Machine Learning for Utility Prediction in Argument-Based Computational Persuasion

Ivan Donadello[1], Anthony Hunter[2], Stefano Teso[4], Mauro Dragoni[3]
[1] Free University of Bozen-Bolzano, Italy
[2] University College London, United Kingdom
[3] Fondazione Bruno Kessler, Italy
[4] University of Trento, Italy
ivan.donadello@unibz.it, anthony.hunter@ucl.ac.uk, stefano.teso@unitn.it, dragoni@fbk.eu

Abstract

Automated persuasion systems (APS) aim to persuade a user to believe something by entering into a dialogue in which arguments and counterarguments are exchanged. To maximize the probability that an APS is successful in persuading a user, it can identify a global policy that will allow it to select the best arguments it presents at each stage of the dialogue whatever arguments the user presents. However, in real applications, such as for healthcare, it is unlikely the utility of the outcome of the dialogue will be the same, or the exact opposite, for the APS and user. In order to deal with this situation, games in extended form have been harnessed for argumentation in Bi-party Decision Theory. This opens new problems that we address in this paper: (1) How can we use Machine Learning (ML) methods to predict utility functions for different subpopulations of users? and (2) How can we identify for a new user the best utility function from amongst those that we have learned? To this extent, we develop two ML methods, EAI and EDS, that leverage information coming from the users to predict their utilities. EAI is restricted to a fixed amount of information, whereas EDS can choose the information that best detects the subpopulations of a user. We evaluate EAI and EDS in a simulation setting and in a realistic case study concerning healthy eating habits. Results are promising in both cases, but EDS is more effective at predicting useful utility functions.

Introduction

Persuasion is an activity that involves one party trying to induce another party to believe something. It is an important and multifaceted human facility. Often, it can involve a dialogue in which arguments and counterarguments are exchanged between the persuader and the persuadee. For example, a doctor can use arguments to persuade a patient to eat a more healthy diet, and the patient can give counterarguments based on their understanding and preferences. Also, the doctor may provide counterarguments to attempt to overturn misconceptions held by the patient.

An automated persuasion system (APS) is a software system that takes on the role of a persuader, and a user is the persuadee (Hunter et al. 2019). It aims to use convincing arguments in order to persuade the user. Whether an argument is convincing depends on its own nature, on the context of the argumentation, and on the user's characteristics. An APS maintains a model of the user to choose the best moves in the dialogue (Hadoux and Hunter 2019). During the turn of the APS, it presents an argument to the user and then user can select their counterarguments from a menu. An APS can also use a natural language interface to allow users input their argument in free text (Chalaguine and Hunter 2020).

In order for an APS to choose the moves it makes in the dialogue (i.e., the arguments it presents), it needs to use a strategy. Within the literature on computational models of arguments, key approaches to strategies are: **Mechanism design** where it is a one-step process rather for multi-step dialogues (Rahwan and Larson 2008; Rahwan, Larson, and Tohmé 2009; Fan and Toni 2012); **Planning systems** to optimize choice of arguments based on belief in premises (Black, Coles, and Bernardini 2014; Black, Coles, and Hampson 2017) or minimize the number of moves made (Atkinson, Bench-Capon, and Bench-Capon 2012); **Machine learning** of strategies such as Reinforcement Learning (Monteserin and Amandi 2013; Rosenfeld and Kraus 2016b; Rach, Minker, and Ultes 2018; Katsumi et al. 2018; Riveret et al. 2019; Alahmari 2020) and transfer learning (Rosenfeld and Kraus 2016a); **Probabilistic methods** to select a move based on what an agent believes the other is aware of (Rienstra, Thimm, and Oren 2013), to approximately predict the argument an opponent might put forward based on data about the moves made by the opponent in previous dialogues (Hadjinikolis et al. 2013), using a decision-theoretic lottery (Hunter and Thimm 2016), using POMDPs when there is uncertainty about the internal state of the opponent (Hadoux et al. 2015); and **Local strategies** based for example on the concerns (i.e., issues raised or addressed by an argument) of the persuadee (Hadoux and Hunter 2019; Chalaguine and Hunter 2020).

Apart from local strategies based on concerns, the above proposals do not take the user's needs into account. Yet, if an APS is to persuade someone to accept some arguments, then how those arguments relate to their needs is fundamental in deciding how to argue with them. This can be captured by considering the utility of each argument from the point of view of both the APS and the user (i.e., we have a utility function for each agent). For this we can harness games in extensive form (Osborne and Rubinstein 1994). For argumentation, this has led to Bi-party Decision Theory (BDT)

(Hadoux, Hunter, and Polberg 2018) that generates a policy that is a good compromise in maximizing the APS and user utilities. However, BDT requires a utility function to be constructed for groups of users. We address this by using Machine Learning (ML) methods that learn utility functions from data about subpopulations of users. Then, an appropriate utility function for a new user (i.e., which subpopulation the user belongs to) has to be determined. This can be performed by asking questions. However, a user might be reluctant to answer too many questions (e.g., because they become bored, or it takes too much effort). So the challenge is to ask fewer questions and still get a useful utility function. We address these novel problems by studying the following research questions: **RQ1** How do we use ML-methods to identify a utility function for each subpopulation of a set of users? **RQ2** How do we optimize the number of questions we ask a new user to identify an appropriate utility function?

Differently from **Recommender Systems** (RSs) (Ricci, Rokach, and Shapira 2015), our proposal is based on BDT and, therefore, it does not provide "one-shot" recommendations based only on users' utilities but a dialogue that considers both users' and system (i.e., biparty) utilities. BDT explicitly models the persuader's and user's needs. This allows a better customization of the APS according to the specific domain and subpopulation. **Conversational RSs** (CRSs) (Jannach et al. 2021) go beyond standard "one-shot" RSs by creating a dialogue with users. However, as far as we know, no method tackles the problem of learning a utility function in a BDT setting for CRSs. **Reinforcement Learning** (RL) (Rosenfeld and Kraus 2016b) is orthogonal to our proposal but it is data inefficient as it involves large amounts of interaction. Our method is less data hungry as it only requires some utility elicitation from users with questionnaires. We addresses the limitations of BDT by developing two ML methods that compute the utility functions from a dataset of user needs. These ML methods allow the optimization of the number of questions to ask. The evaluation using synthetic data reveals promising results and answers to RQ1 and RQ2.

Bi-party Decision Theory

Decision trees (DT) are an important approach to making optimal decisions (Peterson 2009). In computational persuasion, a DT is used for representing all possible dialogues between two agents: each path from the root to a leaf alternates *decision nodes* with *chance nodes*. The former are associated with the persuader (a.k.a. the proponent) and represent the argument to be posed by the APS. The latter are associated with the persuadee (a.k.a. the opponent) and represent the argument posed by the user. Each arc (n, n_i) is labelled with the argument posited by the corresponding agent that has been selected at node n. Figure 1 contains an example of DT for the persuasion goal of reducing the red meat consumption. A *policy* associates a decision node with the best argument to pose by the proponent by considering the points of view of both the proponent and the opponent. In Bi-party Decision Theory, these viewpoints are modelled using two *utility functions* U^p and U^o. These provide each leaf with a utility value that represents the benefit, for the corresponding agent, of accepting the APS arguments in the dialogue from the root to that leaf

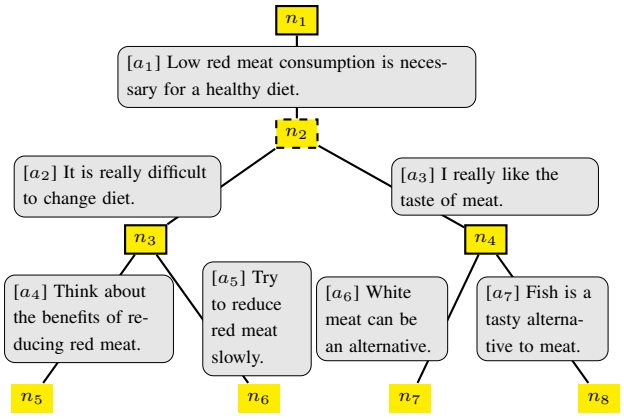

Figure 1: A decision tree for a persuasive dialogue for the red meat persuasion goal. Proponent (decision) nodes are solid boxes and opponent (chance) nodes are dashed boxes.

A **bimaximax policy** maximizes U^p at a decision node and U^o at a chance node. Some notation is necessary before a formal definition of the policy. Let T be a DT, L be a labeling function that associates an argument to each arc (n, n_i) between two nodes of T and $\mathsf{Children}(T, n)$ be the set of children of n. The $\mathsf{AMax}(T, U, n)$ function returns the children of n with highest utility U: $\mathsf{AMax}(T, U, n) = \{n' \in \mathsf{Children}(T, n) \mid \forall n'' \in \mathsf{Children}(T, n), U(n') \geq U(n'')\}$ Let $\delta \in \mathbb{R}$ be a discount factor that decreases the utility of longer branches. Indeed, shorter dialogues can be more persuasive than longer ones that require more attention from the user. Definition 1 defines the bimaximax policy.

Definition 1. *A* **bimaximax policy** *for* (T, L, U^p, U^o, δ) *is* $\Pi : \mathsf{Nodes}(T) \rightarrow \mathsf{Nodes}(T)$ *defined as follows using the calculation of the* $Q^p : \mathsf{Nodes}(T) \rightarrow \mathbb{R}$ *and* $Q^o : \mathsf{Nodes}(T) \rightarrow \mathbb{R}$ *functions.*

- *If* n *is a leaf node, then* $Q^p(n) = U^p(n)$ *and* $Q^o(n) = U^o(n)$.
- *If* n *is a chance node, and* $n_i \in \mathsf{AMax}(T, Q^o, n)$*, then* $Q^o(n) = \delta \times Q^o(n_i)$ *and* $Q^p(n) = \delta \times Q^p(n_i)$.
- *If* n *is a decision node, and* $n_i \in \mathsf{AMax}(T, Q^p, n)$*, then* $Q^o(n) = \delta \times Q^o(n_i)$ *and* $Q^p(n) = \delta \times Q^p(n_i)$ *and* $\Pi(n) = \langle n_i, L(n, n_i)\rangle$.

For the example in Figure 1, the APS aims to persuade the user to adopt healthy alternatives to red meat, the leaf utility values are then $U^p(n_5) = 2$, $U^p(n_6) = 3$, $U^p(n_7) = 4$, $U^p(n_8) = 6$. The user is interested in consuming white meat and to slowly reduce his/her consumption of red meat: $U^o(n_5) = 2$, $U^o(n_6) = 6$, $U^o(n_7) = 7$, $U^o(n_8) = 4$. The utility values are then propagated to the higher levels of the tree as follows, where $Q^{p/o}(n)$ denotes $(Q^p(n), Q^o(n))$: $Q^{p/o}(n_1) = (3, 6)$, $Q^{p/o}(n_2) = (3, 6)$, $Q^{p/o}(n_3) = (3, 6)$, $Q^{p/o}(n_4) = (6, 4)$, with $\delta = 1$, $\Pi(n_1) = (n_2, L(n_2))$, $\Pi(n_3) = (n_6, L(n_6))$ and $\Pi(n_4) = (n_8, L(n_8))$.

Definition 1 does not provide a criterion for the choice of

n_i when $|\mathsf{AMax}(T, Q, n)| > 1$. We therefore adopt a random choice that improves the variability of the APS messages.

Simulated Dialogues

The bimaximax policy defines a rule for choosing the label/argument to pose given a decision node n. Given a chance node n_i, the next decision node n_j is selected by the user via, for example, a menu. However, when real users are not available the next decision node has to be computed according to some *simulated opponent policy* Π^o. This policy can be computed by statistical methods that rely on datasets of conversations with users, by harvesting arguments from the web or with the use of questionnaires. Here, we simply define the opponent policy as $\Pi^o(n_i) = \langle n_j, L(n_i, n_j)\rangle$ with $n_j \in \mathsf{AMax}(T, Q^o, n_i)$. In our example $\Pi^o(n_2) = (n_3, L(n_3))$. We implement Definition 1 in a procedure, called $\mathrm{Bimaximax}$, that propagates up the leaf utilities at the parents, with a post-order tree traversal, and outputs Π, Π^o. A *simulated dialogue procedure*, Algorithm 1, consists in the computation of the policies Π, Π^o (line 2) and in alternating between the proponent and opponent arguments (given by Π, Π^o) from the root to a leaf (lines from 4 to 9). The output path $\boldsymbol{p}$ contains the proponent/opponent nodes from the root to a leaf. In our example, $\boldsymbol{p} = \langle n_1, n_2, n_3, n_6\rangle$.

Algorithm 1: $\mathrm{SimDialogue}(T, L, \boldsymbol{u}^p, \boldsymbol{u}^o, \delta)$

Input: $T, L, \boldsymbol{u}^p, \boldsymbol{u}^o, \delta$
Output: The path $\boldsymbol{p} = \langle \mathrm{root}(T), n_1, \ldots, n_{\mathsf{height}(T)}\rangle$

1: $\boldsymbol{p} := \langle\rangle$
2: $\Pi, \Pi^o := \mathrm{Bimaximax}(T, L, \boldsymbol{u}^p, \boldsymbol{u}^o, \delta)$
3: $n := \mathrm{root}(T)$
4: **while** $\mathrm{isNotLeaf}(n)$ **do**
5: **if** $\mathrm{isDecision}(n)$ **then**
6: $\langle n, l_n\rangle := \Pi(n)$
7: **else**
8: $\langle n, l_n\rangle := \Pi^o(n)$
9: $\boldsymbol{p}.\mathrm{append}(n)$
10: **return** $\boldsymbol{p}.\mathrm{append}(n)$

Let $\boldsymbol{u}^p = \langle U^p(n_1), U^p(n_2), \ldots\rangle$ and $\boldsymbol{u}^o = \langle U^o(n_1), U^o(n_2), \ldots\rangle$ be the proponent and opponent utility vectors respectively, with $n_i \in \mathsf{leaves}(T)$. The values of $\boldsymbol{u}^p$ can be authored by domain experts from the substantial evidence in healthcare literature on the health benefits of specific behaviour changes, e.g., the value of changing from high red meat consumption to white meat consumption (Abete et al. 2014). Instead, the setting of $\boldsymbol{u}^o$ is an open challenge. In the next section, we define two ML methods able to predict an accurate estimation for $\boldsymbol{u}^o$ from a dataset of users' utilities. These methods can be used also in a dialogue procedure with real users where the policy Π^o will be replaced, for example, by a menu in a smartphone app that implements the APS.

Machine Learning for Utility Prediction

Let T be a DT, a corresponding utility dataset $\mathcal{U}^o$ contains the utility values for a population sample. We encode $\mathcal{U}^o$ as a matrix where each row j contains the utility vector $\boldsymbol{u}^o_j$ of the arguments in $\mathsf{leaves}(T)$ for a given user. Such a dataset can be gathered by using, for example, utility elicitation techniques based on questionnaires (Lenert et al. 1997; Lenert, Sherbourne, and Reyna 2001). Here, we assume that $\mathcal{U}^o$ contains samples of subpopulations (as commonly done in RSs), that is, clusters of users who have similar utilities. For example, according to the leaf arguments in Figure 1, we can have three subpopulations/clusters in $\mathcal{U}^o$: C_1, C_2 and C_3. The users in C_1 like to eat food containing animal proteins, the users in C_2 have difficulties changing their diet habits whereas the users in C_3 are more open minded. Such a dataset can be gathered with cluster sampling techniques (Turk and Borkowski 2005).

We develop two ML methods, trained on $\mathcal{U}^o$, for predicting an estimation of the opponent utility vector $\hat{\boldsymbol{u}}^o$, for a given user, as close as possible to the true vector $\boldsymbol{u}^o$. This implies choosing some features of the users as input to our ML methods to predict $\hat{\boldsymbol{u}}^o$. These can be demographic information or some arguments in $\mathsf{leaves}(T)$. We consider these input utilities as the *evidence* to be obtained in order to predict the rest of the utility vector. Therefore, the predicted utility vector is composed from the evidence $\boldsymbol{u}^o_{1:e}$, with $e \in [1, Le-1]$ and $Le = |\mathsf{leaves}(T)|$, and the predicted utilities: $\hat{\boldsymbol{u}}^o_e = \langle \boldsymbol{u}^o_{1:e}; \hat{\boldsymbol{u}}^o_{e+1:Le}\rangle$. In our example, let us suppose that a user in cluster C_1 has true utilities $\boldsymbol{u}^o = \langle 5, 4, 8, 9\rangle$. A ML method, trained on $\mathcal{U}^o$, can ask as evidence the utilities for n_5, n_6 ($e = 2$ out of 4 possible questions) to predict the utilities of n_7, n_8: 9 and 7, for example. Therefore, $\hat{\boldsymbol{u}}^o_2 = \langle 5, 4, 9, 7\rangle$ and this is a good prediction as the true path returned by $\mathrm{SimDialogue}$ is $\boldsymbol{p} = \langle n_1, n_2, n_4, n_8\rangle$ whereas the path returned by using $\hat{\boldsymbol{u}}^o_2$ is $\hat{\boldsymbol{p}} = \langle n_1, n_2, n_4, n_7\rangle$ that is quite similar to the true one. Indeed, both n_7 and n_8 contain arguments related to the animal proteins cluster.

Our proposal consists of providing $\mathrm{SimDialogue}$ with the opponent utility vectors $\hat{\boldsymbol{u}}^o_e$ predicted with ML methods as input. The evidence e obtained from the user has to be enough to have a good dialogue path $\hat{\boldsymbol{p}}$ but without asking too many questions of the user (otherwise they may disengage). For example, asking 50 utilities of a DT with 60 leaves will bring reliable paths but would be too many questions. More formally, let ML be a machine learning method trained on $\mathcal{U}^o$ to compute the opponent utility vector $\hat{\boldsymbol{u}}^o_e$ from $\boldsymbol{u}^o_{1:e}$, that is, $\mathrm{ML}(\boldsymbol{u}^o_{1:e}) = \hat{\boldsymbol{u}}^o_e$, and let $\mathcal{R}$ be a regret function that measures the difference from the path returned by $\mathrm{SimDialogue}$ with the one returned by $\mathrm{SimDialogue}(\mathrm{ML})$, here the notation $\mathrm{SimDialogue}(\mathrm{ML})$ is a shortcut for $\mathrm{SimDialogue}(T, L, \boldsymbol{u}^p, \mathrm{ML}(\boldsymbol{u}^o_{j,1:e}), \delta))$. Given a test set $\mathcal{U}^o_{TE}$, the optimal number e^* of evidence to ask about is:

$$e^* = \underset{e\in[1,Le-1]}{\mathrm{argmin}} \sum_{j=1}^{|\mathcal{U}^o_{TE}|} \mathcal{R}(\mathrm{SimDialogue}(T, L, \boldsymbol{u}^p, \boldsymbol{u}^o_j, \delta), \mathrm{SimDialogue}(T, L, \boldsymbol{u}^p, \mathrm{ML}(\boldsymbol{u}^o_{j,1:e}), \delta)) + \mathcal{E}(e) \quad (1)$$

with $\mathcal{E}$ a monotonic-increasing function that models the user effort in answering the asked evidence whose parameters can be estimated, for example, by measuring the user disengagement rate. This is to avoid having an APS that asks

all the possible questions as evidence. The following subsections present our ML methods for utility prediction.

Evidence as Amount of Information

An *evidence as amount of information* (EAI) method uses a portion of the arguments in $\mathcal{U}^o$ (limited by the evidence e) to train a supervised regressor to predict the utility values of the other arguments. This is the standard *regression* task performed with *supervised learning* techniques where a training set $\mathcal{X} \times \mathcal{Y}$ is used to compute an estimator function f for $\mathcal{Y}$. The set $\mathcal{X}$ contains feature vectors $\boldsymbol{x}$ whereas $\mathcal{Y}$ contains numeric values y to be estimated from $\boldsymbol{x}$. The training is performed such that the output $\hat{y}$ of $f(\boldsymbol{x})$ has to be as close as possible to the true value y. In our case, given an evidence index e, the set $\mathcal{X}$ is given by the first e arguments in $\mathcal{U}^o$ and $\mathcal{Y}$ is one of the remaining arguments. More formally, the training set is $\mathcal{U}^o_{*,1:e} \times \mathcal{U}^o_{*,e+l}$ with $l \in [1, Le-e]$. The notation $\boldsymbol{A}_{*,i:j}$ indicates the columns from i to j (included) of matrix $\boldsymbol{A}$. Therefore, for each evidence index $e \in [1, Le-1]$ multiple regressions have to be performed from $\mathcal{U}^o_{*,1:e}$: one for predicting the utility values for each remaining argument with index in $[e+1, Le]$. As the ML method for computing f, we use a **Support Vector Regression** SVR that linearly penalizes the predictions $\hat{y}$ that are at least ϵ away from the true value y. The advantage of an EAI method is that many off-the-shelf ML methods for computing f can be used. However, such methods are restricted to the provided portion $\mathcal{U}^o_{*,1:e}$ of the dataset to learn the other utility values, without any possibility of automatically choosing a better set of arguments to ask about. The next method addresses this issue. We opted for SVR (and RF in the next section) because it is a known-working technique that does not require large training set to reach reasonable performance. Our focus is not on the choice of the ML method but on the usefulness of adopting ML for effective, data driven, argumentation.

Evidence as Depth of Searching

An *evidence as depth of searching* (EDS) method has a limited amount of evidence e to be obtained from users, but, differently from the EAI-based approach, the EDS method is given access to the whole dataset $\mathcal{U}^o$. Therefore, the EDS method can search the most appropriate evidence for the regression of the utility function. This search is limited by the quantity e. We call the method **Cluster and RAndom forest MEan Regressor** (CRAMER) and, in brief, it performs clustering on $\mathcal{U}^o$ to find subpopulations. Then a Random Forest (RF) is trained to learn the most important utilities of arguments to ask in order to classify a new user into a subpopulation. The utility vector $\hat{\boldsymbol{u}}^o_e$ is given by the asked utilities and by the centroids of the subpopulation. We detail the method with the running example.

Firstly, CRAMER performs clustering on $\mathcal{U}^o$ and discovers the underlying clusters and computes their centroids. These are the mean of the utilities of each user in the cluster. Numeric examples for the centroids for C_1, C_2 and C_3 are: $\boldsymbol{c}(C_1) = \langle 5.0, 4.0, 9.2, 9.1\rangle$, $\boldsymbol{c}(C_2) = \langle 3.1, 8.5, 7.2, 1.9\rangle$ and $\boldsymbol{c}(C_3) = \langle 8.1, 8.6, 2.7, 7.3\rangle$. The second step of CRAMER is to train a RF classifier on $\mathcal{U}^o$ to learn from user utilities what are the main arguments that characterize a cluster. RF classifiers require a max depth parameter for setting the maximum depth of their classification trees, i.e., decision trees trained for classification tasks. We set this with e, that is the number of arguments to be asked. A high value for max depth could generate classification trees that overfit $\mathcal{U}^o$ (with consequently low performance) and could be too demanding for the users. In our example, if we set $e = 2$, a random forest is able to classify a new user by simply asking the utilities only for n_8 (fish) and n_7 (white meat). Indeed, high utilities for both these arguments mean that the user is in C_1, whereas a low utility for the fish argument classifies the user in C_2. A high utility for the fish and a low one for the white meat argument bring the classification to C_3. Lastly, the method joins the asked (true) utilities with the centroid (inferred) utilities in the predicted cluster to obtain the utility vector $\hat{\boldsymbol{u}}^o_e$ of the user. For example, the utility vector of a user answering with utility value of 8 for the white meat argument and with utility 7 for the fish argument (cluster C_1) is $\hat{\boldsymbol{u}}^o_2 = \langle 5, 4, 8, 7\rangle$.

Empirical Evaluation

The aim of the evaluation is to test the ability of SimDialogue(ML) (prediction of $\hat{\boldsymbol{u}}^o$ with ML and consequent use of $\hat{\boldsymbol{u}}^o$ in Algorithm 1) of returning persuasive arguments as good as the ones returned by SimDialogue. However, as big datasets of APS dialogues with real users are not available, we test SimDialogue(ML) on synthetic datasets with the use of simulations. We use the output of the SimDialogue procedure as a *gold standard* and evaluate how much the output of SimDialogue(ML) differs from the gold standard. The source code and the supplementary material are online at shorturl.at/oyKV3

Simulation Design

We compare SimDialogue and SimDialogue(ML) on different abstract DTs and population samples with their own utilities. Our aim is to perform a fair comparison between the gold standard and SimDialogue(ML) trying to avoid possible bias. Good simulation results reveal possible good results in real cases too. The simulation steps are as follows:

Trees Generation: given a tree height and a list of branching factors as input, a random abstract DT is generated with a breadth-first algorithm. This recursively generates random children of a node according to a branching factor taken from the input list. This is repeated until the input height has been reached. We repeat this process to obtain a set $\mathbb{T} = \{T_1, T_2, \ldots\}$ of abstract DTs. As they have no particular meaning, the labelling L is not necessary. Each DT represents the structure of a possible persuasion dialogue between an APS and a user.

Datasets Generation: for each $T \in \mathbb{T}$, we synthesize a set $\mathbb{U}^o_T = \{\mathcal{U}^o_{T,1}, \mathcal{U}^o_{T,2}, \ldots\}$ of datasets of opponent utilities. Each dataset $\mathcal{U}^o_{T,i} = \{\boldsymbol{u}^o_{T,i}\}_j$ represents samples of subpopulations containing the utility vector $\boldsymbol{u}^o_{T,i,j}$ for a user indexed by j belonging to a particular subpopulation. This allows the EAI and EDS methods to learn utility functions for subpopulations. Each dataset is used

for: i) training by procedure SimDialogue(ML), ii) testing by both SimDialogue(ML) and SimDialogue. Several datasets of sobpopulations allow us to check the results of SimDialogue(ML) in a robust way.

We synthesize a dataset $\mathcal{U}^o_{T,i}$ by: i) creating clusters of users; ii) assigning each user a utility vector $\boldsymbol{u}^o_{T,i,j}$ such that users in the same cluster have similar utilities. Let T be a DT, $C \in \mathbb{N}$ be an input number of clusters and $CW \in \mathbb{R}$ be an input center width, i) we sample C center points (vectors with $\mathsf{leaves}(T)$ dimensions) from a multivariate uniform distribution with values in $[-CW, CW]$. These vectors are the centers of the clusters. ii) For each center vector $\boldsymbol{c}$, we sample m vectors (with $\mathsf{leaves}(T)$ dimensions) from a multivariate normal distribution with mean $\boldsymbol{\mu} = \boldsymbol{c}$ and covariance matrix $\boldsymbol{\Sigma} = \sigma^2_C \boldsymbol{I}$, with $\boldsymbol{I}$ the identity matrix and σ^2_C the cluster variance given as input. We sample the m vectors such as every cluster has roughly the same number of vectors. All these sampled vectors compose the dataset $\mathcal{U}^o_{T,i}$.

Simulation Run: for each pair $\langle T, \mathcal{U}^o_{T,i}\rangle$ and for a given evidence index $e \in [1, Le-1]$, we split $\mathcal{U}^o_{T,i}$ into a training set, for training a given ML method, and a test set. Let j be the index of a sample in the test set, we run $\text{SimDialogue}(T, L, \boldsymbol{u}^p, \text{ML}(\boldsymbol{u}^o_{T,i,j,1:e}), \delta)$ and $\text{SimDialogue}(T, L, \boldsymbol{u}^p, \boldsymbol{u}^o_{T,i,j}, \delta)$. Remember that an EAI method takes a portion of the training set given by e, whereas an EDS method takes the whole training set but can ask only e questions. For a statistical significance of the results, we use the k-fold cross validation technique. The dataset $\mathcal{U}^o_{T,i}$ is split into k parts, $k-1$ parts are used as training set for SimDialogue(ML) and the remaining part is left as test set for both SimDialogue(ML) and SimDialogue. In this way, k splits/folds of the original dataset $\mathcal{U}^o_{T,i}$ are obtained and for each split we run both SimDialogue(ML) and SimDialogue.

Comparison: the paths $\hat{\boldsymbol{p}}_{T,i,j,e,k}$ and $\boldsymbol{p}_{T,i,j,k}$ returned by the SimDialogue(ML) and SimDialogue procedures, respectively, are compared according to the following metrics.

Metrics for the Empirical Evaluation

Given a user in $\mathcal{U}^o$, let $\hat{\boldsymbol{u}}^o_e$ be their estimated utility vector and $\boldsymbol{u}^o$ be the true one. Given $\hat{\boldsymbol{p}} = \text{SimDialogue}(T, L, \boldsymbol{u}^p, \hat{\boldsymbol{u}}^o_e, \delta)$ and $\boldsymbol{p} = \text{SimDialogue}(T, L, \boldsymbol{u}^p, \boldsymbol{u}^o, \delta)$ (hereafter we remove the subscripts T, i, e, k if not necessary), we consider the last (leaf) nodes of these paths: $\hat{n} = \hat{\boldsymbol{p}}_h$, $n = \boldsymbol{p}_h$ with h the height of T, we call n the *true node* and $\hat{n}$ the *predicted node* and we compute the evaluation metrics by only comparing n with $\hat{n}$. This strategy is motivated by the random choice of n_i when $|\mathsf{AMax}(T, Q, n)| > 1$: this randomness can easily select a different branch in T for SimDialogue(ML) w.r.t. SimDialogue for the same utility vector. This side effect would highly penalize the performance if we compared the whole paths resulting in a non-fair comparison.

The aim of the evaluation is to test the ability of SimDialogue(ML) to return similar nodes to SimDialogue. This similarity can be easily tested with the standard accuracy, i.e., whether $n_j = \hat{n}_j$ for a given user indexed with j in $\mathcal{U}^o$. However, as random choices can easily fail this testing, the similarity between n_j and $\hat{n}_j$ is relaxed by measuring an *argument distance* between the arguments in n_j and in $\hat{n}_j$, respectively. This is the $\mathcal{R}$ term in Equation (1). The nodes n_j and $\hat{n}_j$ have a small argument distance if the opponent and proponent utilities of the argument in n_j are close to the corresponding ones of the argument in $\hat{n}_j$. Good performance would indicate that an APS based on SimDialogue(ML) is able to provide arguments as similar as possible (w.r.t. the utilities) to the gold standard arguments returned by SimDialogue. More formally, the **mean argument distance** (Mad) is the mean, over all the dataset samples, of the Manhattan distance between the proponent and opponent utilities of the true node and the predicted one: $Mad = \frac{1}{|\mathcal{U}^o|}\sum_{j=1}^{|\mathcal{U}^o|} |U^p(n_j) - U^p(\hat{n}_j)| + |U^o(n_j) - U^o(\hat{n}_j)|$. Notice that, if we decompose the mean argument distance according to the proponent ($\frac{1}{|\mathcal{U}^o|}\sum_{j=1}^{|\mathcal{U}^o|} |U^o(n_j) - U^o(\hat{n}_j)|$) and the opponent ($\frac{1}{|\mathcal{U}^o|}\sum_{j=1}^{|\mathcal{U}^o|} |U^p(n_j) - U^p(\hat{n}_j)|$) dimensions we obtain the *mean absolute errors* (Mae) which are standard metrics for regression tasks. These metrics depend on T, i, e, k, therefore the output of the simulations is a set of argument distances. These values need to be aggregated to have a global estimation of the performance.

k-fold aggregation The first aggregation is performed on the number of folds k of the k-fold cross validation. For each triple $\langle T, i, e\rangle$ we compute the mean and the standard deviation of the metrics over all the k values obtaining the values $\mu(Mad)_{T,i,e}$ and $\sigma(Mad)_{T,i,e}$.

Evidence aggregation The second aggregation regards the evidence e requested as input to users in order to compute the utility values of the remaining arguments. We want to study the trend of the performance metrics with a increasing amount of evidence. This amount is measured as the percentage of asked utility arguments over all possible arguments/leaves of a DT. This allows us to check whether there exists a percentage of evidence from which the performance is acceptable. We therefore aggregate the mentioned means by averaging over $\mathbb{T}$ and $\mathbb{U}^o_T$ according to the evidence percentage. Let $Ep \in \{0.1, 0.2, \dots 1.0\}$ be the evidence percentage and $E = \lceil Ep \cdot (Le-1))\rceil$ be the amount of utility of arguments to ask for T and for a given Ep, the mean argument distance aggregation is the average of the set $\{\mu(Mad)\}_{T,i,e}$, with $e \in \{1, 2, \dots E\}$. The standard deviation of this aggregations is computed from this set by estimating the pooled variance.

Simulation Setting

Some values of the parameters in the simulation are uniformly (randomly) selected from a set of alternatives. This avoids the bias of having only one value resulting in more general results. These are: the tree height in $\{4, 5, 6\}$, the proponent utility in $\{1, 2, \dots 11\}$, the number of clusters $C \in \{4, 6, 8, 10\}$ and the center width $CW \in \{0.5, 1.0, 2.5, 3.0\}$. This setting for CW, allow us to have different levels of clustering performance avoiding the bias of population samples that are perfectly separable into clusters. The branching factor for generating the children of a node is 2 or 3 for 90% of the time, 4 for 10% of the time. This

allows us to have realistic DTs with a reasonable number of leaves. Other parameters have a single value: the number of synthetic trees ($|\mathbb{T}| = 10$) and datasets ($|\mathbb{U}^o_T| = 10$), the size of $\mathcal{U}^o_{T,i}$ is 2000, the cluster variance σ^2_C is 1.0, the discount factor δ in $\mathrm{Bimaximax}$ is 1 as it is not relevant for the simulations and $k = 5$ for the k-fold cross validation. The hyperparameters for SVR are $C = 1$, $\epsilon = 0.1$ and the radial basis function as a kernel. For the clustering in CRAMER we used KMeans with the number of clusters found through grid search in $\{4, 6, 8, 10\}$. The random forest in CRAMER has 100 estimators with the minimum number of samples required: i) to split a node is 2, ii) to be a leaf is 1.

Results

We measure the performance of $\mathrm{SimDialogue(ML)}$ (with the average mean argument distance) according to different levels of evidence percentage E_p. This allows us to check: i) whether ML methods are able to learn utility functions from data in comparison to non-ML baselines (RQ1) and ii) whether there exists a percentage of evidence from which the performance are acceptable (RQ2). We compare some baselines with the gold standard in the same way done for $\mathrm{SimDialogue(ML)}$:

- **Random Regressor** (RandR) randomly selects integer utility values between the maximum and the minimum utility for each argument in the columns of of $\mathcal{U}^o_{*,e+1:Le}$.
- **Mean Regressor** (MeanR) computes the mathematical mean (over all the users) of the utility values of each argument in the columns of $\mathcal{U}^o_{*,e+1:Le}$.
- **CLUster and MEan Regressor** (CLUMER) performs clustering to find the subpopulations in $\mathcal{U}^o_{*,1:e}$. Once a new user is assigned to a cluster, the centroids of the clusters are used as utility values for the remaining arguments in the columns of $\mathcal{U}^o_{*,e+1:Le}$.

Aggregated performance are reported in Figure 2 whereas Appendix A of the supplementary material provides examples of performance for some random DTs and datasets. As expected, the EAI method and the baselines increase their performance as the evidence percentage E_p increases. Indeed, more asked questions allow more accurate performance. In addition, the increase of the performance is also due to a robustness property of the bimaximax policy: i.e., the system ability to capture regions of high user's utility regardless the other regions. If few leaves have a random utility most probably their utility will not be the highest one. Indeed, during the bottom-up utility propagation phase in $\mathrm{Bimaximax}$, the utility of these few leaves will be discarded in favour of higher utilities from other branches as the bimaximax policy selects the highest utility at each node. These few leaves do not have the power to change the prediction of the true node. As E_p increases, the number of random utilities decreases. This explains also the unexpected performance improvement of RandR as E_p increases.

Discussion for RQ1 The results for the argument distances are promising as with 10% of evidence SVR obtains a Mad of 2.8. This means that the predicted node has proponent and opponent utilities that on average have distance 1.4 from the true node. As the utility values range in a Likert scale from 1 to 11, a 1.4 distance is an acceptable error. Therefore, the ML methods are able to compute the utility values for the population sample by predicting nodes that contain arguments that are very close (from the utilities point of view) to the arguments contained in the true nodes.

Discussion for RQ2 An acceptable number of questions as evidence is not trivial to identify. For example, 10% of evidence could present low performance for small trees (e.g., with 20 leaves only 2 questions are asked) therefore choosing 30% could represent a better choice. However, this could result in a high amount of requested information (e.g., 30 questions for a tree with 100). To this extent, CRAMER gives us a different view. For this method, 10% of evidence is required to identify the subpopulation and the utilities are the centroids of each subpopulation. Therefore, for a tree with 20 leaves, a depth of 2 for the random forest is sufficient to identify 4 subpopulations. In general, a depth of D identifies 2^D subpopulations. It is important to notice that the level of depth of search is a *maximum* level for the random forest, that is, the random forest can use less information to identify subpopulations. This can be seen in the picture as the performance of CRAMER are stable as the depth of search increases. The EAI methods need 50% of evidence to have the same performance of CRAMER.

The advantage of an EDS method relies in its ability of automatically selecting the most appropriate arguments to ask from $\mathcal{U}^o$ obtaining good performance with a low E_p. Differently, an EAI method has only a partial view of the whole information but it increases the performance with higher E_p. The drawback of CRAMER is its rough computation of $\hat{\boldsymbol{u}^o_e}$ given a subpopulation. SVR instead has a more accurate computation based on the error optimization of the predicted utility. This can be seen by its better performance with respect to the EAI method CLUMER. In addition, CLUMER and SVR have similar performance as the former explicitly leverages the cluster structure of the users, the latter achieves the same through the underlying Gaussian kernel.

A Realistic Case Study

We evaluate $\mathrm{SimDialogue(ML)}$ on a realistic case study where the DT contains real arguments regarding the reduction of red meat consumption. Both the DT and the synthetic dataset are provided. This is the first step for implementing an APS for following healthy lifestyles. This was commissioned by the Local Healthcare Department of Trento (Italy), for the deployment in the Trentino area.

Regarding the DT, we gathered the arguments and their relations from diet and behavior change literature (Abidi et al. 2014; Stankevitz et al. 2017). We obtained a DT with 23 leaves and height of 4. Concerning $\boldsymbol{u}^p$, we labeled each leaf with a topic describing the content of the arguments in the corresponding path from the root. In our example, the path n_1, n_2, n_4, n_8 is labeled with "Fish as alternative". Each topic is assigned with a proponent utility according to the topic importance from a diet point of view. Each leaf n is assigned with the proponent utility of the corresponding topic. These topics are: vegetarianism ($U^p(n) = 8$), fish as alternative ($U^p(n) = 6$), white meat as alternative

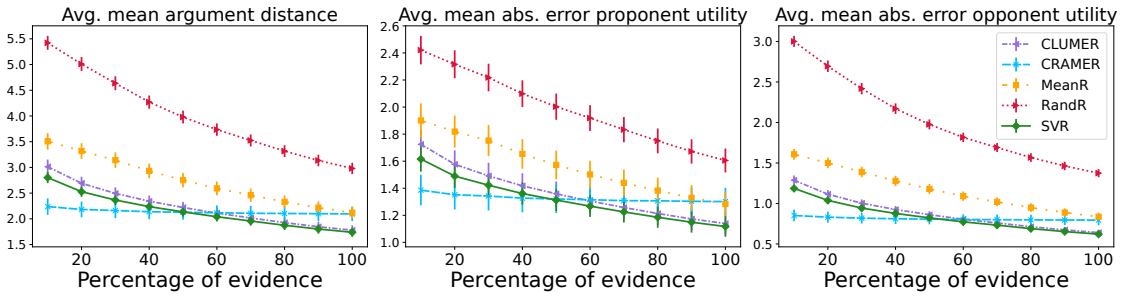

Figure 2: Performance of SimDialogue(ML). Results are aggregated according to the trees, the datasets and the evidence. Vertical bars represent the sampled standard deviation. Best viewed in colors.

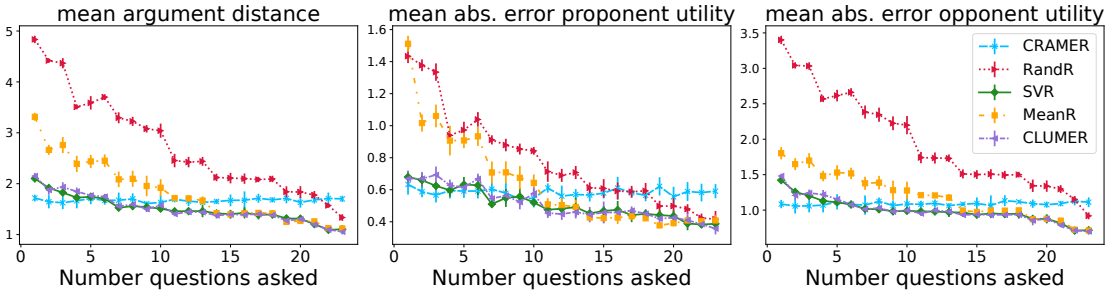

Figure 3: Performance of SimDialogue(ML) for the red meat case study. Results are aggregated according to the k folds. Vertical bars represent the standard deviation. Best viewed in colors.

($U^p(n) = 4$), thinking of alternatives ($U^p(n) = 3$) and reducing red meat consumption ($U^p(n) = 2$).

Regarding the dataset, we created 6 user profiles with their demographics (sex, age, school degree, level of meat consumption and level of physical activity) and utility values according to the profile, e.g, young student or woman in career. Each profile is then transformed into a numeric vector. This dataset of 6 examples is expanded to 2000 examples by repeating and shuffling the 6 example vectors. Then a matrix of gaussian random noise ($\mu = 0$, $\sigma^2 = 1$) is added to the expanded dataset in order to have variability. DT topics and dataset profiles are in Appendix C.

The evaluation has been performed as in the previous section. As there is only one DT and dataset, we present the non-averaged metrics in Figure 3. This realistic case study shows similar findings to the simulations. However, differently from the simulations, here demographic information is available as evidence for all the methods. The Mad is good as with only the demographic information we have a Mad of around 1.7 for CRAMER and around 2.1 for both SVR and CLUMER. Here, the simple clustering and centroids approach of CLUMER is sufficient to achieve the same results of SVR. CRAMER presents stable performance. As the number of questions increases both CLUMER and SVR require 5 questions (out of 22 possible questions, i.e., 23% more of questions) to get a Mad close to the one of CRAMER. The mean absolute error of the proponent utility is much lower compared to the opponent one. This is due to a bias in the dataset as most of the final true nodes have 8 as proponent utility value. This bias does not regard the opponent utility, see the statistic in Appendix D. A qualitative comparison of the arguments returned by CLUMER and SVR for each profile is in Appendix E.

Conclusion

Bi-party Decision Theory is a promising approach for an APS to choose the best arguments to persuade a user. These strategies are based on utility functions. However, no methods deal with the construction of such functions for a new user in an efficient way. In this paper, we addressed this problem by developing two ML models (EAI and EDS) to learn these utility functions from datasets. The evaluation with simulations and with a realistic case study show that both EAI and EDS are able to learn utility functions of subpopulation of users with comparable performance. However, EDS is more efficient as it requires less user information.

As future work, we want to test these methods with real users in a living lab. In addition, a combination of EAI with EDS could improve the performance requiring a lower number of questions to ask about. The use of neural networks for learning the utilities will be studied also in an interactive APS that it adapts the utilities as it interacts with the user (Dragone, Teso, and Passerini 2017).

Acknowledgements

The research of ST was partially supported by TAILOR, an EU Horizon 2020 project, GA No 952215.

References

Abete, I.; Romaguera, D.; Vieira, A. R.; de Munain, A. L.; and Norat, T. 2014. Association between total, processed, red and white meat consumption and all-cause, CVD and IHD mortality: a meta-analysis of cohort studies. *British Journal of Nutrition*, 112(5): 762–775.

Abidi, S.; Vallis, M.; Abidi, S. S. R.; Piccinini-Vallis, H.; and Imran, S. A. 2014. D-WISE: Diabetes Web-Centric Information and Support Environment: conceptual specification and proposed evaluation. *Canadian Journal of Diabetes*, 38(3): 205–211.

Alahmari, S. 2020. *Reinforcement Learning for Argumentation*. Ph.D. thesis, University of York, York, UK.

Atkinson, K.; Bench-Capon, P.; and Bench-Capon, T. J. M. 2012. Efficiency in Persuasion Dialogues. In *ICAART (2)*, 23–32. SciTePress.

Black, E.; Coles, A. J.; and Bernardini, S. 2014. Automated Planning of Simple Persuasion Dialogues. In *CLIMA*, volume 8624 of *Lecture Notes in Computer Science*, 87–104.

Black, E.; Coles, A. J.; and Hampson, C. 2017. Planning for Persuasion. In *AAMAS*, 933–942. ACM.

Chalaguine, L.; and Hunter, A. 2020. A Persuasive Chatbot Using a Crowd-Sourced Argument Graph and Concerns. In *COMMA*, volume 326 of *FAIA*, 9–20. IOS Press.

Dragone, P.; Teso, S.; and Passerini, A. 2017. Constructive Preference Elicitation. *Frontiers Robotics AI*, 4: 71.

Fan, X.; and Toni, F. 2012. Mechanism Design for Argumentation-based Persuasion. In *COMMA*, volume 245 of *Frontiers in Artificial Intelligence and Applications*, 322–333. IOS Press.

Hadjinikolis, C.; Siantos, Y.; Modgil, S.; Black, E.; and McBurney, P. 2013. Opponent Modelling in Persuasion Dialogues. In *IJCAI*, 164–170. IJCAI/AAAI.

Hadoux, E.; Beynier, A.; Maudet, N.; Weng, P.; and Hunter, A. 2015. Optimization of Probabilistic Argumentation with Markov Decision Models. In *IJCAI*, 2004–2010.

Hadoux, E.; and Hunter, A. 2019. Comfort or Safety? Gathering and Using the Concerns of a Participant for Better Persuasion. *Argument & Computation*, 10: 113–147.

Hadoux, E.; Hunter, A.; and Polberg, S. 2018. Biparty Decision Theory for Dialogical Argumentation. In *COMMA*, volume 305 of *Frontiers in Artificial Intelligence and Applications*, 233–240. IOS Press.

Hunter, A.; Chalaguine, L.; Czernuszenko, T.; Hadoux, E.; and Polberg, S. 2019. Towards Computational Persuasion via Natural Language Argumentation Dialogues. In *KI*, volume 11793 of *LNCS*, 18–33. Springer.

Hunter, A.; and Thimm, M. 2016. Optimization of Dialectical Outcomes in Dialogical Argumentation. *International Journal of Approximate Reasoning*, 78: 73–102.

Jannach, D.; Manzoor, A.; Cai, W.; and Chen, L. 2021. A Survey on Conversational Recommender Systems. *ACM Comput. Surv.*, 54(5): 105:1–105:36.

Katsumi, H.; Hiraoka, T.; Yoshino, K.; Yamamoto, K.; Motoura, S.; Sadamasa, K.; and Nakamura, S. 2018. Optimization of Information-Seeking Dialogue Strategy for Argumentation-Based Dialogue System. In *Proceedings of DEEP-DIAL@AAAI'19*, volume abs/1811.10728. ArXiv.

Lenert, L.; Morss, S.; Goldstein, M. K.; Bergen, M.; Faustman, W.; and Garber, A. M. 1997. Measurement of the validity of utility elicitations performed by computerized interview. *Medical Care*, 35(9): 915–920.

Lenert, L. A.; Sherbourne, C. D.; and Reyna, V. 2001. Utility elicitation using single-item questions compared with a computerized interview. *Medical Decision Making*, 21(2): 97–104.

Monteserin, A.; and Amandi, A. 2013. A reinforcement learning approach to improve the argument selection effectiveness in argumentation-based negotiation. *Expert Systems with Applications*, 40: 2182–2188.

Osborne, M.; and Rubinstein, A. 1994. *A Course in Game Theory*. MIT Press.

Peterson, M. 2009. *An Introduction to Decision Theory*. Cambridge University Press.

Rach, N.; Minker, W.; and Ultes, S. 2018. Markov Games for Persuasive Dialogue. In *COMMA*, volume 305 of *Frontiers in Artificial Intelligence and Applications*, 213–220.

Rahwan, I.; and Larson, K. 2008. Pareto Optimality in Abstract Argumentation. In *AAAI*, 150–155. AAAI Press.

Rahwan, I.; Larson, K.; and Tohmé, F. A. 2009. A Characterisation of Strategy-Proofness for Grounded Argumentation Semantics. In *IJCAI*, 251–256.

Ricci, F.; Rokach, L.; and Shapira, B., eds. 2015. *Recommender Systems Handbook*. Springer. ISBN 978-1-4899-7636-9.

Rienstra, T.; Thimm, M.; and Oren, N. 2013. Opponent Models with Uncertainty for Strategic Argumentation. In *IJCAI*, 332–338. IJCAI/AAAI.

Riveret, R.; Gao, Y.; Governatori, G.; Rotolo, A.; Pitt, J.; and Sartor, G. 2019. A probabilistic argumentation framework for reinforcement learning agents - Towards a mentalistic approach to agent profiles. *Autonomous Agents and Multi-Agent Systems*, 33(1-2): 216–274.

Rosenfeld, A.; and Kraus, S. 2016a. Providing Arguments in Discussions on the Basis of the Prediction of Human Argumentative Behavior. *ACM Transactions on Interactive Intelligent Systems*, 6(4): 30:1–30:33.

Rosenfeld, A.; and Kraus, S. 2016b. Strategical Argumentative Agent for Human Persuasion. In *ECAI*, volume 285 of *Frontiers in Artificial Intelligence and Applications*, 320–328. IOS Press.

Stankevitz, K.; Dement, J.; Schoenfisch, A.; Joyner, J.; Clancy, S. M.; Stroo, M.; and Østbye, T. 2017. Perceived barriers to healthy eating and physical activity among participants in a workplace obesity intervention. *Journal of Occupational and Environmental Medicine*, 59(8): 746–751.

Turk, P.; and Borkowski, J. J. 2005. A review of adaptive cluster sampling: 1990–2003. *Environmental and Ecological Statistics*, 12(1): 55–94.