ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/335290953

Evaluating Hierarchies through A Partially Observable Markov Decision Processes Methodology

Preprint · August 2019

CITATIONS
0

READS
14

4 authors, including:

Weipeng Huang
University College Dublin
5 PUBLICATIONS **3** CITATIONS
SEE PROFILE

Guangyuan Piao
Bell Labs Dublin Ireland
28 PUBLICATIONS **144** CITATIONS
SEE PROFILE

Raul Moreno Salinas
University College Dublin
16 PUBLICATIONS **160** CITATIONS
SEE PROFILE

Some of the authors of this publication are also working on these related projects:

Turning coffee into research papers View project

All content following this page was uploaded by Weipeng Huang on 31 August 2019.

The user has requested enhancement of the downloaded file.

arXiv:1908.07031v1 [stat.ML] 19 Aug 2019

Evaluating Hierarchies through A Partially Observable Markov Decision Processes Methodology

Weipeng Huang[a,*], Guangyuan Piao[b], Raul Moreno[a], Neil Hurley[a]

[a] *Insight Centre for Data Analytics, School of Computer Science, O'Brien Building for Science, Belfield, Dublin 4, Ireland*
[b] *Insight Centre for Data Analytics, Data Science Institute, IDA Business Park, Galway, Ireland*

Abstract

Hierarchical clustering has been shown to be valuable in many scenarios, e.g. catalogues, biology research, image processing, and so on. Despite its usefulness to many situations, there is no agreed methodology on how to properly evaluate the hierarchies produced from different techniques, particularly in the case where ground-truth labels are unavailable. This motivates us to propose a framework for assessing the quality of hierarchical clustering allocations which covers the case of no ground-truth information. Such a quality measurement is useful, for example, to assess the hierarchical structures used by online retailer websites to display their product catalogues. Differently to all the previous measures and metrics, our framework tackles the evaluation from a decision theoretic perspective. We model the process as a bot searching stochastically for items in the hierarchy and establish a measure representing the degree to which the hierarchy supports this search. We employ the concept of Partially Observable Markov Decision Processes (POMDP) to model the uncertainty, the decision making, and the cognitive return for searchers in such a scenario. In this paper, we fully discuss the modeling details and demonstrate its application on some datasets.

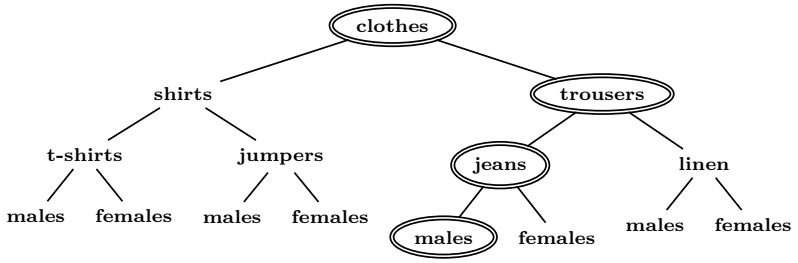
*Corresponding author
Email addresses: weipeng.huang@insight-centre.org (Weipeng Huang), guangyuan.piao@insight-centre.org (Guangyuan Piao), raul.sallinas@insight-centre.org (Raul Moreno), neil.hurley@insight-centre.org (Neil Hurley)

Keywords: Hierarchical Cluster Evaluation, Decision under Uncertainty, Partially Observable Markov Decision Process

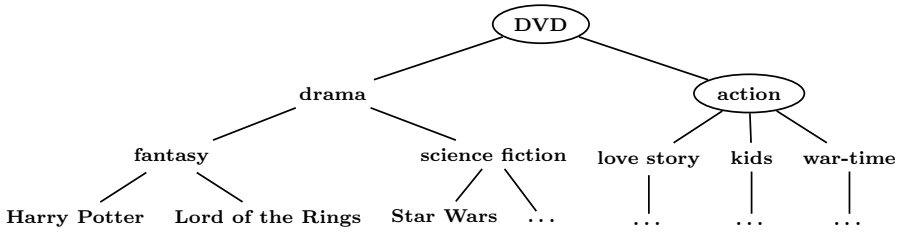
1. Introduction

Hierarchical clustering analysis has been applied in many E-commerce and scientific applications. The process generates a collection of nested clusters that group the data in a connected structure (Balcan et al., 2014), forming a hierarchy. The hierarchy is represented by a tree data structure where each node contains a number of data items. Such a structured organization of the items is useful for tasks such as efficient search. Searchers can navigate through the catalogue of items by choosing a path through the hierarchy and can thus avoid the cost of a linear search through the entire catalogue of items. If the hierarchy is well organized into coherent clusters, then finding the correct path to the required item is easy. Clearly, there are multiple ways of clustering the same items and organizing their hierarchies. However, evaluating the quality of a hierarchy still needs more substantial study, especially for data that lacks a ground-truth hierarchy. The evaluation of hierarchical structures is the focus of this paper.

For example, consider on-line retailers and the customers who access them. When looking for specific items, customers are in essence navigating the hierarchies hosted by the website. Different clusters and hierarchies will probably provide divergent levels of user experience with respect to searching and navigating, even for the same users. For example, imagine that a female jeans is mistakenly placed in a branch of a certain hierarchy, labeled "clothes → trousers → jeans → males" . The user will expect that the jeans are contained along the branch "clothes → trousers → jeans → females", and thus will surely fail to retrieve the required item in that terrible hierarchy (e.g. Figure 1a). As another example, Figure 1b shows that a searcher may be confused and possibly go to the wrong route when trying to find the Harry Potter DVD. Is it more so an action movie than a drama movie? In this second example, it may well be the



(a) An example which illustrates a terrible hierarchy that places a female jeans in the branch of double-bordered ellipses



(b) An example which illustrates a hierarchy that confuses the searchers given that the target is a Harry Porter DVD

Figure 1: Simple hierarchy examples

case that each cluster contains a coherent set of items, but their hierarchical organization makes it difficult for a searcher to choose the correct path. Such observations motivate an evaluation function that accounts for the structure as much as the item to cluster assignments.

In the scenario that we consider here, all items stored in the sub-tree rooted at a node in the hierarchy are available for consideration by a searcher who stops at that node. For instance, a searcher stopping at the *drama* node could consider in turn all the DVDs, "Harry Potter", "Lord of the Rings", "Star Wars", and so on. The further the searcher descends in the hierarchy, the fewer items that need to be considered once the searcher stops to search, thus increasing search efficiency, provided that a correct path in the hierarchy is taken.

Why another evaluation function? There are many evaluation functions already proposed to measure the quality of flat clusterings, with or without ground-truth data. Any of these measures can be applied to a hierarchical clus-

tering, once an appropriate cut of the hierarchy is chosen. Such measures do give feedback on the coherence of the clusters and/or their agreement with a ground truth. However, there is relatively little research that has proposed measures that take the structural organization of the hierarchy into account and we believe that this is critical in order to properly evaluate hierarchies in the context that we have in mind. In particular, we see two specific uses for the methodology that we propose in this paper. Firstly, the measure that we propose can be used as a standalone quality measure for evaluating hierarchies generated from different algorithms. Secondly, as many hierarchical clustering algorithms require the setting of hyper-parameters that have a significant influence on the structures found by the algorithm, we offer the proposed objective as a means for guiding hyper-parameter tuning for hierarchical clustering algorithms. For instance, this measure (or an approximation) can act as an objective function or a regularization factor for tuning the hyper-parameters for the hierarchical clustering algorithms. As demonstrated in (Kobren et al., 2017), many hierarchical clustering algorithms preserve hyper-parameters and might possibly be further improved with our function.

1.1. Problem Statement

As regards the motivation, the problem is simply stated: assume we host a catalogue of items on a website. Having run many hierarchical clustering algorithms in order to find a good organization of the catalogue, we would like to find out which hierarchy is the best one to use. Thus, it is necessary to compare the hierarchies with respect to certain properties, in particular when the data is large and manual checking is implausible. Specifically, we seek a manner to map each hierarchy to a real-valued quality score, with which the hierarchies can be ordered in terms of their relative merits with respect to these properties.

Technically, we detail the problem as follows. Consider a dataset consisting of a collection of items $\mathcal{D} = \{d_n\}_{n=1}^{N}$, which may correspond to documents or products in a catalogue. A hierarchical clustering is a set of nested clusters of

the dataset arranged in a tree $\mathcal{T}$. Each node of the tree has associated with it a sub-set or cluster of the collection, and the child nodes of any node are associated with a partition of the parent's cluster. The root node is associated with the entire collection, and if an item belongs to any particular node in $\mathcal{T}$, it also belongs to all of its ancestor nodes. Assume that the hierarchy consists of a set of clusters C. For $c, c' \in C$, we write $(c, c') \in \mathcal{T}$ to denote a directed edge in $\mathcal{T}$, such that c is the parent of c'. Furthermore, we define that $\mathcal{C}(c)$ returns the set of children of c. The goal is to seek a measure mapping $\mathcal{T}$ to a real value, which reflects the quality of the hierarchical arrangement, from the point of view of supporting efficient search for a *target* item in the collection. In our scenario, a searcher starts from the root and traverses the hierarchy until stopping and searching for the target through all the items stored at that node. The further the searcher descends the hierarchy, the fewer items need to be searched at the stopping point. However, during the navigation, the searcher may be unsure about which child node contains the target and therefore may encounter difficulties in choosing the right node to which to navigate. A good hierarchy should ease the navigation task and reduce the search effort.

1.2. Summary of Modeling

This work aims to arrive at a quantitative quality measure for a given hierarchical arrangement of a catalogue of items, where in a typical use-case, the items are products in a large catalogue offered by an on-line retailer. Our scenario is that of a search bot seeking a specific target item and so we develop the measure by modeling a simplified search process but one which we believe is sufficient to capture the important features of the hierarchy that determine its suitability to support efficient search.

In practice, real searches may follow arbitrarily complex search patterns. Users may start a search, abandon it, start again but remembering what they observed in their previous search, backtrack along a search path and so on. It is not our goal to provide a model of all possible search strategies but rather to measure the hierarchy's ability to support the decisions that typically must be

made during search. In this regard, a hierarchical arrangement that allows fast filtering of the catalogue is preferred i.e. one whose depth is logarithmic rather than linear in the catalogue size. But this is not enough. At decision points in which the search space is reduced by choosing a particular path, with high probability, it should be possible to make a good decision that focuses attention to the part of the hierarchy that contains the target item.

It is intuitive to model this process of decision making under uncertainty as a Markov Decision Process (MDP). MDPs are concerned with the problem of determining a decision policy that optimizes the cumulative reward obtained when applying this policy over some time horizon. In an initial model, we proposed such an MDP (Moreno et al., 2017) to tackle the problem of evaluating hierarchies. In our model, the quality of the hierarchy corresponds to the expected cumulative reward that a search bot will obtain when searching for target items, using a particular search policy, where the reward is a function of whether and how efficiently the target is found. Noteworthy in this initial effort is that our quality measure depended not only on the hierarchy itself, but also on the policy used to determine the action at each decision point.

The work in this paper proposes a novel framework by extending the original MDP model by proposing a reasonable search policy, which we posit provides a good model for how a real searcher may be expected to behave. This removes the policy as an input to the proposed quality measure. Key to our new framework, is that we explicitly model the searcher's lack of knowledge about the environment and this leads us to extend from an MDP framework, to a Partially Observed MDP (POMDP) framework. The POMDP allows us to directly model that a searcher is not able to directly observe the correct path. In a POMDP, the searcher's uncertainty about the state space is modeled as a belief function and the model specifies how belief is updated when observations are made. By tracking how the searcher's observations lead to an update in her belief, we are in a position to apply the machinery of POMDPs to address the question of what policy a searcher should use that best exploits her current beliefs. In our opinion, this extension to the initial model brings us closer to a

pure measure of a hierarchy's innate quality, rather than a measure of an individual searcher's ability to navigate the hierarchy. Finally, we coin the measure Hierarchy Quality for Search (HQS).

1.3. Organization of the Paper

In the rest of the paper, we show the related works in Section 2. After this, we discuss the model in Sections 3 and 4 and the policy choice in Section 5. Then, a mini section Section 6 summarizes the design: the requirements, the challenges and the solutions, which have been addressed individually in the earlier sections. Section 7 carries out a number of case studies to analyze the measure. Finally Section 8 concludes the paper.

2. Related Work

Examining the state-of-the-art, we can distinguish external and internal measures for evaluating hierarchical clustering. External measures evaluate the hierarchy in comparison to a ground-truth hierarchy and include measures such as Shannon's entropy and F-score (Steinbach et al., 2002). Internal measures compare clusters without using a ground-truth. This class includes the modified Hubert Gamma statistic, Calinski-Harabasz index, Dunn's index, and Silhouette index, etc. (Liu et al., 2013). These measures are originally designed for flat clustering and lack the principal for evaluating the hierarchical arrangement (Johnson et al., 2013).

Recently Johnson et al. (2013) extended the Rand index to evaluate the hierarchy structure while their approach relies on the availability of ground-truth clusters. Before this, Cigarran et al. (2005) proposed a goal-oriented measure considering the content of the cluster, the hierarchical arrangement and the navigation cost. We observe that hierarchical clustering measures have hardly addressed the need to understand the convenience for the search and navigation efficiency provided by a hierarchy, accounting for the cognitive cost of choosing a correct path at each branch of the hierarchy. In our previous

work (Moreno et al., 2017), we introduce the concept of using MDPs to model the evaluation. This paper extends from that work in two main directions. The first is that we use a POMDP to model the searcher's lack of the environment knowledge which is closer to a real-world scenario. Secondly, we specify a specific stochastic search policy to model typical search behavior, while the existing work considered the policy as an input parameter to the measure.

There exist plenty of high quality works addressing the issue of acting optimally in POMDPs, such as (Cassandra et al., 1994; Boutilier and Poole, 1996; Meuleau et al., 1999). Also, many solutions to finding near-optimal policies exist for reducing the computational cost without heavily violating the pay-off, e.g., (Kearns et al., 2002; Brafman and Tennenholtz, 2002; Kearns and Singh, 2002; Fard and Pineau, 2011; Gosavi and Purohit, 2011). With regard to online policies in POMDPs, Ross et al. (2008b) elaborates on three main approaches: heuristic search, Monte Carlo sampling, and branch-and-bound pruning. Later, Somani et al. (2013) proposed an improved on-line policy that decreases the size of policy search trees via regularization. These methods allow more a longer look-ahead than is feasible in our problem. There is a similar optimal stopping problem to the one we introduce here, named "Asset selling", which was solved by Sakaguchi (1961). However, that problem assumes that the decision maker knows the distribution of the sequentially incoming offers and the cost is constant; thus, one can use calculus to compute the optimal expected reward. This also does not perfectly fit to our problem.

We realize that Fern et al. (2007, 2014) employed POMDP to help people with dementia to best achieve their goals. These works inspire us to select a myopic policy in solving our model, as a myopic configuration is shown sufficient in real-world applications. Our work is similar to this literature in terms of using POMDP to solve the problems in a specific domain although the target tasks are distinct.

3. Scenario and Model

In this section, we describe the scenario being considered. A rational automatic bot needs to search for a certain item contained in a hierarchy. At each decision point, the bot is allowed to perform one move *down* the hierarchy or look up the target at the current location. Before looking into a certain cluster, the bot is unsure whether or not the target is contained in this cluster. No doubt, after the search and navigation, the bot will receive a positive reward if it finds the target or a penalty otherwise. In this search scenario, we specifically disallow backtracking. Our justification is that, firstly, if backtracking were enabled, the bot might face the dilemma of spending a long time seeking just one item in a large, poorly structured hierarchy, which would make our evaluation really inefficient or even intractable; and secondly, the search strategy need only be sufficiently complex to allow the quality of the hierarchy to be assessed and we argue that following a single path, with possible re-starting in the case that the target is not found, is sufficient to assess a hierarchy's ability to support a bot's correct decision making.

The bot must make decisions based on its imperfect knowledge of the environment. We imagine another role, which we call the *oracle*. The oracle knows the entire hierarchy and, in particular, knows the exact location of the target. The oracle can compute the true long term reward obtained by the bot in carrying out its search. We call this long term reward the *oracle value*. The bot, on the other hand, makes its decisions based on the expected reward over its beliefs. The relationship between the bot and the oracle is analogous to that between the student and the teacher in an examination scenario. A student obtains a certain degree of knowledge and using it, tries her best to answer the questions, while a teacher knows all the answers to the questions and will mark the paper objectively. It means that the value that the bot seeks to optimize in the POMDP framework, is not necessarily identical to the final reward as determined by the oracle. A student who applies an optimal policy to fully exploit her prior knowledge may still receive a terrible outcome if her knowledge

is imperfect.

This point illustrates that prior knowledge plays a vital role in the student's performance in her examination and similarly in the success of the bot's search strategy. In our model we will propose a *guidance function*—which we call the η function—that compresses the information about the hierarchy that will be generated and conveyed to the bot. Such a function serves as the domain knowledge to the bot which will assist its decision making. The knowledge encapsulated in η will be produced by the attributes of the hierarchy and it makes η a factor for assessing the quality of the hierarchy. For a bad hierarchy, the η function might point the bot to a rather wrong position. Returning to our first example where a female jeans has been mistakenly placed in the branch "dress $\rightarrow$ trousers $\rightarrow$ jeans $\rightarrow$ males" in a certain hierarchy (see Figure 1a), even though the bot may choose a path of "dress $\rightarrow$ trousers $\rightarrow$ jeans $\rightarrow$ females" that optimizes its expected reward over its belief that this is the correct path, it will definitely fail to retrieve the item in that terrible hierarchy and its true reward, as computed by the oracle, will be minimal.

4. POMDP Specification

A conventional POMDP is solely an MDP with additional observations, belief state and belief estimator (Cassandra et al., 1994; Kaelbling et al., 1998). Some classes of POMDPs also involve an internal state about intents, feedback, or memory of actions etc; but such complexity is less commonly seen (Aberdeen and Baxter, 2002; Thomson and Young, 2010; Lam and Sastry, 2014). Although Kaelbling et al. (1998) argues that a belief state can be anything, most research works adopt the setting that it is a probability distribution over the states. Fixing the search target of the bot as item x, we model the search of the hierarchy as the POMDP $\mathcal{P}_x = \langle S, A, T, R, Z, O, \gamma \rangle$, such that

- S is the finite state space.
- A is the set of possible actions.

- $T : S \times A \times S \mapsto [0, 1]$ is the transition function where $T(s, a, s')$, also written as $p(s' \mid s, a)$, represents the probability of moving to state s' from s using action a.
- $R : S \times A \times S \mapsto \mathbb{R}$ assigns the expected immediate reward, $R(s, a, s')$, to the resulting state s' after the bot selecting action a in state s.
- O is the observation set.
- $Z : S \times A \times O \mapsto [0, 1]$ is the probability function for the observations. We have $Z(s, a, o) = p(o \mid a, s)$ which reads as the probability of observing $o \in O$ when reaching the state s through action a.
- γ is the discount factor.

Next, we discuss the specification of the model.

4.1. States

States in the POMDP represent the status of the search at a particular point during the search process after an action is taken. The target x is fixed at some leaf of the hierarchy and will therefore be found if a search is carried out at any internal node along the single fixed path from the root to that leaf. At any particular point, the bot is exploring a certain node, c, in the hierarchy. Hence, our model defines that each state is a joint event consisting of 1) the physical location c of the bot, and 2) a Boolean variable indicating whether the bot is still in the right path:

$$S = \bigcup_{c \in \mathcal{T}} \{\langle c, 0\rangle, \langle c, 1\rangle\} \cup \{\langle \emptyset, 0\rangle, \langle \emptyset, 1\rangle\}, \tag{1}$$

where $\emptyset$ is the terminal point which will be reached after the bot chooses to stop and search. Accordingly, $\langle \emptyset, 1\rangle$ is the state representing that the bot stops and the stopping node contains the item, likewise $\langle \emptyset, 0\rangle$ represents stopping but not finding the item.

4.2. Actions

Before introducing the action set, it is important for us to discuss the concept *guidance function*.

4.2.1. guidance function

A guidance function $\eta : 2^{\mathcal{D}} \times 2^{\mathcal{D}} \times \mathcal{D} \to [0, 1]$ provides the search bot with evidence as to the correct path on which to search for an item $x \in \mathcal{D}$. Specifically, $\eta(c, c', x)$ represents the bot's belief that the target x is contained in a cluster c', given that it is in cluster c, where $(c, c') \in \mathcal{T}$. We have that

$$\forall c' \notin \mathcal{C}(c) : \ \eta(c, c', x) = 0 \qquad \text{and} \qquad \sum_{c' \in \mathcal{C}(c)} \eta(c, c', x) = 1 \, .$$

From now on, we omit x in the η function, since the discussions about η will always concentrate on one specific x.

The purpose of the η function is to summarize the information available in the hierarchy which will guide the bot's behavior. It is represented as a discrete probability density function that determines how probable it is that a certain child may be the correct node given its parent node is on the right path. The function can be viewed as the prior domain knowledge that the bot learns about the hierarchy. We design it on top of the similarities between the target item and the different clusters.

Specifically, we write $\eta : 2^{\mathcal{D}} \times 2^{\mathcal{D}} \times \mathcal{D} \mapsto [0, 1]$ for any pair $(c, c') \in \mathcal{T}$ such that

$$\eta(c, c', x) = p(x \in c' \mid x \in c) \triangleq \frac{e^{\mathcal{S}(x, c')/\delta}}{\sum_{c'' \in \mathcal{C}(c)} e^{\mathcal{S}(x, c'')/\delta}} \, . \tag{2}$$

Here, $\mathcal{S}(x, c)$ is the similarity function between the item x and the cluster c. This can be any kind of similarity that is used in measuring clustering results. Leaving the choice of similarity function open adds flexibility to the measure by allowing the users to customize the comparison for various hierarchies of specific datasets. For instance, a simple choice is the inverse or the negative of Euclidean distance, assuming items can be mapped to points in $\mathbb{R}^n$; Bayesian methods could use a similarity based on a distribution density; and if items

are represented as Term Frequency and Inverse Document Frequency (TF-IDF) vectors of textual data, then cosine similarity might be appropriate, and so on. As we can see, the η function returns a higher probability score for the node c' that is "closest" to x among the siblings. The parameter δ is a "temperature" parameter in this well known Boltzmann function. We adopt such a function in order to handle cases where none of the children is similar to the target. For instance, suppose there are two child clusters A and B of parent C such that $\mathcal{S}(x, A) < s_\epsilon$ and $\mathcal{S}(x, B) < s_\epsilon$ for some similarity value $s_\epsilon \approx 0$, and yet $\mathcal{S}(x, A) \ll \mathcal{S}(x, B)$. A simple normalisation of similarity values would result in $\eta(C, A) \approx 1$, while in reality, for two such terrible clusters, it would be better to have $\eta(C, A) \approx \eta(C, B)$. Deep in the hierarchy, the bot should only choose to descend to a cluster when the evidence that it contains the target is strong. A δ parameter that increases with the depth of the tree ensures that the similarity values between two clusters must be more and more distinct deeper in the tree before one cluster is preferred over another. Therefore, δ can be defined as a function over the depth of the hierarchy, such that $\delta_t \triangleq \delta_{base}\nu^t$ where $\nu \in [1, \infty)$. Setting $\nu = 1$ makes δ time-invariant.

4.2.2. Action Set

It is well known that optimal policies for MDPs (and consequently POMDPs) are deterministic. Yet in reality all searchers do not behave in the same way, even given the same feedback. As our measure should represent the quality of the hierarchy for a typical searcher, we capture the many possible search behaviors by modeling that the bot searches stochastically. Nevertheless, our measure should focus on rational search behavior, rather than on fully random search. The proposed framework allows us to capture this rationality by exposing part of the search behavior to rational decision making. Hence, we impose randomness in the manner in which child nodes are selected by a searcher and expose only the decision of whether to descend through the hierarchy or not—i.e. whether or not to exploit the structure of the hierarchy—to the bot's decision-making logic. We design the bot to move randomly according to the guidance function

$\eta(\cdot)$. Specifically, given a set of children $\mathcal{C}(c)$ of parent c, the bot moves to a child c' with probability $\eta(c, c')$. Thus, when the bot decides to descend the hierarchy by another step, it moves stochastically according to the guidance function.

With the above rationale, the action set contains only two actions: descend $\mathrm{a_d}$ and stay $\mathrm{a_s}$, corresponding to the choices of descending the hierarchy to another level, or stopping and searching for the target at the current node. The bot can choose to stop and search at any decision point and this is the only possible action for the states related to the leaf nodes in the hierarchy. When $\mathrm{a_s}$ is performed, the bot reaches the terminal point $\emptyset$.

4.3. Transitions

If the bot chooses to stay and search, the transition is fixed and the bot can fully observe the outcome. That is, given that it has reached $\emptyset$, it knows whether it is in $\langle \emptyset, 0 \rangle$ or $\langle \emptyset, 1 \rangle$. For the descent action, the bot estimates the transition i.e. it computes $\hat{T}$ such that $\hat{T}(s, a, s') = \hat{p}(s' \mid s, a)$. The transition $\hat{T}$ is controlled by an η function that determines the selection of which child node to move to. Note that in traditional MDP and POMDP problems, the unknown transitions are handled by the reinforcement learning (RL) methodology, such that the bot can explore by choosing certain actions and can approximate the transition probabilities by the ratio of the number of ending states over all the attempts (Duff, 2002; Ross et al., 2008a; Ng et al., 2012). Our bot lacks such machinery, and so sticks to its prior domain knowledge. As a consequence, the domain knowledge is the main force that drives the bot to succeed or fail in its task.

Let g_c be the Boolean variable associated with a node c encapsulated in a state s. We decompose the transition probability of applying $\mathrm{a_d}$ for the various cases. For $c' \in \mathcal{C}(c)$, $s = \langle c, g_c \rangle$ and $s' = \langle c', g_{c'} \rangle$,

$$\begin{aligned} \hat{p}(s' \mid s, \mathrm{a_d}) &= p\left(\langle c', g_{c'} \rangle \mid \langle c, g_c \rangle, \mathrm{a_d}\right) \\ &= p\left(c' \mid c, \mathrm{a_d}\right) \hat{p}\left(g_{c'} \mid g_c\right) = \eta(c, c') \hat{p}\left(g_{c'} \mid g_c\right) \end{aligned} \quad (3)$$

where the last step follows from the stochastic descent process described in the previous section. On the other hand, the η function is also used for the estimate $\hat{p}(g_{c'} \mid g_c)$ and has the following formulation:

$$\begin{aligned} \hat{p}(g_{c'} = 1 \mid g_c = 0) &= 0 \quad \hat{p}(g_{c'} = 1 \mid g_c = 1) = \eta(c, c') \\ \hat{p}(g_{c'} = 0 \mid g_c = 0) &= 1 \quad \hat{p}(g_{c'} = 0 \mid g_c = 1) = 1 - \eta(c, c') \end{aligned} \tag{4}$$

Combining, the estimated transition probabilities are only positive when the next state encapsulates a child node in the hierarchy and, furthermore, indicates that the path is correct, only if the current state is already on the correct path. Overall, for $\mathrm{a_d}$, we get, for all $c \in \mathcal{T}$, $c' \in \mathcal{C}(c)$:

$$\begin{aligned} p(\langle c', 1\rangle \mid \langle c, 1\rangle, \mathrm{a_d}) &= \eta(c, c')^2 \\ p(\langle c', 0\rangle \mid \langle c, 1\rangle, \mathrm{a_d}) &= \eta(c, c')(1 - \eta(c, c')) \\ p(\langle c', 1\rangle \mid \langle c, 0\rangle, \mathrm{a_d}) &= 0 \\ p(\langle c', 0\rangle \mid \langle c, 0\rangle, \mathrm{a_d}) &= \eta(c, c')\,. \end{aligned}$$

4.4. Rewards

In our POMDP, only when the bot stops and searches is non-zero reward obtained; navigating earns 0 reward. Moreover, stopping at and searching in a node with fewer items gives the bot a higher cognitive pay-off which provides the motivation for the bot to traverse down the tree. In particular, we have

$$R(s, a, s') = \begin{cases} 0 & a = \mathrm{a_d} \\ r(c) & s' = \langle \emptyset, 1\rangle \\ -1 & s' = \langle \emptyset, 0\rangle \end{cases}$$

where c are the locations encapsulated in s. We assign -1 to searching at a wrong node. One can customize $r(\cdot)$ as long as it is monotonically decreasing with the size of the input cluster to be searched. In particular, we use the following function to approximate the ease of searching within a collection of items:

$$r(c) = 1 - \left(e^{\frac{|c|}{N}} - 1\right)/(e - 1)$$

where $|c|$ is the number of items at c and $N = |\mathcal{D}|$ is the size of the dataset.

4.5. Observations

The observations after exploring (i.e. after choosing $\mathrm{a_d}$) are the locations of the bot in the hierarchy, the children, some abstract summaries from the children, and so on. Considering that this information is fixed and unique for each location that the bot arrives at, we pack all this information into one indexed observation. This observation directly affects our belief update function. After choosing $\mathrm{a_s}$, the observations are whether or not the target is found in the current node.

4.6. Observation Probabilities

Recall that $Z(s, a, o) = p(o \mid s, a)$ and the observation is the location of the bot. It is trivial to see $p(o \mid \langle c', g_{c'} \rangle, \mathrm{a_d}) = \mathbb{1}(c' = o)$, i.e. when the bot navigates to a new node c', the observation tells it the exact location that it has arrived at.

4.7. Belief Update

As states are not directly observable, the bot maintains its belief state b as a probability function over the states, which it updates after every action. Belief updates can be represented using the *belief update function*, τ, where $b' = \tau(b, a, o)$, is the updated belief when observation o is made after action a is applied in belief state b. By Bayes' Theorem,

$$b'(s') = p(s' \mid a, o, b) = \frac{p(o \mid s', a) \sum_s p(s' \mid s, a) b(s)}{p(o \mid a, b)}\,, \tag{5}$$

where $p(o \mid a, b)$ can be regarded as the normalization factor (Kaelbling et al., 1998; Ross et al., 2008b).

In our scenario, the belief update after $\mathrm{a_s}$ is trivial. For $\mathrm{a_d}$, the probability $p(o \mid \mathrm{a_d}, b)$ can be expanded as

$$p(o \mid \mathrm{a_d}, b) = \sum_{s'} p(o \mid s', a) \sum_s p(s' \mid s, a) b(s)\,.$$

Assuming that the bot is in location c and decides to move, we consider the following equation for each possible child $c' \in \mathcal{C}(c)$, such that the new state is $s' = \langle c', 0\rangle$ or $s' = \langle c', 1\rangle$:

$$b'(s') \propto \mathbb{1}(c' = o) \sum_s p(s' \mid s, \mathrm{a_d}) b(s)\,.$$

Specifically,

$$\begin{aligned} b'(\langle c', 0\rangle) &\propto \mathbb{1}(c' = o) \sum_s p(\langle c', 0\rangle \mid s, \mathrm{a_d}) b(s) \\ &= p(\langle c', 0\rangle \mid \langle c, 1\rangle) b(\langle c, 1\rangle) + p(\langle c', 0\rangle \mid \langle c, 0\rangle) b(\langle c, 0\rangle) \\ &= \eta(c, c')[b(\langle c, 0\rangle) + (1 - \eta(c, c')) b(\langle c, 1\rangle)] \\ &= \eta(c, c')[1 - \eta(c, c') b(\langle c, 1\rangle)]. \end{aligned}$$

Similarly, we have

$$b'(\langle c', 1\rangle) \propto \eta(c, c')^2 b(\langle c, 1\rangle)\ ,$$

and, as discussed earlier, for other states, s'', not related to c', $b'(s'') = 0$. To compute the normalizing constant, when $a = \mathrm{a_d}$, we have that

$$p(o = c' \mid b, \mathrm{a_d}) = \eta(c, c')\left[1 - \eta(c, c') b(\langle c, 1\rangle) + \eta(c, c') b(\langle c, 1\rangle)\right] = \eta(c, c')\,,$$

which gives a final belief update rule of

$$b'(\langle c', 1\rangle) = \eta(c, c') b(\langle c, 1\rangle) \qquad b'(\langle c', 0\rangle) = 1 - \eta(c, c') b(\langle c, 1\rangle)\ . \tag{6}$$

Note that $p(o \mid b, \mathrm{a_s}) = 1$ for reaching the terminal state, such that $o = \emptyset$.

Let us denote by $s = s_t$ the state reached after t update steps, with similar subscripting for b and c. The root node is c_0. Throughout our analysis, we assume that the target x is certainly contained inside the hierarchy and hence, the bot always starts in state $\langle c_0, 1\rangle$. It follows that $b(\langle c_0, 1\rangle) = 1$.

Through induction, we arrive at a simple expression for the belief state when the search has reached node c_T at time stamp, $T > 0$.

$$b_T(\langle c_T, 1\rangle) = \prod_{t=1}^{T} \eta(c_{t-1}, c_t) \qquad b_T(\langle c_T, 0\rangle) = 1 - \prod_{t=1}^{T} \eta(c_{t-1}, c_t)$$

while $b_T(s) = 0$ for all other states s. In interpretation, once the bot physically reaches a node c, the belief state of $\langle c, 1\rangle$ is simply the probability of reaching this node from the root, and that of $\langle c, 0\rangle$ is the residual, $1-b_T(\langle c, 1\rangle)$. Moreover, since there is no backtracking, there is exactly one belief state associated with each node $c \in \mathcal{T}$, which, if $\{c_0, \ldots, c_T = c\}$ is the unique path from the root to c, is fully determined by the value $b_c \triangleq b_T(\langle c_T, 1\rangle)$ and we can simply write, for $c' \in \mathcal{C}(c)$, the belief update as $b_{c'} = \eta(c, c')b_c$.

4.8. Discount Factor

In our setting, the POMDP can only take a finite sets of steps, since the process must terminate after a leaf node is reached. Without an infinite time horizon, it is sufficient to set $\gamma = 1$.

4.9. Value Function

The most commonly used objective that drives policy selection in an MDP is the discounted long term reward, $\sum_{t=0} \gamma^t R_t$ where R_t is the reward obtained at time stamp t. An MDP typically seeks a policy to maximize the *value function*, V, corresponding to the expected discounted long term reward, where the expectation is over all states that may be explored by the sequence of actions specified by the policy. To solve a POMDP is to search for a policy π that maximizes the value function, $V^\pi(b)$, corresponding to the cumulative expected reward *over the beliefs*. In particular, let

$$R_B(b, a) = \sum_s b(s) \sum_{s'} p(s'|s, a)R(s, a, s')\,.$$

Then, the belief value function of a policy π is given by

$$V^\pi(b) = \sum_a \pi(b, a) \left[R_B(b, a) + \gamma \sum_s b(s) \sum_{s'} p(s' \mid s, a) \sum_o p(o \mid s', a)V^\pi(\tau(b, a, o)) \right]$$

where $\pi(b, a)$ is the probability that action a is selected in belief state b. The optimal value $V^*(b)$ with a corresponding deterministic policy to choose action a^* can be written as a Bellman equation:

$$V^*(b) = \max_{a \in A} Q(b, a) \qquad a^* = \operatorname*{argmax}_{a \in A} Q(b, a)$$

where

$$Q(b,a) = R_B(b,a) + \gamma \sum_s b(s) \sum_{s'} p(s' \mid s,a) \sum_o p(o \mid s',a) V^*(\tau(b,a,o))$$
$$= R_B(b,a) + \gamma \sum_o p(o \mid b,a) V^*(\tau(b,a,o)) \,.$$

This last expression shows that the POMDP may be interpreted as an MDP over belief states with $p(o \mid b,a)$ the transition probability for moving from belief state b to belief state $\tau(b,a,o)$.

Given our problem with its specific reward function, let T be the step at which a terminal state $\langle \emptyset, 0 \rangle$ or $\langle \emptyset, 1 \rangle$ is reached. The cumulative reward is given by

$$\sum_{t=0}^{T-1} R_t = \begin{cases} -1 & s_T = \langle \emptyset, 0 \rangle \\ r(c_{T-1}) & s_T = \langle \emptyset, 1 \rangle \end{cases} .$$

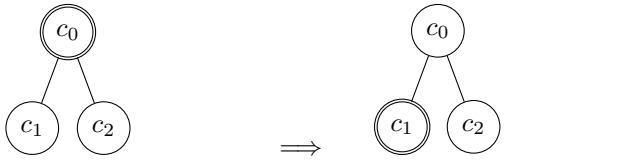
Given a policy π learned over the POMDP, it is natural to examine V of this policy in the underlying MDP (Singh et al., 1994), that arises from the POMDP when the states are fully known. For our model, this is exactly the oracle value we discussed earlier. In our case, V corresponds to knowing after each step whether the bot is still on the right path. Despite this, the policy is chosen to maximize the belief value $V^\pi(b)$.

4.10. Example of the POMDP

Consider a search over the three-node hierarchy presented in Figure 2, for which there are eight possible states:

$$\langle c_0, 1 \rangle, \langle c_0, 0 \rangle, \langle c_1, 1 \rangle, \langle c_1, 0 \rangle, \langle c_2, 1 \rangle, \langle c_2, 0 \rangle, \langle \emptyset, 1 \rangle, \langle \emptyset, 0 \rangle \,.$$

The POMDP for this simple tree yields belief states b_{c_0}, b_{c_1}, b_{c_2} when the bot is at the corresponding node, and the trivial belief states at the fully observed terminal states $\langle \emptyset, 1 \rangle$ and $\langle \emptyset, 0 \rangle$. The bot moves using the guidance function values $\tilde{\eta} \triangleq \eta(c_0, c_1)$ and $\eta(c_0, c_2) = 1 - \tilde{\eta}$ to determine the next node when the action $\mathrm{a_d}$ is selected. The set of reachable belief states is represented in an



(a) Location of the bot at time 1

(b) Location of the bot at time 2

Figure 2: A very simple tree and the flow of the bot's movement, i.e. the bot begins at the root node c_0 and then chooses to move, but randomly arrives at c_1

AND-OR tree in Figure 3 (see a similar figure in Ross et al. (2008b)). In this figure, an action must be chosen at an OR node, the choice of which leads to the set belief states, over all possible observations, that must be considered at the AND nodes. Expected rewards, $R(b, a)$ are represented on the arcs from OR- to AND-nodes, while the transition probabilities $p(o \mid b, a)$ are represented on the arcs from AND- to OR-nodes. Working from the leaf nodes back to the root, we can read from the tree that action $\mathrm{a_s}$ at node c_0, would lead to an expected reward of

$$Q(b_{c_0}, \mathrm{a_s}) = b_{c_0} r(c_0) + (1 - b_{c_0})(-1) = b_{c_0}(r(c_0) + 1) - 1$$

while action $\mathrm{a_d}$ at c_0 would lead to an expected reward of

$$\begin{aligned} Q(b_{c_0}, \mathrm{a_d}) &= \tilde{\eta}(b_{c_1}(r(c_1) + 1) - 1) + (1 - \tilde{\eta})(b_{c_2}(r(c_2) + 1) - 1) \\ &= b_{c_0}\left[\tilde{\eta}^2(r(c_1) + 1) + (1 - \tilde{\eta})^2(r(c_2) + 1)\right] - 1 \end{aligned}$$

where the second expression above uses Equation (6) to obtain $b_{c_1} = \tilde{\eta} b_{c_0}$ and $b_{c_2} = (1 - \tilde{\eta}) b_{c_0}$.

4.11. Hierarchy Quality for Search

The HQS is defined by the weighted mean of the expected return for searching all items with a certain policy. In particular, we apply the policy discussed next section to compute the expected reward V_x of searching x. Eventually, it shows that

$$\mathrm{HQS} = \mathrm{E}_x[V_x]\,.$$

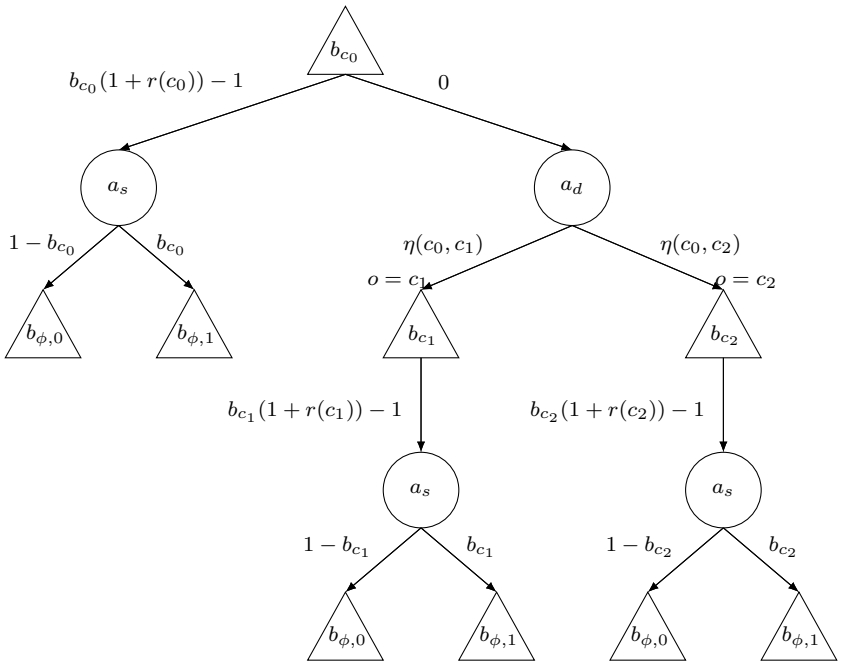


Figure 3: An AND-OR tree of reachable belief states from node c_0 of the three-node hierarchy of Figure 2

The entire flow is shown in Figure 4.

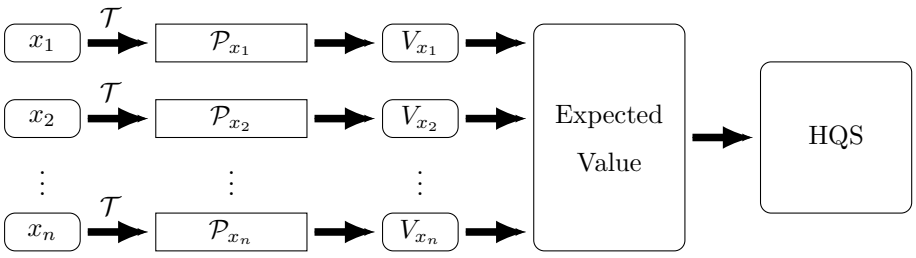


Figure 4: The flow chart of the HQS

5. Solving the POMDP

There are two main categories of POMDP solvers. The first category consists of off-line methods that either learn the whole environment prior to determining

the policy or that learn it through reinforcement learning (RL). The second category consists of on-line planners that solve the POMDP in a real-time decision making setting with the assumption that there is no control over all the states. We appeal to on-line planners as they better simulate the situation that the bot makes decisions as the search proceeds. Also off-line methods would make the process intractable for the evaluation task.

One remarkable on-line planner POMCP (Silver and Veness, 2010), is a Monte-Carlo approach that also integrates RL into its real-time decision making. With a sufficiently large number of samples, POMCP is able to plan a policy that earns a good underlying MDP value regardless of the belief states.

This contradicts with what we expect from the role of belief in our model. If the guidance function is weak, then we expect that this should be reflected in a belief function that leads to a poor reward for the underlying MDP, as it should tend to lead the bot on a path not containing the target. A sophisticated policy that overcomes this weakness, is not of interest in our setting. Hence, the bot chooses the Real-Time Belief Space Search (RTBSS) planner to seek the policy (Paquet et al., 2005a,b; Ross et al., 2008b). In our configuration of this planner, at each decision point, the bot is restricted to learn the required information only of the immediate children, limiting it to only one layer down in the hierarchy. From the point-of-view of the planner, this corresponds to *two* look-aheads, where the bot can compare stopping and searching at the current node, with the two steps of descending to a node on the next layer down and then stopping.

5.1. RTBSS

Algorithm 1 demonstrates the original RTBSS procedure as presented in (Ross et al., 2008b). It heavily relies on the function Expand(b, a) in Algorithm 2 to explore the POMDP. The Boolean function IsLeaf(b) returns `true` if the only belief states reachable from b with non-zero probability are the terminal states. RTBSS is a greedy algorithm that explores a lower-bound on the optimal $V(b)$ using a diversity of actions within a limited number of look-

aheads and selects the policy that maximizes this lower-bound. Considering that the look-aheads are capped, this can also be described as a myopic policy and so follows the proposal in (Fern et al., 2007) to use myopic heuristics for approximating the Q value for each belief-action pair to alleviate the intractable computations in a POMDP.

Algorithm 1: RTBSS

input : d, the maximum look-aheads which is fixed to 2 in our settings

output: π, the policy function

Initialize b

repeat

  $L, a \leftarrow \textsc{Expand}(b, d)$

  $\pi(b, a) \leftarrow 1$

  Execute a and perceive o

  $b \leftarrow \tau(b, a, o)$

until $\textsc{IsLeaf}(b)$;

Algorithm 2: EXPAND

input : b, the current belief state

input : d, the number of levels to explore, must be ≥ 0

output: L^*, optimal lower bound

output: a^*, optimal action

$L^* \leftarrow -\infty$

if $d = 0$ *or* $\textsc{IsLeaf}(b)$ **then**

  $L(a) \leftarrow R(b, a)$

else

  for $a \in A$ **do**

    $L(a) \leftarrow R(b, a) + \gamma \sum_{o \in O} p(o \mid b, a) \textsc{Expand}(\tau(b, a, o), d - 1)$

$L^* \leftarrow \max L(a)$

$a^* \leftarrow \text{argmax}_a \, L(a)$

We present the specialization of the RTBSS planner to our setting in Algorithm 3. At each decision point t, the algorithm calculates $Q(b_t, \mathrm{a_s})$, the Q-value of stopping and searching at the current node and an estimate for descending, $\hat{Q}(b_t, \mathrm{a_d})$, obtained by assuming that the child nodes are leaf nodes, or equivalently, by taking the value of a second descent step, $Q(b_{t+1}, \mathrm{a_d})$ to be 0. In particular, we have

$$Q(b_t, \mathrm{a_s}) = b_{c_t} r(c_t) + (1 - b_{c_t})(-1) = b_{c_t}(r(c_t) + 1) - 1$$

and,

$$\begin{aligned}
\hat{Q}(b_t, \mathrm{a_d}) &\triangleq \sum_{o \in O} p(o \mid b_t, \mathrm{a_d}) Q(\tau(b_t, \mathrm{a_d}, o), \mathrm{a_s}) \\
&= \sum_{c' \in \mathcal{C}(c_t)} p\left(o = c' \mid b_t, \mathrm{a_d}\right) Q(\tau(b_t, \mathrm{a_d}, o), \mathrm{a_s}) \\
&= \sum_{c' \in \mathcal{C}(c_t)} \eta\left(c_t, c'\right) Q(\tau(b_t, \mathrm{a_d}, o), \mathrm{a_s}) \\
&= \sum_{c' \in \mathcal{C}(c_t)} \eta\left(c_t, c'\right) \left(b_{c'}(r(c') + 1) - 1\right).
\end{aligned}$$

Our simplified algorithm provides a belief-based policy for descending through the hierarchy, which at any given node determines whether to descend further or to stop. We measure the quality of the hierarchy as the oracle value, i.e. the value of this policy in the underlying MDP. Since the reward function always outputs -1 whenever the policy leads to a wrong path, to compute this oracle value, we only need to focus on the correct path. This observation makes running the evaluation much more efficient than otherwise. In particular, now using $\{c_0, c_1, \ldots, c_T\}$ to denote the *correct* path containing the target item x, the oracle value is simply:

$$\begin{aligned}
V_x &= r(c_T) \prod_{t=1}^{T} \eta\left(c_{t-1}, c_t\right) + (-1) \cdot \left(1 - \prod_{t=1}^{T} \eta(c_{t-1}, c_t)\right) \\
&= \left[r(c_T) + 1\right] \left(\prod_{t=1}^{T} \eta(c_{t-1}, c_t)\right) - 1\,, \qquad (7)
\end{aligned}$$

given c_0 is the root node of the hierarchy.

Algorithm 3: SIMPLIFIED RTBSS POLICY SPECIFIED FOR HQS

Initialize b such that $b(\langle c_0, 1\rangle) \leftarrow 1$
$t \leftarrow 1$
repeat
    Compute $Q(b, \mathrm{a_s})$ and $\hat{Q}(b, \mathrm{a_d})$
    if $\hat{Q}(b, \mathrm{a_d}) > Q(b, \mathrm{a_s})$ **then**
        $a^* \leftarrow \mathrm{a_d}$
    else
        $a^* \leftarrow \mathrm{a_s}$
    $\pi(b, a^*) \leftarrow 1$
    $b \leftarrow \tau(b, a, o)$ `// focus on the` o `in the right path only`
    $t \leftarrow t + 1$
until ISLEAF(b) *or* $\pi(b, \mathrm{a_s}) = 0$
$\pi(b, \mathrm{a_s}) \leftarrow 1$ `// The bot can only choose to stay at a leaf node`

One can intuitively see that the bot moves randomly according to $\eta(\cdot)$, and when it stops at some time step T, the probability of obtaining a positive reward is simply the probability that the guidance function led it along the right path. Its stopping point is determined by the calculation of its expected reward over its belief that it is on the right path. Note that, given $0 \leq r(\cdot) \leq 1$, it follows that $-1 \leq V_x \leq 1$ which matches our intuition that if there are fewer items stored in the node at which the bot stops and the uncertainty of reaching this node is smaller, the bot achieves a higher value. The stopping point T, is the key output of the belief-based policy obtains that determines the oracle value.

5.2. Time Complexity

Solving a finitely horizontal POMDP is PSPACE-complete (Papadimitriou and Tsitsiklis, 1987), while luckily it does not apply to our case. Consider the HQS calculation for a single target x. Let $\mathcal{F}(\mathcal{S})$ represent the complexity of calculating the similarity between x and a cluster. Let us reasonably assume that each non-root

node has at least one sibling in the hierarchy. For the data with N entries and corresponding hierarchy with M^* nodes, the maximum M^* is $2N-1$. This holds when the tree splits one data point as a leaf and all others remain as one cluster, until all points become leaves. Least optimally, the searcher needs to estimate the return at all nodes for a certain target, which will be in $O(M^*)$. However, the policy can still be pruned as reaching a wrong node finally receives the reward -1. Accordingly the reward computation can concentrate on the path wherein each node contains the target. Denote the number of children of the t^{th} parent in the right path by N_t. The complexity will then follow $O(\sum_t N_t) = O(M^* N)$ which is thus $O(N^2)$. Hereafter, the guidance function requires $O(N^2 \mathcal{F}(\mathcal{S}))$ computations given that $O(\mathcal{F}(\mathcal{S}))$ is the complexity for calculating the similarity for a data point to a cluster where both are with D dimensions. Even though, the average case for the height of a tree is always logarithmic. We can write the average complexity of searching for a target as $O(a \log_a M) = O(a \log_a N)$ where a is a constant for the number of children. The average case for the HQS is therefore $O(a \log_a N \times N\mathcal{F}(\mathcal{S})) = O(N \log N \mathcal{F}(\mathcal{S}))$. The polynomial result concludes that HQS is practically applicable.

6. HQS Design Summary

To summarize the motivation and methodology behind our proposed HQS measure, Table 1 lists the challenges we have addressed and how these have been handled within our proposed framework.

7. Experimental Study

This section is devoted to some case studies. For convenience, we select $N = 12$ items from the Amazon data[1] (McAuley et al., 2015). This dataset consists of images, descriptions, user ratings and reviews for goods in the Amazon catalogue. In this study, we use the textual data associated with each item,

[1] `http://jmcauley.ucsd.edu/data/amazon`

Table 1: Design of HQS

Aspect	Requirement	Challenge	Solution
Practical implementation	Tractability	Solving a POMDP is universally intractable	Online planner to guarantee sub-optimality
Practical implementation	Flexibility in the guidance function	Different traditional measures may fit specific datasets better than others	Leave the similarity choice to the users
Practical implementation	Fair probabilities from the feedback	It could happen that cluster qualities are equally bad as the bot descends. For example, the similarities of x to random cluster A and cluster B, respectively, return 10^{-50} and 10^{-70}, but then A will have the probability $\eta = 1$	Use Boltzmann distribution and tune the temperature parameter to control this situation—temperature can be also a function over time (current depth of the hierarchy) t, e.g. $\delta_t \triangleq \delta_{base}\nu^t$ where $\nu \in [1, \infty)$
Measuring need	Uncertainty in the bot's search	Stochastic policies are hardly studied; There is no way to judge it	Impose the randomness into the bot's moving actions
Measuring need	Limit the bot not to be too smart	E.g., POMCP ignores the external belief and guides the bot to find the target with a high probability	Greedy optimization—RTBSS

and base the guidance function on a TF-IDF similarity score. The motivation is to choose commonly seen daily life products and manually construct a number

of hierarchies, varying their quality. Restricting to just 12 items ensures that the hierarchies can be intuitively assessed by inspection. We also limit the maximum depth of the hierarchy to be $\log_2(N) = \log_2(12) \approx 3$ since each internal node should be split to at least two children.

This approach can help us quickly identify how the HQS works by assessing whether it orders the hierarchies in the expected manner.

7.1. The Selected Items

Some numerical details about the items are displayed in Tables A.8 and A.9 which are in the appendix. In the latter analysis, we refer items with their indices that are decided in Tables A.8 and A.9. The left column is the title of the item, and the second column contains the top 10 terms in the title and the description of the item, with the corresponding TF-IDF score in the parenthesis. As the examples all come from the large fashion category, the TF-IDF score is computed only on this subset of the Amazon data. However, when computing the similarities, we still consider all the features without any feature selection techniques given the small number of items.

Since the data is represented by TF-IDF vectors, which we write as $\boldsymbol{v}_x$ for item x. We define the similarity function $\mathcal{S}(x, c)$ as follows

$$\mathcal{S}(x,c) = \begin{cases} 1 & c = \{x\} \\ \frac{1}{c-1} \sum_{x' \in c \setminus \{x\}} \cos(\boldsymbol{v}_x, \boldsymbol{v}_{x'}) & \text{otherwise} \end{cases}.$$

It is in spirit similar to average link in the Agglomerative Clustering (Day and Edelsbrunner, 1984). We deem it necessary to exclude x itself from computing the similarities over a non-singleton cluster, since the maximum cosine similarity between x and the other items is only 0.11 in the selected samples, and hence the cosine of x with itself would dominate the similarity score.

For the guidance function in our experiments, we set $\delta = .01$ which helps us increase the weight of the best cluster and ensure that the cluster with largest similarities to the datum, has a dominant η value. In these cases, the tree depths are small and none of the similarities of items-to-groups is not as terrible as we

described in Table 1. Therefore, we don't consider a time-variant δ, i.e. we set $\delta_{base} = .01$ and $\nu = 1$ in $\delta_{base}\nu^t$.

This shows the advantages of allowing the users to customize the similarity function and the guidance function to fit the use-case scenario.

7.2. Tested Hierarchies

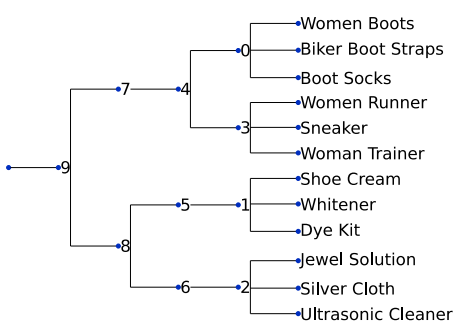
Now, we would like to analyze the HQS scores with respect to the ground-truth hierarchy and four other hierarchies as shown in Figure 5, where Hierarchy A is the ground truth obtained from the labels provided in the Amazon dataset. Hierarchies B and C are randomly generated hierarchies. Finally, hierarchy D is constructed to be deliberately poor. All items in each leaf cluster have different ground-truth labels. For example, "Whitener", "Biker Boot Straps", and "Ultrasonic Cleaner" in the cluster 0 of hierarchy D are all in separate clusters according to the Amazon organization. Ideally, the ground truth hierarchy A should provide the best score and hierarchy D should provide the worst score among these five hierarchies. In Section 7.3, we show the HQSs of these 5 hierarchies and then show the detailed steps of how HQS works along with hierarchies with different structures. Additionally, hierarchy E is constructed on purpose which shows that a hierarchy which is rather different from the ground-truth hierarchy can also be identified as a good hierarchy.

Table 2: HQS results with respect to the five different hierarchies.

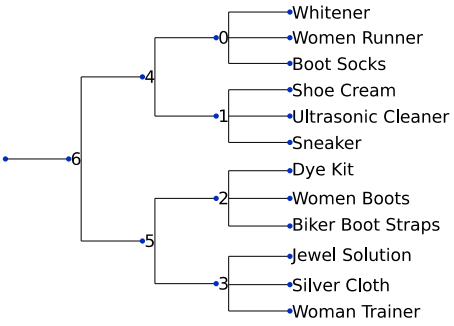
Hierarchy	A	B	C	D	E
HQS	.5715	-.2801	-.3097	-.8477	.3266

7.3. Analysis

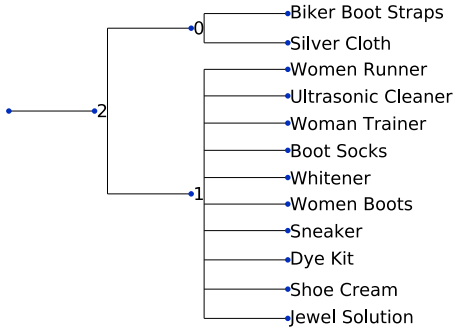
Table 2 shows the HQS results of the 5 different hierarchies in Figure 5. As we can see from the table, the ground truth hierarchy (hierarchy A) provides the highest HQS score (0.5715), followed by randomly generated hierarchies B and C. Hierarchy D provides the worst HQS score (-0.8477). Thus, in this example,



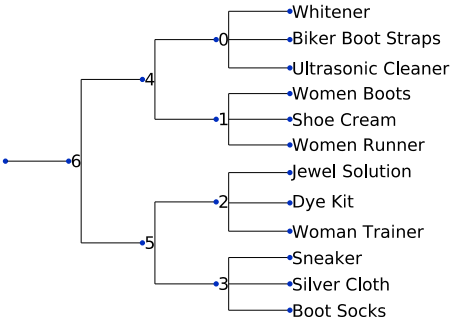
(a) Hierarchy A



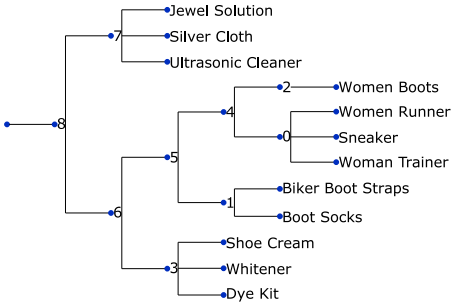
(b) Hierarchy B



(c) Hierarchy C



(d) Hierarchy D



(e) Hierarchy E

Figure 5: Five different hierarchies of the Amazon data for investigating their HQSs. Hierarchy A is the ground truth hierarchy from the data itself, hierarchies B and C are the randomly generated ones, hierarchy D is deliberately generated with all items in each leaf cluster belong to different labels, and hierarchy E is a good hierarchy which is clustered differently to the ground-truth. The number at each node is the corresponding index.

the HQS scoring scheme agrees with our intuition. Finally, hierarchy E obtains a relatively high HQS (0.3266) as we expected. We believe this is a reasonable quality ranking of the hierarchies for the five hierarchies.

Hence, we specify how the policy propagates step-by-step for hierarchies A and B as an example. After analyzing each case, we discuss the characteristics of the HQS.

7.3.1. Hierarchy A

First of all, we detail how the HQS analyzes the ground truth hierarchy Figure 5a. This hierarchy is generated given the ground truth chained labels of the items. For instance, node 4 refers to "Clothing, Shoes & Jewelry", node 7 refers to "Women", and 5 refers to "Shoe Care & Accessories" etc.

Let us check through the process for item x_0, "Women Boots". It starts with the root node, where $b(\langle c_9, 1\rangle) = 1$. Hence, staying and searching now earns an estimate $Q(b_0, \mathrm{a_s}) = 0$. Now, the similarities x_0 with the child nodes are as follows:

$$\mathcal{S}(x_0, c_7) = .061 \qquad \mathcal{S}(x_0, c_8) = .0277$$

and it follows that

$$\eta(c_9, c_7) = .9987 \qquad \eta(c_9, c_8) = .0013\,.$$

Apparently, both nodes have 6 items and so that the reward for staying at both nodes are equal, 0.6226. It follows that

$$\begin{aligned}
\hat{Q}(b_0, \mathrm{a_d}) &= \sum_{o \in \{c_7, c_8\}} p(o \mid b, \mathrm{a_d}) Q(\tau(b_0, \mathrm{a_d}\, o)) \\
&= \eta(c_9, c_7)\big\{\eta(c_9, c_7)[r(c_7) + 1] - 1\big\} \\
&\quad + \eta(c_9, c_8)\big\{\eta(c_9, c_8)[r(c_8) + 1] - 1\big\} \\
&= .6203\,.
\end{aligned}$$

Since $Q(b_0, \mathrm{a_s}) < \hat{Q}(b_0, \mathrm{a_d})$, the bot selects action $\mathrm{a_d}$, and it stochastically moves to the right node with probability .9986. From c_7 to c_4, it is a straight

move as the estimates will be equal and should be encouraged to explore a better chance for a higher total value. The next move is more interesting. Let us review that cluster c_0 refers to "Boots" and c_3 refers to "Fashion Sneakers" under the category chain "All – Clothing, Shoes & Jewelry – Women" (c_9, c_7, c_4). Now, the similarities return $\mathcal{S}(x_0, c_0) = .06, \mathcal{S}(x_0, c_3) = .061$ and we obtain $\eta(c_4, c_0) = .4158$ and $\eta(c_4, c_3) = .5829$. Which suggests that x_0 more strongly belongs to cluster c_3 than to c_0, the cluster it is assigned to. In fact, if we look at cluster c_3, its items "Women Runner", "Women Sneaker" and "Women Trainer" are highly close to "Women Boots" in some sense. It turns out that, after computing the Q-values of $\mathrm{a_s}$ and $\mathrm{a_d}$, the bot decides to stop at node c_4, since it is not sufficiently confident that another descent step will lead to greater reward. Finally, the oracle value V_{x_0} is .6203, a good score, but short of the maximum reward which captures the lack of clarity between the clusters at the lower levels of the hierarchy for this data item. This suggests that our method gives a rather reasonable measurement in this case.

Another interesting example is x_7, the "Dye Kit". It starts navigating wrong actually at the first level. The similarities $\mathcal{S}(x_7, c_7)$ and $\mathcal{S}(x_7, c_8)$ return .0367 and .0297 respectively. It belongs to c_8 while the measurement prefers c_7. In fact, "Dye Kit" is close to the shoe items in cluster c_7, but also close to the items "Shoe Cleaner" and "Whitener" in cluster c_8, as they are also cleaning tools (for jewelry). As it stands, the bot will be guided to the wrong node and hence will obtain a negative reward. It can be argued that this is due to a weakness of the similarity measure, as much as the clustering but it could well be the case that the situation would be avoided with more data items.

A summary of all items' single HQS score is shown in Table 3. The bot is not confident to descend to the bottom of the hierarchy for all the cases, and this is reflected in the associated score. The overall average HQS is then .5715 and the average stopping level is 3.42. This hierarchy is good enough to help the bot descend while maintaining a good HQS.

Table 3: Summary of HQS for Hierarchy A

x	Stop at	$V(x)$
0	3	0.620389
1	4	0.746515
2	4	0.826451
3	4	0.811422
4	3	0.599822
5	4	0.517870
6	4	0.792900
7	2	-0.679057
8	2	0.561567
9	3	0.431666
10	4	0.812053
11	4	0.816681

Table 4: Summary of HQS for Hierarchy B

x	Stop at	$V(x)$
0	1	0.000000
1	3	-0.991150
2	3	-0.988855
3	2	-0.862852
4	3	-0.955350
5	3	-0.736097
6	3	0.759935
7	3	0.658529
8	2	-0.995267
9	1	0.000000
10	3	0.749206
11	1	0.000000

7.3.2. Hierarchy B

The randomly generated hierarchy B is depicted in Figure 5b. In this case, the final HQS is -0.2801 and the average stopping time-stamp is 2.333 out of 3. Examining Table 4, we see that there are six items earning a negative reward as the hierarchy guides them to the "wrong" node. Three more items earn a score of 0, as the bot gets confused at the root level and so that it prefers searching at the root. The remaining three items gain good results.

Interestingly, a random hierarchy does not necessarily achieve an absolute 0 HQS. The measurement carried out by HQS is apparently not linear.

7.3.3. Hierarchy C

Hierarchy C, depicted in Figure 5c achieves its final HQS score of -0.3097 and the average stopping time-stamp is 1.667 out of 2, with item scores shown in Table 5. Similarly to hierarchy B, there are six items earning a negative

reward as the hierarchy has only two items achieving positive scores which are the two in the small cluster, "Biker Boot Straps" and "Silver Cloth" obtain .889234 and 0.878909 respectively. These two items benefit from the fact that the sibling clusters are too diversified, and hence the similarities are diluted compared with the small node which has less diversity. In spite of the two items earning good values, the overall HQS score unveils the fact that the hierarchy is terrible. It indicates that although HQS begins with calculating the values of the bot searching for each item, its final score considers the hierarchy as a whole. Even a terrible hierarchy might enable efficient search for a few items, while there are many more poorly scoring items. Moreover, many items earn 0 as the bot will be confused at the root level and so it prefers searching at the root.

Table 5: Summary of HQS for Hierarchy C

x	Stop at	$V(x)$
0	2	-0.751922
1	2	-0.999094
2	1	0.000000
3	1	0.000000
4	1	0.000000
5	2	0.889234
6	2	-0.856630
7	2	-0.933072
8	2	0.878909
9	1	0.000000
10	2	-0.964455
11	2	-0.978953

Table 6: Summary of HQS for Hierarchy D

x	Stop at	$V(x)$
0	3	-0.668088
1	2	-0.740265
2	3	-0.707562
3	3	-0.995524
4	3	-0.651974
5	3	-0.999896
6	2	-0.791041
7	2	-0.807062
8	2	-0.995364
9	2	-0.925345
10	2	-0.989777
11	3	-0.900619

7.3.4. Hierarchy D

For hierarchy D (Figure 5d with item scores shown in Table 6), the overall HQS score of -0.8477, and average stopping time-stamp of 2.5 out of 3, reflects the fact that this is a terrible hierarchy. It is our expected worst case and HQS successfully reveals this by returning a score close to the minimum, the worst performing one among the exhibited hierarchies. This indicates that HQS downgrades a random hierarchy largely.

7.3.5. Hierarchy E

Finally, we construct a "good" hierarchy (Figure 5e with item scores shown in Table 7), while its style is very different to the ground-truth hierarchy. It separates items about jewelry and the items associated with shoes at the first level. Overall, there is only one item, "Shoe Cream" which receives a negative value $-.7972$. At the first decision point, $\eta(c_8, c_7)$ is .8894 which leads the bot to move to the wrong node with a dominant probability. The item, given its textual representation, tends to be better associated with the cleaning toolkits. Moreover, "Dye Kit", "Whitener", and "Biker Boot Straps" each achieves a HQS score of 0. For "Dye Kit" and "Whitener", it is rather reasonable as it could be seen better grouped with the jewelry cleaning tools as they belong to the same group according to the ground truth. Interestingly, "Biker Boot Straps" receives the similarities 0.0379 and 0.0375 for c_6 and c_7 respectively. Thus, the bot stops at the root node. Finally, other items perform reasonably well. This implies that the HQS does not devalue hierarchies with different structures to the ground-truth hierarchy as long as the hierarchy provides efficiency for search.

7.4. Discussion of the Results

Based on the studies, we can discover the following characteristics.

The temperature parameter in the Boltzmann distribution has to help the bot find a leading η value within a certain scope, in particular at a higher level. The only exception is that the clusters are all just extremely close to the datum.

Table 7: Summary of HQS for Hierarchy E

	Stop at	V(x)
0	3	0.610799
1	3	-0.797205
2	4	0.672706
3	1	0.000000
4	3	0.533369
5	1	0.000000
6	2	0.833914
7	1	0.000000
8	2	0.423173
9	1	0.000000
10	2	0.834673
11	4	0.807958

One can observe that, the bot will only desire to descend when there is a leading η among those for the children, at the current node. Otherwise, the expected belief-based value of (a_d, a_s) will be simply smaller than staying at the current node. However, we tend not to remove all this kind of cases for close η weights between the siblings. This may guide us select the temperature parameter.

HQS will highly recommend merging the very similar sibling clusters into one. Imagine that there are two close siblings, which will reduce the chance of obtaining a leading η weight (though this is better than receiving a leading η to the wrong child). With the two extremely close clusters merged, HQS will appreciate that as it helps the bot descend to a deeper level (in the right path) and achieve better reward.

Finally, a HQS score of 1 is the limit, as the relative size of the clusters in which items are found tends to zero, of what can be achieved by a perfect hierarchy in which all items are successfully found by the bot. In practice,

even the highest quality hierarchies achieve scores less than unity. Hence, it is worth pointing out that positive scores less than 1, do not indicate that the hierarchy is terrible; in fact, HQS scores should be used as a means of comparing different hierarchies, rather than as an absolute measurement of how good a single hierarchy is.

8. Conclusion

This paper has presented a POMDP for modeling the uncertainty and the decision making for searchers with regard to search and navigation in a hierarchy. This model can be extended to measuring the quality of the hierarchies. Encouragingly, the current experiments achieve a sensible grading for the performances generated by several algorithms. We deem it the first step towards creating a generic quantitative measure for evaluating hierarchies, accounting for the quality of hierarchical structure along with the item to cluster assignments. Also, we hope this opens more research on this less active topic—measuring the hierarchy, which may possibly promote creating novel methods in the hierarchical clustering.

A number of questions should be further investigated. The model and the policy are improvable so that one could work on the better models and policies. Another critical task is to explore the input parameters for the model, e.g., the reward function for staying and searching inside a node needs more solid function design.

Acknowledgements

W. Huang and N. Hurley have been supported by Science Foundation Ireland (SFI) by grant SFI/12/RC/2289 and SFI/12/RC/2289_P2. G. Piao and R. Moreno were supported by SFI under grant number SFI/12/RC/2289.

References

Aberdeen, D., Baxter, J., 2002. Scalable internal-state policy-gradient methods for pomdps, in: Proceedings of the Nineteenth International Conference on Machine Learning, San Francisco, CA, USA. pp. 3–10.

Balcan, M.F., Liang, Y., Gupta, P., 2014. Robust Hierarchical Clustering. The Journal of Machine Learning Research 15, 3831–3871.

Boger, J., Poupart, P., Hoey, J., Boutilier, C., Fernie, G., Mihailidis, A., 2005. A decision-theoretic approach to task assistance for persons with dementia, in: IJCAI, Citeseer. pp. 1293–1299.

Boutilier, C., Poole, D., 1996. Computing Optimal Policies for Partially Observable Decision Processes Using Compact Representations, in: AAAI, pp. 1168–1175.

Brafman, R.I., Tennenholtz, M., 2002. R-max - a general polynomial time algorithm for near-optimal reinforcement learning. Journal of Machine Learning Research 3, 213–231.

Cassandra, A.R., Kaelbling, L.P., Littman, M.L., 1994. Acting Optimally in Partially Observable Stochastic Domains, in: AAAI, pp. 1023–1028.

Cigarran, J.M., Peñas, A., Gonzalo, J., Verdejo, F., 2005. Evaluating hierarchical clustering of search results, in: String Processing and Information Retrieval: 12th International Conference, SPIRE 2005, pp. 49–54.

Day, W.H., Edelsbrunner, H., 1984. Efficient algorithms for agglomerative hierarchical clustering methods. Journal of classification 1, 7–24.

Duff, M.O., 2002. Optimal Learning: Computational Procedures for Bayes-adaptive Markov Decision Processes. Ph.D. thesis. University of Massachusetts at Amherst.

Fard, M.M., Pineau, J., 2011. Non-Deterministic Policies In Markovian Processes. Journal of Artificial Intelligence Research 40, 1–24.

Fern, A., Natarajan, S., Judah, K., Tadepalli, P., 2007. A decision-theoretic model of assistance., in: IJCAI, pp. 1879–1884.

Fern, A., Natarajan, S., Judah, K., Tadepalli, P., 2014. A decision-theoretic model of assistance. Journal of Artificial Intelligence Research 50, 71–104.

Gosavi, A., Purohit, M., 2011. Stochastic Policy Search for Variance-penalized Semi-Markov Control, in: Proceedings of the 2011 Winter Simulation Conference, pp. 2865–2876.

Johnson, D.M., Xiong, C., Gao, J., Corso, J.J., 2013. Comprehensive Cross-Hierarchy Cluster Agreement Evaluation, in: AAAI (Late-Breaking Developments).

Kaelbling, L.P., Littman, M.L., Cassandra, A.R., 1998. Planning and Acting in Partially Observable Stochastic Domains. Artificial Intelligence 101, 99–134.

Kearns, M., Mansour, Y., Ng, A.Y., 2002. A Sparse Sampling Algorithm for Near-Optimal Planning in Large Markov Decision Processes. Machine learning , 193–208doi:10.1023/A:1017932429737.

Kearns, M., Singh, S., 2002. Near-optimal Reinforcement Learning in Polynomial Time. Machine Learning 49, 209–232.

Kobren, A., Monath, N., Krishnamurthy, A., McCallum, A., 2017. A Hierarchical Algorithm for Extreme Clustering, in: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM. pp. 255–264.

Lam, C.P., Sastry, S.S., 2014. A pomdp framework for human-in-the-loop system, in: IEEE 53rd Annual Conference on Decision and Control (CDC), IEEE. pp. 6031–6036.

Liu, Y., Li, Z., Xiong, H., Gao, X., Wu, J., Wu, S., 2013. Understanding and Enhancement of Internal Clustering Validation Measures. IEEE Transactions on Cybernetics 43, 982–994.

McAuley, J., Targett, C., Shi, Q., Van Den Hengel, A., 2015. Image-based Recommendations on Styles and Substitutes, in: Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM. pp. 43–52.

Meuleau, N., Kim, K.E., Kaelbling, L.P., Cassandra, A.R., 1999. Solving pomdps by searching the space of finite policies, in: Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence, Morgan Kaufmann Publishers Inc.. pp. 417–426.

Moreno, R., Huáng, W., Younus, A., O'Mahony, M., Hurley, N.J., 2017. Evaluation of Hierarchical Clustering via Markov Decision Processes for Efficient Navigation and Search, in: Experimental IR Meets Multilinguality, Multimodality, and Interaction. CLEF 2017, Proceedings, Springer. pp. 125–131. doi:10.1007/978-3-319-65813-1.

Ng, B., Boakye, K., Meyers, C., Wang, A., 2012. Bayes-Adaptive Interactive POMDPs.

Papadimitriou, C.H., Tsitsiklis, J.N., 1987. The Complexity of Markov Decision Processes. Mathematics of operations research 12, 441–450.

Paquet, S., Tobin, L., Chaib-Draa, B., 2005a. An Online POMDP Algorithm for Complex Multiagent Environments, in: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems, ACM. pp. 970–977.

Paquet, S., Tobin, L., Chaib-draa, B., 2005b. Real-Time Decision Making for Large POMDPs, in: Conference of the Canadian Society for Computational Studies of Intelligence, Springer. pp. 450–455.

Ross, S., Chaib-draa, B., Pineau, J., 2008a. Bayes-Adaptive Pomdps, in: Advances in neural information processing systems, pp. 1225–1232.

Ross, S., Pineau, J., Paquet, S., 2008b. Online Planning Algorithms for POMDPs. Journal of Artificial Intelligence Research 32, 663–704.

Sakaguchi, M., 1961. Dynamic Programming of Some Sequential Sampling Design. Journal of Mathematical Analysis and Applications 2, 446–466.

Silver, D., Veness, J., 2010. Monte-Carlo Planning in large POMDPs, in: Advances in neural information processing systems, pp. 2164–2172.

Singh, S.P., Jaakkola, T., Jordan, M.I., 1994. Learning without State-Estimation in Partially Observable Markovian Decision Processes, in: Machine Learning Proceedings 1994. Elsevier, pp. 284–292.

Somani, A., Ye, N., Hsu, D., Lee, W.S., 2013. DESPOT: Online POMDP Planning with Regularization, in: Advances in Neural Information Processing Systems, NIPS, pp. 1772–1780.

Steinbach, M., Karypis, G., Kumar, V., 2002. A Comparison of Document Clustering Techniques, in: TextMining Workshop at KDD2000 (May 2000).

Thomson, B., Young, S., 2010. Bayesian update of dialogue state: A pomdp framework for spoken dialogue systems. Computer Speech & Language 24, 562–588.

Appendix A. Amazon Item Details

Table A.8: Selected Items from Amazon: Part 1

index	short name	top 10 terms
0	Women Boots	Harness (0.2455), term (0.2186), long (0.1984), Womens (0.181), durability (0.1709), boots (0.1647), inch (0.1547), Crushed (0.1514), element (0.1465), tougher (0.1465),
1	Shoe Cream	Meltonian (0.2913), waxes (0.2768), cream (0.2169), cloth (0.1914), afterwards (0.1653), Misc (0.158), staining (0.1489), honored (0.1489), creamy (0.1489), terrific (0.1489),
2	Women Runner	Ascend (0.4741), Wave (0.3866), MIZUNO (0.237), EU (0.2266), SZ (0.2135), lends (0.2089), Mizuno (0.2049), UK (0.1822), Running (0.1822), China (0.1659),
3	Whitener	Whitener (0.5887), Sport (0.3376), chalky (0.2943), restores (0.2502), KIWI (0.2324), formula (0.218), scuffs (0.218), polish (0.1974), covers (0.1974), Kiwi (0.1925),
4	Sneaker	Coach (0.4753), signature (0.2487), leather (0.2173), preeminent (0.1759), Poppy (0.1759), Barrett (0.1759), emerged (0.1759), resulting (0.1682), Scribble (0.1682), coach (0.1682),
5	Biker Boot Straps	to (0.2226), 6in (0.2192), are (0.2111), clips (0.1902), in (0.1884), Straps (0.188), sold (0.1716), prevent (0.1564), Boot (0.15), SP6 (0.1402),

Table A.9: Selected Items from Amazon: Part 2

index	short name	top 10 terms
6	Jewel Solution	cleaning (0.2535), components (0.2389), metals (0.2297), precious (0.1925), solution (0.1867), free (0.1512), accumulate (0.1482), Biodegradable (0.1482), titanium1 (0.1482), injectors (0.1482),
7	Dye Kit	TRG (0.4005), included (0.2872), Turquoise (0.2277), Detailed (0.2141), Everything (0.2117), coats (0.2105), dye (0.1958), instructions (0.1937), Self (0.1912), Dye (0.1852),
8	Silver Cloth	silver (0.3919), tarnishing (0.3315), tarnish (0.189), Silvershield (0.1713), Cadet (0.1713), shining (0.1414), drawer (0.1414), trade (0.1389), Tarnish (0.1389), will (0.1277),
9	Woman Trainer	Ryka (0.5414), Rythmic (0.406), Womens (0.1547), the (0.148), Athena (0.1353), cardio (0.1353), fittest (0.1353), Rhythmic (0.1353), kickboxing (0.1353), gain (0.1294),
10	Ultrasonic Cleaner	Professional (0.3063), ultrasonic (0.2914), grade (0.2771), NUMWPT (0.2082), Qt (0.2082), Joy4Less (0.2082), automotive (0.1925), transducer (0.1875), Heater (0.1875), controls (0.1834),
11	Boot Socks	dead (0.3354), Cuffs (0.3036), gorgeous (0.2605), drop (0.2396), lace (0.2396), season (0.2363), Add (0.2348), cuffs (0.234), put (0.2248), Socks (0.2194),

View publication stats