条件検索におけるランキング学習とそれに基づく条件推薦

佐々岡哲哉† 田島 敬史† 清田 陽司††

† 京都大学大学院情報学研究科 〒 606–8501 京都府京都市左京区吉田本町
†† 株式会社 LIFULL 〒 102–0083 東京都千代田区麹町 1-4-4
E-mail: †sasaoka@dl.soc.i.kyoto-u.ac.jp, †tajima@i.kyoto-u.ac.jp, ††kiyotayoji@lifull.com

あらまし 本稿では，様々な属性についての条件を指定してアイテムの検索を行う条件検索において，望ましいアイテムの発見を容易にする二つの手法を提案する．一つ目は検索条件に依存するランキングである．過去に同じ条件や類似する条件で検索した他のユーザが検索解中のどのアイテムにクリックを行ったかのデータからランキングを学習する．検索条件間の類似度は，検索で指定された属性条件の集合をベクトルで表現したものの間のコサイン類似度で求め，属性条件集合のベクトル化は，過去の検索条件の間での，どれだけどのようなアイテムをクリックしたかの類似度を用いて学習する．二つ目は検索条件の推薦である．検索条件が与えられた際，その条件を満たさないために解とはならないが，上述のランキング手法ではスコアが高くなるようなアイテムを見つけ，それらが解に含まれるようになる検索条件を推薦する．その際，当初の検索条件から離れ過ぎると意図に反する可能性が高いことも考慮する．

キーワード 情報推薦，情報検索，ランキング学習，機械学習，条件検索，不動産検索，クエリ推薦，クエリ類似度

1 はじめに

なんらかの「検索」機能を提供するシステムは，主に，データベースシステムと情報検索システムに分けられる．データベースシステムでは，多数の属性を持つアイテムの集合から，これらの属性に関する明確な条件を指定して，条件を満たすアイテムの一覧を取得するような検索が典型的である．一方，情報検索システムでは，キーワードなどを用いて曖昧な情報要求を表現し，アイテムを適合する確率が高いと思われる順にランキングして表示するような検索が典型的である．

しかし，多数の属性に関する検索条件を指定して，これらの条件を満たすアイテムを好ましさの順にランキングして表示したいという，両者の中間のようなシステムも存在する．代表的な例としては，賃貸不動産物件検索，ホテル検索，中古車検索などが挙げられる．そのような検索システムにおいて，ユーザが望ましいアイテムを発見することを困難にする主要因として以下の二つが挙げられる．

一つ目は，最適なアイテムが見つかるような検索条件を適切に指定することの困難さである．例えば，賃貸物件検索であれば，ユーザは，賃料，床面積，駅からの距離などの様々な条件を指定して検索を行うが，属性数が多い場合，様々な属性を総合的に見てもっとも好ましいと思われるアイテムに適切に絞り込めるような検索条件を指定するのは容易ではない．検索条件が緩すぎると，検索解の数が多過ぎて全てに目を通せず，検索条件が厳しすぎると，ある属性についてはわずかに条件を満たさないが総合的に見ると最適であるようなアイテムが解から漏れる．このような条件調整の困難さは，キーワードによる情報検索における適切なキーワードの発見の困難さとは異なる問題である．

二つ目は，検索結果を適切にランキングすることの困難さである．属性条件による検索では，検索解を特定の属性の値（例えば，賃貸物件検索なら，賃貸料，床面積，登録日など）でソートして表示させることが多い．しかし，一つの属性にもとづくソートでは，その属性に関して突出したアイテムが最上位に来ることになり，バランスの取れたアイテムを上位に出すことが難しい．

そこで，本研究では，多数の属性に関する条件を指定して検索を行い，結果を好ましさの順にランキングするような検索システムにおいて，上記 2 点の問題を解決して，ユーザが好ましいアイテムを探し当てるのを容易にするための手法を提案する．

まず，二つ目の適切にランキングすることの困難さを解決するために，検索条件に依存するランキングの手法を提案する．この手法では，過去に同じ条件や類似する条件で検索した他のユーザが検索解中のどのアイテムにクリックを行ったかのデータからランキングを学習する．ここで，同じ条件による検索だけではなく類似する条件による検索も用いるのは，多数の属性に関する条件を指定できる検索システムでは，検索が多様であり，完全に同じ検索が過去にもなされているとは限らないためである．

しかし，「類似する検索」も用いるためには，検索条件間の類似度を定める必要がある．提案手法では，各属性についての条件のリストである検索条件をベクトルで表現し，このベクトル間のユークリッド距離に基づいて類似度を定める．また，属性条件のリストをベクトル化する方法については，過去に実行された検索条件の間の類似度を，その検索の解においてどれだけ同じようなアイテムにクリックが行われたかで定義し，この類似度とユークリッド距離で定めた類似度が一致するようなベクトル化を学習することで与える．

次に，一つ目の適切な検索条件を指定することの困難さを解決するために，検索条件の推薦手法を提案する．検索条件が与

えられた際，その条件を満たさないために解とはならないが，上述の本研究で提案するランキング手法ではスコアが高くなるようなアイテムを見つけ，それらが解に含まれるようになる検索条件を推薦する．また，その際，ユーザが当初入力した検索条件から離れるほど，ユーザの意図に反する検索となってしまう危険性が高まることも考慮した上で，検索条件を推薦する．

検索条件ではなくアイテム自体を直接推薦することも考えられる．しかし，ユーザの当初の検索条件を満たさないが推薦すべきと思われるアイテムが多数ある場合，ユーザがそれらを一つ一つずつ確認するよりも，検索条件を推薦して，ユーザがそれを受け入れるかどうかを判断する方が，ユーザは効率よく意図に反する推薦を除外することができる．

なお，本稿のこれ以降では，賃貸不動産物件を例に用いて説明するが，提案手法は，属性条件を指定して検索し結果を好ましさでランキングするような検索システム一般に適用可能である．

2 関 連 研 究

本研究では，様々な属性の条件を指定する検索における，ランキング学習とクエリ推薦の手法を提案する．そこで，本章では，ランキングに関する関連研究と，当初与えたクエリで適切な解が得られない場合の対応策に関する関連研究について説明する．また，本研究では条件検索においてクエリ類似度を求める手法も提案するため，クエリ類似度に関する関連研究についても述べる．

2.1 ランキングに関する関連研究

情報検索システムにおいて，解のランキングは最も重要な機能であり，数多くの研究が行われている．

中でも特に近年，機械学習を用いてランキング関数を生成するランキング学習に関する研究が多く行われている．文献 [3] は，ユーザーのクリックログデータを利用する．ユーザーのクリックログデータからクエリに対するアイテムの順序を生成し，その順序関係をもとに SVM による学習を行いランキングを生成する．また，文献 [5] では，クリックログデータに加えて，クエリチェーンも利用してランキングを生成する．あるクエリを入力した後別のクエリを入力した時，二番目のクエリに関連するページも最初のクエリに関連していると考え，それをランキングに利用するというものである．

また，複数の条件を指定して検索を行う際のランキングの手法としては，各条件の優先度をユーザに指定させ，この情報に基づいてランキングを行う手法が文献 [8] で提案されている．この研究では，レシピ検索システムを対象としており，ユーザは食材のリストを入力してそれらを使用するレシピを検索する．その際，各食材をどれだけ用いたいかの優先度を-10 から 10 までの 21 段階で指定でき，各レシピを構成する食材の優先度を加算したものをそのレシピの評価値として算出し，ランキングに利用する．

この手法は，食材のように含まれるか否かのどちらかしかない項目の優先度については，ユーザの意図をある程度うまく結果に反映させることができるが，数値情報を扱う場合，項目同士の優先度の差が，どの程度ランキングに反映されるのかの理解はより難しい．例えば，家賃の優先度を 5，広さの優先度を 2 と入力したあるユーザが，家賃 5 万円広さ 5 畳の物件が家賃 6 万円広さ 8 畳の物件よりも上位にランキングされることを想定していたとしても，実際のランキングでは逆順にランキングされ意図に反したランキングとなる可能性がある．また，建物構造を木造や鉄筋構造などの複数の選択肢から選ぶようなな項目がある場合，その項目内の選択肢同士での優先度に加え，他の項目との優先度も考慮する必要が出てくるため，検索条件を指定する作業が非常に複雑になってしまう．以上の理由から，優先度をユーザに指定させる手法は，賃貸物件検索のように多数の数値属性を扱う検索には不向きである．

属性条件を指定する検索においても，本研究の提案手法と同様に，ユーザに優先度を指定させるのではなく，ユーザの行動ログを用いてランキングを学習する手法も提案されている．文献 [10] では，商品検索においてユーザがこれまでに商品に対して行ったアクションの情報を，ニューラルネットワークを用いたランキング学習である RankNet [1] を拡張したものに学習させる手法を提案している．多くの検索システムではユーザがそのアイテムを選択し詳細情報にアクセスしたかどうかという情報しかないのに対し，EC サイトでは，「対象商品をクリックし詳細ページを閲覧した」「対象の商品をカートに入れた」「対象の商品を購入した」という三つのアクションが存在する．そこで，この研究では，これらのアクションを区別することでランク学習の性能を向上させることを目指している．しかし，この研究は，全ての検索に対して一つの学習器を作成し，入力する特徴量に検索条件を含めている．例えば，アイテムの属性値と検索条件の属性値の差を特徴量に持たせたり，価格などの特に重要な検索条件を特徴量に追加したりしている．しかし，不動産物件や中古車の検索では，どの属性がどの程度重要かはユーザごとに大きく異なる．そこで，本研究では，検索条件ごとに学習器を作成して検索条件に依存したランキングを実現し，また，検索条件同士の類似度を用いることで，そのランキングの精度を上げることを目指す．

2.2 適切ではないクエリへの対応策

Web 検索では，ユーザが求めるページにたどり着かない場合，より適切と思われるクエリを提示することで支援するアプローチが広く用いられている [4], [9]．Qiaozhu らは，複数の意図に解釈しうるクエリが入力された場合，ユーザの意図した検索結果に絞り込めると思われるクエリを提示する手法を提案している [4]．提示するクエリは，過去の全ユーザのクエリとクリックデータを表す二部グラフを用いて算出される．二部グラフは，クエリを表すノードと，ユーザがクリックしたサイトの URL を表すノードにより構成され，ノード間のエッジは，そのクエリで検索した際にその URL をクリックした場合に引かれる．エッジの重みはクリックした回数とする．この二部グラフを用いて，重みの大きいエッジを介してつながるクエリを推薦

する．

しかし，この手法ではより多く検索されている検索意図のクエリばかりが推薦されてしまうという問題がある．例えば，DVD というクエリが入力されたとする．この時，DVD ディスク自体について検索している，または，DVD のレンタルについて検索しているという 2 種類の検索意図が考えられる．しかしこの手法では，より検索頻度が多い，DVD レンタルという検索意図に関するクエリばかりが推薦されてしまう．この問題を解決するため，今井ら [9] は二部グラフのクエリノードをクラスタリングすることで，頻度の高い検索意図に偏らないような推薦を行う手法を提案した．

Web 検索などキーワードをクエリとする検索において，あるページにたどり着きたい時にユーザが入力する単語の種類は比較的限られていて，クエリと URL の二部グラフを構成するとエッジの重みが大きい部分と小さい部分がはっきりと分かれる．一方で数値条件による条件検索では，同じアイテムがアクセスされた時に入力されていた検索条件はばらつきがあり，全く同じ検索条件となることはキーワード検索と比べ少ない．そのため，条件検索においては，クエリとアイテムの二部グラフを構成した場合，エッジの重みが小さくなってしまい，この手法は条件検索には適さない．

また，適切でないクエリを入力したユーザに，クエリから想定されるユーザの興味をいくつか提示し，より適切に絞り込んだ検索を行わせるという手法も研究されている [6]．この手法では，書籍のメターデータと，クリックスルーログを用いて，クエリだけでは十分に表現されていないようなユーザの潜在的な意図を LSA を用いて学習する．ユーザが入力したキーワードをもとに，ユーザの潜在的な意図に関するキーワードを提示し，検索結果をより簡単に絞り込めることが可能となる．求めているアイテムを的確に表すキーワードを思いつく困難さがうまく検索できない原因となる，書籍検索などのシステムでは非常に有効である．

一方で，物件検索のような条件検索では，クエリは絞り込みのための数値やカテゴリーであり，どのようなクエリを入力すればいいかを思いつかないことが問題なのではなく，データベースの中にどのようなアイテムが存在するのかがわからず，検索条件の細かい調整が困難であることが問題となる．このような検索システムでは，Web 検索や書籍検索に比べ，ユーザの検索意図は，クエリに反映されやすく，したがって，クエリに十分に表現されていない潜在的な意図を推定することよりも，クエリに顕在的に表現された意図を汲み取って，データベース内にどのようなアイテムが存在するか，どのような検索条件が適切なのかをユーザに知らせることが重要となる．

2.3 クエリ間類似度に関する関連研究

クエリ間類似度を求める研究も多く行われている．Zhao らは，クエリ間の類似度を，クエリとその時にどのサイトをクリックしたのかを記録したクリックスルーデータに加え，クリックした時のタイムスタンプまで考慮することでクエリ間の意味的な類似性をより正確に捉える手法を提案した [7]．また，Guo らは，マイナーな検索意図にも対応できるように，検索した意図に基づいてクエリ間の類似度を判断する手法を提案した [2]．この研究では，クエリ同士で同じものをクリックしたかどうかを正規化したものと，検索結果のスニペットにある単語を利用して，クエリの潜在意図をトピックモデルで学習する．

3 提案手法

この章では，本研究で提案するランキング手法とそれに基づく条件推薦手法について記述する．まず，与えられた検索条件に対して，その検索条件と同じまたは類似する過去の検索におけるユーザのアクションの情報からアイテムを評価するランキング関数を学習し，これを検索結果のランキングに利用する．それと同時に，検索条件を満たさず検索結果に表示されなかったアイテムに対しても，同じランキング関数を適応し高スコアとなるアイテムを発見する．そして，それらも含めて，学習したランキング関数で評価の高いものが上位 10 件以内に表示されるようようになる検索条件を推薦する．ただし，その際，推薦する検索条件が現在入力されている検索条件から遠のくほどユーザの意図とは異なるものになりうるということも考慮する．以下，3.1 節で，どのように評価関数を作成するのかについて述べ，3.2 節でどのように検索条件を評価し推薦するかについて記述する．

3.1 ランキングの作成

本稿では，検索条件ごとに学習器を生成し，検索条件を入力したユーザが好む項目に高い評価値を与える．学習には，ランキング学習で使用される RankNet [1] を使用する．RankNet は，ニューラルネットワークを使用するランキング学習方法で，アイテムのペアを受け取り，どのアイテムを上位にランク付けするかを学習する．その結果，任意のアイテムを入力すると，その項目の評価値を出力するランキング関数が生成される．評価値は学習対象によって異なる．学習対象が EC サイトの場合，購入回数を評価値とし，学習対象が Web 検索の場合，実際にクリックしてサイトにアクセスした回数を評価値とする．入力は，アイテムペアの特性とどちらが上位にランク付けされているかのみであり，クエリ情報は使用されない．

RankNet の入力にクエリの情報を加えることで，任意のクエリに対応する一つの学習器を作成することもできる．例えば，文献 [10] では，商品の説明文や商品のカテゴリーの説明文の特徴ベクトルと，クエリの単語頻度ベクトルとの間の内積を，入力する特徴に加えている．しかし，本研究で主要な応用の例として考えている賃貸物件検索などにおいては，検索条件ごとにユーザがアイテムのどの造成をどれだけ重視するかは大きく異なると考える．例えば，家賃が 4 万円以下という条件だけで検索したユーザは，とにかく安い物件を探しており，一方で，家賃の条件を入力せず，広さの下限や駅からの距離の上限を入力したユーザは，家賃よりも広さや駅からの距離を重視していると考えられる．そこで本研究では，その検索条件を入力するユーザはどの属性をどれだけ重視するのかという情報を利用するた

め，クエリごとに学習器を生成し，ランキング関数を導出する．

RankNet では，アイテム I_i, I_j の特徴ベクトル x_i, x_j を入力とし，それぞれに対応する実数値 $s_i = f(x_i), s_j = f(x_j)$ が出力される．この時，アイテム I_i が I_j よりも上位にランキングされる確率は以下のシグモイド関数で与えられると考える．

$$P_{ij} = \frac{1}{1 + exp[-\{s_i - s_j\}]} \tag{1}$$

そして，RankNet の損失関数は，通常，交差エントロピー損失を用いて以下のように設定する．

$$L_{ij} = -\bar{P}_{ij} \log P_{ij} - (1 - \bar{P}_{ij}) \log(1 - P_{ij}) \tag{2}$$

P_{ij} は上で述べたように，アイテム I_i が I_j よりも上位にランキングされる確率である．また，$\bar{P}_{ij}$ とは，学習データ内で I_i が I_j よりも上位にランキングされたかどうかを表し，I_i が I_j よりも上位にランキングされる場合に 1，同ランクの場合に 0.5，下位にランキングされる場合に 0 となる．

本研究では，検索条件ごとに学習器を作成する．また，一般的な RankNet はクリック数などをランキングのためのアイテムの評価値として学習するが，本調査では賃貸物件がクリックされて詳細ページが表示された回数を評価値として扱う．したがって，学習対象の検索条件を q とし，損失関数を以下のように定義する．

$$L^q_{ij} = -\bar{P}^q_{ij} \log P^q_{ij} - (1 - \bar{P}^q_{ij}) \log(1 - P^q_{ij}) \tag{3}$$

ここで，P^q_{ij} は検索条件 q の際に，アイテム I_i が I_j よりも多くクリックされる確率であり，$\bar{P}^q_{ij}$ は学習データ内で検索条件 q の解において I_i が I_j よりも多くクリックされたかどうかを表し，I_i の方が多くクリックされる場合に 1，同数の場合に 0.5，I_j の方が多い場合に 0 となる．すなわち，行動ログ内にある検索条件 q の解においてアイテム I_i にクリックが行われる回数を y^q_i とすると，$\bar{P}^q_{ij}$ は次のように表せる．

$$\bar{P}^q_{ij} = \begin{cases} 1 & \text{if} \quad y^q_i > y^q_j \\ \frac{1}{2} & \text{if} \quad y^q_i = y^q_j \\ 0 & \text{if} \quad y^q_i < y^q_j \end{cases} \tag{4}$$

しかし，物件検索のような数値属性を含む多数の属性に対する条件を指定できる条件検索において，完全に検索条件が一致する検索のログを学習に十分なほど確保するのは困難である．そのため，学習器を作成する検索条件を絞り込み，それらの学習した学習器と検索条件の類似度を利用して任意の検索条件についてのランキング関数を作成する．その際，検索条件同士の類似度を考慮し，検索条件が似通っているものほど評価に大きな影響を与え，遠いものほど影響を与えないようにする．この時，学習器を作成した検索条件の集合を Q_n，検索条件 q の学習器によるアイテム I の評価値を $f_q(I)$ とし，検索条件 q_1, q_2 の類似度を $sim(q_1, q_2)$ として，任意の検索条件 q のアイテム I についての評価値を以下のように定義する．

$$\sum_{q_1 \in Q_n} f_{q_1}(I) \frac{sim(q, q_1)}{\sum_{q_2 \in Q_n} sim(q, q_2)} \tag{5}$$

以下，3.1.1 項で，まず，類似度の求め方について述べ，その後，学習させる検索条件をどのようにして選ぶかを 3.1.2 項で述べる．

3.1.1 検索条件間の類似度

我々の目的は，与えられた検索を入力したユーザがどのアイテムを好ましいと思うかを，検索ログ中の類似する検索の情報を用いて推定することなので，検索条件間の類似度の定義にあたっては，それが「検索条件を入力したユーザが好ましいと思うアイテムの類似度」を表すようにすべきである．よって，検索ログ中の検索条件同士の類似度であれば，その検索への解の中でどのアイテムにどれだけクリックがなされたかの類似度によって定義することが考えられる．

しかし，現在，ユーザに与えられているクエリの検索条件については，どのようなアイテムにクリックがなされるかは，まだわからない．検索ログ中に同じ検索条件があれば，その際にどのアイテムにどれくらいクリックがなされたかの情報を使うことが考えられるが，3.1 節でも述べたように，数値属性を含む多数の属性の条件を指定する検索では，検索の種類の数が多く，現在与えられているクエリの検索条件と同じものが検索ログ中に存在するとは限らない．

そこで，本研究では，どのアイテムにクリックがなされたかがわかっている検索ログ中の検索条件について，それらの間の類似度を，同じようなアイテムをクリックする傾向があるのかによって定義し，これを学習データとして学習することで，クリックの情報がない任意の検索条件の間についても類似度を求められるようにすることを考える．そうすれば，その結果を用いて，現在与えられているクエリの検索条件と検索ログ中の検索条件との間の類似度も求められるようになる．

より具体的には，検索条件をベクトル化し，検索条件の間の類似度は，そのベクトルの間のユークリッド距離に基づいてで定義することを考える．そして，検索条件をどのようにベクトル化するかについては，検索ログ中の検索条件についてどのようなアイテムをどれだけクリックしたかに基づいた検索条件間類似度とベクトル化した検索条件同士のユークリッド距離に基づいた検索条件間類似度が一致するようにベクトル化の方法を学習する．そのようなベクトル化が学習できたら，クエリが与えられた際に，その検索条件をベクトル化することで，任意の検索条件との類似度を求めることができる．

よって，まず，検索ログ中に存在する検索条件 q_a, q_b の間の類似度 $\bar{sim}(q_a, q_b)$ を以下のように定義する．

$$\bar{sim}(q_1, q_2) = cos(q\star_1, q\star_2) \tag{6}$$

ここで，$q\star$ とは検索条件 q のクリックベクトルとして定義し，どのようなアイテムをクリックする傾向にあるかを示している．具体的には $q\star$ は以下のように定義する．

$$q^\star = \frac{q'^\star}{|q'^\star|} \tag{7}$$

$$q'^\star = (Count(q, G_1), \ldots, Count(q, G_m)) \tag{8}$$

$Count(q, G_1)$ は検索条件 q で検索したユーザがグループ G_1 に

属するアイテムをクリックした回数を指し，m はグループの数である．アイテムのグループは同じようなアイテムをまとめたものであり，アイテムの全ての属性が同じ範囲にあれば同じグループに所属するとみなす．例えば，5,0000 ~ 60,000 を一つの金額帯，20 ~ 25m^2 を一つの広さ帯とすると，物件 A が 59,000 円・24m^2 ，物件 B が 52,000 円・21m^2，物件 C が 61,000 円・26m^2 の時，物件 A と B が同じグループ，物件 C は別のグループという扱いになる．

続いて，検索条件 q のベクトル化である条件ベクトル $\mathbf{q}$ を以下のように定義する．

$$\mathbf{q} = (w_1 \cdot v_1, \ldots, w_n \cdot v_n) \tag{9}$$

ここで，n は各アイテムが持つ属性の数，v_i は各属性に対して q で定義されている条件を数値化したもの，w_i は検索条件の類似度を求めるにあたって，各属性をどの程度重視するかの重みである．

このとき，条件ベクトル $\mathbf{q}$ を用いた検索条件 q_1, q_2 間の類似度を $d(a,b)$ をベクトル a,b のユークリッド距離として以下のように定める．

$$sim(q_1, q_2) = 1 - \frac{d(\mathbf{q_1}, \mathbf{q_2})}{|\mathbf{w}|} \tag{10}$$

この w_i を，検索ログ中の検索条件 q_a, q_b に対して $\bar{sim}(q_a, q_b)$ と $sim(\mathbf{q_a}, \mathbf{q_b})$ とが一致するように学習する．

v_i は，その属性に対して指定されている条件の厳しさを数値化したものであり，検索ログ中のクエリのうち，その条件と同じか，または，より緩い条件を課していたクエリの比率により与えられる．例えば，賃貸物件の駅からの距離について，

- 検索ログ中の 10%のクエリが徒歩 20 分以内と指定
- 検索ログ中の 25%のクエリが徒歩 15 分以内と指定
- 検索ログ中の 50%のクエリが徒歩 10 分以内と指定
- 検索ログ中の 15%のクエリが徒歩 05 分以内と指定

していたとする．この時，

- 「徒歩 20 分以内」の厳しさは 0.1
- 「徒歩 15 分以内」の厳しさは 0.1 + 0.25 = 0.35
- 「徒歩 10 分以内」の厳しさは 0.1 + 0.25 + 0.5 = 0.85
- 「徒歩 05 分以内」の厳しさは 0.1 + 0.25 + 0.5 + 0.15 = 1

となる．また，何も検索条件を課していない場合の厳しさは常に 0 と定義する．

なお，本研究では各属性について，上限と下限のうちの一方だけを考慮する．例えば，賃貸物件検索では，駅からの距離については上限のみを，広さについては下限のみを考慮する．また，「オートロックあり」などのように，真偽値を取る属性については，真を 1，偽を 0 とする．

また w_i は，C_Q を学習データ内に存在する全ての検索条件の組み合わせとし，以下の値を最小とするように算出する．

$$\sum_{(q_a, q_b) \in C_Q} (\bar{sim}(q_a, q_b) - sim(\mathbf{q_a}, \mathbf{q_b}))^2 \tag{11}$$

3.1.2 学習させる検索条件を選ぶ方法

先にも記述したように，賃貸物件検索などでは検索条件の種類は多く，全てのクエリに対して，そのクエリの検索条件に対応した学習器を作成するのは現実的ではない．そこで，本研究では，n 個の検索条件に対して学習器を作成し，それ以外の検索条件に対しては，それら n 個の検索条件との類似度に基づいて，n 個の学習器の結果に重みを付けて集約することで代用する．本研究では $n = 100$ とする．

検索ログから，学習器を作成する検索条件を選ぶ際には，頻度が高い検索条件で，かつ，できるだけ互いに似ていないようなものを以下のような貪欲法で選ぶ．まず，最も頻度が高い検索条件を選ぶ．次に，既に選択した検索条件の集合を Q_n，また，検索条件 q が検索された回数を c_q とすると，以下の値が最も大きいものを学習器を作成する検索条件に加える．

$$\max_{q \in (Q \setminus Q_n)} \sum_{q' \in Q_n} \frac{c_q}{sim(q, q')} \tag{12}$$

これを検索条件の数が必要な個数になるまで繰り返す．

これらの選択された検索条件について生成した学習器と，検索条件間の類似度を用いて，その他の検索条件 q に対するアイテム I_i の評価値を以下の式で求める．

$$\sum_{q_1 \in Q_n} f_{q_1}(I_i) \frac{sim(q, q_1)}{\sum_{q_2 \in Q_n} sim(q, q_2)} \tag{13}$$

ここで $f_q(x_i)$ とは，検索条件 q について作成した学習器がアイテム I_i について出力する評価値である．

3.2 条件推薦

3.1 節で作成したランキング関数を用いて，検索条件を満たさないために検索解に含まれなかったアイテムについても評価を行い，ユーザにとってより良いと思われるアイテムがあれば，それが検索結果に含まれるようになるように検索条件の推薦を行う．提示する検索条件の定め方について二つの手法を提案し実験で比較を行う．

3.2.1 手法 1：ユーザの検索条件変更の許容度を全体に付与

ユーザがクエリとして検索条件 q を指定しているとする．この時，検索条件 q' を満たすアイテムの中で，f_q による評価値が i 番目に高いと評価されたアイテムを $I_{rank^{q',q}=i}$ と表すと，その検索条件 q' がどれだけ q を入力したユーザにとって良いものであるかを以下の値で評価する．

$$\sum_{k=1}^{10} \frac{1}{k} f_q(I_{rank^{q',q}=k}) \tag{14}$$

手法 1 では，この値に対してさらに，提案する検索条件がどれだけユーザの意図に沿っているかの確率をかけたものを用いる．ユーザの意図に沿っているかの推定には，ユーザの検索ログを用いる．現在入力している検索条件を q とし，検索条件 q' がそのユーザの意図に沿っている確率を考える．まず，検索条件 q, q' をベクトル化したものの差である $\mathbf{q} - \mathbf{q}'$ を求め，検索ログ中に存在する検索条件の推移におけるベクトルの差のうち，これを上回っているものの比率を，q から q' への推移がユーザ

の意図に反しない確率と考える．

検索条件の推移におけるベクトルの差とは，ユーザが検索結果内に満足のいくアイテムを見つけることができず検索条件を変更したときの，変更前の検索条件と変更後の検索条件のベクトルの差のことをいう．例えば，$\mathbf{q}-\mathbf{q}'=(0.2,0.1,0.1)$ であったとする．そして，検索ログには，検索条件の推移として，$q_{a1}\rightarrow q_{b1}$，$q_{a2}\rightarrow q_{b2}$，$q_{a3}\rightarrow q_{b3}$ の三つが存在し，それぞれ，$\mathbf{q}_{b1}-\mathbf{q}_{a1}=(0.1,0.05,0.08)$，$\mathbf{q}_{b2}-\mathbf{q}_{a2}=(0.15,0.075,0.08)$，$\mathbf{q}_{b3}-\mathbf{q}_{a3}=(0.3,0.1,0.05)$ であったとする．このとき，$\mathbf{q}-\mathbf{q}'<\mathbf{q}_{b1}-\mathbf{q}_{a1}$，$\mathbf{q}-\mathbf{q}'<\mathbf{q}_{b2}-\mathbf{q}_{a2}$ は成り立つが，$\mathbf{q}-\mathbf{q}'<\mathbf{q}_{b3}-\mathbf{q}_{a3}$ は成り立たない．ここで，$\mathbf{A}<\mathbf{B}$ とは，ベクトル $\mathbf{A}$ の全ての要素が，ベクトル $\mathbf{B}$ の対応する要素を下回ることを表している．したがってこの場合，検索ログ中に存在する検索条件の推移に関するベクトルの差を下回る確率は 2/3 となる．よって，このとき検索条件 q を入力したユーザに対する検索条件 q' の評価値は以下のようになる．

$$\frac{2}{3}\sum_{k=1}^{10}\frac{1}{k}f_q(I_{rank^{q',q}=k}) \tag{15}$$

すなわち，ユーザの検索条件変更の許容度，つまり，検索条件 q' が検索条件 q を入力したユーザの意図に沿っている確率を $P(q'|q)$ と表すと，提案手法 1 では，検索条件 q を入力したユーザに対する検索条件 q' の評価をを以下で定義する．

$$P(q'|q)\sum_{k=1}^{10}\frac{1}{k}f_q(I_{rank^{q',q}=k}) \tag{16}$$

この評価値が一番大きな検索条件をユーザに推薦する．

3.2.2 手法 2：ユーザの検索条件変更の許容度をアイテムに個別付与

提案手法 1 では, ユーザの検索条件変更の許容度である $P(q'|q)$ を検索条件の評価全体にかけた．提案手法 2 では，アイテムごとに，そのアイテムが解に含まれるような検索条件のうちで，ユーザが入力した元の検索条件に一番近いものへの変更許容度を算出し，該当アイテムの値にかける．つまり，アイテム I_i が出る検索条件のうち，一番 q に近い検索条件を $q_{near}(q,I_i)$ とすると，q を入力したユーザに対する検索条件 q' の評価値を以下の式で定義する．

$$\sum_{k=1}^{10}\frac{1}{k}P(q_{near(q,I_{rank^{q',q}=k})}|q)\cdot f_q(I_{rank^{q',q}=k}) \tag{17}$$

この評価値が一番大きな検索条件をユーザに推薦する．

4 実　　験

3 章で述べたランキングと条件推薦の手法の有効性を検証するために，実験を行う．ランキングの有効性の検証では，ユーザの入力した検索条件と検索ログに基づいて，実際にその検索条件を入力したユーザがとったアクションの順に並び替えられたアイテムの順序が，提案手法のランキング結果とどれだけ一致しているかを評価し，これをベースラインとなるランキング手法と比較する．また，検索条件の推薦の有用性の検証においても，ユーザの入力した検索条件とその後の検索ログの情報を利用し，そのユーザが検索条件を変更した後にアクションを行ったアイテムが解に含まれるようになるような検索条件をどれだけ提示できているかを検証する．

4.1 実 験 概 要

本研究で用いる不動産データは，株式会社 LIFULL から提供を受けた物件と検索ログのデータを利用する．実験に用いたデータログ中の検索数は 139,517 件，クリック数は 165,608, クリックに関わった物件数は 113,579 であった．

4.1.1 ランキングの比較実験

提案手法では，物件の情報を特徴ベクトルに変換したものをデータとして用いる．特徴ベクトルの各項目は，徒歩距離・占有面積・家賃・築年数・建物構造・敷金礼金が合計何ヶ月分か・特徴の数・新築・2 階以上・南向き・駐車場の有無・オートロック・エアコンの有無・バストイレ別・追い焚き機能・フローリング・ペット可の 12 項目である．数値データの属性は，アイテム集合全体のその属性値について，標準化したものを特徴ベクトルの要素として扱う．また，また，一つでも抜け落ちている項目がある物件データは学習データから取り除いた．

比較対象のベースラインとして，その検索条件でクリックしたユーザがクリックした回数が多い順に並び替えたクリック順と，賃貸料などの数値項目のうち一つの項目でソートしたランキング手法４種の計５つを用いる．属性のソートは以下の４種類の数値項目を対象する：賃料＋管理費・築年月・駅からの徒歩距離・専有面積.

ランキングの評価として，MAP を用い，正解はそのユーザが実際にその検索またはそれ以降の検索条件でクリックした物件とする．また，今回は検索結果の 1 ページ目に表示される数であるトップ 10 までについて評価する．

以上のようにして，本研究で提案するユーザが入力した検索条件の類似度に基づいたランキングの有効性の検証を行う．

4.1.2 検索条件推薦の検証実験

検索条件推薦の検証実験では，ユーザの入力した検索条件とその後の検索ログの情報を利用する．本実験では，そのユーザが検索条件の変更後に最終的にクリックしたアイテムが解に含まれるような検索条件をどれだけ提示できているかを検証する．

比較対象として，ユーザが変更した条件検索のうち一番多いものを用いる．例えば，5 人のユーザが検索条件 q_1 で検索を行ったが検索一覧に満足のいくアイテムが表示されなず，そのうち 2 人が q_2 を検索条件に入力し，残りの 3 人は q_3 を検索条件に入力したとする．

この時，比較するベースラインの手法である “ユーザが変更した条件検索のうち一番多いもの” とは，検索条件 q_1 に対する検索条件 q_3 である．

どちらの条件検索が優れているのかを評価は，推薦した検索条件に表示されるアイテムの上位 10 個が，推薦前の検索条件を入力したユーザにとってどれだけ価値のあるものかに基づいて行う．また，その検索条件を入力したユーザがそのアイテム

提案手法	**0.031**
クリック順	**0.0351**
徒歩距離順	0.000114
家賃順	0.000103
広さ順	0.000272
築年数順	0.0000977

図 1　MAP によるランキング手法の評価結果

に対してどれだけ価値を感じるかは，その検索条件を入力したユーザが検索した検索条件に関わらずそのアイテムをクリックした回数に基づいて評価する．すなわち，検索条件 q を入力したユーザに対する検索条件 q' の評価を $score(q'|q)$, アイテム I_i に対する評価を $score(I_i|q)$ と表し，以下のように考える．

$$score(q|q_{before}) = \sum_{I_i \in top10(q)} score(I_i|q_{before}) \tag{18}$$

$$score(I_i|q) = \sum_{q' \in Q_{after}(q)} y_i^{q'} \tag{19}$$

ここで，$top10(q)$ とは，検索条件 q で検索したときに表示される上位 10 個のアイテムの集合である．今回の実験では，ランキング手法が評価に与える影響をなるべく少なくするために，検索条件 q を満たすアイテムをクリック数が多い順に並び替えたものとする．また，y_i^q は，検索条件 q で検索した際にアイテム I_i がクリックされた回数，$Q_{after}(q)$ は検索条件 q を入力したユーザがそれ以降入力した全ての検索条件の集合であり，重複があったとしても重複した数だけ検索条件を要素に含むとする．

この評価値を基にして，ベースラインとなる比較手法と，本研究の提案手法のどちらの条件推薦がユーザにとって有意義な推薦になっているのかを検証する．

4.2 実験結果

4.2.1 ランキングの実験結果

代表的な検索条件を選択し，RankNet [1] を用いた学習器を作成した．今回用いたランキング手法はその検索条件におけるクリック数上位 20 件を提案するランキング関数で並び替えたものである．比較対象のベースラインとして，代表的な 4 つの数値項目（賃料＋管理費・築年月・駅からの徒歩距離・専有面積）をそれぞれを一つの項目でソートしたランキング手法を用いた．また，クリック数の多い順で並び替えたランキングも比較のベースラインとして用いた．以上 5 つの手法をベースラインとし，提案手法との比較実験を行った．MAP による評価の結果は，図 1 のとおりとなった．

単純なクリック順によるソートがやはり良い結果となった．しかし，これは，本来評価したかった “ユーザの好みの物件である物件が上位に表示されるランキングであること” よりも，“ユーザの検索結果に表示された物件が上位に表示されるランキングであること” つまり，ユーザの目につく頻度が高い物件を上位にランキングしたことによる影響が強く残ったためではないかと考える．このため，クリックされる物件は，ユーザは自分の嗜好に合う物件であることも重要であるが，同時に，検索結果一覧の上位に表示された物件であるという条件も満たす必要がある．その検索条件でクリックされた回数が多い順で上位に並び替えられた物件は，この “ユーザの検索結果に表示された” を強く満たすため，クリック順でのランキングの MAP は高い数値を示したのだと考える．

	提案手法 1	提案手法 2	ベースライン
正規化した評価値の平均値	**26.56**	23.52	22.24

図 2　条件推薦における各手法の結果

一方で，提案手法によるランキングが有効に働いたケースも存在する．

検証データのうち，提案手法のランキングと，クリック順に並び替えたベースラインのランキングの AP (Average Precision) が異なる値をとったのは，203 件存在する．このうち，ベースラインが優位な値となったのは，このうち 107 件であり，提案手法のランキングが優位な値となったのが 96 件であった．

4.2.2 検索条件推薦の実験結果

実験で用いる条件推薦を行う検索条件は，なるべく多様性があり，かつ，ユーザが入力する機会も多い検索条件になるように，4.2.1 節で取得した 100 個の検索条件を利用した．

提案手法で推薦する検索条件は，与えられた検索条件を q_{before} とした時，式 (18) で示した値が最も高くなるような q である．今回利用した検索ログの数は，139,517 件存在し，そのうち，検索条件は 1,814 種類であった．この検索条件の中から，一番式 (18) の値を高くする検索条件を推薦する．また，比較手法のベースラインは，その検索条件を入力した後に入力する回数が多い検索条件を推薦する検索条件として扱うような手法をとる．

推薦する検索条件の評価は，推薦した検索条件に表示されるアイテムの上位 10 個が，推薦前の検索条件を入力したユーザにとってどれだけ価値のあるものかに基づいて行い，式 (18), (19) の値を推薦する検索条件の評価値として考える．

実験の評価は，この 100 個の検索条件について，それぞれの手法が提示する検索条件を評価することで検索条件の推薦の有用性を検証する．

式 (16) で示した提案手法 1，式 (17) で示した提案手法 2，ベースラインそれぞれについて，検索条件ごとに最も良かった手法の成績を 1 となるように正規化した評価値の平均は，図 2 のようになった．提案手法 1 が一番高い値となり，次いで提案手法 2 が高い値をとった．このことから，提案手法による条件の推薦が有効な手法であるということが言える．

5 まとめ

本研究では，賃貸不動産物件検索のような，多くの属性に対する条件を指定して行う検索を対象とし，満足のいくアイテム

をユーザが探し当てることを容易にするための二つの手法を提案した．

一つ目は検索条件に依存するランキングである．過去に同じ条件や類似する条件で検索した他のユーザが検索解中のどのアイテムにクリックしたかのデータからランキングを学習する．同様の手法はキーワード検索では既に存在するが，属性条件による検索では，検索の類似度をどのように定義するかが問題となる．本研究では，検索条件をベクトルで表現したもののユークリッド距離を求め，この距離が小さいほど類似度が高くなるように定義した．検索条件のベクトル化は，過去にどのような条件で検索した人の間で，クリックするアイテムが類似していたかに基づき学習する．比較対象のベースラインとして，主要な数値検索条件でソートをした順にランキングをする手法と，入力された検索条件で検索した時にクリックされた総回数でランキングした手法を用いる．数値のソートに用いる検索条件項目は，駅からの徒歩距離・家賃・広さ・築年数の 4 項目である．MAP による評価を行った結果，提案手法・クリック回数によるソート・駅からの徒歩距離・家賃・広さ・築年数についてのソートの評価値はそれぞれ，0.031・0.0351・0.000114・0.000103・0.000272・0.0000977 となり，提案手法とクリック回数によるソートでのランキングが優れた値となった．特にクリック回数によるランキングは高い評価値を取ったが，テストデータの検索条件それぞれをについての評価を確認すると，提案手法の方が優れた値を取ったものが 96 件存在し，ベースラインの方が優れた値を取ったものが 107 件存在するなど，場合によって提案手法が良いランキングを提供することがわかった．

二つ目は，検索条件推薦の手法である．ユーザが検索条件を繰り返し変更して，最終的に自分の満足のいくアイテムを解に含む検索条件にたどり着くという手間を省くため，ユーザにとって良いと思われるアイテムが解に含まれるような検索条件を推薦する．上述のランキング手法の中で作成したアイテムの評価関数を利用して，初めにユーザが入力した検索条件の解には含まれていなかったが，高い評価地となるアイテムを発見し，これを解として含むような検索条件を推薦する．その際，ユーザが当初入力していた検索条件から外れるほど，ユーザの意図に反するものになる可能性が高いことを考慮し，なるべくユーザにとって良いアイテムが出て，かつ，なるべく元々ユーザが指定していたものに近い検索条件を推薦する．検索条件の推薦については，二つの手法を提案した．一方は，推薦する検索条件の評価全体にその検索条件がユーザの意図に沿っている確率をかけたもの．もう一方が，推薦する検索条件に含まれるアイテムそれぞれにそのアイテムがそれぞれ出てき得る検索条件がユーザの意図に沿っている確率をかけたものである．検索条件ごとに最も良かった手法の成績を 1 となるように正規化した評価値の平均をとったところ，前者の提案手法が一番高い値，後者の提案手法が次いで高い値となり，提案手法のいずれもがベースラインを上回った．

6 謝　　辞

本研究では，国立情報学研究所の IDR データセット提供サービスにより株式会社 LIFULL から提供を受けた「LIFULL HOME’S データセット」を利用した．また，本研究は JST CREST (JPMJCR16E3)，JSPS 科研費 18H03245 の支援を受けたものである．

文　　献

[1] Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. Learning to rank using gradient descent. In *Proceedings of the 22nd international conference on Machine learning*, pp. 89–96, 2005.

[2] Jiafeng Guo, Xueqi Cheng, Gu Xu, and Xiaofei Zhu. Intent-aware query similarity. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, pp. 259–268, 2011.

[3] Thorsten Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 133–142, 2002.

[4] Qiaozhu Mei, Dengyong Zhou, and Kenneth Church. Query suggestion using hitting time. In *Proceedings of the 17th ACM conference on Information and knowledge management*, pp. 469–478, 2008.

[5] Filip Radlinski and Thorsten Joachims. Query chains: learning to rank from implicit feedback. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pp. 239–248, 2005.

[6] Deng Yi, Yin Zhang, Haihan Yu, Yanfei Yin, Jing Pan, and Baogang Wei. Improving multi-faceted book search by incorporating sparse latent semantic analysis of click-through logs. In *Proceedings of the 12th ACM/IEEE-CS joint conference on Digital Libraries*, pp. 249–258, 2012.

[7] Qiankun Zhao, Steven CH Hoi, Tie-Yan Liu, Sourav S Bhowmick, Michael R Lyu, and Wei-Ying Ma. Time-dependent semantic similarity measure of queries using historical click-through data. In *Proceedings of the 15th international conference on World Wide Web*, pp. 543–552, 2006.

[8] 塩澤秀和, 三田村祐介ほか. 食材の優先度を考慮した料理レシピの検索. 情報処理学会研究報告ヒューマンコンピュータインタラクション (HCI), Vol. 2007, No. 41 (2007-HCI-123), pp. 51–57, 2007.

[9] 今井良太, 戸田浩之, 関口裕一郎, 望月崇由, 鈴木智也, 今井桂子. Web 検索サービスにおける多義的なクエリ推薦手法. *DBSJ Journal*, Vol. 9, No. 1, pp. 1–6, 2010.

[10] 清水仁, 岩田具治. 商品検索の検索ログを用いたマルチタスク学習. 人工知能学会全国大会論文集 第 34 回全国大会 (2020), pp. 1E4GS904–1E4GS904. 一般社団法人 人工知能学会, 2020.