Assume that we are at state (E, H). Now, we discuss the algorithms for choosing e for both Optimal Decision Tree problem and Multiple Intent Re-Ranking Problem.

1 Optimal Decision Tree

1.1 Greedy Algorithm

This algorithm is based on the classic greedy objective discussed in [8], [6], [1], [5], [7]. At each step, the greedy algorithm will pick the element which makes the decision tree as balanced as possible. So we choose the element e such that:

$$e = argmax_{e \in U \setminus E} \min(|\{i | i \in H, r_i(e) = 1\}|, |\{i | i \in H, r_i(e) = 0\}|)$$

Then we update E and H as follows:

$$E \leftarrow E \cup \{e\}$$

$$H \leftarrow H \setminus \{i | r_i(e) = 0\}$$

1.2 Static Algorithm

The static algorithm that we are using is from a paper by Azar and Gamzu paper [3]. By the term "static" we are referring to this fact that the algorithm is not dependent on the elements' feedback. Let T_e be the set of scenarios that have feedback 1 on test e. Now, we define $T_e(i) = T_e$ if and only if $r_i(e) = 0$ (or equivalently $i \notin T_e$) and $T_e(i) = [m] \setminus T_e$ if and only if $r_i(e) = 1$ (or equivalently $i \in T_e$) . Then we define $f_i(S) = |\bigcup_{e \in S} T_i(e)| \cdot \frac{1}{m-1}$. In static algorithm, we choose element e such that:

$$e = argmax_{e \in U \setminus E} \sum_{i \in H} p_i \cdot \frac{f_i(e \cup E) - f_i(E)}{1 - f_i(E)}$$

Then we update E and H as follows:

$$E \leftarrow E \cup \{e\}$$

$$H \leftarrow H \setminus \{i | f_i(E) = 1\}$$

1.3 Ad-Static Algorithm

This is a modified version of the static algorithm discussed above. Basically everything is the same except for updating the set H:

$$H \leftarrow H \setminus \{i | r_i(e) = 0\}$$

1.4 ML-Based Algorithm

This is a "machine learning" algorithm that for a parameter K, uses k-Means [2] to *a priori* partition U into K clusters. Each cluster $c_j, j \in [1, K]$ is initially given a weight $w_{c_j} = 1$. To choose the next element $e \in U \setminus E$, a cluster $j \in [1, K]$ is first chosen by sampling non-uniformly according to w_{c_j}. Next, an element $e \in c_j$ is chosen uniformly at random. If $r_{i^*}(e) = 1$, the c_j is rewarded by setting $w_{c_j} = 2w_{c_j}$, else c_j is penalized by setting $w_{c_j} = 0.5w_{c_j}$. Again based on e's feedback we update E and H:

$$E \leftarrow E \cup \{e\}$$

$$H \leftarrow H \setminus \{i | r_i(e) = 0\}$$

1.5 Adaptive Submodular Ranking Algorithm

Lets define $L_e(H) = argmin(|\{i | i \in H, r_i(e) = 1\}|, |\{i | i \in H, r_i(e) = 0\}|)$, which means that $L_e(H)$ corresponds to the lighter branch of the algorithm for a test e. More specifically for any test e and set of compatible scenarios H if test e has outcome 1 for $H_1 \subseteq H$ and 0 for $H \setminus H_1$, then $L_e(H)$ will be equal to H_1 if and only if $|H_1| < |H|/2$ and will be equal to $H \setminus H_1$ otherwise. f_is have the same definition as before. Now, our algorithm at any state $(E.H)$ choose element e such that:

$$e = argmax_{e \in U \setminus E} \left(\sum_{j \in L_e(H)} p_j + \sum_{i \in H} p_i \cdot \frac{f_i(e \cup E) - f_i(E)}{1 - f_i(E)} \right)$$

2 Multiple Intents Re-Ranking

2.1 Greedy Algorithm

At each step, the greedy algorithm will pick the element with the maximum probability of having feedback 1. In other words, we choose the element e such that:

$$e = argmax_{e \in U \setminus E} \sum_{i \in H : r_i(e)=1} p_i$$

Then we update E and H as follows:

$$E \leftarrow E \cup \{e\}$$

$$H \leftarrow H \setminus \{i | r_i(e) = 0\}$$

2.2 Static Algorithm

We are going to use the algorithm in [4] for the general case. Let S_i be the set of relevant results for user (scenario) i.

```
while U \ E ≠ ∅ do
    for i ∈ [m] do
        if |S_i ∩ E| < K_i then
            ρ(i) = 1 / (K_i − |S_i ∩ E|)
        end
        else
            ρ(i) = 0
        end
    end
    e ← argmax_{e∈U\E} Σ_{i:e∈S_i} ρ(i)
    E ← E ∪ {e}
end
```

while $U \setminus E \neq \emptyset$ **do**

- **for** $i \in [m]$ **do**
  - **if** $|S_i \cap E| < K_i$ **then**
    - $\rho(i) = \frac{1}{K_i - |S_i \cap E|}$
  - **end**
  - **else**
    - $\rho(i) = 0$
  - **end**
- **end**
- $e \leftarrow argmax_{e \in U \setminus E} \sum_{i: e \in S_i} \rho(i)$
- $E \leftarrow E \cup \{e\}$

end

2.3 Ad-Static Algorithm

This is a modified version of the static algorithm discussed above. Basically everything is the same except for updating the set H:

$$H \leftarrow H \setminus \{i | r_i(e) = 0\}$$

2.4 ML-Based Algorithm

This is exactly the same as ML-based algorithm for Optimal Decision Tree problem.

2.5 Adaptive Submodular Ranking Algorithm

Lets $L_e(H)$ have the same definition as before (i.e. $L_e(H) = argmin(|\{i|i \in H, r_i(e) = 1\}|, |\{i|i \in H, r_i(e) = 0\}|))$. We define $f_i(S) = \min(|S \cap S_i|, K_i)/K_i$. Now, at any state $(E.H)$ choose element e such that:

$$e = argmax_{e \in U \setminus E} \left(\sum_{j \in L_e(H)} p_j \ + \ \sum_{i \in H} p_i \cdot \frac{f_i(e \cup E) - f_i(E)}{1 - f_i(E)} \right)$$

References

[1] M. Adler and B. Heeringa. Approximating optimal binary decision trees. *Algorithmica*, 62(3-4):1112–1121, 2012.

[2] David Arthur and Sergei Vassilvitskii. k-means++: The advantages of careful seeding. In *SODA*, pages 1027–1035, 2007.

[3] Y. Azar and I. Gamzu. Ranking with submodular valuations. In *SODA*, pages 1070–1079, 2011.

[4] Y. Azar, I. Gamzu, and X. Yin. Multiple intents re-ranking. In *STOC*, pages 669–678, 2009.

[5] V. T. Chakaravarthy, V. Pandit, S. Roy, P. Awasthi, and M. K. Mohania. Decision trees for entity identification: Approximation algorithms and hardness results. *ACM Transactions on Algorithms*, 7(2):15, 2011.

[6] S. Dasgupta. Analysis of a greedy active learning strategy. In *NIPS*, 2004.

[7] A. Guillory and J. Bilmes. Average-Case Active Learning with Costs. In *Algorithmic Learning Theory*, pages 141–155. Springer Berlin / Heidelberg, 2009.

[8] S. R. Kosaraju, T. M. Przytycka, and R. S. Borgstrom. On an Optimal Split Tree Problem. In *Proceedings of the 6th International Workshop on Algorithms and Data Structures*, pages 157–168, 1999.