**The HUC Recommender Algorithm**

**Vectors:**

**Pi = <s, c, p, e>** *(a profile for user i)*

s = <a1, …, a7>, where ai is a **social attribute** (relationship-status,age-range, gender, education, likes, political-affiliation, social status)

c = <c1, …, c5>, where ci is a **cultural attribute** (religion, current-location, home-town, timezone/time-period, language)

p = <p1, …, p4> where pi  is a **psychological attribute** (birthday, friends-birthday, movies, upcoming-events)

e = <e1, …, e5> where ei  is a **economic attribute** (currency, work-history, profession, residential-category)

**Sets:**

I = {x:x is an item/product}

Ir ={x:x ∈ I AND recommended to Pu}

U = {x: x = Pu, where Pu is a profile for user u}

O = {x: x ∈ Uo, where 0 is an old user profile}

R = {x: x ∈ U AND x is *an active* user}

N= {x: x ∈ Un, where n is a new user profile}

M = {x: x ∈ R AND **Similarity(xi, xj) ≧ n**, where n is 0.5, ∀ i, j}

*Check if user u has previously expressed interest in product i:*

x ∈ Boolean, isActive(u,i) = x, u ∈ U AND i ∈ I

**Derivative expressions:**

R ⊆ U, U = O ⋃ N and M ⊆ R

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | ComputeSimilarProfile(u, O):  INPUT: u (profile of user)  O (set of old profiles)  OUTPUT: M (set of profiles similar to profile u)  Begin  C ← **computeProfileClassSet**(u,O)  R ← {r: r ∈ C and **isActive**(r,I)};  k ← 0.5; *(Jaccard coefficient threshold for highly similar profiles)*  M ← {m: m ∈ R AND **JaccardSimilarity**(m, u) ≥ k, ∀m};  return M;  End |

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | Entropy(O)  INPUT: O(set of old profiles)  OUTPUT: Entropy Value  Begin  Entropy ← — (|O+|/|O|) Log2 (|O+|/|O|) — (|O-|/|O|) Log2 (|O-|/|O|)  # O+ number of profiles in O with positive actions  return Entropy  End |

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | computeProfileClassSet(u,O)  INPUT: u(profile of user)  O(set of old profiles)  OUTPUT: Od(set of profiles in same class as u )  Begin  T ← {x:x is a user attribute}  O ← {x: x = <v1,...,vn>, vi ∈ Dom(ai), ai ∈ T, i=1,2,...,n, n=|T|}  V[O] ← {x:x ∈ DomO(ai), ai ∈ T and ∃va[i]:x = va[i] ∈ tuple, some tuple ∈ O } # all distinct values of ai in O  ga[i] ← Entropy(O)- x ∈ V[o](|Ox|/|Oa[i]|.Entropy(Ox))  Go ← {<x,y>: y=ga[i] i=1,2,...,n, n=|T|, ∃ai:ai= x ∈ T }  Od ← {p:p ∈ O, and ∀pa[i]  ∈ p ∃ai : ga[i]=maxy(G0)}  return Od  End |

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | **isActive**(u,i)  INPUT: u(profile of user)  i(item or set of items)  OUTPUT: bool(True or False)  Begin  t 🠔 00:00;(time period)  p 🠔 u clicks item i on time t  bool 🠔[p ➝ True OR ∽p ➝ False]  ∅  ≠  return bool  End |

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | JaccardSimilarity(ui,uj)  INPUT: ui(profile of user)  uj(profile of user)  OUTPUT: k(similarity value)  Begin  k ←[(ui ⋂ uj)/ (ui ⋃ uj)] ([Mahmud Ridwan](https://www.toptal.com/resume/mahmud-ridwan),2014)  return k  End |

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | ComputeItemsActionedbySimilarProfile(M,I,u)  INPUT: M(set of profiles similar to profile u)  u(profile of user)  I(set of items actioned by M)  OUTPUT: It(set of items actioned by T)  Begin  freqSet ← **Apriori**(M,I,u)  T ← {t:(t∈ M AND (**isActive**(t,freqSet)) }  It ← {i: **isActive**(T,i), i ∈ (I U freqSet)}  return It  End |

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | Apriori(M,Im,u)  INPUT: M(set of profiles similar to profile u)  Im(set of sets of items actioned by M)  u(profile of user)  OUTPUT: F(set of items frequently actioned by M)  Begin  k ← 25%;(support threshold)  Im ← {X:X∈ℙ(I) AND isActive(M,X),∀X}  FrequentSets ← {X:Support(X,|Im|)>=k, X∈Im}  F ← {i: i ∈ X, X ∈ FrequentSets,∀X}  return F  End |

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | Support(X,|Im|)  INPUT: X(set of items)  |Im|(number of items transacted)  OUTPUT: Sratio(support ratio of X)  Begin  count=0  for all i in Im  if(i==X)  count=count+1  Sratio= (count/|Im|)\* 100  return Sratio  End |

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | ComputeRecommendations(M,I,u)  INPUT: M(set of profiles similar to profile u)  I(set of items actioned by M)  u(profile of user)  OUTPUT: Ir (items recommended to u)  Begin  wanted\_Items ← ComputeItemsActionedbySimilarProfile(M,I,u)  Ir ← I ⋃ wanted\_Items  return Ir  End |