaps if there are not, she might edit her original query to one that sacrifices the golf in favour of the city centre location. This is a hit-and-miss process that can be improved by the intervention of a recommender system. The recommender system: observes the user's actions (her queries); infers constraints on the user's preferred products; uses these inferences to deduce which queries a user is likely to try next; and advises the user to avoid those that cannot be satisfied.

Imagine for instance a user whose current query requests hotels that have air-con and golf, and the recommender knows that, while the user's query is satisfiable, there are no such hotels located in the city centre but there are some that also have swimming pools. In this situation the system could help the user in the search process by supplying exactly this information: this will dissuade the user from fruitlessly trying to add to her query a request for a city centre location, and it will enable her to realise that she can satisfy a desire for a pool. Moreover, if it is satisfiable

to have air-con, golf and a sauna but the recommender has inferred from earlier actions that a sauna is less preferable than a pool, then it is better to tell the user that there are no hotels with air-con and golf in the centre but there are some that have a pool rather than to tell her that there are no hotels with air-con and golf in the centre but there are some that have a sauna.

Obviously the recommender could tell the user the satisfiability of all the possible edits to her query, but such a list would typically not be of much use, as it could potentially be very long. Hence the recommender must put in place methods to minimise the quantity of information that is passed to the user during their interaction. In other words, the recommender must pass to the user the information that has the greatest value for the user.

The information that has the greatest value is here considered to be that which minimises the total quantity of information exchanged and the interaction length, while still finding the 'best' product(s) for the user. A more general approach, which we have not investigated yet, would take a broader view of cognitive load: in this case, we would seek not just to minimise the quantity of information and the interaction length, but also the user memory and deduction costs.

The minimisation of the quantity of information transmitted by the advisor to the user is particularly important in mobile applications, which are among the targets of our research, where the physical constraints of the device impose a narrow communication channel (small screen size and limited input capability), and the context of usage further imposes that the interaction be as quick and simple as possible.[14]

In this paragraph and the next, we take the opportunity to compare the use that Information Recommendation makes of what it learns about its user with the use that classical product recommendation makes. In the Adaptive Place Advisor, for example, the user preference information is used for product selection.[9] The advisor infers feature weights and defaults, and uses them in product retrieval. Similarly, in the work of Pu *et al.*, the system uses its knowledge of users and of the product space to select sets of products, albeit sets that it hopes will provoke the user into volunteering further preferences.[10] In collaborative filters also, the user model (the ratings profile) is for product retrieval and ranking.

But in Information Recommendation, the user preference information is used to guide the user's search rather than to retrieve or rank products. So, while the ultimate end is still that of helping the user find the best products, the primary goal of our proposed techniques is to *support* the user in finding them autonomously rather than in finding them for the user. Work on question selection in dynamic dialogues can be seen as an example of Information Recommendation. The system dynamically selects questions to elicit user preferences. Its goal is to choose a sequence of questions that most effectively homes in on desirable products. In most such work, questions are selected based on the user's partial query and the product distribution. But, Schmitt's *simVar* system also builds and uses simple user models.[8] Reilly *et al.*'s use of the query history to dynamically recommend compound critiques can

also be regarded as Information Recommendation.[5] In their work, the system shows the user some products which the user can critique, but it also advises the user by displaying dynamically-computed critiques that are known to be satisfiable, which the user can select.[15]

Below we describe Information Recommendation in more detail.

2.1. *The products*

In this paper, we assume that the products are modeled with a collection of Boolean-valued features $V = \{F_1, \ldots, F_n\}$. The features are intended to relate to a set of products that the user is interested in choosing between; for example, in choosing a hotel room, one feature might be whether the hotel has a swimming pool.

Define a configuration α to be a mapping from $\{1, \ldots, n\}$ to $\{1, 0\}$. A configuration α can also be thought of as a selection of features: all features F_i such that $\alpha(i) = 1$. For convenience we will write a configuration such as $(1, 0, 1) = \big(\alpha(1), \alpha(2), \alpha(3)\big)$ as $f_1 \bar{f}_2 f_3$.

In Information Recommendation, configurations can be thought of as queries over the set of features. If a user issues query q and if $f_i \in q$, this means that the user is interested in products that have the i^{th} feature. In accordance with most Web-based product search systems, $f_i \notin q$ means only that the user has not (yet) declared any interest in feature F_i; it does not mean that the user wants products that lack the i^{th} feature. So, for example, if q is $f_1 \bar{f}_2 f_3$, the user wants a product that has features f_1 and f_3 and has said nothing yet about f_2.

A subset of the configurations correspond to products that are available to the user. A query is *satisfiable* if and only if there exists a product which has all the features in the query; otherwise, it is *unsatisfiable*. Users cannot be expected to know in advance which queries are satisfiable and which are not, although they may have incomplete knowledge of this.

2.2. *The dialogue*

In the kind of system envisaged in (Ref. 11), the user submits an initial query, typically one that is quite under-specified: 'to test the water'. In our experiments (Section 7) we use an empty initial query. Let this query be known as the *current query*, q.

The recommender system does not know the user's preferences and does not ask about them. It may only infer them from the sequence of queries that the user submits. As the dialogue proceeds, the recommender system will infer constraints on the user's preferences and express them as statements in a language $\mathcal{L}$. We will denote the current set of statements by Φ. Initially Φ may contain a set of 'background' assumptions. In particular, we will want to express the idea that including a feature in a query is at least as good as not including it. Statements will be added to Φ as the dialogue proceeds. For example, if the user's query requests a certain

feature, we may plausibly infer that this feature is more important than the ones not included in the query.

The interaction between the user and the recommender system proceeds as follows:

(i) The recommender system analyzes current query q, with particular regard to differences between q and the queries the user might alternatively have submitted. The system induces some additional constraint on the user's preferences and adds corresponding statements to Φ.
(ii) The recommender system generates a set of candidate next possible queries and prunes this set to those that are satisfiable and undominated (see below). It advises the user to confine her next query to this set.
(iii) The user chooses and submits her next query. This becomes the new current query q. In the experiments reported in Section 7, we arrange that the user always chooses one of the queries that the system advises (although this might not be so in practice).

Steps (i)–(iii) are repeated until the user is satisfied with q or the set of undominated, satisfiable candidates is empty, in which case as far as the recommender system is concerned q cannot be bettered. At this point, the user can request to see the products that satisfy q.

The goal of the recommender system is to give the advice that has the greatest value. We consider this to be that which minimises the total quantity of advice given and the dialogue length, while guiding the user to the best product.

During step (ii) above, the recommender system computes the following three sets of queries:

- *Candidates:* Candidate queries are ones which are close, in a particular sense, to the current query. Each is a low-cost edit to the current query. For example, if $f_i \notin q$, the set of candidates will include the query that results from adding just feature f_i to q.
- *Satisfiables:* The recommender system should never include unsatisfiable queries in its advice: they make interaction length longer without leading the user to the best product. Hence, the system eliminates from *Candidates* those queries which are unsatisfiable; the remaining queries are called the *Satisfiables*.
- *Undominated*: The system could advise the user to confine her query to *Satisfiables*. However, this set can be large. Hence, the system eliminates from *Satisfiables* each query which is dominated by (i.e., worse than) some other member of *Satisfiables*; the remaining set of queries is called *Undominated*. The dominance relation is based on what is induced in step (i) above. The rationale is to exclude from the system's advice queries that, on the basis of what the system has induced about the user's preferences, it thinks the user would regard as inferior.

In effect, the system's advice to the user is to confine her next query to a set which

the system knows are satisfiable and believes the user is likely to try next (since, according to what the system has inferred about the user's preferences, they are ones that are not dominated by other satisfiable queries).

We will now explain how to obtain the three sets of queries.

Generating the candidates

Real user behaviour in query editing tends to proceed with modifications of limited 'reach'. Hence, following (Ref. 11), we define *Candidates* as the set of queries which we obtain by applying three editing operations, *Add*, *Switch* and *Trade*, to the current query. These operations are defined as follows. Given current query q and $f_i \notin q$, operation $Add(q, f_i)$ adds just feature f_i to q, giving the new query $q \cup \{f_i\}$, which we sometimes write as q^i. $Switch(q, f_i, f_j)$ where $f_i \in q, f_j \notin q, i \neq j$ discards feature f_i in favour of feature f_j, giving the new query $(q \setminus \{f_i\}) \cup \{f_j\}$. Finally, $Trade(q, f_i, f_j, f_k)$ where $f_i \in q, f_j \notin q, f_k \notin q, i \neq j, i \neq k, j \neq k$ discards feature f_i and introduces features f_j and f_k.

There are, of course, other edits that could be in this set of candidates, such as trading two features in q for three not in q. It is possible that such an edit would turn out to be satisfiable and not dominated by any other edit. It is helpful, however, in a practical system to confine the recommender's advice to a set of readily-understandable edits, particularly ones that the user herself is likely to contemplate. Other edits, such as deletion of a feature from q, are excluded because they would produce a new query which, although satisfiable if q is satisfiable, would be dominated by q itself. Finally, we note that, although $Trade(q, f_i, f_j, f_k) = Add(Switch(q, f_i, f_j), f_k)$, the *Trade* operation is not redundant in Information Recommendation. There are scenarios in which the recommender could advise the user to try the *Trade* operator (if it results in a satisfiable query that is not dominated by any other edits to q) but could not advise the user to first try the *Switch* operator, as a precursor to then doing an *Add*; the latter advice would not be suitable in scenarios where the *Switch* results in a query that, while satisfiable, is dominated by one of the other edits.

Checking satisfiability

As defined earlier, a query is satisfiable if and only if there exists a product which has all the features present in the query. If products are stored explicitly in a database, satisfiability of a candidate query can be checked by a scan of the database. For configurable products, where the set of products is represented as a set of solutions to a Constraint Satisfaction Problem, satisfiability of a candidate query can be checked by determining if the CSP has solutions containing all the features in the query (which can be checked by checking satisfiability of an augmented CSP).

Checking for dominance

The final pruning of the satisfiable candidate queries is performed using one of the two instances of the framework for dominance of preferences that we will explain in Sections 3, 4 and 5. In either case, $q \in$ *Satisfiables* is pruned if it is strictly dominated, i.e., dominated according to relation $\succ_\Phi$, by $q' \in$ *Satisfiables*.

3. A Framework for Dominance of Preferences

We assume a set Ω of possible configurations (as defined in Section 2). We would like to generate some kind of (partially ordered) preference relation $\succcurlyeq$ on Ω, based on previous information we have received regarding the user's preferences. This section describes a framework for generating such a relation. We define two instances of the framework in Sections 4 and 5.

In order to make non-trivial inferences regarding the user's relative preferences over configurations, we will have to make some assumptions. We assume:

- A set of models $\mathcal{M}$, each of which is intended to represent a possible user (or way the user could be). Associated with each $M \in \mathcal{M}$ is a total pre-order $\succcurlyeq_M$ on configurations, i.e., a reflexive, transitive and complete relation (so for all configurations α and β, we have either $\alpha \succcurlyeq_M \beta$ or $\beta \succcurlyeq_M \alpha$ or both).
- A formal language $\mathcal{L}$ whose statements express constraints on the user's preferences.
- A relation $\models$ between $\mathcal{M}$ and $\mathcal{L}$. For $M \in \mathcal{M}$ and $\varphi \in \mathcal{L}$, we interpret $M \models \varphi$ to mean that φ holds for the preferences of M.

Given a particular set $\Phi \subseteq \mathcal{L}$ of statements, we consider the orderings on configurations which hold for every model satisfying statements Φ. Formally, we define relation $\succcurlyeq_\Phi$ on configurations as follows: $\alpha \succcurlyeq_\Phi \beta$ if and only if $\alpha \succcurlyeq_M \beta$ for all M satisfying (every member of) Φ. It follows that $\succcurlyeq_\Phi$ is a pre-order (a reflexive and transitive relation) on configurations. Now, $\alpha \succcurlyeq_\Phi \beta$ means that every user who agrees with Φ considers that configuration α is at least as desirable as configuration β (assuming this particular model of users). It is possible that we also have $\beta \succcurlyeq_\Phi \alpha$, in which case every user considers that α and β are equally desirable. We define the relation $\succ_\Phi$ to be the strict part of $\succcurlyeq$, so that $\alpha \succ_\Phi \beta$ if and only if $\alpha \succcurlyeq_\Phi \beta$ and $\beta \not\succcurlyeq_\Phi \alpha$. Relation $\succ_\Phi$ is irreflexive and transitive. We say that *Given* Φ, α *strictly dominates* β, if $\alpha \succ_\Phi \beta$, that is, if all users (represented by models M in $\mathcal{M}$) agreeing with Φ regard α as at least as preferable as β (i.e., $\alpha \succcurlyeq_M \beta$), and at least one such user regards α as strictly preferable to β (i.e., $\beta \not\succcurlyeq_M \alpha$).

4. Sum of Weights Model of the User

In this section we consider our first kind of model of the user's preferences, where it is assumed that a user assigns a weight to each feature, and configurations are compared on the sum of weights of the associated set of features.

4.1. *Models*

The set of models is the set of all vectors of weights $w = (w_1, \ldots, w_n)$, where w_i is a non-negative real number. w_i is the weight assigned to feature F_i. Given a weights vector w, the overall value $w(\alpha)$ of a configuration α is the sum of weights of the features included in α, i.e., $w(\alpha) = \sum_{i:\alpha(i)=1} w_i$, which also can be written as $\sum_i w_i\alpha(i)$. This is used to define the ordering on configurations. We define the preference relation $\succcurlyeq_w$ for model w by $\alpha \succcurlyeq_w \beta$ if and only if $w(\alpha) \geq w(\beta)$, i.e., if and only if $\sum_i w_i\big(\alpha(i) - \beta(i)\big) \geq 0$. Thus $\succcurlyeq_w$ is a total pre-order on configurations.

4.2. *Constraint language*

The language consists of statements of the form $\alpha \geq \beta$, where α and β are configurations.

4.3. *Dominance relation*

Weight vector w is defined to satisfy $\alpha \geq \beta$ if $\alpha \succcurlyeq_w \beta$. Let Φ be a set of statements. The definitions in Section 3 lead to the following definition of $\succcurlyeq_\Phi$, the induced preference relation given constraint statements Φ:

For configurations α and β, $\alpha \succcurlyeq_\Phi \beta$ if and only if $\alpha \succcurlyeq_w \beta$ for all weight vectors w satisfying Φ. Dominance relation $\succ_\Phi$ is then the strict part of $\succcurlyeq_\Phi$. When we want to make explicit comparisons with the second instance of the framework (Section 5), we will use the notation $\succcurlyeq_\Phi^{sw}$ for $\succcurlyeq_\Phi$ when the Sum of Weights model is considered.

Example 1. Let Φ be the pair of statements: $f_1\bar{f}_2f_3 \geq \bar{f}_1f_2f_3$, and $f_1f_2\bar{f}_3 \geq \bar{f}_1f_2f_3$. Let α be the configuration $f_1\bar{f}_2\bar{f}_3$, and let β be the configuration $\bar{f}_1f_2f_3$. Weights vector w satisfies the constraint $f_1\bar{f}_2f_3 \geq \bar{f}_1f_2f_3$ if and only if $w(f_1\bar{f}_2f_3) \geq w(\bar{f}_1f_2f_3)$, i.e., $w_1 + w_3 \geq w_2 + w_3$, which holds if and only $w_1 \geq w_2$. By similar reasoning, w satisfies Φ if and only if $w_1 \geq w_2$ and $w_1 \geq w_3$. Also, w satisfies $\alpha \geq \beta$ if and only if $w_1 \geq w_2 + w_3$. Thus Φ does not entail $\alpha \geq \beta$, so we do not have $\alpha \succcurlyeq_\Phi^{sw} \beta$, since, for example, weights vector w with $w_1 = 4$, $w_2 = 2$ and $w_3 = 3$ satisfies Φ but does not satisfy $\alpha \geq \beta$. □

Example 2. Suppose now that there are four features, and let Ψ be the pair of statements $f_1\bar{f}_2f_3f_4 \geq \bar{f}_1f_2f_3f_4$ and $f_1f_2\bar{f}_3\bar{f}_4 \geq f_1\bar{f}_2f_3\bar{f}_4$. With the sum of weights semantics this implies $f_1f_2\bar{f}_3f_4 \geq \bar{f}_1f_2f_3f_4$, since the first statement implies $w_1 \geq w_2$, and the second statement implies $w_2 \geq w_3$, implying $w_1 \geq w_3$, which implies that the third statement is satisfied. We therefore have $f_1f_2\bar{f}_3f_4 \succcurlyeq_\Psi^{sw} \bar{f}_1f_2f_3f_4$. In fact, we have strict dominance: $f_1f_2\bar{f}_3f_4 \succ_\Psi^{sw} \bar{f}_1f_2f_3f_4$ since we do not have $\bar{f}_1f_2f_3f_4 \succcurlyeq_\Psi^{sw} f_1f_2\bar{f}_3f_4$. □

4.4. *Computation of preference*

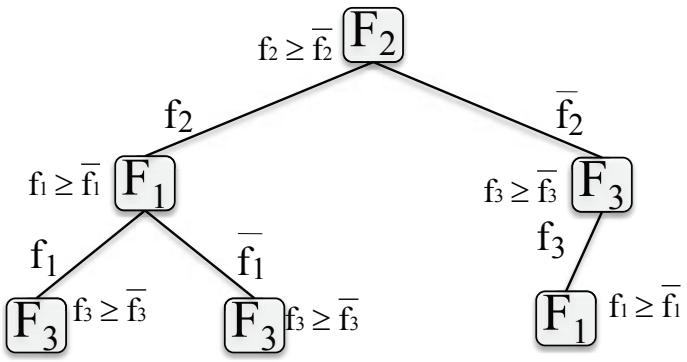
We wish to determine if $\alpha \succcurlyeq_\Phi \beta$, for given configurations α and β. Let *Pos* be the set of constraints $w_i \geq 0$, for $i = 1, \ldots, n$, representing the non-negativity of the weights (and corresponding to the assumption that including a feature is always at least as good as not including it). The definition implies that $\alpha \succcurlyeq_\Phi \beta$ if and only if the linear constraints $\Phi \cup Pos$ (over real-valued variables w_i) entail the constraint $\sum_i \big(\alpha(i) - \beta(i)\big) w_i \geq 0$.

For implementation of this using a Linear Programming solver, it can be convenient to express it as a linear optimisation problem. Define A_{min} to be the minimum value of $\sum \big(\alpha(i) - \beta(i)\big) w_i$ subject to constraints $\Phi \cup Pos$. It can be easily shown that $\alpha \succcurlyeq_\Phi \beta$ if and only $\Phi \cup Pos$ entails $\sum_i \big(\alpha(i) - \beta(i)\big) w_i \geq 0$ if and only if $A_{min} \geq 0$.

5. Comparative Preferences Model of the User

5.1. *Models*

In our second approach, models are a kind of generalised lexicographic order, called *cp-trees*,[16,17] which are similar to search trees used for solving constraint satisfaction problems. Figure 1 gives an example of a cp-tree. Each node is labeled with a variable (i.e., a feature, in the current context). The root is labeled by the most important variable, F_2 in this example. Each node is associated also with a preference ordering of the values of the variable. This local ordering in the case of the nodes in the example is $f_i \geq \bar{f}_i$, where f_i means F_i is included ($F_i = 1$) and $\bar{f}_i$ means that F_i is not included ($F_i = 0$). This ordering captures the requirement that including a feature is never worse than not including it.



$$f_1 f_2 f_3 \geq f_1 f_2 \overline{f_3} \geq \overline{f_1} f_2 f_3 \geq \overline{f_1} f_2 \overline{f_3} \geq f_1 \overline{f_2} f_3 \geq \overline{f_1 f_2} f_3 \geq f_1 \overline{f_2 f_3} \equiv \overline{f_1 f_2 f_3}$$

Fig. 1. A cp-tree σ, along with its associated ordering $\succcurlyeq_\sigma$ on configurations.

Two configurations α and β are compared first on the most important variable. If they do not agree on this variable then the comparison is settled: in the example, if α contains feature F_2 and β does not, then α is better than β. This happens, for example, if α is $\overline{f_1} f_2 \overline{f_3}$ and β is $f_1 \overline{f_2} f_3$. Otherwise, α and β agree on the most important variable. The user may then have a next most important variable (labeling a child node); this can depend on the value assigned to the most important variable (signified by the value on the edge from parent to child). For this reason, this model allows *conditional preferences*. If there is no such next important variable, then α and β are considered equally preferable according to this cp-tree. Thus the cp-tree σ generates a total pre-order $\succcurlyeq_\sigma$ on configurations.

Note that each node in the cp-tree in Figure 1 is associated with a single variable. In fact, we can allow a more general representation, where at most γ variables (where, in this paper, $\gamma = 1$, 2 or 3) are associated with a node, along with a total pre-order over the assignments to that set of at most γ variables. For example, if a node is associated with the pair of variables $Y = \{F_2, F_3\}$ then the local ordering is over assignments to Y, and might be e.g., $f_2 f_3 \geq f_2 \bar{f}_3 \equiv \bar{f}_2 f_3 \geq \bar{f}_2 \bar{f}_3$ (so that $f_2 \bar{f}_3$ and $\bar{f}_2 f_3$ are equivalent in the local total pre-order $\geq$, i.e., both $f_2 \bar{f}_3 \geq \bar{f}_2 f_3$ and $f_2 \bar{f}_3 \leq \bar{f}_2 f_3$). Let $\mathcal{M}(\gamma)$ be the set of cp-trees over Ω, where the set of variables associated to a node involves at most γ variables. A 1-cp-tree over Ω is defined to be an element of $\mathcal{M}(1)$, i.e., a cp-tree with a single variable being associated with each node (as in Figure 1). The local ordering associated with a node associated with feature F_i, must then be $f_i \geq \bar{f}_i$, since including a feature is always at least as good as not including it. For full definitions of cp-trees and their associated total pre-orders see (Ref. 16).

5.2. *Constraint language*

The language will include statements that compactly express comparative (and sometimes conditional) preferences among configurations. There is a substantial and fast growing literature on this topic in the AI and Philosophy of Science communities.[18,19,13,20,21,22,16]

The language includes comparative preference statements φ of the form $p \geq q \parallel T$, where P, Q and T are subsets of the set of features V, and p is an assignment to P (i.e., a function from P to $\{0, 1\}$), and q is an assignment to Q. Informally, the statement $p \geq q \parallel T$ represents the following: p is preferred to q if T is held constant. More formally, the preference relation $\succcurlyeq$ satisfies the statement $p \geq q \parallel T$ if and only if $\alpha \succcurlyeq \beta$ for all configurations α and β such that (i) α extends p (i.e., α restricted to the subset of the features in P equals p), (ii) β extends q, and (iii) α and β agree on T: $\alpha(F_i) = \beta(F_i)$ for all $F_i \in T$.

A very important kind of statement is one expressing the constraint that one configuration, α, is preferred over another, β. This can be written as $\alpha \geq \beta \parallel \emptyset$; we also write such a preference statement as $\alpha \geq \beta$.

Since including a feature is at least as good as not including it, we always include,

in the set Φ, the statement $f_i \geq \overline{f_i} \parallel V \backslash \{F_i\}$ for each feature F_i. Hence our language is strongly related to *Conditional Importance Networks*.[22]

5.3. *Dominance relation*

We define the set of models of users to be the $\mathcal{M}(\gamma)$ for some $\gamma = 1, 2$, or 3. Each cp-tree σ generates a total pre-order $\succcurlyeq_\sigma$ on configurations. Let σ be a cp-tree and let φ be a statement in the constraint language. σ satisfies φ if and only if $\succcurlyeq_\sigma$ satisfies φ. In the same way as in Section 4, we define, for a given set of statements Φ, $\succcurlyeq_\Phi$ (or more precisely, $\succcurlyeq_\Phi^{cp_\gamma}$), in the following way: $\alpha \succcurlyeq_\Phi \beta$ holds if and only if $\alpha \succcurlyeq_\sigma \beta$ holds for all cp-trees σ in $\mathcal{M}(\gamma)$ satisfying Φ. The definitions immediately imply that relations $\succcurlyeq_\Phi^{cp_\gamma}$ are monotonic in both Φ and γ (increasing in Φ and decreasing in γ): if $\gamma \geq \gamma'$ and $\Phi \subseteq \Psi$ then $\alpha \succcurlyeq_\Phi^{cp_\gamma} \beta$ implies $\alpha \succcurlyeq_\Psi^{cp_{\gamma'}} \beta$.

Example 1 continued. With the cp-tree semantics, when $\gamma = 1$, the pair of statements Φ ($f_1\bar{f}_2f_3 \geq \bar{f}_1f_2f_3$, and $f_1f_2\bar{f}_3 \geq \bar{f}_1f_2f_3$) implies the preference statement $f_1\bar{f}_2\bar{f}_3 \geq \bar{f}_1f_2f_3$, so we have $f_1\bar{f}_2\bar{f}_3 \succcurlyeq_\Phi^{cp_1} \bar{f}_1f_2f_3$ (and indeed we have $f_1\bar{f}_2\bar{f}_3$ strictly dominates $\bar{f}_1f_2f_3$, i.e., $f_1\bar{f}_2\bar{f}_3 \succ_\Phi^{cp_1} \bar{f}_1f_2f_3$). The reason is that, for any 1-cp-tree σ satisfying $f_1\bar{f}_2f_3 \geq \bar{f}_1f_2f_3$, the most important feature must be either F_1 or F_3. (If F_2 were the most important feature, then we would not have $f_1\bar{f}_2f_3 \succcurlyeq_\sigma \bar{f}_1f_2f_3$, because the local ordering is $f_2 \geq \bar{f}_2$, since the presence of a feature is never worse than its absence.) Similarly, if 1-cp-tree σ satisfies $f_1f_2\bar{f}_3 \geq \bar{f}_1f_2f_3$, then the most important feature must be either F_1 or F_2. Hence for any 1-cp-tree σ satisfying Φ, F_1 is the most important feature. The root node then determines the preference ordering of the pair of configurations $f_1\bar{f}_2\bar{f}_3$ and $\bar{f}_1f_2f_3$: since the local ordering of this node must be $f_1 \geq \bar{f}_1$, we have $f_1\bar{f}_2\bar{f}_3 \succeq_\sigma \bar{f}_1f_2f_3$. Hence we have $f_1\bar{f}_2\bar{f}_3 \succcurlyeq_\Phi^{cp_1} \bar{f}_1f_2f_3$. The qualitative and lexicographic nature of the cp-trees semantics ensures this inference, in contrast with the numerical sum of weights method, which did not. □

Example 2 continued. Recall that Ψ is the pair of statements $f_1\bar{f}_2f_3f_4 \geq \bar{f}_1f_2f_3f_4$ and $f_1f_2\bar{f}_3\bar{f}_4 \geq f_1\bar{f}_2f_3\bar{f}_4$. In contrast with the sum of weights semantics, Ψ does not imply $f_1f_2\bar{f}_3f_4 \geq \bar{f}_1f_2f_3f_4$. To show this we can construct a 1-cp-tree σ with F_4 as the most important (root node) variable, and where, given f_4, F_3 is more important than F_1 which is more important than F_2, and given $\bar{f}_4$, F_2 is more important than F_3 which is more important than F_1. σ then satisfies Ψ, but not $f_1f_2\bar{f}_3f_4 \geq \bar{f}_1f_2f_3f_4$.

A key issue here is that cp-trees can represent *conditional* preferences: the preferences can be different given f_4 from those given $\bar{f}_4$. In contrast, the sum of weights semantics assumes preferential independence, so preferences are not conditional at all, which is why the inference holds for the sum of weights semantics. □

The pair of examples shows that the two preference dominance techniques are

incomparable: $\succcurlyeq_{\Phi}^{cp_1}$ can sometimes include preferences not included in $\succcurlyeq_{\Phi}^{sw}$, and vice versa.

5.4. *Computation of preference*

Given set of statements Φ and configurations α and β, we can determine in polynomial time whether or not $\alpha \succcurlyeq_{\Phi} \beta$ holds, using the algorithm given in (Ref. 16), as shown by Theorem 1 in (Ref. 16).

6. Inducing Constraints on Preferences in Information Recommendation

Now that we have explained the two instances of the framework for dominance of preference, it remains for us to return to Information Recommendation to explain what the system induces in step (i) above (Section 2.2), when it observes the user's queries. We explain this below for each of the two preference models.

6.1. *Inducing constraints in the Sum of Weights model*

Add(q, f_i)

If the user has added feature f_i to query q, giving rise to new query q^i, then statements $q^i \geq q^j$ are induced for all $f_j \notin q$, $i \neq j$ unless $Add(q, f_j) = q^j$ is unsatisfiable. This assumes that the new query is preferred to other satisfiable queries that could have been generated by adding other features. This implies that the weight vector satisfies the linear inequality $w_i \geq w_j$. However, we do not infer $q^i \geq q^j$ in all cases. In particular, we do not infer it if $Add(q, f_j)$ is unsatisfiable. Users may have (incomplete) knowledge of which queries are unsatisfiable: if she knows a query is unsatisfiable, then she will not submit it. We 'play it safe': when q^j is unsatisfiable, in case the user knows this, we do not assume that the query that she does submit has higher weight than this unsatisfiable query.

Switch(q, f_i, f_j)

If the user switches f_i for f_j, then we infer that $w_i \leq w_j$; and for all $f_k \notin q$ we can infer $w_j \geq w_k$ unless $Switch(q, f_i, f_k)$ is unsatisfiable.

Trade(q, f_i, f_j, f_k)

If the user trades f_i for f_j and f_k, then we infer that $w_i \leq w_j + w_k$; and for all $j', k' \notin q$ such that $\{j, k\} \neq \{j', k'\}$ and $j' \neq k'$, we infer $w_j + w_k \geq w_{j'} + w_{k'}$ unless $Trade(q, f_i, f_{j'}, f_{k'})$ is unsatisfiable.

Note that we have been quite conservative in what we have inferred. In the event of observing $Trade(q, f_i, f_j, f_k)$ for example, we might also have inferred that the selected *Trade* is better than *all* other satisfiable *Trade* operations, $Trade(q, f_{i'}, f_{j'}, f_{k'})$ for $\{i, j, k\} \neq \{i', j', k'\}$ instead of just other ways of trading f_i. Using the same reasoning, we might have inferred that a *Trade* is better

than all satisfiable *Add* and *Switch* operations; and that a *Switch* is better than all satisfiable *Add* operations.

We might make these inferences if we attribute ever greater rationality to the user. In this paper, we are not going to attribute these higher levels of rationality to the user, and hence we will infer only what we stated earlier. An important observation is that it is not a problem to our proposed methods if we assume that users are less rational than they really are. The reason this does not pose a problem to our proposed methods is that assuming users are less rational than they really are will result only in us making fewer deductions when observing their moves; it will not result in us drawing incorrect inferences. In fact, it is more dangerous to assume a fully rational user, who can really take the best move, since this will cause us to draw inferences that may be incorrect.

6.2. *Inducing constraints in the Comparative Preferences model*

Add(q, f_i)

Again consider the situation where the user has chosen to add feature f_i rather than feature f_j (which is another feature different from f_i not present in q). For this model, there are alternative statements one might induce from this decision by the user. We consider two, each being a kind of counterpart for the constraint $w_i \geq w_j$ induced for the sum of weights approach. It is an advantage of the Comparative Preferences model that it can express nuances that the Sum of Weights model cannot.

- **Basic:** Let q be equal to the current query, let q^i be the current query q with the feature f_i added, and let q^j be q with the feature f_j added. A basic, somewhat conservative, approach is to just model the preference of feature i over feature j by the preference statement: $q^i \geq q^j || \emptyset$, i.e., $q^i \geq q^j$, which just expresses a preference for q^i over q^j.
- **Importance:** Alternatively, and less conservatively, we can induce $f_i \geq \overline{f_i} || V \setminus \{F_i, F_j\}$, which says that the presence or not of the feature F_i is more important than the choice of F_j. Thus, whatever the state of the feature F_j in the query, the user will prefer F_i to be present in the query so that (if possible) this feature is included in the best product.

Note too that in either case we ensure that the recommender system 'plays it safe' when inducing preference statements, in the same way that we explained for the Sum of Weights model, by not inducing preferences over unsatisfiable queries.

Switch(q, f_i, f_j)

If the user switches f_i for f_j then we can induce statements stating the superiority of the feature f_j over the feature f_i, and also over the features f_k that were not chosen instead of f_j.

- **Basic:** Let q be the current query, let q^j_{-i} be the current query q with the feature f_j added and the feature f_i removed, i.e., *Switch*(q, f_i, f_j).

We model the preference of feature f_j over feature f_i by the preference statement $q^j_{-i} \geq q||\emptyset$, i.e., $q^j_{-i} \geq q$. Similarly, we model the preference of feature f_j over feature f_k for any k such that q^k_{-i} is satisfiable by $q^j_{-i} \geq q^k_{-i}$.

- **Importance:** Here we induce the preference statement $f_j \geq \overline{f_j}||V \setminus \{F_i, F_j\}$, and the preference statement $f_j \geq \overline{f_j}||V \setminus \{F_k, F_j\}$ for any k such that q^k_{-i} is satisfiable.

Trade(q, f_i, f_j, f_k)

If the user trades F_i for F_j and F_k then we can induce statements stating the superiority of the combination of features f_j and f_k over the feature f_i, and also over the combinations of features f_m and f_n that were not chosen.

- **Basic:** Let q^{jk}_{-i} be the current query q with the features f_j and f_k added and the feature f_i removed, i.e. *Trade*(q, f_i, f_j, f_k), and let q^{mn}_{-i} be q with the features f_m and f_n added and the feature f_i removed. We model the preference of the combination of features f_j and f_k over the feature f_i by the preference statement $q^{jk}_{-i} \geq q||\emptyset$, i.e., $q^{jk}_{-i} \geq q$. We model the preference of the combination of features f_j and f_k over the combination of features f_m and f_n, with q^{mn}_{-i} being satisfiable, by the preference statement $q^{jk}_{-i} \geq q^{mn}_{-i}$.
- **Importance:** We induce the preference statement $f_j f_k \geq \overline{f_j f_k}||V \setminus \{F_i, F_j, F_k\}$, which says that the presence or not of the combination of features f_j and f_k is more important than the choice of F_i. We also induce the statement $f_j f_k \geq \overline{f_j f_k}||V \setminus \{F_j, F_k, F_m, F_n\}$, for any $m, n \notin q$ such that $\{j, k\} \neq \{m, n\}$ and $m \neq n$.

7. Experiments

In this section, we report experiments with simulated users that demonstrate the feasibility of using both the Sum of Weights model and the Comparative Preferences model within the Information Recommendation system. It is a common practice to use simulated interactions to initially test alternative algorithms for conversational systems.[23,24] Simulations can pinpoint the main deficiencies of the algorithms and can be used to compare a large number of alternative approaches, as in our case. Experiments with real users cannot be used to extensively test alternative dialogue control algorithms, even if it is clear that the ultimate evaluation of the effectiveness of a conversational system has to be made online.

We use two separate product databases, that we scraped from the Web, each describing hotels by their amenities expressed as Boolean features such as *airport shuttle*, *pets permitted*, *restaurant on-site*, etc. The Marriott-NY database records 9 features about 81 hotels; many offer the same amenities, and so there are 36 distinct products in the database. The Trentino-10 database records 10 features for 4056 hotels, of which 133 are distinct.

The simulated users in our experiments behave in the following somewhat idealized way: within a dialogue, they do not try queries that they have tried earlier in

the dialogue; they are aware of their own preferences and never choose a next query that would be inferior to the current one; and they take heed of all advice given, i.e., if the recommender system tells them to confine their next query to a certain set, then they do so; indeed they choose the best possible query from this set. In the terminology of (Ref. 11), these are *optimizing users*.

For a simulated user to make choices about which among the queries in the recommender's advice is the best one for it to submit next, the simulated user must be assigned a set of *true preferences*. These are generated randomly. We arrange that they are known to the simulated user and used by that user for query selection, but they are not known to the recommender system, which knows only what it induces about user preferences and adds to Φ when observing user query behaviour.

In the same way that the recommender system can represent induced constraints on users' preferences in either the Sum of Weights model or the Comparative Preferences model, equally the simulated users' true preferences can be represented in either model. If the true preferences are represented in the Sum of Weights model, then recommenders that induce constraints on preferences in the Sum of Weights model may have an advantage over recommenders that are using the Comparative Preferences model, and vice versa. We control for this problem by showing results below that pair both ways of representing true preferences with recommenders that use both ways of representing induced preferences.

In (Ref. 11), the initial query in each dialogue was non-empty but small, and was generated randomly but in such a way as to be compatible with the user's true preferences. In the dialogues that we use in our experiments here, the initial query in each dialogue is empty. While this is less realistic (real users usually make a few selections in Web search forms before first submitting), we found this to be the best way of ensuring that our comparisons of the two preference models are done fairly.

In the experiments, one recommender system uses the Sum of Weights model; six use the Comparative Preferences model, differing first on which of the two alternative preference statements they infer (Basic or Importance), and on their value for γ (1, 2 or 3). For each pairing of a user with a recommender system, we ran 500 simulated dialogues. In total then, we are reporting results for 2 databases $\times$ 2 ways of representing true preferences $\times$ 7 recommenders $\times$ 500 dialogues, which is 14000 runs of the system.

We stress that a recommender that does not remove from its advice those queries that are dominated would suggest to the user a large number of next possible queries, i.e., all those that are satisfiable. This has been a common approach in earlier conversational systems and it is current practice in many conventional Web applications which allow the user to specify a preferred value for a feature (e.g., using a checkbox) and then show the number of products that satisfy that condition. The whole advantage of Information Recommendation rests on being able to prune the dominated queries, showing only the undominated ones, on the basis that these best match the user's preferences. Hence, in the experiments we compare the pruning rates achieved by using the Sum of Weights model with those achieved by the six

recommender systems that use the Comparative Preferences model. The pruning rate is defined as follows:

$$pruning\ rate = \frac{|Satisfiables \setminus Undominated|}{|Satisfiables|} \times 100 \quad (1)$$

"Eq. (1)" shows the extent to which an approach eliminates what it takes to be inferior satisfiable candidate queries from its advice. In general, the shorter the advice the better, as this reduces the set of options the user has to look through.

7.1. *Representing true preferences in the Sum of Weights model*

In our first set of experiments, we set the user's true preferences by randomly generating weight vectors over product features. The pruning rates in this case are shown in Table 1.

Table 1. The pruning rates (true preferences represented in Sum of Weights model)

	γ =1	γ =2	γ =3
Marriott-NY			
Comp. Prefs. Basic	87·50	14·48	12·65
Comp. Prefs. Importance	87·50	87·49	87·42
Sum of Weights		87.38	
Trentino-10			
Comp. Prefs. Basic	87·49	16·51	13·98
Comp. Prefs. Importance	87·42	87·57	86·72
Sum of Weights		85.72	

Table 2. The average number of steps per dialogue (true preferences represented in Sum of Weights model)

	γ =1	γ =2	γ =3
Marriott-NY			
Comp. Prefs. Basic	6·004	5·998	6·000
Comp. Prefs. Importance	6·008	6·006	5·998
Sum of Weights		6.001	
Trentino-10			
Comp. Prefs. Basic	6·726	6·784	6·798
Comp. Prefs. Importance	6·748	6·756	6·790
Sum of Weights		6.790	

The table shows that, in most settings, the Comparative Preferences approach is pruning non-optimal queries very slightly more than the Sum of Weights approach (the exceptions being Basic with $\gamma = 2$ or 3 when the pruning is very much less). For example, the Comparative Preferences model using *Basic* preference statements and with $\gamma = 1$ eliminates 87.5% of satisfiable candidates in dialogues about the Marriott-NY database, whereas the Sum of Weights approach prunes 87.38%.

Table 3. The same final query rate (true preferences represented in Sum of Weights model)

	γ =1	γ =2	γ =3
Marriott-NY			
Comp. Prefs. Basic	98·00	98·80	99·40
Comp. Prefs. Importance	99·00	98·80	99·60
Trentino-10			
Comp. Prefs. Basic	92·60	96·20	98·40
Comp. Prefs. Importance	92·00	93·00	97·40

Table 4. The average shortfalls (true preferences represented in Sum of Weights model)

	γ =1	γ =2	γ =3
Marriott-NY			
Comp. Prefs. Basic	0·0005	0·0000	0·0000
Comp. Prefs. Importance	0·0003	0·0001	0·0002
Sum of Weights		0.0000	
Trentino-10			
Comp. Prefs. Basic	0·0070	0·0010	0·0005
Comp. Prefs. Importance	0·0070	0·0050	0·0020
Sum of Weights		0.0005	

Table 5. The computation time (true preferences represented in Sum of Weights model) in millisecond(ms)

	γ =1	γ =2	γ =3
Marriott-NY			
Comp. Prefs. Basic	46·10	1529·11	2937·20
Comp. Prefs. Importance	44·92	205·61	2466·01
Sum of Weights		59.73	
Trentino-10			
Comp. Prefs. Basic	195·70	6978·69	7493·94
Comp. Prefs. Importance	129·56	726·56	11053·50
Sum of Weights		289.53	

The table also shows that, on the whole, with the Comparative Preference model, the amount of pruning increases as the preference statements induced become less conservative (from Basic to Importance). For example, in the *Trentino-10* part of Table 1, with $\gamma = 2$, pruning goes from 16.51% Basic to 87.57% Importance. (The very slight exception to this for the Trentino-10 $\gamma = 1$ case is probably due to random variation in tie breaking.)

Furthermore, we see that the parameter γ, (the maximum number of variables that are associated with a node in a cp-tree), affects the degree of pruning. Specifically, as γ increases, the number of queries pruned tends to decrease. For example, in the *Marriott-NY* part of Table 1, with preference statements Importance, pruning goes from 87.50% ($\gamma = 1$) to 87.49% ($\gamma = 2$) to 87.42% ($\gamma = 3$). This is a reflection of the monotonicity with respect to γ observed above. (However, pruning is to do with strict dominance, which is not necessarily monotonic with respect to

γ, but very often will be, because of the monotonicity of dominance.) The effect is especially marked in the Basic model where the pruning rate falls from nearly 90% to around 16.5% or less. When γ is increased from 1 to 2, many queries of the *Trade* form become undominated in the Basic model, because of the more expressive preference relations which can be represented by cp-trees with $\gamma = 2$ (allowing more than one feature to be assigned at a node). With the stronger Importance preference form, these *Trade* queries are still dominated.

What is also of concern from a practical point of view is the average length of the advice that the system gives, i.e., the number of options the user has to choose from; this is inversely related to the pruning rate. Except in the cases where pruning is very low (Basic with $\gamma = 2$ or 3), advice from the Sum of Weights recommenders is very slightly longer than it is in the case of the Comparative Preferences recommenders, being around 10 for both datasets.

Table 2 shows dialogue lengths are very similar in the case of all recommenders: around 6 steps on average for Marriott-NY, and around 6.8 steps for Trentino-10.

Of course, it is not enough to know that one approach prunes more than another, or gives shorter advice, or gives rise to shorter dialogues. If it were doing so to the detriment of other factors, in particular the ability of the user to reach the best product, then the extra pruning would be of little value.

We have measured the extent to which the final queries that the user reaches in a dialogue (and hence the final product that she might choose) agree across the different recommenders. We find (see Table 3) that the Comparative Preferences approaches agree with the Sum of Weights approach between 91 and 99% of the time, and the more an approach prunes, the less this agreement is. For example, for Trentino-10, Basic $\gamma = 1$ agrees with Sum of Weights 92.6% of the time; this rises to 96.2% for $\gamma = 2$; and it falls to 92% for Importance $\gamma = 1$.

Since true preferences are represented in the Sum of Weights model, we can also measure the amount by which the utility of the product that the user ultimately chooses falls short of the utility of the best product that she could have reached, normalized by the difference between the products of highest and lowest utility: see Table 4. Unsurprisingly, these follow a similar pattern to the percentage agreements reported in the previous paragraph. The values are very close to zero, ranging from 0 to 0.007.

Table 5 shows the computation time taken by the different implementations of the pruning. When $\gamma = 1$, the time taken by the different implementations of the pruning is roughly similar—for example, around 0.2 seconds for a dialogue with the basic comparative preferences pruning for the Trentino dataset—with the sum of weights linear programming algorithm taking a little longer than the two others. We can see the computation time increasing very strongly with γ, from $\gamma = 1$ to $\gamma = 2$. For example, for the Basic-Trentino the time increases by more than 30 times from 195ms ($\gamma = 1$) to 6978ms ($\gamma = 2$). This is partly because the complexity of the dominance algorithm is exponential in γ.

Overall, for this experimental setup it seems that it is better to use the more restrictive set of models corresponding to $\gamma = 1$, at least for the Basic form, because it then generates much greater pruning (than the models with $\gamma = 2$ or 3), leading to manageable sets of options for the user, and is computationally cheaper. However, there may be situations (e.g. other datasets) where the more cautious reasoning corresponding to $\gamma = 2$ or 3 might pay off in terms of the final quality of solutions.

7.2. *Representing true preferences in the Comparative Preferences model*

In our second set of experiments, the user's true preferences are set by randomly generating cp-trees over product features; more precisely, we use 1-cp-trees, i.e., with $\gamma = 1$. The pruning rate results in this case are shown in Table 6.

Table 6. The pruning rates (true preferences represented in Comparative Preferences model)

	γ =1	γ =2	γ =3
Marriott-NY			
Comp. Prefs. Basic	85·77	14·28	14·28
Comp. Prefs. Importance	85·78	85·77	85·75
Sum of Weights		85.73	
Trentino-10			
Comp. Prefs. Basic	86·81	15·03	14·94
Comp. Prefs. Importance	86·81	86·79	85·93
Sum of Weights		85.15	

Table 7. The computation time (true preferences represented in Comparative Preferences model) in millisecond(ms)

	γ =1	γ =2	γ =3
Marriott-NY			
Comp. Prefs. Basic	29·67	276·20	1076·61
Comp. Prefs. Importance	28·63	134·77	1776·30
Sum of Weights		39.35	
Trentino-10			
Comp. Prefs. Basic	38·52	1578·89	1780·70
Comp. Prefs. Importance	37·37	184·20	3086·39
Sum of Weights		55.72	

The results in Table 6 pattern in a very similar way to the ones in Table 1. For example, again, in most settings, the Comparative Preferences approach is pruning non-optimal queries very slightly more than the Sum of Weights approach; also, with the Comparative Preference model, the amount of pruning increases as the preference statements induced become less conservative (from Basic to Importance); and, as γ increases, the number of queries pruned tends to decrease.

With users' true preferences represented as cp-trees, the pruning rate (Table 6) is roughly the same as when they are represented by weight vectors (Table 1), a little less for the $\gamma = 1$ case. The consequences of this for these experiments is slightly longer advice (around 13 queries on average for both datasets and in all settings except Basic with $\gamma = 2$ or 3, where pruning is very low) and shorter dialogues (around 3.7 steps for Marriott-NY and around 3.9 steps for Trentino-10). Table 7 shows that computation time in these settings shows a similar pattern as when the users' true preferences are represented by vectors of weights. Furthermore, in every dialogue, the product that the user ultimately chose was the optimal product for her true preferences (whereas we saw very small shortfalls in utility in the experiment described in the previous section). The dialogue involves features being added incrementally to the empty initial query—with the most important (according to the true cp-tree) first—until the optimal product is reached.

We were interested to see whether mis-matches between the ways in which true preferences and induced preferences are represented would have any effect. One might expect if induced preferences are represented in the same model as the true preferences, then they can more accurately capture the true preferences, resulting in greater pruning. But we are not seeing this in our experiments. We are seeing that, for most settings, irrespective of the way in which true preferences are represented, using the Comparative Preferences approach for induced preferences results in very slightly greater pruning.

8. Other Kinds of Models of Preferences

In this paper we focus on two kinds of models of user preferences, one based on a weighted sum, the other based on comparative preferences. In this section we briefly discuss some other possible sets of models.

8.1. *Larger sets of models*

One simple idea is to only assume that the user's preference relation is a total pre-order, so we consider the set $\mathcal{M}$ of models to just be the set of all total pre-orders on configurations (this is similar to what is done for the standard inference for CP-nets[13], for example). In particular, let Φ be a set of statements of the form $\alpha_i \geq \beta_i$, for configurations α_i and β_i, $(i = 1, \ldots, k)$. Then, for configurations α and β, we have if $\alpha \succcurlyeq_\Phi \beta$ (α dominates β) if and only if $\alpha \succcurlyeq \beta$ holds for every total pre-order $\succcurlyeq$ satisfying (every member of) Φ. This is very conservative, and perhaps not appropriate for this kind of application, since the inference from Φ is very weak. In particular, it can be shown using a standard argument that $\alpha \succcurlyeq_\Phi \beta$ if and only if (α, β) is in the transitive closure of Φ (viewing Φ as a set of pairs of configurations). More explicitly, $\alpha \succcurlyeq_\Phi \beta$ if and only if there exists a sequence $\delta_1, \ldots, \delta_r$ of configurations, with $\delta_1 = \alpha$, and $\delta_r = \beta$, and, for each $j = 1, \ldots, r-1$, $\delta_j \geq \delta_{j+1}$ is in Φ.

The cp-tree model is parameterised by γ, where the lower the value of γ, the more restricted the models are, and hence the stronger the dominance relation. One could also consider a parameterised generalisation of the Sum of Weights model, where the overall weight function is a sum of component weight functions, where each component function depends on at most γ variables. This is a kind of Generalised Additive Independence (GAI) representation.[25] The Sum of Weights model in the current paper corresponds to the case when $\gamma = 1$. Even for $\gamma = 2$, the set of models is much more expressive, and so will lead to a substantially more conservative (weaker) dominance relation.

8.2. *Strengthening the pruning*

In some circumstances, we may find that neither the sum of weights approach nor the cp-tree approach leads to as strong pruning as we would like, because the set of undominated configurations is still too large. It is natural to then consider ways that we can strengthen these dominance relations. One idea is to prune first with one of the dominance relations (e.g., with $\succcurlyeq_\Phi^{sw}$) and then to prune the remaining $\succcurlyeq_\Phi^{sw}$-undominated configurations with the other ($\succcurlyeq_\Phi^{cp_1}$). A less *ad hoc* approach is to consider preference orderings that can be generated by both 1-cp-trees and by weights vectors: these are a form of lexicographic ordering. In the context of these particular boolean features, such an ordering is generated by a sequence of features, for instance, F_5, F_1, F_4. The associated ordering has, for example, α dominating β for any α and β such that $\alpha(F_5) = f_5$ and $\beta(F_5) = \bar{f}_5$. We can achieve the same ordering in the sum of weights system by choosing, e.g., $w_4 = 1$, $w_1 = 2$ and $w_5 = 4$, and $w_i = 0$ for other features F_i. Alternatively, we can generate the same ordering on configurations with a 1-cp-tree that has the same ordering, F_5, F_1, F_4, of features in each branch.

In general, if the set of models is very large then we will tend to get weak inferences, and the set of undominated elements may well be very large. On the other hand, if we consider a very restricted set of models then it could happen that none of these may correspond closely to the user's preferences. This can lead to equivalences between items that the user does not regard as equivalent; this also reduces the strict dominance relation. To give an extreme example, suppose that we assume the Sum of Weights model, but that in fact the user's true preferences are the cp-tree depicted in Figure 1. If we are given all the associated preferences between configurations, shown at the bottom of Figure 1 ($f_1f_2f_3 \geq f_1f_2\bar{f}_3 \geq \cdots$), then we will infer that $w_1 = w_3 = 0$, which leads to a relatively weak strict dominance relation. The choice of set of models therefore needs to balance these considerations, so that the set of models is restricted enough to make the dominance relation sufficiently strong, but not so restricted that too many equivalences are generated.

9. Conclusions

There has been a lot of excellent theoretical work produced on comparative preference formalisms in recent years including award winning papers[13,26]; however, development towards applications has been lagging somewhat. A major contribution of this paper is to show how a comparative preferences approach can be adapted for a conversational recommender system. For this kind of application it is important that the preference language allows the expression of direct comparisons between configurations (that one configuration is preferred to another); and also that the inference technique is both efficient and strong enough in its inferences (or else the pruning of possibilities is too weak). The language, inference method and algorithm described in (Ref. 16) fit these requirements, as does the Sum of Weights approach.[11] We have shown that it is possible to implement this inference method so that it is efficient in a practical (rather than theoretical) sense, as part of a form of conversational recommender system. We have tested our method on datasets involving real hotel data, and shown that it leads to strong pruning of possibilities, but without eliminating the best options, even when 'best' is defined based on a different semantics, i.e. when the user's true preferences are represented in a model that is different from the one in which the recommender system represents them.

An attractive feature of the comparative preferences approach is its expressiveness. In the current work, it has allowed us to choose between differently nuanced induced preferences. In future work, it would also allow general conditional preference statements to be expressed by the user, such as *If the hotel is not in the city centre, then I'd like there to be an on-site restaurant.* Such statements can further strengthen the pruning capability, and could potentially be re-used for different searches.

In future work, we will extend these approaches for other kinds of recommender systems, including for non-boolean features, for other inference procedures (as discussed in Section 8), and for configurable products, where the set of possibilities is expressed implicitly as the solutions of a Constraint Satisfaction Problem.

Acknowledgements

This material is partly supported by the Science Foundation Ireland under Grant No. 08/PI/I1912.

References

1. G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 6, pp. 734–749, 2005.
2. S. S. Anand and B. Mobasher, "Intelligent techniques for web personalization," in *Intelligent Techniques for Web Personalization*, S. S. Anand and B. Mobasher, Eds. Springer, 2005, pp. 1–36.
3. D. Bridge, M. Göker, L. McGinty, and B. Smyth, "Case-based recommender systems," *The Knowledge Engineering review*, vol. 20, no. 3, pp. 315–320, 2006.

4. F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, Eds., *Recommender Systems Handbook*. Springer, 2011.
5. J. Reilly, K. McCarthy, L. McGinty, and B. Smyth, "Dynamic critiquing." in *Procs. of the 7th European Conference on Case-Based Reasoning*. Springer, 2004, pp. 763–777.
6. D. McSherry, "Retrieval failure and recovery in recommender systems," *Artificial Intelligence Review*, vol. 24, no. 3-4, pp. 319–338, 2005.
7. F. Ricci, D. Cavada, N. Mirzadeh, and A. Venturini, "Case-based travel recommendations," in *Destination Recommendation Systems: Behavioural Foundations and Applications*, D. R. Fesenmaier *et al.*, Eds. CABI, 2006, pp. 67–93.
8. S. Schmitt, "*simVar*: A similarity-influenced question selection criterion for e-sales dialogs," *Artificial Intelligence Review*, vol. 18, pp. 195–221, 2002.
9. C. A. Thompson, M. Göker, and P. Langley, "A personalized system for conversational recommendations," *Journal of Artificial Intelligence Research*, vol. 21, pp. 393–428, 2004.
10. P. Pu, P. Viappiani, and B. Faltings, "Increasing user decision accuracy using suggestions," in *Procs. of the SIGCHI conference on Human Factors in computing systems*. ACM Press, 2006, pp. 121–130.
11. D. Bridge and F. Ricci, "Supporting product selection with query editing recommendations," in *RecSys '07: Proceedings of the 2007 ACM conference on Recommender systems*. New York, NY, USA: ACM, 2007, pp. 65–72.
12. J. Figueira, S. Greco, and M. Ehrgott, *Multiple Criteria Decision Analysis - State of the Art Surveys*. Springer International Series in Operations Research and Management Science Volume 76, 2005.
13. C. Boutilier, R. I. Brafman, C. Domshlak, H. Hoos, and D. Poole, "CP-nets: A tool for reasoning with conditional *ceteris paribus* preference statements," *Journal of Artificial Intelligence Research*, vol. 21, pp. 135–191, 2004.
14. F. Ricci, "Mobile recommender systems," *International Journal of Information Technology and Tourism*, vol. 12, no. 3, 2011.
15. L. McGinty and J. Reilly, "On the evolution of critiquing recommenders," in *Recommender Systems Handbook*, F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, Eds. Springer, 2011, pp. 419–453.
16. N. Wilson, "An efficient deduction mechanism for expressive comparative preference languages," in *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI-09)*, 2009, pp. 961–966.
17. ——, "An efficient upper approximation for conditional preference," in *Proceedings of ECAI-06*, 2006, pp. 472–476.
18. G. H. von Wright, *The logic of preference*. Edinburgh University Press, 1963.
19. S. O. Hansson, "Preference logic," in *Handbook of Philosophical Logic*, D. Gabbay and F. Guenthner, Eds. Kluwer, 2001, pp. 319–393.
20. N. Wilson, "Extending CP-nets with stronger conditional preference statements," in *Proceedings of AAAI-04*, 2004, pp. 735–741.
21. R. Brafman, C. Domshlak, and E. Shimony, "On graphical modeling of preference and importance," *Journal of Artificial Intelligence Research*, vol. 25, pp. 389–424, 2006.
22. S. Bouveret, U. Endriss, and J. Lang, "Conditional importance networks: A graphical language for representing ordinal, monotonic preferences over sets of goods," in *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI-09)*, 2009, pp. 67–72.
23. L. McGinty and B. Smyth, "Adaptive selection: An analysis of critiquing and preference-based feedback in conversational recommender systems," *International Journal of Electronic Commerce*, vol. 11, no. 2, pp. 35–57, 2006.

24. Q. N. Nguyen and F. Ricci, "Replaying live-user interactions in the off-line evaluation of critique-based mobile recommendations," in *RecSys '07: Proceedings of the 2007 ACM conference on Recommender systems*. New York, NY, USA: ACM Press, 2007, pp. 81–88.
25. F. Bacchus and A. Grove, "Graphical models for preference and utility," in *Proceedings of the Eleventh Conference of Uncertainty in Artificial Intelligence (UAI-95)*, 1995, pp. 3–10.
26. F. Koriche and B. Zanuttini, "Learning conditional preference networks with queries," in *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI-09)*, 2009, pp. 1930–1935.