Partially Observable Markov Decision Process Modelling for Assessing Hierarchies

Weipeng Huang   WEIPENG.HUANG@INSIGHT-CENTRE.ORG
Insight Centre for Data Analytics, University College Dublin, Ireland

Guangyuan Piao   GUANGYUAN.PIAO@INSIGHT-CENTRE.ORG
Insight Centre for Data Analytics, National University of Ireland, Galway

Raul Moreno   RAUL.MORENO.SALINAS@GMAIL.COM
Neil Hurley   NEIL.HURLEY@INSIGHT-CENTRE.ORG
Insight Centre for Data Analytics, University College Dublin, Ireland

Editors: Sinno Jialin Pan and Masashi Sugiyama

Abstract

Hierarchical clustering has been shown to be valuable in many scenarios. 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. This measurement is useful, e.g., to assess the hierarchical structures used by online retailer websites to display their product catalogues. Our framework is one of the few attempts for the hierarchy 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 Partially Observable Markov Decision Processes (POMDP) to model the uncertainty, the decision making, and the cognitive return for searchers in such a scenario.

Keywords: POMDP Applications, Evaluating Hierarchical Clustering Results

1. Introduction

Hierarchical clustering (HC) 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 organisation 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 organised 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 organising 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.
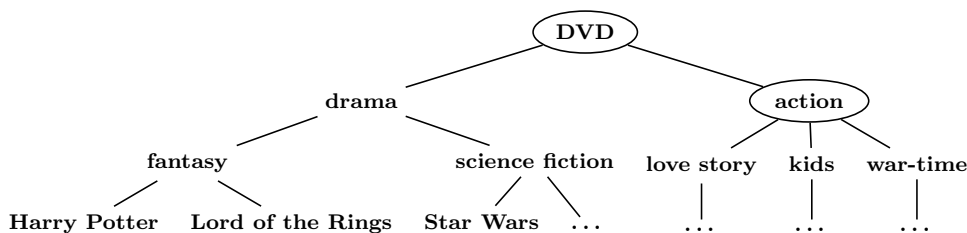
Figure 1: An example which illustrates a hierarchy that confuses the searchers given that the target is a Harry Porter DVD

We consider on-line retailers and the customers who access them, as an example. 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, labelled "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 hierarchy. As another example, Fig. 1 shows that a searcher may be confused and possibly search for a Harry Potter DVD in the wrong route. Is it more so an action movie than a drama movie? In this second example, it may well be the case that each cluster contains a coherent set of items, but their hierarchical organisation 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.

There are many evaluation functions already proposed to measure the quality of regular (non-hierarchical) clustering results, with or without ground-truth data. Any of these measures can be applied to a HC, 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 scoring the structural organisation of the hierarchy. This is critical in order to properly evaluate hierarchies in the context that we have in mind. In particular, one can extend it to perform the hyper-parameter tuning for a number of HC algorithms. As demonstrated in Kobren et al. (2017), many HC algorithms depend on hyper-parameters and might possibly be further improved with our function.

Problem Statement   Let us consider a set of items $X = \{x_n\}_{n=1}^{N}$. HC is a set of nested clusters of the dataset arranged in a tree $\mathcal{H}$. Each node of the tree has associated with it a sub-set or cluster of the collection, and the children 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 $\mathcal{H}$, it also belongs to all of its ancestor nodes. The goal is to seek a function mapping $\mathcal{H}$ 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 summary, 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 modelling a simplified search process but one which is sufficient to capture the important features of the hierarchy that determine its suitability to support efficient search.

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 optimises the cumulative reward obtained when applying this policy over some time horizon. Moreno et al. (2017) proposed such an MDP to tackle the problem of evaluating hierarchies. In that 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. Notice that, this quality measure depends not only on the hierarchy itself, but also on the policy used to determine the action at each decision point.

Contribution We propose a novel framework extending the MDP model with a reasonable search policy proposed, which provides a solid model for measuring the efficiency of item retrieval in the hierarchy. Our framework explicitly models the searcher's lack of knowledge about the environment. This leads us to extend from an MDP model, to a Partially Observable MDP (POMDP) model. We coin the measure Hierarchy Quality for Search (HQS).

2. Related Works

When assessing general clustering results, one can appeal to measuring the results with or without the ground-truths (Liu et al., 2013; Steinbach et al., 2000). A ground-truth hierarchy should contain multiple layers assigning nodes to each level of the hierarchy. Thus, we do not consider the Dendrogram Purity (e.g., used in Heller and Ghahramani (2005); Kobren et al. (2017)) as it evaluates a hierarchy with regard to only one layer of cluster assignment. There are many tools for evaluating the quality of the clustering but they lack the capability to evaluate the hierarchical arrangement (Johnson et al., 2013).

Cigarran et al. (2005) proposed a prototype considering the content of the cluster, the hierarchical arrangement and the navigation cost. Unfortunately, it is hard to develop this idea as neither detailed procedures nor experiments were presented in this short paper. More recently, Johnson et al. (2013) proposed the Hierarchical Agreement Index (HAI), which borrows its concept from the Rand Index, to compare the structure to a ground-truth hierarchy. Moreno et al. (2017) introduced the concept of using MDPs to model the evaluation. They observed that HC 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. However, it is a prototype rather than a finished product. Our work extends these ideas to a POMDP (Kaelbling et al., 1998) model, which improves the previous work by 1) modelling

the uncertainty in the search process with belief states; 2) specifying a policy to solve the problem so that the strong assumptions (the bot knows the position of the items universally) in the previous work can be relaxed. We consider this work an important contribution to the topic of evaluating hierarchical structures with no help from ground-truths.

There exist abundant research addressing the issue of acting optimally in POMDPs, such as (Cassandra et al., 1994; Boutilier and Poole, 1996; Meuleau et al., 1999; Ross et al., 2008), to name but a few. Seeking a better policy is not the main focus in the paper; hence, we adopt the existing (on-line) policies. On the other hand, we find the works (Shani et al., 2005; Zheng et al., 2018; Ie et al., 2019) employ Reinforcement Learning (RL) to model the user behaviours for improving the performance in recommender systems, which are inspiring. These are somewhat close to our work while in a different application domain.

3. POMDP Specification

We develop a POMPD model of a bot searching a hierarchy, where, at each node, it must make the decision to search or to descend further. The bot cannot backtrack, once a descent step has been performed. Simply put, a POMDP is an MDP with additional observations, belief state and belief estimator (Cassandra et al., 1994; Kaelbling et al., 1998). 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 $T(s, a, s') \triangleq p(s' \mid s, a)$, represents the probability of moving to state s' from state 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 selects 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. In our model, it is sufficient to set $\gamma = 1$ and hence we will omit further explanation about this part.

STATES

POMDPs use states to represent the status of the search at a particular point during the search process. Our model defines that each state is a joint event consisting of 1) the physical location (node) c of the bot, and 2) a *boolean* variable indicating if the bot is in the right path:

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

where $\emptyset$ is the terminal point reached after the bot chooses to search. Accordingly, $\langle \emptyset, 1 \rangle$ is the state representing that the bot stops and the stopping node contains the item, likewise $\langle \emptyset, 0 \rangle$ depicts the opposite case.

Actions

We first discuss the *guidance function* concept, which is a core component in our model. Then, we specify the action set.

Guidance Function   A guidance function $\eta : 2^X \times 2^X \times X \to [0, 1]$ provides the search bot with evidence as to the correct path on which to search for an item $x \in X$. Specifically, $\eta(c, c', x)$ represents the bot's estimate that the target x is contained in a cluster c', given that it is in cluster c, for $(c, c') \in \mathcal{H}$. We have

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

where $\mathcal{C}(c)$ returns the children of c. The η function summarises the information (prior domain knowledge) available in the hierarchy which will guide the bot's behaviour. It is represented as a discrete probability density function. In particular, we write for any parent-child pair $(c, c') \in \mathcal{H}$ that,

$$\eta(c, c', x) = p(x \in c' \mid x \in c) \triangleq \frac{\exp\{\mathrm{sim}(x, c')/\delta\}}{\sum_{c'' \in \mathcal{C}(c)} \exp\{\mathrm{sim}(x, c'')/\delta\}} \tag{2}$$

where $\mathrm{sim}(x, c)$ is the similarity function between the item x and the cluster c.[1] Additionally, $p(x \in c \mid x \notin c)$ is always 0 as it is impossible to retrieve the item in a node whose parent does not contain it. Leaving the choice of similarity function open adds flexibility to the measure for adjusting to diverse datasets and applications. The η function should return a higher probability score for the node c' that is "closest" to x among the siblings. The parameter δ is a *temperature* parameter in the Boltzmann function. We disclose more insights about this function in the supplemental materials about how to control the function as the path goes deeper.

Action Set   As our measure should represent the quality of the hierarchy for a typical searcher, we capture the possible search behaviours by modelling that the bot searches stochastically. Nevertheless, the measure focuses on rational search behaviour, rather than on fully random search. The proposed framework allows us to capture this rationality by exposing part of the search behaviour to rational decision making. Hence, we impose randomness in the manner in which children are selected by a searcher and expose only the decision of whether or not to explore 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, the bot moves to a child c' with probability $\eta(c, c')$, i.e. based on the probability of the item being stored at the target location. With the above rationale, the action set contains only two actions: descend a_d and search a_s, corresponding to the choices of descending the hierarchy to another level, or stopping and searching for the target at the current node.

---

1. In the rest of the paper, we omit x in the guidance function, since the discussions about the guidance function will always concentrate on one specific item.

Transitions

The transition is fixed and the bot can fully observe the outcome, when the bot searches. That is, the bot knows whether it is in $\langle \emptyset, 0 \rangle$ or $\langle \emptyset, 1 \rangle$ reaching the state $\emptyset$ when applying action a_s.

For the descent action, the bot estimates the transition i.e. it computes $\hat{T}$ such that $\hat{T}(s, a, s') = \hat{p}(s' \mid s, a)$. In regular MDP and POMDP problems, the unknown transitions are handled by the 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; Ng et al., 2012). In our task, we have to design for bot's transition distribution rather than to learn them from the data. Thus, the "domain knowledge" should play the role for driving 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 a_d for the various cases. For $c' \in \mathcal{C}(c)$, $s = \langle c, g_c \rangle$ and $s' = \langle c', g_{c'} \rangle$,

$$\hat{p}(s' \mid s, a_d) = p\left(\langle c', g_{c'} \rangle \mid \langle c, g_c \rangle, a_d\right) = p\left(c' \mid c, a_d\right) \hat{p}\left(g_{c'} \mid g_c\right) = \eta(c, c')\hat{p}\left(g_{c'} \mid g_c\right)$$

where the last step follows from the stochastic descent process described in the previous section. The η function is also used for the estimate $\hat{p}(g_{c'} \mid g_c)$ such that, $\hat{p}(g_{c'} = 1 \mid g_c = 1) = \eta(c, c')$. Overall, for a_d, we get, for all $c \in \mathcal{H}$, $c' \in \mathcal{C}(c)$:

$$\begin{aligned}
p(\langle c', 1 \rangle \mid \langle c, 1 \rangle, a_d) &= \eta(c, c')\hat{p}(g_{c'} = 1 \mid g_c = 1) = \eta(c, c')^2 \\
p(\langle c', 0 \rangle \mid \langle c, 1 \rangle, a_d) &= \eta(c, c')\hat{p}(g_{c'} = 0 \mid g_c = 1) = \eta(c, c')(1 - \eta(c, c')) \\
p(\langle c', 1 \rangle \mid \langle c, 0 \rangle, a_d) &= \eta(c, c')\hat{p}(g_{c'} = 1 \mid g_c = 0) = 0 \\
p(\langle c', 0 \rangle \mid \langle c, 0 \rangle, a_d) &= \eta(c, c')\hat{p}(g_{c'} = 0 \mid g_c = 0) = \eta(c, c').
\end{aligned} \tag{3}$$

Rewards

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

$$R(s, a, s') = \begin{cases} 0 & a = a_d \\ r(c) & s' = \langle \emptyset, 1 \rangle \\ -1 & s' = \langle \emptyset, 0 \rangle \end{cases} \quad and \quad r(c) = 1 - \left(e^{|c|/N} - 1\right)/(e - 1)$$

where c is the location encapsulated in s. We assign -1 to searching at a wrong node. One can customise $r(\cdot)$ as long as it is monotonically decreasing with the size of the input cluster to be searched. The selected $r(c)$ approximates the ease of searching within a set of items where $|c|$ is the number of items at c.

Observations and Observation Probabilities

The observations after the bot descends are its own location in the hierarchy, the children, and some abstract summaries from the children, etc. This information is fixed and unique for

each location that the bot arrives at. Hence one can integrate it into one indexed observation that directly affects our belief update function. Recall that $Z(s, a, o) = p(o \mid s, a)$. It is trivial to see $p(o \mid \langle c', g_{c'} \rangle, a_d) = \mathbb{1}\{o = c'\}$, i.e. the bot can detect its exact location. After performing a_s, the observations are whether or not the target is found in the current node.

Belief Update

As states are not fully observable, the bot maintains its belief state b as a probability function at any specific time. Belief updates can be written 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,

$$b'(s') = p(s' \mid a, o, b) = \frac{p(o \mid s', a) \sum_s p(s' \mid s, a)b(s)}{p(o \mid a, b)} . \tag{4}$$

Considering that a_d is performed at c, then for each possible child $c' \in \mathcal{C}(c)$, such that the new state is $s' = \langle c', 0 \rangle$ or $s' = \langle c', 1 \rangle$, we have

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

To be more specific,

$$b'\left(\langle c', 0 \rangle\right) \propto \mathbb{1}\{o = c'\} \sum_s p(\langle c', 0 \rangle \mid s, a_d)b(s)$$

$$= p(\langle c', 0 \rangle \mid \langle c, 1 \rangle)b(\langle c, 1 \rangle) + p(\langle c', 0 \rangle \mid \langle c, 0 \rangle)b(\langle c, 0 \rangle)$$

$$= \eta(c, c')[1 - \eta(c, c')b(\langle c, 1 \rangle)] . \tag{5}$$

$$b'\left(\langle c', 1 \rangle\right) \propto \eta(c, c')^2 b\left(\langle c, 1 \rangle\right) , \tag{6}$$

where Eq. (6) is obtained in a similar way to Eq. (5). As discussed earlier, for other states, s'', not related to c', $b'(s'') = 0$. To compute the normalising constant $p(o \mid b, a_d)$, when $a = a_d$, we have that

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

which gives a final belief update rule of

$$b'\left(\langle c', 1 \rangle\right) = \eta(c, c')b\left(\langle c, 1 \rangle\right) \qquad \text{and} \qquad b'\left(\langle c', 0 \rangle\right) = 1 - \eta(c, c')b\left(\langle c, 1 \rangle\right) . \tag{7}$$

Note that $p(o \mid b, a_s) = 1$ 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 sub-scripting 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 $b(\langle c_0, 1 \rangle) = 1$.

By induction, we arrive at a simple expression for the belief state when the search has reached node c_T at time T where $T > 0$, $b_T(\langle c_T, 1 \rangle) = \prod_{t=1}^{T} \eta(c_{t-1}, c_t)$ and $b_T(\langle c_T, 0 \rangle) = 1 - \prod_{t=1}^{T} \eta(c_{t-1}, c_t)$ while $b_T(s) = 0$ for all other states s. That is, once the bot 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, $1 - b_T(\langle c, 1 \rangle)$. Furthermore, since there is no backtracking, there is exactly one belief state associated with each node $c \in \mathcal{H}$, which, if $\{c_0, \ldots, c_T = c\}$ is the unique path, 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 $b_{c'} = \eta(c, c')b_c$.

Value Function

The most commonly used objective that drives policy selection in an MDP is the value V (discounted long term reward), s.t. $V \triangleq \sum_{t=0} \gamma^t R_t$ where R_t is the reward obtained at time stamp t. Let use denote a policy by π. Solving a POMDP aims to maximise the value function, $V^\pi(b)$, corresponding to the cumulative expected reward *over the beliefs* (Singh et al., 1994). In particular, $R_B(b, a) = \sum_s b(s) \sum_{s'} p(s'|s, a) R(s, a, s')$. It follows that 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:

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

$$\text{where} \qquad Q(b, a) = R_B(b, a) + \gamma \sum_o p(o \mid b, a) V^*(\tau(b, a, o)) \,.$$

This 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 $\tau(b, a, o)$.

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

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

It is natural to examine the value V^π achieved by the policy π learned over the POMDP in the underlying MDP (Singh et al., 1994), that arises from the POMDP when the MDP states are fully observable. V^π may be interpreted as the value that an *oracle* that knows the target's location would assign to the bot's policy. We base the HQS measure on this oracle value, noting that the bot can only attempt to maximise the belief value $V^\pi(b)$.

3.1. Example of the POMDP

Due to the space limit, we provide a walk-through example in the supplemental document (Appendix A.) which shows how a POMDP will be constructed given a simple hierarchy.

3.2. Hierarchy Quality for Search

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. In particular, let T denote the depth at which the bot chooses a_s and let $\{c_0, c_1, \ldots, c_T\}$ now denote the *correct* path containing the target item x to depth T. Then

$$V_x = r(c_T) \prod_{t=1}^{T} \eta(c_{t-1}, c_t) + (-1) \times \left(1 - \prod_{t=1}^{T} \eta(c_{t-1}, c_t) \right) = (r(c_T) + 1) \left(\prod_{t=1}^{T} \eta(c_{t-1}, c_t) \right) - 1 \,.$$

Intuitively, the bot moves randomly according to $\eta(\cdot)$, and when it stops at depth T, the probability of obtaining a positive reward is simply the probability that the guidance function led it along the right path. Given $0 \leq r(\cdot) \leq 1$, we obtain $-1 \leq V_x \leq 1$ 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 determining the oracle value.

Eventually, we define HQS as follows.

> **Definition 1 (HQS)** *The* HQS *is a function over the set of data $X = \{x_n\}_{n=1}^{N}$ and the hierarchy $\mathcal{H}$, such that* $\mathrm{HQS}(X, \mathcal{H}) \triangleq \frac{1}{N} \sum_{n=1}^{N} V_{x_n}^{\pi}(\mathcal{H})$ *where $V_{x_n}^{\pi}(\mathcal{H})$ is the long term expected reward for the bot to search item x_n in $\mathcal{H}$, obtained by taking policy π.*

4. Solving the POMDP

One can categorise POMDP solvers to off-line and on-line policies. Off-line methods either learn the whole environment prior to determining the policy or learn it through RL. On-line planners solve the POMDP in a real-time decision making context, focusing on the information in the current states (Ross et al., 2008). We appeal to on-line planners as they better simulate the situation of a bot making decisions as the search proceeds. Also, off-line methods would make the process less tractable and thus unsuitable for the evaluation task.

One notable on-line Monte-Carlo based planner, POMCP (Silver and Veness, 2010), is able to earn a good underlying MDP value regardless of the quality of the belief states. This *contradicts* 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. Hence, the bot is coded with the Real-Time Belief Space Search (RTBSS) planner to seek the policy (Paquet et al., 2005a,b; Ross et al., 2008). In RTBSS, 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. This corresponds to *two* look-aheads for an on-line planner, where the bot can compare stopping and searching at the current node, with that searching at the next layer after one descent.

4.1. RTBSS

We leave the details of the original RTBSS procedure as presented in (Ross et al., 2008) to the supplemental document. 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. It selects the policy that maximises this lower-bound.

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

to be 0. In particular,

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

$$\hat{Q}(b_t, a_d) \triangleq \sum_{c' \in \mathcal{C}(c_t)} p\left(o = c' \mid b_t, a_d\right) Q(\tau(b_t, a_d, o), a_s)$$

$$= \sum_{c' \in \mathcal{C}(c_t)} \eta\left(c_t, c'\right) \left(b_{c'}(r(c') + 1) - 1\right).$$

Algorithm 1 SIMPLIFIED RTBSS POLICY specified for HQS

1: Initialise b such that $b(\langle c_0, 1 \rangle) \leftarrow 1$
2: **while** ISNOTLEAFBELIEFSTATE(b) and $\pi(b, a_s) = 0$ **do**
3:     Compute $Q(b, a_s)$ and $\hat{Q}(b, a_d)$
4:     $a^* \leftarrow$ **if** $\hat{Q}(b, a_d) > Q(b, a_s)$ **then** a_d **else** a_s
5:     $\pi(b, a^*) \leftarrow 1$
6:     $b \leftarrow \tau(b, a, o)$ // It focuses on the o in the right path only
7: $\pi(b, a_s) \leftarrow 1$ // The bot can only choose to search at a leaf node

---

Our simplified algorithm (in Algorithm 1) provides a belief-based policy for descending through the hierarchy, which at any given node determines whether to descend further or to stop. The simplified algorithm helps HQS become less computationally demanding and achieves a polynomial time complexity.

Proposition 2 *The worst case complexity of computing* HQS *for N items is $O(N^3 \mathcal{F}(\text{sim}))$ where $\mathcal{F}(\text{sim})$ is the complexity of the similarity function.*

The sketch of the proof can be found in the supplemental document.

5. Experimental Study

We show how HQS works through a case study using a small dataset and compare it with an existing approach. Then, we will discuss on scaling the HQS.

5.1. Case Study on Amazon

This study uses the textual information for the items from the Amazon[2] (McAuley et al., 2015), and bases the guidance function on a TF-IDF similarity score. We select commonly seen daily life products and manually construct a number of hierarchies, varying their quality. To ensure that the hierarchies can be intuitively assessed by inspection, we restrict to just 12 items. This approach can help us quickly assess HQS procedure by assessing if it orders the hierarchies as expected.

The Selected Items Some TF-IDF details about the items are displayed in Table S1 and S2 (in the supplemental document). We refer to items by their indices as specified in that table. More details are also revealed in the supplemental document.
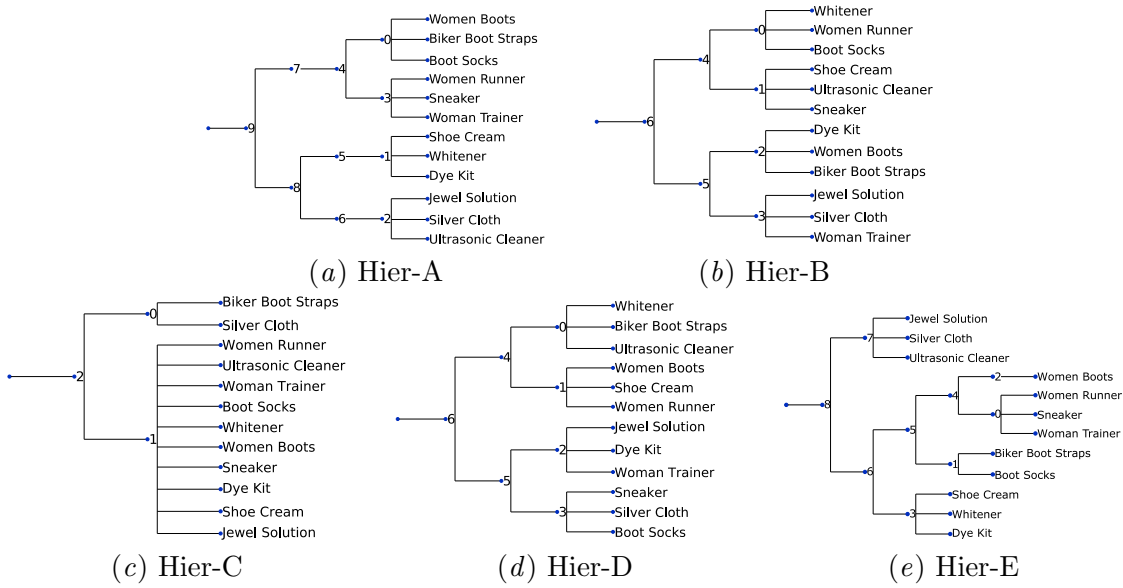
---

2. http://jmcauley.ucsd.edu/data/amazon

Figure 2: Five different hierarchies

Since the data is represented by TF-IDF vectors, which we write as $\boldsymbol{v}_x$ for item x, we define the similarity function by

$$\text{sim}(x, c) = \begin{cases} 1 & c = \{x\} \\ \frac{1}{|c\backslash\{x\}|} \sum_{x' \in c\backslash\{x\}} \cos(\boldsymbol{v}_x, \boldsymbol{v}_{x'}) & \text{otherwise} \end{cases}.$$

This similarity is in spirit similar to the average link of Agglomerative Clustering (AC) (Day and Edelsbrunner, 1984). It is important to exclude x itself from computing the similarities over a non-singleton cluster, since the maximum cosine similarity between any (x, y) pair is only around 0.11 in the selected items, and hence $\cos(\boldsymbol{v}_x, \boldsymbol{v}_x)$ would dominate the similarity score. For the guidance function in our experiments, we set $\delta = 0.01$ which helps us increase the weight of the best cluster and ensure that the cluster with largest similarities to the item, has a dominant guidance value.

Tested Hierarchies We analyse the HQS scores for five hierarchies in Fig. 2, where Hier-A is the ground-truth obtained from the labels provided in the `Amazon` dataset. Hier-B and Hier-C are randomly generated hierarchies. Hier-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 cluster 0 of Hier-D are all in separate clusters according to the `Amazon` organisation. Finally, Hier-E is constructed on purpose to show that a good hierarchy is achievable even different from the ground-truth.

5.1.1. Walk-through Analysis

The second column of Table 1 shows the HQS results of the five different hierarchies. The ground truth hierarchy, A, provides the highest HQS score (0.5715), followed by a relatively high score for Hier-E as expected. Hier-D provides the worst HQS score (-0.8477). In this

Table 1: HQS results of the five hierarchies (A–E in Fig. 2) and the value V for the twelve items (indexed from 0–11) in each hierarchy.

	HQS	0	1	2	3	4	5	6	7	8	9	10	11
A	.572	.620	.747	.827	.811	.600	.518	.793	-.679	.562	.432	.812	.817
B	-.280	.0	-.991	-.989	.863	-.955	-.736	.760	.659	-.995	.0	.749	.0
C	-.310	-.752	-.999	.0	.0	.0	.889	.857	-.933	.879	.0	-.965	-.979
D	-.848	-.668	-.740	-.708	-.996	-.652	-1.00	-.791	-.607	-.995	-.925	-.990	-.901
E	.327	.611	-.797	.673	.0	.533	.0	.834	.0	.423	.0	.835	.808

example, the HQS scoring scheme agrees with our intuition and we believe this ranking is reasonable. Table 1 also displays the values for each item obtained by RTBSS in the five hierarchies. Next, we specify how the policy propagates step-by-step for those hierarchies.

Hier-A   This hierarchy is generated given the ground truth 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., as shown in Table S1 and S2.

For item x_0, "Women Boots", staying and searching at the root earns an estimate $Q(b_0, a_s) = 0$. Now, the similarities x_0 with the child nodes are $\text{sim}(x_0, c_7) = 0.061$ and $\text{sim}(x_0, c_8) = 0.0277$. It follows that $\eta(c_9, c_7) = 0.9987$ and $\eta(c_9, c_8) = 0.0013$. Clearly, both nodes have 6 items and so, the rewards for staying at both nodes are equal, 0.6226. It follows that $\hat{Q}(b_0, a_d) = \sum_{o \in \{c_7, c_8\}} p(o \mid b, a_d) Q(\tau(b_0, a_d o)) = 0.6203$. Since $Q(b_0, a_s) < \hat{Q}(b_0, a_d)$, the bot performs action a_d, and it randomly moves to the right node with probability 0.9986. From c_7 to c_4, it is a straight move as the estimates will be equal and the bot should explore a better chance for a higher value. Let us review that cluster c_0 refers to "Boots" and c_3 refers to "Fashion Sneakers" under the category chain "All $\rightarrow$ Clothing, Shoes & Jewelry $\rightarrow$ Women" (c_9, c_7, c_4). It shows $\text{sim}(x_0, c_0) = 0.06, \text{sim}(x_0, c_3) = 0.061$ and we obtain $\eta(c_4, c_0) = 0.4158$ and $\eta(c_4, c_3) = 0.5829$. This suggests that x_0 more strongly belongs to cluster c_3 than to c_0, the cluster it is assigned to. Regarding c_3, its items "Women Runner", "Women Sneaker" and "Women Trainer" are highly close to "Women Boots" in some sense. Upon computing the Q-values, the bot decides to stop at node c_4, since it is not confident that another descent step will lead to greater reward. Finally, the oracle value V_{x_0} is 0.6203, 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 item. This suggests that our measurement is rather reasonable in this case.

Another interesting example is x_7, "Dye Kit". It starts navigating wrong at the first level. The similarities $\text{sim}(x_7, c_7)$ and $\text{sim}(x_7, c_8)$ return 0.0367 and 0.0297 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 jewellery). As it stands, the bot will be guided to the wrong node and hence receives a negative reward. It could well be the case that the situation would be improved with more data items.

The bot is not confident to descend to the bottom of the hierarchy for every case which is reflected in the associated score. In summary, Hier-A is sufficiently good to help the bot descend and acquires a good HQS of 0.5715.

Hier-B In this case, the final HQS is -0.2801. Table 1 demonstrates that there are 6 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 it prefers searching at the root. Interestingly, a random hierarchy does not necessarily achieve an absolute HQS of zero. The HQS measurement is apparently not linear.

Hier-C Its HQS is -0.3097. Similarly to Hier-B, there are 6 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 0.8892 and 0.8789 respectively. The similarities are diluted compared with the small node, since the sibling clusters are too diversified. However, HQS unveils the fact that the hierarchy is poor for item retrieval. Although we compute the value for the bot searching for each item, HQS considers the hierarchy as a whole. Thus, even a poor hierarchy might enable efficient search for a few items, while there are many more poorly scoring items.

Hier-D For this hierarchy, the overall HQS score of -0.8477, reflects the fact that this is a hierarchy within which the bot has difficulty in searching. HQS successfully reveals this by returning a score close to the minimum, the worst performing one among the exhibited hierarchies. This implies that HQS correctly downgrades a random hierarchy.

Hier-E Finally, we construct a "good" hierarchy (Fig. 2(e)) while its style is very different to the ground-truth. It separates the items about jewellery and the items associated with shoes at the first level. Overall, there is only one item, "Shoe Cream" which receives a negative value -0.7972. At the first decision point for this item, $\eta(c_8, c_7)$ is 0.8894 which leads the bot to the wrong node with a dominant probability. The item's textual representation tends to be better associated with the cleaning toolkit. 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. It implies that the HQS acknowledges hierarchies with different structures to the ground-truth as long as the hierarchy is efficient for search.

5.1.2. COMPARISON TO EXISTING APPROACH

We show that HQS outperforms HAI (Johnson et al., 2013) on our example hierarchy.

HAI defines a distance function $d(x, y, \mathcal{H}) = |c_{x,y}|/N$ where $c_{x,y}$ is the closest ancestor node of items x and y when such an ancestor is not a leaf node; otherwise the distance returns 0. The HAI metric requires a ground-truth hierarchy and we denote it by $\mathcal{H}_{gt}$. Therefore, HAI is formulated as

$$\text{HAI}(\mathcal{H}, \mathcal{H}_{gt}) = 1 - \frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} |d(x_i, x_j, \mathcal{H}) - d(x_i, x_j, \mathcal{H}_{gt})|.$$

We conduct the evaluation over the hierarchies B–E with Hier-A as ground truth. Their HAI scores are $0.64, 0.46, 0.58, 0.85$ respectively. Both HQS and HAI rank Hier-E and Hier-B as first and second, excluding the ground-truth. However, the HAI score of 0.58 for Hier-D, which was generated to serve as the worst example, is better than the score of Hier-C (which should not happen) and also numerically close to that of Hier-B.

5.2. Discussion of Scaling HQS

HQS can easily be run in parallel, since the POMDPs for the items are completely independent. In this section, we demonstrate that sampling can further help with the runtime efficiency–with sufficient samples, one can achieve an approximated HQS close to the true HQS. With the number of items sampled, the discrepancy between the sampled HQS and the true HQS decreases exponentially and the runtime speeds up linearly.

Again, we use `Amazon` but keep $37,826$ samples and 100 dimensions with PCA. The hierarchy is simply generated using AC and contains $75,651$ nodes with a maximum height 65. The prototype of HQS is implemented with Python.[3] Tree is a relatively memory-intense data structure and the data size is also a consideration. Hence, RAM prevents us from examining larger (in another order of magnitude) trees.

Let us now down-sample the items for computing the oracle values. Fig. 3 displays the empirical densities of the HQS for the whole set of `Amazon` items, fitted using `StatsPlots`[4]. The black dashed line is the true HQS with all items computed, while the others have sampled the corresponding percentages of items. It is apparent that the density curve with more samples gets closer to the one of the entire set of data. Fig. 4 exhibits the L_1 errors between the true HQS and the sampled HQS, which are divided by the absolute true HQS. It is clear that the errors decrease drastically as more items are sampled. After 30% sampling, the errors are less than 5% of the absolute value of the true HQS. As expected, the HQS of the cases with fewer samples are less accurate. Finally, the runtime is plotted in Fig. 5, and is shown to follow approximately a linear relation.
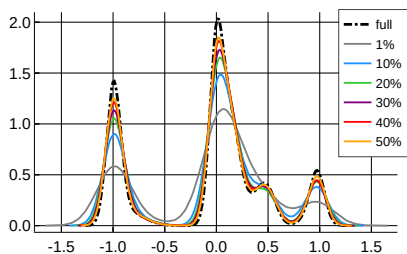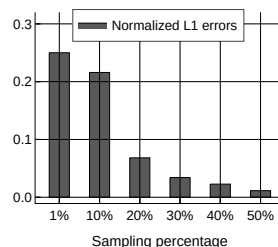

Figure 3: Empirical density
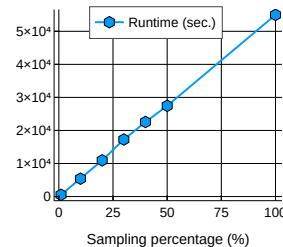
Figure 4: Normalised L_1 errors

Figure 5: Runtime of the sampled cases

Overall, HQS can benefit from that it is completely parallelisable. In addition, sampling techniques can also help improve the efficiency. We would like to emphasise that the runtime efficiency is apparently improvable with better programming code and machines.

6. Conclusion

We have proposed HQS, an approach for assessing the quality of hierarchical clusters which needs no ground-truth information. It employs POMDP for modelling the uncertainty and

---

3. The experiments are conducted in a computer equipped with AMD Ryzen 7 1700X@2.2GHz 8-Core Processor and 64GB of RAM, using 16 threads. See https://github.com/weipeng/pyhqs for the code.

4. https://github.com/JuliaPlots/StatsPlots.jl

the decision making for searchers with regard to search and navigation in a hierarchy, and extends the model to measure the quality of the hierarchies. The experiments show that HQS can order hierarchies reasonably based on their qualities without the ground-truth, and perform better than a state-of-the-art approach which requires the ground-truth hierarchy. For future work, we would like to continue exploring better models and policies, and analyse the properties of the measure theoretically.

Acknowledgments

References

Maria-Florina Balcan, Yingyu Liang, and Pramod Gupta. Robust Hierarchical Clustering. *Journal of Machine Learning Research*, 15(1):3831–3871, 2014.

Craig Boutilier and David Poole. Computing Optimal Policies for Partially Observable Decision Processes Using Compact Representations. In *AAAI*, pages 1168–1175, 1996.

Anthony R. Cassandra, Leslie Pack Kaelbling, and Michael L. Littman. Acting Optimally in Partially Observable Stochastic Domains. In *AAAI*, volume 94, pages 1023–1028, 1994.

Juan M. Cigarran, Anselmo Peñas, Julio Gonzalo, and Felisa Verdejo. Evaluating Hierarchical Clustering of Search Results. In *12th SPIRE 2005*, pages 49–54, 2005.

William HE Day and Herbert Edelsbrunner. Efficient Algorithms for Agglomerative Hierarchical Clustering Methods. *Journal of Classification*, 1(1):7–24, 1984.

Michael O'Gordon Duff. *Optimal Learning: Computational Procedures for Bayes-adaptive Markov Decision Processes*. PhD thesis, Univ. of Massachusetts at Amherst, 2002.

Katherine Heller and Zoubin Ghahramani. Bayesian Hierarchical Clustering. In *ICML*, pages 297–304, 2005.

Eugene Ie, Vihan Jain, Jing Wang, Sanmit Narvekar, Ritesh Agarwal, Rui Wu, Heng-Tze Cheng, Morgane Lustman, Vince Gatto, Paul Covington, Jim McFadden, Tushar Chandra, and Craig Boutilier. Reinforcement Learning for Slate-based Recommender Systems: A Tractable Decomposition and Practical Methodology. Technical report, 2019. arXiv preprint arXiv:1905.12767 [cs.LG].

David M. Johnson, Caiming Xiong, Jing Gao, and Jason J. Corso. Comprehensive Cross-Hierarchy Cluster Agreement Evaluation. In *AAAI (Late-Breaking Developments)*, 2013.

Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. Planning and Acting in Partially Observable Stochastic Domains. *Artificial Intelligence*, 101:99–134, 1998.

Ari Kobren, Nicholas Monath, Akshay Krishnamurthy, and Andrew McCallum. A Hierarchical Algorithm for Extreme Clustering. In *SIGKDD*, pages 255–264. ACM, 2017.

Yanchi Liu, Zhongmou Li, Hui Xiong, Xuedong Gao, Junjie Wu, and Sen Wu. Understanding and Enhancement of Internal Clustering Validation Measures. *IEEE Transactions on Cybernetics*, 43(3):982–994, 2013. ISSN 2168-2267.

Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. Image-based Recommendations on Styles and Substitutes. In *SIGIR*, pages 43–52. ACM, 2015.

Nicolas Meuleau, Kee-Eung Kim, Leslie Pack Kaelbling, and Anthony R Cassandra. Solving POMDPs by Searching the Space of Finite Policies. In *UAI*, pages 417–426, 1999.

Raul Moreno, Wěipéng Huáng, Arjumand Younus, Michael O'Mahony, and Neil J Hurley. Evaluation of Hierarchical Clustering via Markov Decision Processes for Efficient Navigation and Search. In *CLEF 2017, Proceedings*, pages 125–131. Springer, 2017.

Brenda Ng, Kofi Boakye, Carol Meyers, and Andrew Wang. Bayes-Adaptive Interactive POMDPs. In *AAAI*, 2012.

Sébastien Paquet, Ludovic Tobin, and Brahim Chaib-Draa. An Online POMDP Algorithm for Complex Multiagent Environments. In *AAMAS*, pages 970–977. ACM, 2005a.

Sébastien Paquet, Ludovic Tobin, and Brahim Chaib-Draa. Real-Time Decision Making for Large POMDPs. In *Conference of the Canadian Society for Computational Studies of Intelligence*, pages 450–455. Springer, 2005b.

Stéphane Ross, Joelle Pineau, Sébastien Paquet, and Brahim Chaib-Draa. Online Planning Algorithms for POMDPs. *Journal of Artificial Intelligence Research*, 32:663–704, 2008.

Guy Shani, David Heckerman, and Ronen I. Brafman. An MDP-based Recommender System. *Journal of Machine Learning Research*, 6(Sep):1265–1295, 2005.

David Silver and Joel Veness. Monte-Carlo Planning in large POMDPs. In *Advances in neural information processing systems*, pages 2164–2172, 2010.

Satinder P. Singh, Tommi Jaakkola, and Michael I. Jordan. Learning without State-Estimation in Partially Observable Markovian Decision Processes. In *Machine Learning Proceedings 1994*, pages 284–292. Elsevier, 1994.

Michael Steinbach, George Karypis, and Vipin Kumar. A Comparison of Document Clustering Techniques. In *TextMining Workshop at KDD2000*, 2000.

Guanjie Zheng, Fuzheng Zhang, Zihan Zheng, Yang Xiang, Nicholas Jing Yuan, Xing Xie, and Zhenhui Li. DRN: A Deep Reinforcement Learning Framework for News Recommendation. In *WWW*, pages 167–176, 2018.