Preference Dominance Approaches for Conversational Recommender Systems

Walid Trabelsi and **Nic Wilson** and **Derek Bridge**[1] and **Francesco Ricci**[2]

Abstract. Conversational recommender systems typically involve iteratively showing the user a small set of options for them to choose between. In order to choose an appropriate set to display at each stage, it is useful to have information regarding the user's relative preference between options: specifically whether an option is dominated by others with respect to the user's preference relation. This paper describes a simple semantic framework for preference dominance, and two instances of the framework are developed for a recent kind of conversational recommender system. The first is based on a basic quantitative preferences formalism, where products are compared using sums of weights of features, and the input language involves comparisons between options. The second is a qualitative preference formalism, where models are a kind of generalised lexicographic order, and the inputs are comparative preference statements (in a language generalising CP-nets). 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 products. The two approaches have been implemented, with encouraging experimental results.

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]. These systems infer user preferences from data gathered either explicitly, e.g. in the form of product ratings, or implicitly by observing user behaviour. The prediction algorithms used within recommender systems include collaborative filtering, content-based filtering and case-based reasoning [1, 6].

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. Conversational recommenders 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. This is also an opportunity for the recommender system to guide the user by asking questions, giving advice, displaying candidate products, and giving explanations [11, 6, 13, 12]. In 2007, Bridge & Ricci introduced a new kind of *conversational recommendation*, which they call *Information Recommendation* [7]. We describe Information Recommendation in Section 5. We use it as the setting in which we explore different models of user preferences.

[1] Department of Computer Science, University College Cork, Ireland, email: {w.trabelsi,n.wilson}@4c.ucc.ie; d.bridge@cs.ucc.ie
[2] Faculty of Computer Science, Free University of Bozen-Bolzano, email: fricci@unibz.it

Conversational recommender systems typically involve iteratively showing the user a small set of options (e.g., products) for them to choose between. In order to choose an appropriate set to display at each stage, from a much larger collection of options, it is useful to have information regarding which options are likely to be preferred to others by the user, based on previous responses they have 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.

This paper describes, in Section 2, a simple semantic framework for this kind of reasoning, where a set of models of the user is assumed, with an associated preference ordering, along with a satisfaction relation between models and preference statements in an appropriate language. Based on such a framework, given a set of input statements, we infer that one choice is preferred to another if this preference holds for all models satisfying the input statements. Two instances of the framework are developed. The first, described in Section 3, is based on a simple quantitative preferences formalism, involving a sum of weights, with an associated language of linear inequalities. The second (see Section 4) is a qualitative preference formalism, where models are a kind of generalised lexicographic order, and the inputs are comparative preference statements (in a language generalising CP-nets [3]). A crucial feature of both methods is that deductions can be made efficiently, since this procedure of testing dominance needs to be applied for many pairs of products.

Both the quantitative and the qualitative approaches are developed in Section 5 for Information Recommendation [7]. Implementations of the methods have been experimentally tested, as described in Section 6.

2 A Framework for Dominance of Preferences

We assume a set Ω of outcomes, representing possible choices, for example, products that a user is interested in buying. We would like to generate some kind of (partially ordered) preference relation on Ω, based on previous information we have received regarding the user's preferences. This section describes a simple semantic framework for generating such a preference relation. We will define in Section 3 and Section 4 two instances of the framework.

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

- A set of models $\mathcal{M}$: each of these 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 outcomes, i.e., a reflexive, transitive and complete relation (so for all outcomes α and β, we have either $\alpha \succcurlyeq_M \beta$ or $\beta \succcurlyeq_M \alpha$ (or both).

- A formal input preference language $\mathcal{L}$, used to represent statements induced from the user's responses.
- 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.

For any particular situation the assumption is that we will be given a particular set $\Phi \subseteq \mathcal{L}$ of input statements. We can then consider the preference orderings on outcomes which hold for every model satisfying input statements Φ. Formally, we define relation $\succcurlyeq_\Phi$ on outcomes 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 outcomes. Now, $\alpha \succcurlyeq_\Phi \beta$ means that every user who agrees with Φ considers that outcome α is at least as desirable as outcome β (assuming this particular model of users). It is possible that we also have $\beta \succcurlyeq_\Phi \alpha$, Define 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 $\alpha \succ_\Phi \beta$, i.e., if all users M agreeing with Φ regard α as at least as preferable as β (i.e., $\alpha \succcurlyeq_M \beta$), and at least one user M regards α as strictly preferable to β (i.e., $\beta \not\succcurlyeq_M \alpha$).

Outcomes generated from features. In this paper we consider a set of outcomes with a particular structure. We assume a collection of features $V = \{F_1, \ldots, F_n\}$. Define a configuration α to be a mapping from $\{1, \ldots, n\}$ to $\{1, 0\}$. Configuration α can also be thought of a set of features, i.e., all features F_i such that $\alpha(i) = 1$. 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. We assume here that having a feature is always at least as good as failing to have it. The set of outcomes is then a subset of the set of configurations, corresponding to choices (e.g., products) that are available to the user. For convenience we will also write a configuration such as $(1,0,1)$ as $f_1 \bar{f}_2 f_3$.

3 Sum of Weights Model of the User

In this section we consider our first kind of model of users, where it is assumed that a user assigns a weight to each feature, and outcomes are compared on the sum of weights of the associated set of features. This is a very commonly used model for preference representation, specifically, in Multi-Attribute Utility Theory (MAUT) [8].

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 can be considered as a weighting 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, i.e., $w(\alpha) = \sum_{i:\alpha(i)=1} w_i$. 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., iff $\sum_i w_i(\alpha(i) - \beta(i)) \geq 0$. Thus $\succcurlyeq_w$ is a total pre-order on configurations.

Preference language: this consists of statements of the form $\alpha \geq \beta$, where α and β are outcomes. Weight vector w is defined to satisfy $\alpha \geq \beta$ if $\alpha \succcurlyeq_w \beta$. Let Φ be a set of input preference statements. The definitions in Section 2 lead to the following definition of $\succcurlyeq_\Phi$, the induced preference relation given preference statements Φ:

For outcomes 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 comparing with the second semantics, we also use the notation $\succcurlyeq^{sw}_\Phi$ for $\succcurlyeq_\Phi$.

Example 1. Let Φ be the pair of preference statements: $f_1 \bar{f}_2 f_3 \geq \bar{f}_1 f_2 f_3$, and $f_1 f_2 \bar{f}_3 \geq \bar{f}_1 f_2 f_3$. Let α be the outcome $f_1 \bar{f}_2 \bar{f}_3$, and let β be the outcome $\bar{f}_1 f_2 f_3$. Weights vector w satisfies $f_1 \bar{f}_2 f_3 \geq \bar{f}_1 f_2 f_3$ if and only if $w(f_1 \bar{f}_2 f_3) \geq w(\bar{f}_1 f_2 f_3)$, 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 $\alpha \succcurlyeq^{sw}_\Phi \beta$, since, for example, weights vector w with $w_1 = 4$, $w_2 = 2$ and $w_3 = 3$ satisfies Φ but does not satisfy $\alpha \geq \beta$.

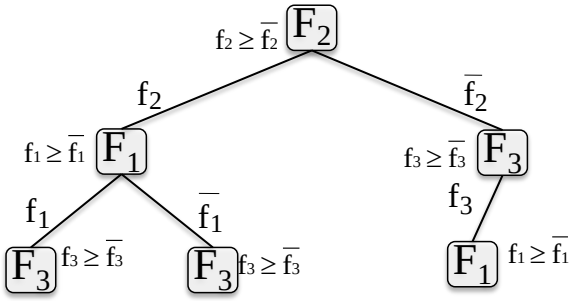
Example 2. Suppose now that there are four features, and let Ψ be the pair of preference statements $f_1 \bar{f}_2 f_3 f_4 \geq \bar{f}_1 f_2 f_3 f_4$ and $f_1 f_2 \bar{f}_3 \bar{f}_4 \geq f_1 \bar{f}_2 f_3 \bar{f}_4$. With the sum of weights semantics this implies $f_1 f_2 \bar{f}_3 f_4 \geq \bar{f}_1 f_2 f_3 f_4$, 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 $f_1 f_2 \bar{f}_3 f_4 \succcurlyeq^{sw}_\Psi \bar{f}_1 f_2 f_3 f_4$.

Computational aspects: 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 $\sum_i (\alpha(i) - \beta(i)) w_i \geq 0$.

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 $\sum(\alpha(i) - \beta(i)) w_i$ subject to constraints $\Phi \cup Pos$. It can be easily shown that $\alpha \succcurlyeq_\Phi \beta$ if and only $\Phi \cup Pos$ entails $\sum_i (\alpha(i) - \beta(i)) w_i \geq 0$ if and only if $A_{min} \geq 0$.

4 Comparative Preferences Model of the User

In this scenario, models are a kind of generalised lexicographic order, called *cp-trees* [16], which are similar to search trees used for solving constraint satisfaction problems. Figure 1 gives an example of a cp-tree. There is a most important variable, F_2, and an ordering of its values. In order to be consistent with the requirement that including a feature is never worse than not including it, this local ordering is $f_2 \geq \bar{f}_2$, where f_2 means F_2 is included ($F_2 = 1$) and $\bar{f}_2$ means that F_2 is not included ($F_2 = 0$).



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

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

Two configurations α and β are compared first on this most important variable. If they do not agree on this variable then the comparison is settled: If α contains feature F_2 and β does not, then α is better than β. This happens, for example, if α is the outcome $\overline{f_1} f_2 \overline{f_3}$ 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; which this is can depend on the value assigned to the most important variable (thus 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 outcomes.

Note that a 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 e.g., $\gamma = 1$, 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., $f_2 f_3 \geq f_2 \bar{f}_3 \equiv \bar{f}_2 f_3 \geq \bar{f}_2 \bar{f}_3$. 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. 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. Full definitions of cp-trees are given in [16].

The Language: The preference language will include preference statements that compactly express comparative (and sometimes conditional) preferences. There is a substantial and fast growing literature on this topic in the AI and Philosophy of Science communities, see e.g., [14, 9, 3, 15, 5, 4, 16].

Each statement φ in the language has an interpretation φ^*, which is a relation between outcomes, giving the direct implications of φ regarding preferences between outcomes. We say that total pre-order $\succcurlyeq$ satisfies φ iff $\succcurlyeq$ extends φ^*, i.e., if $(\alpha, \beta) \in \varphi^*$ implies $\alpha \geq \beta$.

The language will include only comparative preference statements φ of the form $p \geq q \parallel T$, where P, Q and T are sets of features, 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. Formally, the semantics of this statement is given by the relation φ^* which is defined to be the set of pairs (α, β) of outcomes such that α extends p (i.e., α restricted to P equals p), and β extends q, and α and β agree on T: $\alpha(T) = \beta(T)$. A very important kind of input statement is one expressing a preference for one outcome, α, over another, β. This can be written as $\alpha \geq \beta \parallel \emptyset$; as above 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 of input statements, the preference statement $f_i \geq \overline{f_i} \parallel V \setminus \{F_i\}$ for each feature F_i. Hence our language is strongly related to *Conditional Importance Networks* [4].

Comparative preference 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 outcomes. Let σ be a cp-tree and let φ be an input comparative preference statement. σ satisfies φ if and only if $\succcurlyeq_\sigma$ satisfies φ, which is if and only if $\succcurlyeq_\sigma$ extends φ^*. In the same way as in Section 2 and Section 3, we define, for a given set of input comparative preference 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.

Example 1 continued. With the cp-tree semantics, when $\gamma = 1$, the pair of statements Φ implies the preference statement $f_1 \bar{f}_2 \bar{f}_3 \geq \bar{f}_1 f_2 f_3$, so we have $f_1 \bar{f}_2 \bar{f}_3 \succcurlyeq_\Phi^{cp_1} \bar{f}_1 f_2 f_3$. The reason is that, for any 1-cp-tree σ satisfying $f_1 \bar{f}_2 f_3 \geq \bar{f}_1 f_2 f_3$, 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 $f_1 \bar{f}_2 f_3 \succcurlyeq_\sigma \bar{f}_1 f_2 f_3$, 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 $f_1 f_2 \bar{f}_3 \geq \bar{f}_1 f_2 f_3$, then the most important feature must be either F_1 or F_2. Hence for any 1-cp-tree satisfying Φ, F_1 is the most important variable. The top node then decides pair of outcomes $f_1 \bar{f}_2 \bar{f}_3$ and $\bar{f}_1 f_2 f_3$. Since the local ordering of this node must be $f_1 \geq \bar{f}_1$, we have $f_1 \bar{f}_2 \bar{f}_3 \succeq_\sigma \bar{f}_1 f_2 f_3$. Hence we have $f_1 \bar{f}_2 \bar{f}_3 \succcurlyeq_\Phi^{cp_1} \bar{f}_1 f_2 f_3$. The qualitative and lexicographic nature of the cp-trees semantics ensures this inference, in contrast with the numerical sum of weights method.

Example 2 continued. In contrast with the sum of weights semantics, Ψ does not imply $f_1 f_2 \bar{f}_3 f_4 \geq \bar{f}_1 f_2 f_3 f_4$. To show this we can construct a 1-cp-tree σ with F_4 as the most important (top) 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 $f_1 f_2 \bar{f}_3 f_4 \geq \bar{f}_1 f_2 f_3 f_4$.

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.

Computation of Preference: Given set of preference statements Φ, and outcomes α and β we can determine in polynomial time whether or not $\alpha \succcurlyeq_\Phi \beta$ holds, using the algorithm given in [16], as shown by Theorem 1 in [16].

5 Development of Approaches for a Conversational Recommender System

We compare the Sum of Weights Model and the Comparative Preferences Model in the conversational product recommender system application first described in [7]. In this setting, a user repeatedly edits and resubmits a query until she finds a product that she wants. The recommender system: observes the user's actions; infers user preferences; uses the preference statements to deduce which queries a user is likely to try next; and advises the user to avoid those that are unsatisfiable. In [7], users are represented by the Sum Of Weights Model.

Below we describe this recommender system in more detail; we relate it to the framework for dominance of preferences presented above; and we show how users can alternatively be represented by the Comparative Preferences Model.

In this setting, outcomes Ω, over which preferences are expressed, are queries over a set of features $F_1, \ldots, F_n$ (rather than e.g., products). If a user issues query q and if $f_i \in q$, this means that the user is interested in products that have the ith 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 ith feature.

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 although they may have incomplete knowledge of this.

The dialogue

In the kind of system envisaged in [7], the user submits an initial query, typically one that is quite under-specified: 'to test the water'. In our experiments we use an empty initial query.

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 preference statements expressed in preference language $\mathcal{L}$. We will denote the current set of statements by Φ. Initially Φ contains a set of 'background' assumptions. In particular, we wish to express that including a feature in a query is at least as good as not including it. For the Sum of Weights Model, initially $\Phi = Pos$, i.e. the set of constraints $w_i \geq 0$, for $i = 1, \ldots, n$. For the Comparative Preferences Model, initially Φ contains the preference statement $f_i \geq \overline{f_i} \parallel V \setminus \{F_i\}$ for each feature F_i.

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

1. The recommender system analyzes q, with particular regard to differences between current query q and the previous query the user submitted. (In the case where q is the very first query, the previous query is the empty set.) The system induces some additional preference statements to add to inputs Φ.
2. 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.
3. The user chooses and submits her next query. This becomes the new current query q. In the experiments reported in the next section, we arrange that the user always chooses one of the queries that the system advises (although this might not be so in practice.)

Steps 1–3 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 minimizes the total quantity of advice given and the dialogue length, while guiding the user to the best product.

During step 2 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, for each $f_i \notin q$, the set of candidates will include the query that results from adding just feature f_i to q.
- *Satisfiables:* The recommender system never includes 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 1 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 the following, we will 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 [7], 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 $f_i \notin q$, then $Add(q, f_i)$ 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.

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. (It may also have other features, not present in the query.) To produce *Satisfiables* requires eliminating unsatisfiable queries from *Candidates*. 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 can be performed using one of the two strict dominance approaches described earlier. 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*. This dominance relation is based on the set Φ of formal preference statements which are induced in step 1 of the dialogue.

It remains for us to explain what the system induces in step 1 above, when it observes the user's queries. We explain this below for each of the two preference models. Due to space limits, we confine our treatment only to what can be inferred when the system observes the user adding a feature f_i to a query q, $Add(q, f_i)$. However, our system induces preference statements for *Switch* and *Trade* as well as for *Add*.

Generating induced preference statements for sum of weights model: If the user has added feature f_i to query q, 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 value of the new query must be at least as much as the value of 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.

Generating induced preference statements for comparative preferences model: Again consider the situation where the user has chosen to add feature f_i instead of feature f_j. There is more than one way one might induce a comparative preference statement from this decision by the user. We consider two, each being a kind of coun-

terpart for the constraint $w_i \geq w_j$ induced for the sum of weights approach.

Basic: Let q be 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 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.

6 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 conversational recommender system that we described above. 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 about 4056 hotels, of which 133 are distinct.

The simulated users 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 [7], these are *optimizing users*.) The user's true preferences are represented either in the Sum of Weights Model by randomly generating weight vectors over product features or in the Comparative Preferences Model by randomly generating cp-trees over product features. While this vector or cp-tree is known to the simulated user and used for query selection, it is not known to the recommender system, which knows only what it induces and adds to Φ when observing user query behaviour.

In the experiments, we pair the users with each of several recommender systems. 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.

The experiments allow us to compare the pruning rate 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:

$$\textit{pruning rate} = \frac{|\textit{Satisfiables} \setminus \textit{Undominated}|}{|\textit{Satisfiables}|} \times 100$$

It shows the extent to which an approach eliminates what it takes to be inferior satisfiable candidate queries from its advice. Other things being equal, the shorter the advice the better, as this reduces the choice the user has to make. The results in the case where the users' true preferences are represented in the Sum of Weights Model are shown below.

	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		

The table shows that, in nearly all settings, the Comparative Preferences approach is pruning non-optimal queries a little more than the Sum of Weights approach. For example, in the *Marriott-NY* part of the table, 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, where the Sum of Weights approach prunes 87.38%. The tables also show 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 the table with $\gamma = 2$, pruning goes from 16.51% Basic to 87.57% Importance. (The very slight exception to this for the Trentino-$\gamma = 1$ 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 the table 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. 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. 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. Space limits preclude the inclusion of detailed results but we find 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 $\gamma = 1$.

Where the user's true preferences are represented in the Sum of Weights Model, we have also measured 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. 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 0.008.

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. The computation time increases exponentially with γ; for example, the Basic-Trentino-$\gamma = 3$ dialogue takes on average around 7.5 seconds.

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 preferences input form, because, it generates much greater pruning, leading to manageable sets of options for the user, and is computationally cheaper. However, in other situations the more cautious reasoning corresponding to $\gamma = 2$ or 3 might pay off in terms of the final quality of solutions.

We have shown results in the case where the user's true preferences are represented in the Sum of Weights Model. Space limits prevent us from showing what happens when their preferences are modeled using cp-trees. Suffice to say that the numbers are very similar and follow similar patterns to those reported above.

7 Discussion

There has been a lot of excellent theoretical work produced on comparative preference formalisms in recent years, for example, the award winning papers [3, 10]; 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 outcomes (that one outcome is preferred to another); and also that the inference technique is both efficient and rather adventurous in its inferences (or else the pruning of possibilities is too weak). The language, inference method and algorithm described in [16] fit these requirements. 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 (based on a sum of weights of included features).

An attractive feature of the comparative preferences approach is that 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, and for configurable products, where the set of possibilities is expressed implicitly as the solutions of a Constraint Satisfaction Problem. We will also consider other inference procedures. For instance, the weighted sum model can be easily extended to a sum of functions of more than one variable (a GAI representation [2]).

Acknowledgements

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

REFERENCES

[1] Gediminas Adomavicius and Alexander 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*, **17**(6), 734–749, (2005).

[2] F. Bacchus and A. Grove, 'Graphical models for preference and utility', in *Proc. UAI-95*, pp. 3–10, (1995).

[3] 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*, **21**, 135–191, (2004).

[4] Sylvain Bouveret, Ulle Endriss, and Jérôme Lang, 'Conditional importance networks: A graphical language for representing ordinal, monotonic preferences over sets of goods', in *Proc. of IJCAI*, pp. 67–72, (2009).

[5] R. Brafman, C. Domshlak, and E. Shimony, 'On graphical modeling of preference and importance', *Journal of Artificial Intelligence Research*, **25**, 389–424, (2006).

[6] Derek Bridge, Mehmet Göker, Lorraine McGinty, and Barry Smyth, 'Case-based recommender systems', *The Knowledge Engineering review*, **20**(3), 315–320, (2006).

[7] Derek Bridge and Francesco Ricci, 'Supporting product selection with query editing recommendations', in *RecSys '07: Proceedings of the 2007 ACM conference on Recommender systems*, pp. 65–72, New York, NY, USA, (2007). ACM.

[8] 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.

[9] S. O. Hansson, 'Preference logic', in *Handbook of Philosophical Logic*, eds., D. Gabbay and F. Guenthner, 319–393, Kluwer, (2001).

[10] Frédéric Koriche and Bruno Zanuttini, 'Learning conditional preference networks with queries', in *Proc. IJCAI 2009*, pp. 1930–1935, (2009).

[11] David McSherry, 'Retrieval failure and recovery in recommender systems', *Artificial Intelligence Review*, **24**(3-4), 319–338, (2005).

[12] Pearl Pu, Paolo Viappiani, and Boi Faltings, 'Increasing user decision accuracy using suggestions', in *Procs. of the SIGCHI conference on Human Factors in computing systems*, pp. 121–130. ACM Press, (2006).

[13] Francesco Ricci, Dario Cavada, Nader Mirzadeh, and Adriano Venturini, 'Case-based travel recommendations', in *Destination Recommendation Systems: Behavioural Foundations and Applications*, eds., D. R. Fesenmaier et al., 67–93, CABI, (2006).

[14] G. H. von Wright, *The logic of preference*, Edinburgh University Press, 1963.

[15] N. Wilson, 'Extending CP-nets with stronger conditional preference statements', in *Proceedings of AAAI-04*, pp. 735–741, (2004).

[16] N. Wilson, 'An efficient deduction mechanism for expressive comparative preference languages', in *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI-09)*, pp. 961–966, (2009).