原著論文

Original Paper

属性値間の関連性を用いた属性抽出の精度向上

Improving the Accuracy of Attribute Extraction using the Relatedness between Attribute Values

ボレガラ ダヌシカ
Danushka Bollegala
東京大学大学院 情報理工学研究科
Graduate School of Information and Technology, The University of Tokyo
danushka@iba.t.u-tokyo.ac.jp, http://www.iba.t.u-tokyo.ac.jp/~danushka

谷 直紀 *1
Naoki Tani
（同　上）
tani@mi.ci.i.u-tokyo.ac.jp

石塚 満
Mitsuru Ishizuka
（同　上）
ishizuka@i.u-tokyo.ac.jp,http://www.miv.t.u-tokyo.ac.jp/ishizuka/

keywords: attribute extraction, information extraction, web mining

Summary

Extracting attribute-values related to entities from web texts is an important step in numerous web related tasks such as information retrieval, information extraction, and entity disambiguation (namesake disambiguation). For example, for a search query that contains a personal name, we can not only return documents that contain that personal name, but if we have attribute-values such as the organization for which that person works, we can also suggest documents that contain information related to that organization, thereby improving the user's search experience. Despite numerous potential applications of attribute extraction, it remains a challenging task due to the inherent noise in web data – often a single web page contains multiple entities and attributes. We propose a graph-based approach to select the correct attribute-values from a set of candidate attribute-values extracted for a particular entity. First, we build an undirected weighted graph in which, attribute-values are represented by nodes, and the edge that connects two nodes in the graph represents the degree of relatedness between the corresponding attribute-values. Next, we find the maximum spanning tree of this graph that connects exactly one attribute-value for each attribute-type. The proposed method outperforms previously proposed attribute extraction methods on a dataset that contains 5000 web pages.

1. ま え が き

Web が普及するに伴い，Web 上に大量の情報が存在するようになった．その内容は多岐にわたり，企業株価からおすすめのレストランまで，ほぼ全ての分野の情報を見ることができる．また，Wikipedia や Blog，Social Networking Service (SNS) などを通じて，一般のユーザから提供される情報も増え続けている．そのため，人が何か調べ物をするときには，検索エンジンを用いてウェブ上の情報を利用することが当たり前になっている．

しかし，個々のユーザが独自に情報発信をしているため，関連する情報が様々な Web ページに分散して保存されてしまっている．また，Web 上のドキュメントは新聞や雑誌などのように整ったものばかりではなく，内容や形式も多様である．そのため自分の欲しい情報を手に入れるためには時間や手間といった大きなコストを支払わなければならない．

この問題を解決する一つの方法として，Web から必要な情報を抽出してデータベースを作成しておくことが考えられる．商品情報のデータベースがあれば価格や性能の比較が容易に行なえるようになり，買い物などが楽になる．人物データベースが作成できれば，検索エンジンで様々な Web ページを閲覧して情報を集める必要がなくなり，データベースを人名で検索するだけでその人に関する情報を全て取得することができるようになるであろう．また，ユーザの「オバマ大統領の出身地は？」のような質問にもデータベースを用いて的確に回答することが可能になる．つまり必要な情報を取得する情報抽出 (Information Extraction, IE) タスクにおいても，ユーザの質問に的確な回答を返す (QA) タスクにおいても，Web をデータベースとして扱う技術が重要であると言える．

Web をデータベースとして扱うために必要となるのが属性抽出 (Attribute Extraction, AE) 技術である．属性とは，ある種類のエンティティに共通して備わっているとされる性質や特徴のことで，人物の属性を考えると，身長や体重，生年月日などが対応する．この属性に対応する値のことを属性値と呼ぶ．例えば，“バラク・オバマ”

†1　現在は，楽天株式会社に勤務．

というエンティティは “職業” という属性を持っており，その属性値は “政治家” である．エンティティが与えられた時にそのエンティティの所有している属性を抽出するタスクと，エンティティ・属性が与えられたときに対応する属性値を抽出するタスクの２つを属性抽出と呼ぶ．また，エンティティではなくクラス (“人間” などの上位語) を与えられたときにクラスの属性を抽出することをクラス属性抽出 (ClassAE)，エンティティの属性を抽出することをインスタンス属性抽出 (InstanceAE) と分類する事もある．

Web からエンティティに対する属性値を抽出する場合に解決しなければならない課題が複数存在する．Web 上ではテキストで書かれた非構造的な情報を処理する必要がある．そのため正規表現のような事前に指定したパターンだけで抽出できない属性値が存在する．尚，複数の人物に関する属性情報が一つのページに書かれている場合，昔の情報が更新されていない場合，間違った属性情報が記載されている場合なども扱う必要がある．これらの原因から間違った属性値が多く抽出されてしまい，属性抽出の精度が下がってしまうという問題がある．例えば，Web People Search Evaluation Workshop (WePS) という，Web 上のデータを対象とした属性抽出の評価型ワークショップでは，最も良い属性値抽出手法の F-スコアは 0.12 と低く，属性抽出の難しさが分かる [Sekine 09].

では，どのようにすれば属性抽出の精度を向上させることができるのだろうか．ここで，人間が文書から属性を抽出する方法について考えてみよう．例えば，バラク・オバマの誕生年を知りたいとする．バラク・オバマの職業が “政治家” であると分かっている場合には，もし同じページに「2001 年生まれ」という言葉が書かれていたとしても，これがバラク・オバマの誕生年であると考える人はいないだろう．なぜなら，アメリカでは 25 歳以上にならないと政治家になれないため，一人の人物が “職業” : “政治家” と “誕生年” : “2001 年” という 2 つの属性値を同時に持つことはできないからだ*1．同様に，文書中に “スポーツ選手”, “ノーベル賞”, “金メダル” という単語が出現した場合，我々は「スポーツ選手がノーベル賞を取る確率は低い」という知識を持っているため “職業” : “スポーツ選手”, “受賞歴” : “金メダル” という属性値が正解であると分かる．このように，人間があるエンティティの属性を判断する場合には，1 つの属性値だけを見るのではなく，複数の属性値同士の関連を考慮して正しい属性値を選択しているといえる．

本稿では，ある属性抽出手法を用い，人物に関して抽出した属性値候補の中からその人物の正しい属性値を選択する手法を提案する．従って, 提案手法は単独で属性抽出を行うのではなく，何らかの属性抽出手法と組み合わせてその属性抽出の精度を向上するための手法である．提案手法ではまずある人物に関して抽出された属性値をノードとする重み付き無向グラフを構築する．グラフの２つのノードを結ぶエッジの重みはそれらのノードに対応する属性値間の関連性を表すものとする．次に, ある人物に関する属性値を選択する問題を，上記のように作成したグラフ上で全ての属性をカバーし，かつ，属性値間の関連性が最大となる部分木を選択する問題として定式化する．提案手法の性能を評価するために，Web 上の 5,000 文書を対象にした属性抽出の実験を行った結果，人名からの距離や出現頻度を利用して属性を選択する既存手法に比べ，提案手法はより高い精度で属性値選択が可能であることが分かった．

*1 2012 年時点での状況を想定している

2. 属性値間の関連性

あるエンティティ E には n 個の属性 $A_1, A_2, \ldots, A_n$ が存在すると仮定しよう．例えば，E が人物である場合 A_1 は「氏名」，A_2 は「生年月日」，A_3 は「国籍」といった属性として解釈できる．更に，ある属性値抽出手法を用い，これらの属性に関する属性値を抽出できていると仮定する．属性 A_i に関して抽出した属性値の集合を $\mathcal{S}(A_i)$ で表す．尚，属性 A_i の属性値を表す変数を $a_i \in \mathcal{S}(A_i)$ とする．例えば，属性 A_3(国籍）に関しては a_3 は「日本」という属性値を取ることが可能である．以下，議論を簡単化するため，本論文では E に関する各属性に関して正しい属性値は一つしか存在しないと仮定する．この制約を緩める方法について後ほど述べる．尚，エンティティ E の属性 A_i に関する正しい属性値を a_i^* で表す．上記の記号を用い，E に関する各属性の属性値集合の中からそれぞれ正しい属性値を選択する問題を，属性値間の関連性を表すスコア関数 Γ の値を最大化する属性値の組み合わせを選択する問題として定式化する．つまり，E に関する正しい属性値の組み合わせ $(a_1^*, \ldots, a_n^*)$ は次の式 1 で与えられるものとする．

$$\mathrm{argmax}_{(a_1,\ldots,a_n)\in\mathcal{S}(A_1)\times\ldots\times\mathcal{S}(A_n)}\Gamma(a_1,\ldots,a_n) \qquad (1)$$

もちろん，実世界には同じ属性に関して複数の属性値を持つエンティティが存在する．例えば，二重国籍を持つ人物や，複数の職業を持つ人物はそうである．そのようなケースを扱うためには上記の定式化を２通りの方法で拡張できる．一つの拡張方法として「本国籍」,「副国籍」,「本業」,「副業」など新たな属性を定義すれば，上述した一つの属性に関する正しい属性値は一つしか存在しないという制約を導入することができるので式 1 の定式化をそのまま適用できる．もう一つの拡張方法として Γ の最大値に対応する属性値の組み合わせだけではなく，n-best 出力 (Γ の値が大きい順でソートした場合の上位 n 個の属性値の組み合わせ）を選択することも可能である．

属性値間の関連性を表す関数 Γ を式 2 で示してある通

表 1 式 2 で sim(x,y) として使用する関連度関数の定義と，出現頻度 $h(x)$,$h(y)$ と共起頻度 $h(x \cap y)$ を用いる計算方法．

類似度尺度	定義	計算方法
Jaccard 係数	$\frac{\lvert x \cap y \rvert}{\lvert x \cup y \rvert}$	$\frac{h(x \cap y)}{h(x)+h(y)-h(x \cap y)}$
Dice 係数	$\frac{2\lvert x \cap y \rvert}{\lvert x \rvert + \lvert y \rvert}$	$\frac{2h(x \cap y)}{h(x)+h(y)}$
Overlap 係数	$\frac{\lvert x \cap y \rvert}{\min(\lvert x \rvert, \lvert y \rvert)}$	$\frac{h(x \cap y)}{\min(h(x), h(y))}$
PMI	$\log \frac{P(x,y)}{P(x)P(y)}$	$\log \frac{\frac{h(x \cap y)}{N}}{\frac{h(x)}{N} \times \frac{h(y)}{N}}$

り，各属性値対の関連性の重み付き平均として近似する．

$$\Gamma(a_i, \ldots, a_n) = \sum_{i=1}^{n} \sum_{j=i+1}^{n} w_{ij} sim(a_i, a_j) \quad (2)$$

式 2 で $sim(a_i, a_j)$ は属性値 a_i と a_j 間の関連性を表す関数とする．尚，w_{ij} は属性 A_i と属性 A_j に関する重みであり，$\sum_{i=1}^{n} \sum_{j=i+1}^{n} w_{ij} = 1$ とする．本稿ではどのような属性対にどれくらいの重みを割り当てるかということを事前に指定せず，どの属性対も同様に扱う．そのため本稿では，$w_{ij} = 2/n(n-1)$ とする．

２つの属性値 x,y 間の関連度関数 $sim(x,y)$ として表 1 にまとめてある Jaccard 係数，Dice 係数，Overlap 係数，pointwise mutual information (PMI) の 4 つの関数を使う．表 1 では x の出現頻度 $h(x)$,y の出現頻度 $h(y)$,x と y の共起頻度 $h(x \cap y)$ を使ってこれらの関連度関数を計算する．$h(x \cap y)$ として x と y の AND 検索に関するヒット件数を用いる．これらの関連度尺度は Web から関連度を計算する先行研究では広く使われている [Bollegala 07]. 尚, 表 1 で N は検索エンジンがインデックスしているウェブページ数で $N = 10^{10}$ とした．

出現頻度 h を求めるソースとして，提案手法では Web 検索エンジンと Freebase[*2]の 2 種類のリソースを使用する．検索エンジンを利用する場合には，属性値をクオテーションマークで囲ったクエリを作成し，そのクエリを検索した場合のヒット件数を共起頻度とする．例えば “オバマ”, “大統領” という 2 つの属性値の関連度を求めたい場合には，「“オバマ” “大統領”」というクエリを作成し，Web で検索する．ここで属性値候補をクオテーションマークで囲っているのは，単語が分解して検索されるのを防ぐためである．もしこのクオテーションマークが無いと属性値候補が自動的に分解して検索されてしまい，正しいヒット数を得ることができない．検索エンジンには Yahoo! Search BOSS[*3]を利用した．

もう 1 つのソースである Freebase とは，世界の情報を集めたオープンデータベースであり，3,000 以上の 関係データベースと 1,200 万件以上のエントリーによって構成されている．不特定多数の人々が自由に編集するという点では Wikipedia と同じだが，Wikipedia では情報を文章で記述するのに対して Freebase では情報は属性–属性値のペアで記述されている．そのため，Freebase は属性抽出実験のためのデータセットに適しているといえる．Freebase から属性の共起頻度を使う場合には，同じ人物が 2 つの属性値候補を持っている場合に共起したとみなす．

Web 検索エンジンと Freebase の 2 種類の検索ヒット件数から 4 種類の関連度を計算し，組み合わせることで最終的な関連度を計算する．x,y という 2 つの属性値の関連度 $sim(x,y)$ は

$$sim(x,y) = \alpha sim_{FB}(x,y) + (1-\alpha) sim_{Web}(x,y) \quad (3)$$

と表せる．$sim_{Web}(x,y)$ は Web の検索ヒット件数を元にした関連度であり，$sim_{FB}(x,y)$ は Freebase の検索ヒット件数を元にした関連度である．$\alpha \in [0,1]$ は Web 関連度と Freebase 関連度の割合を調整するためのパラメータである．α が大きくなるほど Freebase 関連度の影響が大きくなる．Freebase は人手で作成されたデータベースであり，信頼度が高い．しかし Web 全体に比べるとエンティティ数が少ないため，カバー率が低いと考えられる．逆に，Web は膨大でカバー率は高いが，間違った属性や他のエンティティの属性が出現する可能性があるため，信頼度は低い．α を変更することでこれらの 2 つのリソースの影響を調整することができる．

3. 属性値の選択

本章では式 1 の関連性スコア Γ を最大化する属性値の組み合わせを選択する問題を考慮する．属性 A_i に関して抽出した属性値の数を $|\mathcal{S}(A_i)|$ で表すと，式 1 では $|\mathcal{S}(A_1)| \times |\mathcal{S}(A_2)| \times |\mathcal{S}(A_n)|$ 個の属性値の組み合わせを考慮しなければならない．この属性値選択問題を効率的に解くために，まず属性値をグラフ構造で表現し，次に作成したグラフ構造上で最大全域木 (Maximum Spanning Tree) を選択する問題として定式化する．

属性値選択問題を解くために，まず，抽出した属性値 a_i をノードとする重み付き無向グラフ $\mathcal{G}$ を構築する．グラフ $\mathcal{G}$ 上で属性値 a_i と a_j に対するノード同士を繋ぐエッジの重みを式 3 で与えられる $sim(a_i, a_j)$ にする．本稿では表 1 で示してあるそれぞれの関連度関数を個別に使用し，式 3 の $sim(a_i, a_j)$ を計算し，それぞれの関連度関数に対するグラフ構造を構築し，その性能評価を行う．

作成するグラフの一例を図 1 に示す．ここではエンティティ“バラク・オバマ” の属性値候補として “職業”: {“政治家”, “柔道家”}, “国籍”: {“アメリカ”, “日本”}, “出身地”: {“ワシントン”, “トロント”} が得られた場合を考える．ノードはそれぞれの属性値を表している．また点線で囲まれたノードは同じ属性に所属している属性値の集合である．式 1 で考慮する全ての属性値の組み合わせを図 1 で線で繋げてある．図 1 を見やすくするため，エッジの重みである式 3 によって計算される関連度 $sim(a_i, a_j)$ の

*2 http://www.freebase.com/
*3 http://developer.yahoo.com/search/boss/

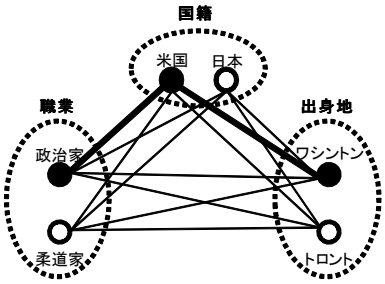

図 1 エンティティ“バラク・オバマ” に関して抽出された属性間の関連度を表すグラフ構造 $\mathcal{G}$ を示す. 抽出された属性値の中には誤りが存在するので式 1 の属性値間の関連度を表すスコア Γ を最大化する属性値の組み合わせを選択する問題として属性値選択問題を定式化する.

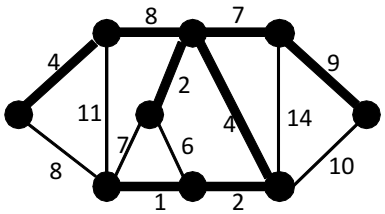

図 2 最小全域木の例

値を省略してある．図 1 では同じ属性に属しているノード間にはエッジが張られていないのは，同じ属性に所属している属性値は 1 つしか選べないという制約があるため，関連度を計算する必要がないからである．

次に，作成したグラフを利用して最も良い属性を見つけ出す方法を説明する．Γ を最大化する属性値の組み合わせを選択する問題をグラフ $\mathcal{G}$ の最大全域木問題に帰着させる．まず最初に，3·1 節では最小全域木問題を説明し，次に 3·2 節では無向グラフの最小全域木を求めるためのアルゴリズムであるプリム法 (Prim's MST Algorithm) を紹介する．最後に，3·3 節ではプリム法を応用して正しい属性値の組み合わせを選択する問題を解く手法を説明する．

3·1 最小全域木問題

連結無向グラフが与えられたとき，全域木とは木 (連結であり閉路がないグラフ構造) になっており，全てのノードを結ぶ部分グラフのことを意味する．1 つのグラフから複数の全域木を構成することができる．ここで，それぞれのエッジに重みをつけ，全域木の重みは木を構成するエッジの重みの合計で表すとすると，重みが最小となる全域木を最小全域木 (Minimum Spanning Tree, MST) と呼ぶ．図 2 に最小全域木の例を示す．黒丸で表したノードを全て通り，かつエッジの重みの合計が最小となっている．

最小部分木が利用されている有名な例としては，ネットワーク敷設工事がある．ある地域に新しくネットワー

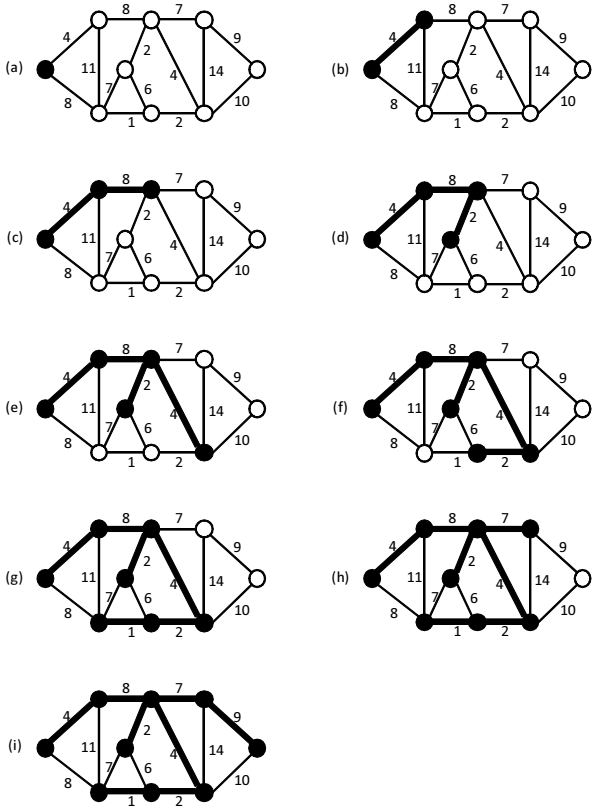

図 3 プリム法の具体例 [Cormen 01]

クを作成する際に，できるだけ費用が少なくなるように全ての世帯にネットワークを敷設したい．この場合，グラフのノードが拠点であり，エッジがネットワークケーブルになる．それぞれのエッジごとにかかる費用が定められており，これはエッジごとに異なる．閉路がないように全ての世帯にネットワークを敷設すると，全域木となる．全域木のなかで費用の合計が最も少ないものが最小全域木であり，最小全域木と一致するようにネットワークを敷設するのがよい．

3·2 プ リ ム 法

無向グラフ $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ が節点集合 $\mathcal{V}$ と各節点対上のエッジ集合 $\mathcal{E}$ によって定義されるとき，プリム法 [Prim 57] はそのような無向連結グラフから MST を構築する方法である．図 3 に示すように，プリム法は，MST に到着する (得られた被覆木がおそらく最小となる) まで 1 つづつエッジを追加して木 $\mathcal{T}$ を成長させていく．このアルゴリズムの背景にあるのは，直感的には，$u \in \mathcal{T}$ と $v \in \mathcal{V} - \mathcal{T}$ との間の最小重みを持つエッジ (u, v) は，MST に含まれるという事実である．太線になっているエッジは成長中の木のエッジであり，木のノードは黒丸で示す．プリム法の各ステップでは，木に属するノードからグラフのカットを決定し，このカットと交差する軽いエッジを木に加える．例えば，図 3 (c) では，左上のエッジがこのカットと交差する軽いエッジであるから，プリム法はこのエッジを選択し木に加える．

プリム法の実行中，木が含まないすべてのノードを key フィールドに基づく優先度つきキュー PQ に置く．各ノード v において，$key[v]$ は v を木 $\mathcal{T}$ に属するあるノードと結ぶエッジの中の最小重みである．ただし，そのようなエッジが存在しない場合には $key[v] = \infty$ と置く．$pred[v]$ は v の 1 つ前の節点が入っている．この情報を使うことで始点 s から節点 v までの経路を復元することができる．プリム法終了後に得られる最小全域木 $\mathcal{T}$ は

$$T = \{(pred[v], v) : v \in \mathcal{V} - \{s\}\}$$

となる．ただし s は最小全域木 T の根である．プリム法の各ステップでは，木の重みの増加を最小にするエッジを選択するので，貪欲法と言える．

プリム法の擬似コードを Algorithm 1 に示す [Cormen 01]．第 1-4 行で各ノード v のキー $key[v]$ を ∞ に初期化する．また，各ノードの親ノードを -1 に設定する．次に開始節点 $s \in \mathcal{V}$ をランダムに選択し，最初に処理するためにキーの値を 0 に設定する（第 5 行）．これは $\mathcal{T}$ が s を根とする木になることを意味する．今回は節点 0 を開始節点として選ぶ．第 6-9 行では優先度つきキュー PQ をすべてのノードを含むように初期化する．第 10-22 行の while ループの各繰り返しが開始される直前では次の 3 つの性質が成立する．

(i). $\mathcal{T} = \{(pred[v], v) : v \in \mathcal{V} - \{s\} - PQ\}$

(ii). 最小全域木の中にすでに置かれたノードは $\mathcal{V} - PQ$ に属するノードである．

(iii). 全てのノード $v \in PQ$ に対して，$pred[v] \neq -1$ ならば，$key[v] < \infty$ かつ $key[v]$ は v と最小全域木 $\mathcal{T}$ にすでに置かれたノードとを結ぶ軽いエッジ $(pred[v], v)$ の重みである．

第 11 行で $(\mathcal{V} - PQ, \mathcal{V})$ と交差する軽いエッジに接続するノード $u \in PQ$ を求める．第 12-21 行の for ループでは u と隣接し，木 $\mathcal{T}$ に属さないすべてのノード v について $key[v]$ と $pred[v]$ を更新する．この更新式によってループ不変式 3 が保存される．ノード間の接続行列からプリム法を用いて MST を計算する場合の計算量は $\mathcal{O}(|\mathcal{V}|^2)$ となる．ここで $|\mathcal{V}|$ はグラフのノード数を表す．

Algorithm 1 プリム法

Input: undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$

```
 1: for all v ∈ V do
 2:     key[v] = ∞
 3:     pred[v] = −1
 4: end for
 5: key[0] = 0
 6: PQ = new Priority Queue
 7: for all v ∈ V do
 8:     insert (v, key[v]) into PQ
 9: end for
10: while PQ is not empty do
11:     u = getMin(PQ)
12:     for all edge(u, v) ∈ E do
13:         if v in PQ then
14:             w = weight(u,v)
15:             if w < key[v] then
16:                 pred[v] = u
17:                 key[v] = w
18:                 decreaseKey(PQ, v, w)
19:             end if
20:         end if
21:     end for
22: end while
```

3·3 プリム法を用いた属性値選択

作成したグラフ $\mathcal{G}$ に対して MST を求めることで，全ての属性を選択しながら関連度の総和が最も大きくなる属性値リストを取得する．ただし，通常の MST 問題とは以下の違いがある．

- MST 問題は重みの総和を最小にするが，今回の問題は重み (関連度) の総和を最大にしたい．
- MST 問題は全てのノードを通る木を作成するが，今回の問題は全ての属性を通る木を作成する．

最初の違いはエッジの重みを減らしたいか増やしたいかの違いである．Prim 法は最小全域木を求めるアルゴリズムなので，今回のように最大全域木を求めたい場合は，重みが最大となるエッジを選んでいけばよい．

一方，2 番目の違いは 1 つの属性が複数の属性値を持つことはないという考えに基づく．そのためには，すでに選択された属性に所属するノードは選択しないという制約を新たに導入する必要がある．ところで，この制約のもとで MST を求める問題は Generalized Minimum Spanning Tree (GMST) と知られており，NP-hard であることが証明されている [Myung 95]．GMST を効率的に解くアルゴリズムは存在しないため，本稿では，プリム法に「すでに選ばれた属性値と同じ属性を持つ属性値候補 (ノード) に接しているエッジの重みをマイナスにする」というヒューリスティックを追加することで近似的に属性値選択問題を解く．今回使用している類似度尺度は全て 0.0 から 1.0 の値を持つため，重みをマイナスにしておけば選択されることはない．エッジの重みを変更し，選択できるエッジがなくなった場合 (選択されていない全てのエッジの重みがマイナスになった場合) に終了する．

Prim 法では全てのノードが選択されることが分かっているため，どのノードを始点にしても正しく動作することが保証されていた．今回は全てのノードが選択されるとは限らないため，選択されないノードを始点にしてしまうと正しい全域木が作成できないという問題が生じる．そのため全てのノードを始点し，MST を求め，その中で最大のものを取ることにした．これらの工夫によって提案手法ではプリム法を $|\mathcal{V}|$ 回繰り返すことになるので提案手法の計算量は $\mathcal{O}(|\mathcal{V}|^3)$ となる．

図 4 に提案手法の具体例を示す．このグラフには 4 つのノードがあるため，それぞれのノードを始点として (a)(b)(c)(d) の 4 種類の木を作成する．木を作成する際

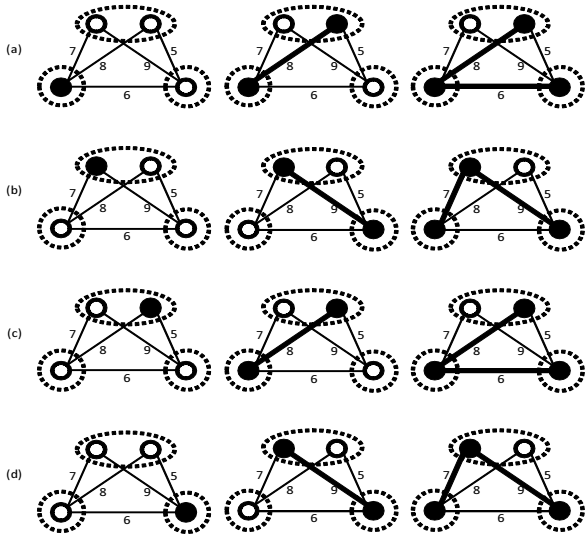

図 4 提案手法の具体例

には，基本的にはプリム法と同じだが，上の 2 つのノードは同じ属性に所属しているためどちらか片方しか選べないことに注意する．また，最小全域木ではなく，最大全域木を求めたいので重みが大きいエッジを選択してゆく．この例では，(b) と (d) の全域木の重みが 16 と最大になるため，どちらかをランダムに選択する．

4. 実 験 結 果

WePS2 に登録されている属性種類の中から提案手法を定量的に評価するため，多くの人物に対して登録されている次の 8 つの属性を使う [Sekine 09]．誕生年（人物が生まれた年），出身地（人物が生まれた地域），国籍（人物の現在の国籍），職業（人物の現在の職業），学位（人物が獲得している学位，BA,BS,MA,MS,PhD など），出身学校（学位を獲得した大学），専攻（どの専攻で学位を獲得したかということ），所属企業（現在務めている会社，大学など）．これらの 8 つの属性に関する属性値が全て登録されている 100 人の人物を Freebase の中からランダムに選択した．次に，Yahoo BOSS Search API を使って，選択したそれぞれの人物の人名をクエリとして Web ページをダウンロードした．ダウンロードしたページの中に同姓同名人物が含まれているものを除外し，それぞれの人物に対し，50 件の Web ページを選択した．以上の過程を経て 100 人の人物に関する 5000 (100×50) 件の Web ページを選択した．本論文で提案している属性値選択手法はそれ自体が属性抽出を行なっておらず，何らかの属性抽出手法を用い，属性値の候補となるものを事前に抽出しておく必要がある．そこで，[Watanabe 09] が提案した属性抽出手法を用いて属性値の候補となるものを抽出した．WePS2 [Sekine 09]（属性抽出の評価型ワークショップ）では渡辺らのシステムが第 5 位の F スコアを報告している．このシステムよりも精度が高いシステムがあるが，残念ながらそれらのシステムの実装は公開されておらず，提案手法を評価するために必要なデータが入手できなかった．

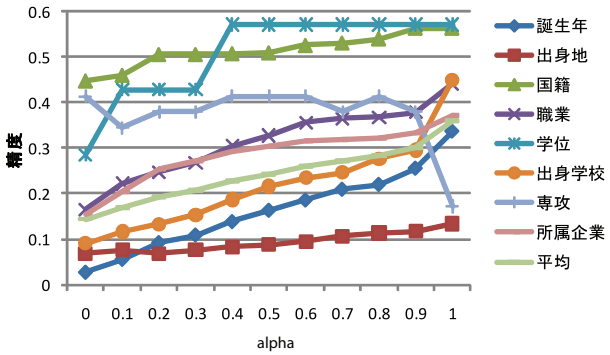

図 5 提案手法で α を調整した場合の，様々な属性に関する属性値選択精度．

渡辺ら [Watanabe 09] は 3 つの手法を使って，与えられたテキストコーパスから属性値候補を選択する手法を提案した．まず，Wikipedia の一覧ページから人手で選択した属性値リストに含まれている属性値がコーパス中に出現している場合は属性値候補として選択する．特に，大学名，企業名，職業名，国名，専攻名などの一覧ページは Wikipedia に登録されているため，これらの属性値の抽出には事前作成した属性値候補リストとのマッチングが有用である．しかし，この方法では，誕生年のような年に関する属性値が抽出できないため，渡辺らは正規表現に基づく属性抽出ルールも提案している．最後に，固有名詞抽出器[*4] を用い，コーパス内に出現する固有名詞（地域名，人名，組織名）を抽出し，属性値候補として選択する．渡辺らの属性抽出手法はそれぞれの属性に関して，カバレージを重視し，多くの属性値を抽出しているため，正しくない属性も多く含んでしまう．そのため, 少ない属性値 候補しか抽出しない手法と比べ, 抽出した属性値候補の中から正しい属性値を選択することが重要かつ困難なタスクとなる．上記の理由から本稿で提案した属性値選択手法を評価するために，渡辺らの手法で抽出した属性値を用いるのが妥当と考えられる．

本稿では属性抽出の精度を計測するための評価尺度としてそれぞれの属性に関する抽出精度を用いる．属性 A の抽出精度は次のように定義する．

$$\frac{\text{属性 } A \text{ に関する正しい属性値が選択された人物の数}}{\text{評価対象の人物の数}}$$

式 3 の Web と Freebase から計算した関連度を統合するパラメータ α の影響を調べるために $[0,1]$ 範囲内で α を調整しながら属性抽出の精度を測定する．Freebase を用いた関連度，sim_{FB} を計算する際には，評価データとして選択した 100 人の人物に関するデータを事前に削除する．Jaccard 係数を関連度関数として用いた場合の，α に

*4 http://nlp.stanford.edu/software/CRF-NER.shtml

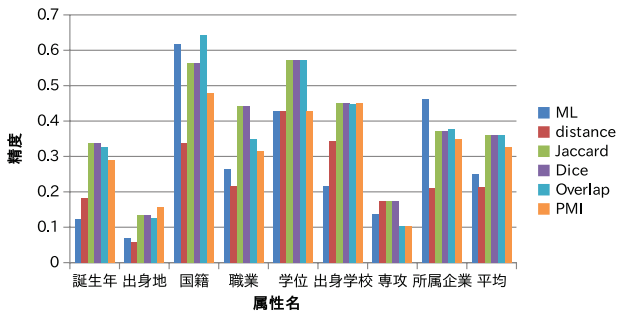

図 6 提案手法と比較手法による属性抽出の精度.

対する精度の変化を図 5 に示す．図 5 で示してある平均精度は対象とする 8 つの属性に関する抽出精度の平均である．図 5 から分かるように α を大きくするに連れ，平均抽出精度が増加し，$\alpha = 1$ の時に最も良い平均抽出精度が得られる．この現象は Jaccard 係数だけではなく，表 1 の全ての関連度関数に対して観測できた．式 3 で $\alpha = 1$ は Freebase のみを用いて属性値間の関連度を計算する場合に該当する．従って，属性値間の関連度を計算するためには Web 上の共起よりも Freebase 中の共起の方が有用であることが分かる．Freebase では既に人物に関する属性値という形で情報が整理されているに対し，Web から関連度を計算する場合は属性値として出現しているかどうか分からないため，Web よりも Freebase を対象にして属性値間の関連性を計測する方が良いと思われる．以上より，残りの実験で式 3 は $\alpha = 1$ として関連度を計算する．

次に，上記選択した 100 人に関する属性値候補を用いて，表 1 にまとめた 4 つの関連度尺度と次に述べる 2 つのベースライン手法を比較する．最初のベースライン手法として渡辺らが提案した，文書内で属性抽出を対象とする人物の人名からの距離を使って属性値選択を行う **distance** 手法を用いる．この手法は「正しい属性値は人名から近い位置に出現する」というヒューリスティックに基づいている．文書内での距離としてスペースで分割した場合のトークン（単語）数を用いた．もう一方のベースラインとして属性抽出を対象とする人物の人名と Web 上で最も多く共起する属性値を正しい属性値とする **ML** (most likely) 手法を用いる．Web 上の共起指標として人名と属性値の AND クエリに関するヒット件数を用いた．実験結果を示す図 6 の平均抽出精度から分かるように，Jaccard 係数，Dice 係数，overlap 係数が PMI や 2 つのベースライン (ML と distance) より良い精度を示している．尚，Jaccrad と Dice 係数は $Jaccard = Dice/(2 - Dice)$ という関係で結ばれているため，図 6 では同等な精度を示している．

5. 関 連 研 究

Web テキストから属性抽出を行うために，Ravi らは HTML タグの情報を利用した [Ravi 08]．まず，入力として，属性を抽出したいクラスと，シードとなる少数の属性を与える．次に，シード属性で検索し，シード属性付近で HTML タグに囲まれているものを属性候補として抽出する．HTML タグとしては，ボールド体 <b>・イタリック体 <i>・表 <td> <th>・リスト <li> などを使用する．最後に，シード属性と属性候補の階層構造を比較し，シード属性と似た構造を持つ属性候補を上位にランキングする．日本語の Web テキストから属性抽出を行った研究として，[Tokunaga 05] が挙げられる．徳永らは，単語の統計的特徴・抽出パターン・HTML タグを組み合わせて，日本語を対象にした属性抽出手法を提案した．まず抽出パターンを利用して，属性の候補となる単語を大量に取得する．次に，Web や新聞記事に出現した回数・HTML タグがつけられた回数などを加味して属性候補をランク付けする．同様に日本語の Web テキストを対象にした研究には [Yoshinaga 06, Yoshinaga 07] などがある．

検索エンジンの履歴 (クエリログ) を対象とする研究も存在する [Durme 07, Pasca 07b, Pasca 07a]．検索履歴は Web テキストと比較すると，存在する量が圧倒的に少ない．また，検索履歴はキーワードの羅列であり，文書に比べてあいまいさも残る．しかし，検索エンジンを利用する人は，なるべく正確な情報を得ようとして最適なクエリを作成するため，検索履歴から有用な情報が取り出すことが可能となる．[Pasca 07b] では，人手で作られたパターンを利用して，検索履歴からクラスと属性を抽出している．抽出パターンとしては “What is the A of I” などを用いる．ここで A は属性，I は属性を抽出したいクラスに属するインスタンスを表す．この場合，” What is the population of Japan ” などの検索履歴が抽出され，“Japan” というインスタンスが “population” という属性を持っていることが分かる．一方，[Pasca 07a] では，弱い教師付き学習を使って属性を抽出している．まず，属性を抽出したいクラスのインスタンスと，正解の属性を与え，インスタンスと同時に検索された単語を属性候補として取得する．次に，属性候補がどのような単語と同時に検索されているかを調べ，正解属性と似た単語と一緒に検索されていれば，その属性候補を抽出する．

Wikipedia[*5] の記事から属性を抽出する研究も非常に盛んである．Wikipedia は通常の Web テキストよりも構造化されており，属性抽出が行いやすい．Nastase らは記事のカテゴリ名からクラス属性を抽出し，さらにカテゴリ間の関係ネットワークを構築した [Nastase 08]．Suchanek らは Wikipedia と Wordnet[*6] を利用して 100 万

*5 http://www.wikipedia.org/
*6 http://wordnet.princeton.edu/

以上のエンティティが登録されたオントロジー Yago を作成した [Suchanek 07]．Wu らはクラス属性を抽出することで Wikipedia の Infobox を自動的に作成することを可能にした [Wu 08]．

Probst らは，製品の説明書からその製品の持っている属性と属性値を教師付き学習によって抽出した [Probst 07]．例えば，ランニングシューズという製品には，“材質”：“ナイロン”，“高さ”：“低い”，“レーシングシステム”：“標準” のような様々な属性がある．これらの属性を利用すれば，製品の価格を比較したり，自分が必要な属性を持った製品を簡単に見つけたりすることが可能になる．また，属性が似た過去の商品と比較することで，その商品の売れ行きも予想することができる．説明書には全ての属性値が記載されているわけではないため，このように自動的に属性値を取得する手法が必要となる．同様に製品情報を対象にした属性抽出には [Raju 09, Popescu 05] などが存在する．しかし，本研究と違って，上述した先行研究ではあるエンティティに関する属性を抽出する際には抽出対象の属性を個別に扱い，抽出を行なっているため属性値間の関連性に着目していない．

6. むすび

本稿ではあるエンティティに対して抽出された属性値候補の中から正しい属性値を選択するための手法を提案した．Freebase から計算した Jaccard 係数を用いることで属性値間の関連性を計測し，属性値をノードとするグラフの最小全域木を求めることで，与えられたエンティティに関する属性値を選択できることが分かった．しかし，まえがきのところで紹介した通り，Web のテキストから属性抽出を行う際に解決しなければならない課題はまだまだ沢山存在しており，今後，提案した属性値選択手法からのフィードバックを属性値抽出システムに与えることで更なる精度向上を目指す．

◇ 参 考 文 献 ◇

[Bollegala 07] Bollegala, D., Matsuo, Y., and Ishizuka, M.: Measuring semantic similarity between words using web search engines, in *Proc. of WWW '07*, pp. 757–766 (2007)

[Cormen 01] Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C.: *Introduction to Algorithms*, MIT Press (2001)

[Durme 07] Durme, M., Pasca, B., and Garera, N.: The role of documents vs. queries in extracting class attributes from text, in *CIKM'07*, pp. 485–494 (2007)

[Myung 95] Myung, Y.-S., Lee, C.-H., and Tcha, D.-W.: On the generalized minimum spanning tree problem, *Networks*, Vol. 26, No. 4, pp. 231 – 241 (1995)

[Nastase 08] Nastase, V. and Strube, M.: Decoding wikipedia categories for knowledge acquisition, in *AAAI'08*, pp. 1219–1224 (2008)

[Pasca 07a] Pasca, M.: Organizing and Searching the World Wide Web of Facts Step Two: Harnessing the Wisdom of the Crowds, in *WWW'07*, pp. 101–110 (2007)

[Pasca 07b] Pasca, M. and Durme, B.: What you seek is what you get: Extraction of class attributes from query logs, in *IJCAI'07*, pp. 2832–2837 (2007)

[Popescu 05] Popescu, A.-M. and Etzioni, O.: Extracting product features and opinions from reviews, in *EMNLP'05*, pp. 339–346 (2005)

[Prim 57] Prim, R. C.: Shortest connection networks and some generalizations, *Bell System Technology Journal*, Vol. 36, pp. 1389–1401 (1957)

[Probst 07] Probst, K., Ghani, R., Krema, M., Fano, A., and Liu, Y.: Semi-supervised learning of attribute-value pairs from product descriptions, in *IJCAI'07*, pp. 2838–2843 (2007)

[Raju 09] Raju, S., Pingali, P., and Varma, V.: An Unsupervised Approach to Product Attribute Extraction, in *ECIR'09*, Vol. 5478, pp. 796–800 (2009)

[Ravi 08] Ravi, S. and Pasca, M.: Using Structured Text for Large-Scale Attribute Extraction, in *CIKM'08* (2008)

[Sekine 09] Sekine, S. and Artiles, J.: WePS 2 Evaluation Campaign: Overview of the Web People Search Attribute Extraction Task, in *Proceedings of the 2nd Web People Search Evaluation Workshop (WePS-09) at WWW'09* (2009)

[Suchanek 07] Suchanek, F., Kasneci, G., and Weikum, G.: Yago: a core of semantic knowledge, in *WWW'07*, pp. 697–706 (2007)

[Tokunaga 05] Tokunaga, K., Kazama, J., and Torisawa, K.: Automatic discovery of attribute words from Web documents, in *IJCNLP'05*, pp. 106–118 (2005)

[Watanabe 09] Watanabe, K., Bollegala, D., Matsuo, Y., and Ishizuka, M.: A Two-Step Approach to Extracting Atrtibutes for People on the Web, in *Proceedings of the 2nd Web People Search Evaluation Workshop (WePS-09) at WWW'09* (2009)

[Wu 08] Wu, F., Hoffmann, R., and Weld, D.: Information extraction from Wikipedia: Moving down the long tail, in *KDD'08*, pp. 731–739, ACM (2008)

[Yoshinaga 06] Yoshinaga, N. and Torisawa, K.: Finding specification pages according to attributes., in *WWW'06*, pp. 1021–1022 (2006)

[Yoshinaga 07] Yoshinaga, N. and Torisawa, K.: Open-Domain Attribute-Value Acquisition from Semi-Structured Texts, in *Proceedings of Workshop on Ontolex*, pp. 55–66 (2007)

〔担当委員：平 博順〕

2012 年 02 月 24 日 受理

著 者 紹 介

ボレガラ ダヌシカ

2005 年東京大学工学部電子情報工学科卒．2007 年同大学院情報理工学系研究科修士課程修了．2009 年同研究科博士課程修了．博士 (情報理工学)．現在: 同研究科・助教．自然言語処理に興味を持つ．WWW, ACL, ECAI などの会議を中心に研究成果を発表．IEEE 会員．

谷直紀

2011 年 東京大学大学院情報理工学系研究科電子情報学専攻修士課程修了．現在，楽天株式会社に勤務中．ウェブを用いた知識取得に興味を持つ．

石塚 満（正会員）

1971 年東京大学工学部卒，1976 年同大学院工学系研究科博士課程修了．工学博士．同年 NTT 入社，横須賀研究所勤務．1978 年東京大学生産技術研究所・助教授 (1980-81 年 Perdue 大学客員准教授)．1992 年同大学工学部電子情報工学科・教授．現在: 同大学院情報理工学系研究科・教授．研究分野は人工知能，Web インテリジェンス，意味計算，生命的エージェントによるマルチモーダルメディア．IEEE, AAAI，電子情報通信学会，情報処理学会等の会員，本会の元会長