クエリ集合における属性改良問題

此島　魁二†　　陳　漢雄†2　　古瀬　一隆†3

†筑波大学 情報学群情報メディア創成学類 〒 305-8571 茨城県つくば市天王台 1-1-1
†2 筑波大学 システム情報系 〒 305-8571 茨城県つくば市天王台 1-1-1
†3 白鴎大学 経営学部 〒323-8585 栃木県小山市駅東通り 2 丁目 2-2
E-mail:　† s1611436@u.tsukuba.ac.jp,　†2 chx@cs.tsukuba.ac.jp,　†3 furuse@fc.hakuoh.ac.jp

あらまし 近年、情報検索の技術を用いてユーザーが好みそうな商品をランキング形式で求めるという研究が活発になされてきている。また、その逆側からの検索としてある商品を好むユーザーを求めるという研究もなされてきている。それらの研究の発展として、ある商品をより多くのユーザーに好まれるように改良する戦略を求める Improvement Query という研究がなされている。本研究では、予算を与えたときにある商品集合の中からもっとも改良効果が得られる商品を求める方法を提案する。また、その高速化について検証を行う。

キーワード　属性改良問題、情報検索、情報推薦

1. はじめに

クエリによる情報検索技術の応用として属性値の改良を提案する Improvement Query(以下 IQ)[1]という研究が先行研究としてなされてきた。 IQ は Top-k query[2]や Reverse k-rank query(以下、RkR)[3][4][5]の応用である。Top-k query はユーザー側からの検索に使用されるような検索手法で、クエリとして一人のユーザーの好みを与えたときに、そのユーザーがもっとも好みそうな k 個の商品を求めるような検索である。具体的には、購入機会が少ないパソコンなどの商品については小さい k の値とし、お菓子などの幅広い商品が購入される可能性のあるような商品については大きい k の値をとることで望まれる検索結果を得ることができる。逆に RkR は商品の販売者側からの検索に利用されるような検索手法で、クエリとして商品を与えたとき、RkR ではその商品を相対的に高く評価する k 人のユーザーを求める。IQ は特に Top-k query の応用である。IQ ではクエリとして商品を与え、全ユーザーに対して Top-k query を計算する。クエリ商品の Top-k query へのヒット数をもとにクエリ商品をある条件内で改良する改良戦略を求める検索手法である。IQ の条件の調整には予算内で Top-k query のヒット数を最大限増加させる調整(Max-Hit IQ)と、与えられた Top-k query のヒット数を満たすように最小限のコストで改良を行う調整(Min-Cost IQ)がある。しかし、そのどちらも実世界での利用にそぐわない面がある。あるユーザーのランキングですでに Top-k query にヒットする商品に関しては改良を行ったとしても改良として認識されないという問題点が存在する。この問題点を解決するために Reverse Rank Improvement Query(以下、RRIQ)[6][7]問題が提起された。オブジェクトのランク上昇量に着目することで前述の問題点を解決することができる。

本研究では、RRIQ のさらなる応用として、クエリ商品が 1 つではなく複数からなる集合を与えられたとする。そのときに、商品集合の中から予算内でもっとも高い改良効果が得られる商品とその改良戦略を求める。具体的には商品の販売者がいくつか同系統の商品を販売しており、その際に予算が一定額あるとする。このとき、販売者はもっとも効果的な改良が得られる商品に予算を使いたい。そのような例について、本研究は最適な改良が得られる商品とその改良を求める。改良戦略の評価には RRIQ に基づき、クエリ商品のランク上昇量を利用する。また、クエリ商品集合にフィルタリングを用いることで高速化することを目的とする。

2. 関連研究

本研究では候補となる改良オブジェクト集合から評価の高い改良を見つけるために、RkR を利用する。定義 2 に RkR の定義を示す。

定義 2(Reverse k-rank query) 商品データ集合を P、ユーザーの好みのデータ集合を U、正の整数 k、クエリ商品 q を与えたときに *Reverse k-rank query* は次のデータ集合 Sを求める。データ集合 Sは次の条件を満たす。$S \subseteq U, |S| = k, \forall u_i \in S: \forall u_j \in (U - S): Arank(u_i, q) \leq Arank(u_j, q)$ が成り立つ。

2.1. Improvement Query

ここでは、Top-k query を例にクエリ検索がどのように行われるかの例を示し、先行研究の IQ が Top-k query を利用してどのように改良戦略を提案するのかを示す。表 1 のような商品の属性値に対する評価の集合、表 2 のようなユーザーの好みベクトルの集合があるとする。このとき、ベクトルの内積値、k=2 としたときの Top-k query は表 3 のようになる。先行研究に倣って、ベクトルの値、内積値ともに小さい方が高い評価としてい

る。

表 1　商品ベクトル集合

	cpu	メモリ	容量
p1	0.5	0.3	0.4
p2	0.3	0.3	0.4
p3	0.2	0.7	0.1

表 2　ユーザーの好みベクトル集合

	cpu	メモリ	容量
u1	0.3	0.3	0.4
u2	0.1	0.2	0.7
u3	0.5	0.4	0.1

表 3 Top-2 query

	u1	u2	u3	u1内のランク	u2内のランク	u3内のランク
p1	0.4	0.39	0.41	3	3	3
p2	0.34	0.37	0.31	2	2	1
p3	0.31	0.23	0.39	1	1	2

このとき、u1 の Top-2 query は{p2,p3}、u2 の Top-2 query は{p2,p3}、u3 の Top-2 query は{p2,p3}となる。ここで改良戦略として$s = \{-0.1, -0.1, -0.1\}$を p1 に適用する。適用後の Top-2 query は表 4 から、u1 は{p1,p3}、u2 は{p2,p3}、u3 は{p1,p2}となる。これより p1 の Top-k query ヒット数が改良前 0 に対して、改良後は 3 に増加している。よってこの改良は効果的であると判断できる。先行研究では予算内で Top-k query のヒット数を最大限増加させる調整(Max-Hit IQ)と、与えられた Top-k query のヒット数を満たすように最小限のコストで改良を行う調整(Min-Cost IQ)の 2 通りの方法が提案された。

表 4 改良戦略適用後の Top-2 query

	u1	u2	u3	u1内のランク	u2内のランク	u3内のランク
p1	0.3	0.29	0.31	1	2	1
p2	0.34	0.37	0.31	3	3	1
p3	0.31	0.23	0.39	2	1	3

3. 提案手法

本研究では IQ で定義された Max-Hit IQ の考え方を参考に属性改良問題を解いていく。また IQ の問題点を解決するために Top-k query のヒット数ではなく、ランク上昇量に基づき、改良戦略の評価を行う。RRIQ 問題は NP 困難であるため([6]の 3.2 参照)、ヒューリスティックに解を求める方法を提案する。本研究では以下の手順で RRIQ 問題を解いていく。

3.1. 問題定義

商品データ集合を P、ユーザーの好みのデータ集合を U、各商品$p_i \in P$が d次元のベクトルで d個の属性値を持ち、$p_i = (p_i^{(0)}, p_i^{(1)}, \dots, p_i^{(d)})$というベクトルで表されるとする。また、各ユーザー$u_j \in U$が d 個の好みを持つ d 次元ベクトル$u_j = (u_j^{(0)}, u_j^{(1)}, \dots, u_j^{(d)})$であり、$\sum_{k=0}^{d} u_j^{(k)} = 1$であるとする。このとき本研究はクエリ商品集合$Q \subset P$を与えたときに、その中から条件を満たすクエリ$q_k \in Q$とその改良戦略を求めることを目的とする。改良戦略の定義を定義 1 に示す。

定義 1(改良戦略)　オブジェクトに対する改良戦略を s とする。改良戦略は d 次元のベクトルとして$s = (s^{(0)}, s^{(1)}, \dots, s^{(d)})$と表す。改良戦略 s をもとのオブジェクトp_iに適用した改良後オブジェクトp_i'を$p_i' = p_i + s$で表す。

ここでのランクは商品ベクトルとユーザーの好みデータのベクトルの内積を評価値として計算し、その評価値の高いものから順に並べたものを指す。本研究では商品ベクトルとユーザーの好みベクトルは各属性値が$0 \leq \ p_i^{(m)} \leq 1,\ 0 \leq \ u_j^{(n)} \leq 1\ (0 \leq m, n \leq d)$を満たすものであるとし、計算の都合上 0 に近いほど良い評価とする。

3.2. 提案手法の流れ

あるユーザーu_iとクエリオブジェクト q を用いて改良戦略の作成を行う。まず、予算を一定間隔で分割したものをδとし、j 番目の属性にのみ改良予算δを加え、その他の属性は改良予算が 0 であるような予算分配を考える。このとき、j 番目の属性が$\Delta p^{(j)}$だけ改良されたとする。$\Delta p^{(j)} * u_i^{(j)}$を計算し、もっともこの値が大きくなる属性を見つける。これを予算が尽きるまで繰り返し、改良戦略候補を作成する。これを全ユーザーに対して行い、改良戦略候補集合を作成する。あるユーザーu_iからのクエリ商品 q の改良前ランクをr_i、改良後ランクをr_i'として、改良戦略の評価を$eval(q')$ (1)で行う。この式はランク上昇量が大きく、また同じランク上昇量でもより上位にいけば大きい値をとるような調整がされている。この一連の流れを Algorithm1 に示す。

$$eval(q') = \frac{1}{|U|} \sum_{i=0}^{|U|} \frac{r_i - r_i'}{r_i'} \quad (1)$$

そして、クエリ商品集合 Q に対してそれぞれにアルゴリズムを適用し、(1)の式を用いて評価を行い、もっとも高いものを最大の改良効果が得られる商品とその改

良戦略として求める。また、アルゴリズムで使われている記号の意味は表 5 の通りである。本研究では評価関数を RkR で絞り込んだ上位 k 人に適用し、高速化を図った。

3.3. 改良戦略算出関数

本研究では、改良戦略の計算に予算を用いる。本研究では先行研究の改良された値に対してコストを計算するという方式ではなく、予算を与えたときに予算から改良される値を算出するという考えを用いる。本研究では簡略化のため、すべての属性で与えた予算の半分が属性値に反映されるようになっているが、本来は属性ごとに違った関数を用いるべきである。例えば、スマートフォンの改良を考えたときに CPU とカメラを強化するのでは与えた予算に対して改良効果は異なってくる。このような異なった関数を用いた実験は今後の課題である。

表 5 記号の意味

記号	意味
q	クエリ商品
P	商品集合
U	ユーザー集合
S	改良戦略候補集合
cost	予算を格納するベクトル
usedcost	消費された予算
l	インデックスを保持する変数
calStrategy	予算から改良戦略を計算する関数
PickBestScoreIndex	最大のスコアを持つインデックスを返す
eval	評価関数

Algorithm 1 Reverse Rank Improvement Query

Require: P, U, q, $budget$
Ensure: Improvement query strategy
  $S \Leftarrow \phi$
  for $u_i \in U'$ **do**
    for $(usedcost = 0; usedcost < budget; usedcost+ = delta)$ **do**
      $Find\,best\,element\,with\,maximum\,rising\,scalar$
      $cost[bestelement]+ = delta$
      $q_i' \Leftarrow calStrategy(cost)$
    end for
    $S.add(q_i')$
    $score_i \Leftarrow eval(q_i')$
  end for
  $k \Leftarrow PickBestScoreIndex(score)$
  return q_k'

4. 実験、考察

提案手法の有用性を示すために、貪欲に改良戦略を作成し適用する方法と提案手法について実験を行い、その結果を示した。実験環境は以下の通りである。

- OS: MacOS Mojave 10.14.5
- CPU: 2.3GHz Intel Core i5
- メモリ: 8GB 2133 MHz LPDDR3
- 開発言語: C++

4.1. 実験データ

本実験では商品の評価データ、ユーザーの好みデータとして一様分布に作成した合成データを用いる。実験データのベクトルの各属性は 0 から 1 の値をとるように調整されており、0 に近いほど高い評価であるとしている。また、本実験では$|P| = 500,\ |U| = 500,\ |Q| = 1$を初期値とし、これらを変量として実験を行った。設定は表 6 の通りである。

表 6 実験データ設定一覧

データ	デフォルト値	値の範囲
$\|P\|$	500	500-2000
$\|U\|$	500	500-2000
次元数d	3	固定
k	$\|U\|/10$	固定
クエリ数$\|Q\|$	1	1,3,5

4.2. 近似解の精度

提案手法による近似解と貪欲法による最適解を比較し、精度を調べる。$|P| = 100, |U| = 100, 200, 300,\ |Q| = 1$として最適解ベクトルと近似解ベクトルの属性値の平均二乗誤差を測定し、グラフにしたものが図 1 である。ユーザー数の変化に関わらず、高い精度が出ていることがわかる。また、同様に$|P| = 100,\ 200\ , 300, |U| = 100,\ |Q| = 1$として平均二乗誤差を測定したものが図 2 となる。

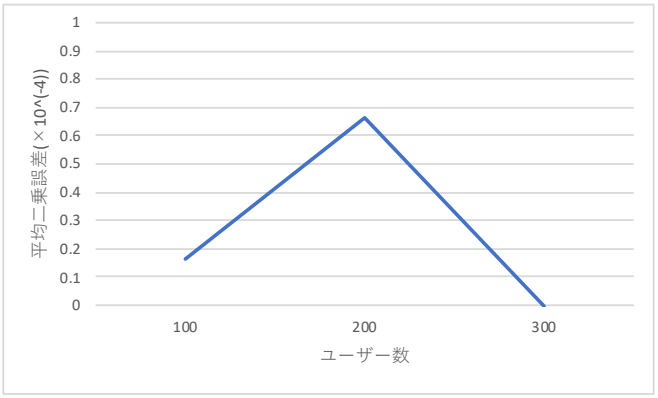

図 1 ユーザー数を変化させたときの精度

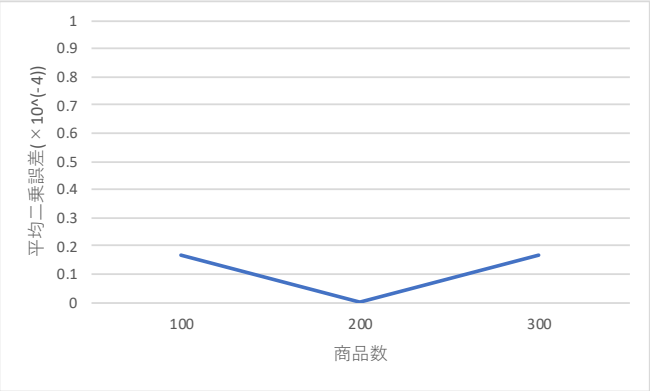

図 2　商品数を変化させたときの精度

こちらも同様に高い精度が出ていることがわかり、これらより商品数、ユーザー数の変化にかかわらず高い精度で近似解を導出できることがわかった。また、貪欲法では$|P| = 100, |U| = 100$ のときに 260.595 秒かかっているのに対して、提案手法では 0.473268 秒で計算が終了しており、高速に近似解を求めることができている。これにより、クエリ集合の中から最大の改良効果が期待できる商品とその改良戦略を求めることができると考えられる。

4. 3. クエリ集合における提案手法の実行速度

図 3 は$|P| = 500,1000,1500,2000$にしながら実験を行った結果である。同様に$|U| = 500,1000,1500,2000$と値を変化させ実験を行った結果が図 4 である。図 3 によれば商品数の増加では実行時間は線形に増加していることがわかる。$|P| = 2000$で実行時間が減少しているのは$|P|$の影響が少ないことによる誤差だと考えられる。また、図 4 によるとユーザー数の増加による実行時間の増加の影響は商品数よりも大きく、非線形に上昇していることがわかる。本実験では評価値の計算をするユーザーを絞り込むために RkR を用いているが、改良戦略を求める前にユーザー集合をフィルタリングすることで計算時間を大幅に短縮できると考えられる。しかし、フィルタリングによってユーザー集合を絞り込めば、近似解の精度が落ちる恐れがある。このことに関しては RkR の絞り込みを行う数を変化させながら精度を測定し、集合の性質に応じた最適な値を見つけるといった実験による検討が必要である。また、クエリ集合に関しても、凸包を用いたフィルタリングなどを検討する必要がある。

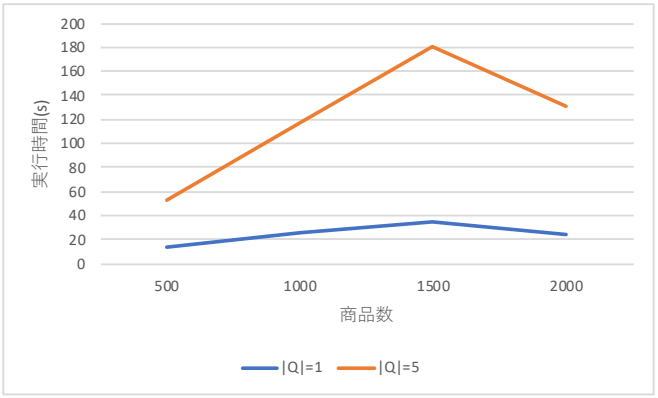

図 3 商品数による変化

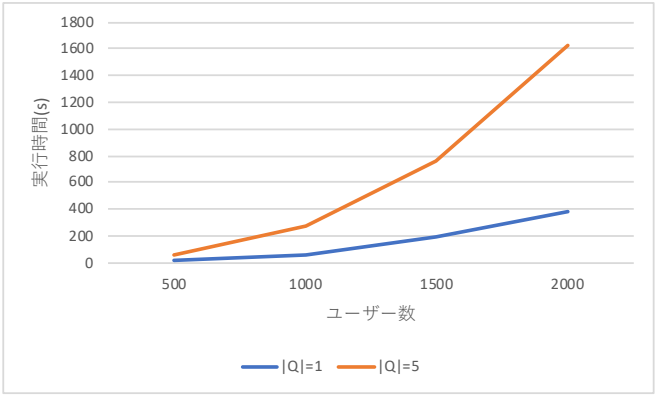

図 4 ユーザー数による変化

5. おわりに

本研究ではクエリ集合における属性改良問題の解を各クエリでの属性改良問題をヒューリスティックな手法を用いて解き、高速化することで現実的な時間で解くことを可能にした。今後の課題としては先にも触れた改良戦略を算出する関数を実世界の事情にあったものになるように調整すること、与えられたクエリ集合にフィルタリングをかけ、計算時間を短縮することが挙げられる。また、一様分布の合成データだけではなく、正規分布のデータ、実データを用いた実験によるさらなる分析、考察が必要である。

謝辞

本研究は JSPS 科研費 JP19K12114 の助成を受けたものである。

参　考　文　献

[1] Yang, G., Cai, Y.: Querying Improvement Strategies. EDBT. pp.294–305, 2017.

[2] Ilyas, I.F., Beskales, G., Soliman, M.A.: A survey of top-k query processing tech- niques in relational database sys-tems. ACM Comput. Surv. 40(4), 11:111:58, 2008.

[3] Zhang, Z., Jin, C., Kang, Q.: Reverse k-ranks query. PVLDB 7(10),785796, 2014.

[4] Dong, Y., Chen, H., Furuse, K., Kitagawa, H.: Aggregate Reverse Rank Queries. DEXA 2016, pp. 87-101, September 5-8, 2016.

[5] Dong, Y., Chen, H., Furuse, K., Kitagawa, H.: Grid-Index algorithm for reverse rank queries. EDBT 2017, pp. 306-317, March 21-24, 2017.

[6] 山下 雅弘.,佐野 ひかり., 陳 漢雄., 古瀬 一隆., 北川 博之. 逆ランク検索による情報改良支援. DEIM Forum 2018 D4-3.

[7] 小栗 大輝., 董 于洋., 陳 漢雄., 古瀬 一隆. 改良コスト最小化の逆情報推薦. DEIM Forum 2018 D3-5.