WWWからの属性・属性値情報ページの検索

吉田　稔 †‡　中川　裕志 †‡
† 東京大学情報基盤センター　‡ CREST
{mino,nakagawa}@r.dl.itc.u-tokyo.ac.jp

1　はじめに

「人間」「ＰＣ」等、ユーザーが探したい**オブジェクト**の種類（カテゴリ）を入力とし、「自己紹介」「ＰＣカタログ」等、「希望のオブジェクトが記述されたページ」を出力とする、新しい Web ページ検索を提案する。

例として、Web 上から、自己紹介ページを検索することを考える。このとき、「自己紹介」というクエリで検索するだけでは発見できないページ（すなわち、「自己紹介」という語を含まないが、自己紹介をしているページ）が多く存在し、このため、既存のテキスト検索エンジンを用いるのみでは、再現率に問題がある。

しかし、システムが検索対象のオブジェクトについて何らかの知識を持っている場合、状況は異なってくる。例えば、人間オブジェクトは、「年齢」「性別」「趣味」等の**属性**で特徴づけることができる。このような属性、あるいはその属性値（「29 歳」等）がページに表記されていれば、そのページが人間のプロフィールを示したものである可能性が高い。本研究では、そのような知識を、**HTML テーブルから抽出されたオントロジー**[1][1] の形で与えることを目標とする。すなわち、実現すべきシステムは、「**オントロジーをクエリとし、それに適合する Web ページを検索するシステム**」となる。

本手法の応用分野としては、例えば、情報抽出システムへ入力する Web ページを収集する為の前処理等に用いることが考えられる。

本タスクは、情報検索あるいは文書分類の一形態として捉えることもできる。しかしながら、既存手法の多くは、クエリ或いは訓練データと、テストデータの間に同一のモデルを仮定している。一方、本タスクは、オントロジーという構造化された知識を用いて、統一的な構造を持たない文書を検索するため、この点において、既存手法をそのまま適用することは難しい。本論文では、オントロジーの属性・属性値構造を効率的にモデル化する手法を提案する。

[1]本研究におけるオントロジーは、属性・属性値の形で与えられる知識ベースであり、通常のオントロジーとは形式が異なることに注意されたい。

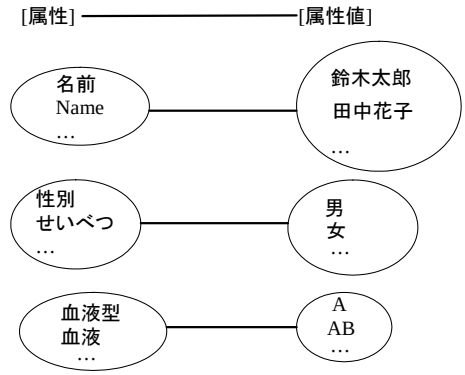

図 1: 人間オントロジーの例

題名	作曲者	価格 (円)
Sonata No. 1	Mozart	2500
Symphony No. 9	Beethoven	1800
Nocturne No. 1	J.S.Bach	1500

図 2: CD を列挙したテーブル

2　問題設定

システムは、Web ページの集合 B, カテゴリー C を表すオントロジー o を入力とし、B 中の Web ページ p を、C への適合度（スコア）に従ってランク付けしたリストを出力する。スコアは、$P(C_+|p)$ で表される。ここで、C_+ は、「カテゴリー C に属する」という事象を表す。

2.1　オントロジー

本研究におけるオントロジーは、カテゴリー C のオブジェクトを属性・属性値で表現した知識ベースである。属性とその属性値はペアとなっており、それぞれは文字列の集合で表現される。図 1 にカテゴリー「人間」のオントロジーの例を示す。

オントロジーは、HTML テーブルの集合から抽出さ

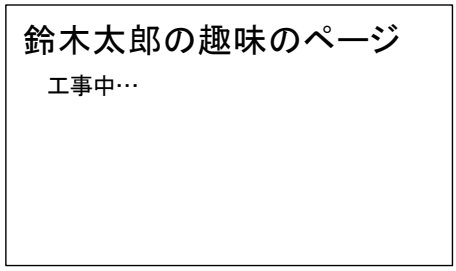

図 3: 問題となるページ例 1

図 4: 問題となるページ例 2

れる。HTML テーブルは、図 2 の例に見られるように、属性と属性値によりオブジェクトを記述する表現形態であり、ここから、あるカテゴリーにおける典型的な属性・属性値を抽出することができる。（詳細については、論文 [1] を参照されたい。）

属性（とその属性値）は、元となる HTML テーブル集合における出現頻度に応じてランク付けされる。そのうち、頻度の高い順に一定数（現在は 10 ペア＝ 20 個）を検索に用いる。

2.2 Web ページ

Web ページやオントロジーのモデル化に際して、一般的な、単語の出現頻度に基づいたベクトル空間モデルや、生成モデルを用いることが考えられる。これらのモデルを用いた場合、オントロジーに於いて出現頻度の高い単語が Web ページ中にも多く出現する程、高いスコアが与えられることになる。（一方、出現頻度の低い単語が出現するほど、スコアは低くなる。）しかしながら、これらのモデルでは、以下のような場合に問題が起きる。

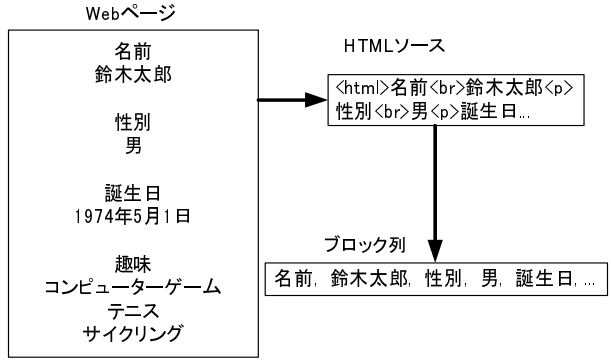

図 5: ブロック列への変換例

ケース 1 図 3 のような場合、「鈴木太郎」と「趣味」という、自己紹介カテゴリに適合する文字列がページの多くの部分を占めるため、スコアが高くなってしまう。

ケース 2 例えば、更新履歴のページのように、日付が多く含まれるページでは、日付が自己紹介カテゴリ（例えば「誕生日」として）に適合する文字列であるため、スコアが高くなる。

ケース 3 図 4 のような場合、人間オブジェクトと関係のない大学名がページの大半を占めるため、スコアが低くなる。

本研究で提案するアルゴリズムは、以上のような欠点を補うため、

- Web ページ全体でなく、その一部を検索 (ケース 3 への対策)
- 「高頻度」でなく「多種類」を高く評価するモデルの採用（ケース 1,2 への対策）

という方針に基づき設計されている。次節では、アルゴリズムを具体的に解説する。

3 アルゴリズム

アルゴリズムは「ブロック列への変換」「部分ブロック列の選択」「スコア計算」という 3 つの手順で動作する。以下、各手順について解説する。

3.1 ブロック列への変換

システムは、最初のフェーズにおいて、与えられた Web ページを**ブロック列**に変換する。ブロック列は、Web ページの HTML ソースファイルを、HTML タグを境界として[2]、**ブロック**と呼ばれる小領域の列に分解

[2] このほか、“:” 等、幾つかの特別な文字についても境界と見做す。

することで得られる。図 5 にブロック列への変換例を示す。

ブロックは、通常の文章における「文」に相当する、Web ページ表現における単位である。例えば、「私の名前は鈴木太郎です。年は 25 です。」という文章では、最初の文が名前に関する情報を、後の文が年齢に関する情報を提供している。この場合、一つの文を一つの情報提供の単位と見做すことができ、句点を情報の区切りとすることができる。これに対し、Web ページ上の文章以外の表現では、図 5 に見られるように、句点で情報を区切ることができない。この場合、HTML タグを区切りとすることが有効となる。

3.2 部分ブロック列の選択

システムは、ブロック列に対してスコアを計算し、それを元の Web ページのランク付けに使用する。このとき、前述のケース 3 の問題（同一ページ内に、「対象カテゴリに関係ある部分」と「関係ない部分」が含まれる問題）に対処するため、ある一定のウインドウサイズ l を設定し、ブロック列の長さが l より大きい場合は、長さ l の部分ブロック列すべてについてスコア計算をし、その最大値を元のブロック列のスコアとして用いる。（すなわち、いわゆるパッセージ検索 [2] と同様のことを行う。）現在、$l = 20$ に設定されている。これは、対象としている属性と属性値すべてが連続したブロック中に存在した場合を想定した値である。

3.3 スコア計算

自己紹介ページを探すとき、ある Web ページ中に「趣味」という単語が存在した場合、ページ中の他の部分を見るまでもなく、自己紹介ページである可能性が高い。このように、与えられたブロック列が対象カテゴリに属することの最も強い証拠となる部分（カテゴリに最も特徴的な部分）を取り出し、それをブロック列のスコアとするのが、本手法の基本的なアイデアである。この場合、スコアは、

$$P(C_+|p) \approx \max_s P(C_+|s),$$

（C_+：p が C に適合する事象、s：p 中の単語或いは単語列[3]）と表される。これは、単語列の存在をルールとし、$P(C_+|s)$ を信頼度とする決定リスト [3] の一種と見做すこともできる。

しかしながら、以上のように、一つの手掛かりのみを用いる方法は、信頼性にやや欠ける。上の例では、「趣味」という単語を含むページが全て自己紹介と判定されることになれば、「趣味の園芸」等、少なからぬ数の自己紹介と無関係の文字列が、誤って利用される可能性が高い。

この問題を、上記のスコア値に**線形補間**を適用することにより回避する。オントロジー中の属性（あるいは属性値）の集合を A とし、A 中の各属性（値）a について

$$\max_s P(a_+|s)$$

なるスコアを計算する。ここに a_+ は、入力ページ p が属性（あるいは属性値）a に適合する事象を表す。これらのスコアを用い、カテゴリ C 全体と p の適合度を、

$$P(C_+|p) \approx \frac{1}{|A|} \sum_{a \in A} \max_s P(a_+|s)$$

で計算する[4]。

$P(a_+|s)$ の値の計算には、オントロジー以外のページ集合 D_o を収集し、それを負例として用いる。

- n_a：s が属性 (値)a 中に現れる回数
- n_o：s が D_o（又は、オントロジー中の a 以外の属性）中に現れる回数

としたとき、$P(a_+|s)$ の値は、以下で計算される。

$$\hat{P}(a_+|s) = \frac{n_a}{n_a + n_o}.$$

現在、D_o としては、入力 Web ページの集合 B を用いている。このとき、$n_o > 0$ が満たされるため、「偶然オントロジー中にのみ出現する単語列」（このとき、$n_o = 0$ により、$\hat{P}(a_+|s) = 1$ となる）を不当に高く評価することを避けられる。

提案手法は、生成モデルやベクトル空間モデルと異なり、同一ページに同一の属性（値）が複数出現した場合でも、そのうちの一回分のみをスコア計算に考慮する。例えば、日付を表す文字列がページ中に複数回登場した場合でも、「誕生日」としてスコア計算に用いられるのはそのうちの一回分のみである。

提案モデルの特長は、文字列 s の**特異性**（s が目的のカテゴリに特徴的に現れるか）と**多様性**（多くの種類の属性が表現されているか）を、前者は確率 $P(C_+|s)$ として、後者は全属性間の線形補間として、単純な形で同時にモデル化できるということである。この単純性により、様々なカテゴリ・Web ページに適用可能な頑強なモデルとなることが期待できる。

4 実験

「人間」と「PC」の 2 種のカテゴリについて実験を行った。各カテゴリについて、Web 上の HTML テーブル[5]から抽出したオントロジーを用意した。

検索対象としたのは、合計 3,317 個の Web ページである。全てのページについて正解・不正解を判定する

[3] 現在、処理効率のため、単語列の長さは 1 又は 2 のみを扱っている。

[4] ここでは、$\forall a \in A_N : a_+ \Leftrightarrow C_+$ を仮定している。

[5] 人間カテゴリでは 1,327 個、PC カテゴリでは 68 個

手法/使用した属性	AV	A	V
提案手法 (*Window*)	**0.591**	0.405	0.247
提案手法 (*Naive*)	0.197	**0.250**	0.102
ベクトル空間モデル	0.0467	**0.152**	0.0128

図 6: 人間カテゴリにおける平均精度

手法/使用した属性	AV	A	V
提案手法 (*Window*)	**0.837**	0.461	0.746
提案手法 (*Naive*)	**0.488**	0.351	0.481
ベクトル空間モデル	0.208	**0.437**	0.282

図 7: PC カテゴリにおける平均精度

のは高コストであるため、pooling[4] を用いて、一部のページのみについて判定を行う。具体的には、実験に用いた各手法が返した、上位３０個のページを集め、正解セットとする。正解セット内のページのみについて正解・不正解（対象カテゴリに適合するか否か）を人手で判定し、正解セット以外のページはすべて不正解と見做す。

ベースラインとして、ベクトル空間モデルによる実装も行った。ベクトルの各次元の値は、単語列[6]の頻度[7]を用い、ベクトル間の距離はコサイン値で定義される。

また、検索に用いる属性（値）選択の影響を調べるため、**AV**（属性・属性値両方を使用）、**A**（属性のみ使用）、**V**（属性値のみ使用）の３つの場合それぞれについて精度を測定し、比較を行った。

図 6 と図 7 に、それぞれの手法の**平均精度** [5] を示す。平均精度とは、以下の式で定義される尺度である。

$$\frac{1}{|D_q|}\sum_{1\leq k\leq |D|} r_k \cdot precision(k)$$

ここに、D_q は適合文書全体、D は文書全体の集合である。$precision(k)$ は、上位 k 個の文書における精度であり、r_k は、ランキング k 位の文書が目的のカテゴリに適合している場合に 1, 適合しない場合に 0 となる値である。

表中、*Window* は、提案手法において、部分ブロック列の選択を行った場合を、*Naive* は、選択を行わなかった場合（ブロック列全体をそのままスコア計算に用いた場合）をそれぞれ表す。

ベクトル空間モデルから *Naive* への精度向上は、前述の「特異性」（オントロジー外の情報の利用）と「多様性」（属性（値）全般に渡った線形補間）をモデルに組み込んだことによる効果と考えられる。

また、部分ブロック列選択を行うことにより、精度の大幅な改善が見られた。これは、日記のように比較的大きなサイズのページ中に、カテゴリに適合する文字列がページ全体に散らばった場合、*Naive* ではスコアが高くなる現象を、*Window* では回避できたことが主な理由であった。

属性（値）の選択による効果を見ると、ベクトル空間モデルが **A** で最も良い結果を出したのに対し、提案手法では、**AV** で最も良い結果となった。これは、属性値を検索に用いなければ、ベクトル空間モデルでも比較的良い精度を出すが、属性値まで含めて検索に利用するためには、提案手法のような工夫が必要となることを示唆している。

[6]提案手法と同一の素性

[7]HTML テーブルと Web ページ間で共通の DF 値を定義することが自明でないため、TFIDF は使用していない。

5 おわりに

HTML テーブルから抽出したオントロジーをクエリとして用いることによる、スペック情報（属性・属性値情報）を載せた Web ページを検索するタスクと、それに対する有効なアルゴリズムを提案した。ベクトル空間モデルとの比較実験では高い精度を実現したが、手法の優位性の実証には、より詳細な実験が必要と思われる。そのほか、様々な種類のオントロジー（特に、1 個や 2 個等、少量の HTML テーブルから抽出されたオントロジー）による実験も、今後の課題である。

参考文献

[1] M. Yoshida, K. Torisawa, and J. Tsujii, 2001. “Extracting Ontologies from World Wide Web via HTML tables.” *Proceedings of PACLING2001*: pp. 332–341.

[2] Salton, G. and Allan, J. and Buckley, C. 1993. “Approaches to Passage Retrieval in Full Text Information Systems.” *Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*: pp. 49–58.

[3] Rivest, R. 1987. “Learning decision trees.” *Machine Learning, 2*:229–246.

[4] Gordon V. Cormack, Christopher R. Palmer, and Charles L. A. Clarke. 1998. “Efficient construction of large test collections.” *In Proceedings of the 21st International ACM SIGIR-98*

[5] Soumen Chakrabarti. 2002. “Mining the Web : Discovering Knowledge from Hypertext Data.” Morgan-Kaufmann Publishers