# 分散環境上のグラフ構造解析を用いた非構造化データの類似度の算出

佐伯　嘉康† 　田島　玲†

† ヤフー株式会社　〒107–6211 東京都港区赤坂 9-7-1 ミッドタウン・タワー
E-mail: †{ysaeki,atajima}@yahoo-corp.jp

あらまし　Linked Data を代表とし，異なるサービス間で管理しているデータ同士を内容に応じ結び付け，類似などの関係を算出し，互いのデータの発見・利用効率を高めようという動きがある．これらの応用には，推薦やオントロジーの構築がある．本稿では，そうした応用の土台となるデータ間の類似度を求める手法として，実サービスで利用されている非構造化データから，1）それが指すエンティティと，エンティティの特徴となる要素を抜き出し，2）要素同士を連結したグラフ構造データに変換し，3）グラフ構造データ間の非対称類似度を求める手法を提案した．検証は，動画情報からの類似動画の発見を想定し，分散環境 Hadoop で動作するグラフ処理フレームワークである Giraph と，実サービスデータを用いて行い，データ間の類似度の算出の結果を複数評価者の目視と計算時間の計測により，定性・定量の両面で評価を実施し，有用性を確認した．

キーワード　グラフマイニング, リコメンデーション, エンティティネットワーク

## 1. 導　　入

Web サービスの多くはエンティティ（実体などと訳される，識別子により一意に識別される独立した存在）を扱う．例えば，映画レビューを扱う Web サービスであれば，映画，レビュー，レビュー記述者などがエンティティである．エンティティは概念であるため，機械可読な形式でデータとなる．これらは Web サービスが管理するデータベースに参照可能な形式で保存され，エンティティデータベース，あるいは，ナレッジベースと呼ばれる．

昨今では，データベースのクラウド化などを背景に，Linked Data (注1) などの Web を通じた異分野間でのエンティティ（を記述したデータ）連携の推進の動きが活発である．エンティティの利用は 1 つの Web サービスに留まらず，複数の Web サービス同士がエンティティを集め，連結し，エンティティ同士の関連などから，重複排除や推薦，オントロジーの構築などのアプリケーションを通し，新たな価値を見出し利用する流れが積極的になっている．

そうした応用の土台として，エンティティ同士の連結のために，ある Web サービスがもつ 1 エンティティと，別の Web サービスがもつ 1 エンティティが，表記ゆれなどを考慮しながら，同一であるかあるいは類似しているかを判断する手法が必要である [1]．問題の例を以下に 2 つ挙げる．

- 表記識別子が同じだが，それが指す対象が複数ありうるものに対応する: ある Web サービスが扱うエンティティの識別子である “*Apple*” と，別のサービスのエンティティ識別子の “*Apple*” はどの程度似ているか．
- 表記識別子が異なるが，内容から類似性が大きいと判断できるものに対応する: 映画 “スタンド・バイ・ミー”（原題:“*Stand by Me*”）の原作にあたる小説のタイトルは “*The Body - Fall From Innocence*” である．これらは類似したエンティティとしたい．

本研究では，複雑かつ一般的な状況として，エンティティの集合がテーブルで表現されている構造化データと，1 つ以上のエンティティを含むテキストデータや Web ページなど非構造化（半構造化）データが混在する場合での，エンティティ間の類似度の算出を想定する．そのような状況で，この問題を解決するためには，まず，エンティティそのものを比較するのではなく，エンティティの特徴を比較する手段が有効である．そして，エンティティ間の類似度を効率的に算出できる形式で表現されている必要がある．そのために，本研究では前処理として，エンティティの特徴を失わず記述に汎用性がある形式として，エンティティの集合と関係をグラフ構造データに変換し表現する．

次に，同一性の表現について，Semantic Web 技術をベースとする Linked Data の場合，OWL (注2) で定義される sameAs（ある 2 つのデータまたはエンティティは同一である），SKOS (注3) で定義される broadMatch / closeMatch / exactMatch / narrowMatch / relatedMatch（それぞれ同一であることの厳密度が異なる）などでエンティティ間の類似度を，人手あるいはエンティティ外にある常識などの知識に基づき記述している．本研究では，グラフ構造データに変換したエンティティ集合を対象に，エンティティ間の非対称類似度（実数）を算出するグラフ処理アルゴリズムを提案する．このアルゴリズムは大規模データを対象とした算出を想定し，分散環境上で動作することを保証する．

検証は，実 Web サービスで用いられているエンティティデータベースを用い，エンティティ集合から得られる組み合わせの類似度の算出の結果が妥当であること，かつ実用的な時間内に完了することを確認することにより，有用性を示す．

(注1)：http://linkeddata.org/

(注2)：http://www.w3.org/2001/sw/wiki/OWL

(注3)：http://www.w3.org/2004/02/skos/

## 2. 関連研究

グラフ構造データの類似性を得る研究として，Zhou らによるグラフの構造とグラフのノードに付随する属性情報を用いた類似度算出に基づくクラスタリング [2] がある．また，Ma らによる分散環境に適したグラフ構造パターンマッチアルゴリズム [3] を用いることで，大規模なグラフ構造データについても類似度算出の計算が可能であることが分かっている．

データの類似度を求める研究として，Herzig らによる，Web 検索クエリや Web ページを対象データとし，それらが指すエンティティ間の類似度を求める研究 [4] がある．この研究では，エンティティに付加されている属性と言語モデルを用い，Web 検索クエリを介して属性集合同士の比較を行っている．また，Blanco らによる Spark システム [5] は，関連付けられたエンティティの集合を Wikipedia などのデータを含んだナレッジベースに格納し，人手による修正や編集を加えた上で Web 検索結果の拡張に応用している．

本研究は，これらの関連する先行研究の成果を基に，多様な実サービスデータを対象とした応用を行う．そのために，1）データのグラフ構造データへの変換を行った上で，エンティティ間の質の差異を考慮した非対称類似度を提案する．2）類似度算出を分散環境上で実行するためのグラフ処理アルゴリズムを提案し，実装する．

## 3. 非構造化データの非対称類似度の算出

本章では，非構造化データと構造化データの集合から得られるエンティティの組について，類似度を算出する．そのために，以下の 2 つの手順を踏む．1）非構造化データから，データが指す唯一の主体となるエンティティと，そのエンティティを特徴付けている複数の要素をプロパティとして抽出し，そのプロパティの集合をグラフ構造データとして定義し，変換する．2）そのデータを元に，エンティティ間で類似度を計算することで，そのエンティティを含むデータ間の類似度を算出する．

本研究でグラフ構造データを採用した理由は，データが扱うエンティティが多種多様になりうる中で，グラフの表現には汎用性があり，プロパティというエンティティに依存しない要素を経由し，エンティティ同士の関連を容易に記述できるためである．

### 3.1 非構造化データからのプロパティ抽出とグラフ構造データへの変換

非構造化データから抽出するプロパティはデータが指すエンティティの種類と内容に基づき，人手により schema.org (注4) などから決定することが望ましい．schema.org のプロパティはエンティティの種類毎にプロパティ名と値の型が定義されているが，複数種類のエンティティにまたがって同じ，あるいは他のプロパティと親子・兄弟関係を持つプロパティが定義されおり，多様な分野のデータ同士の連携を想定しているため，本研究が想定する状況に適している．

(注4)：http://schema.org/

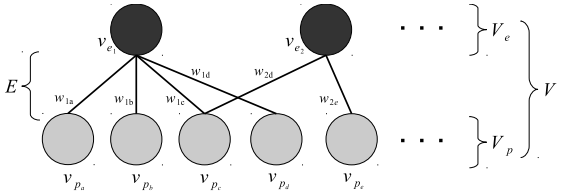


図 1　エンティティとプロパティによるグラフ
ノード $v_e$ と $v_p$，それを結ぶエッジは重み $w$ を持つ

非構造化データからプロパティを抽出する手法として，主にテキストデータに用いられる形態素解析や特徴語抽出，主にマルチメディアデータに用いられるメタデータ抽出などを採用することができる．手法の選択については，扱う非構造化データの分野，形式により異なる．本研究における具体的な手法は第 5. 章の実験の詳細で取り上げる．

以上により得たエンティティとプロパティの集合をグラフ構造データ $G$ として，ノード集合 $V$ とエッジ集合 $E$ により以下のように定義した．また，グラフを図で表すと図 1 のようになる．

$$G = (V, E) \tag{1}$$

$$V = (V_e, V_p) \tag{2}$$

$$\begin{aligned} e_{ep} &= (v_e, v_p, w) \\ &= v_e \overset{w}{\longleftrightarrow} v_p \\ &\quad (e_{ep} \in E, v_e \in V_e, v_p \in V_p, w : weight) \end{aligned} \tag{3}$$

ここで，$v_e$ の集合である $V_t$ は，元の非構造データが示すエンティティの識別子を持つノード（エンティティノードと呼ぶ）の集合とし，$v_p$ の集合である $V_p$ は，エンティティに紐づくプロパティの名前と値のペアを識別子として持つノード（プロパティノードと呼ぶ）の集合である．エンティティノードは，別のエンティティノードとの類似度を値として持つことになる．$e_{ep}$ は，あるエンティティノード $v_e$ から，プロパティノード $v_p$ へのエッジ（エンティティがそのプロパティを持つ）を表し，エンティティはプロパティに対し重み $w$ を持つ．$w$ は類似度を求める際に用い，エンティティから見たプロパティの重要度を表す．

### 3.2 グラフ構造データの非対称類似度の算出

本研究の目的は，ある非構造データ間の類似度を求めることであるため，グラフ構造データ上の類似度は，2 エンティティノード間で求めるとする．エンティティノード $v_{e_i}$ から見た $v_{e_j}$ との類似度は以下の式で求める．

$$similarity_{ij} = sim(v_{e_i}, v_{e_j}) \tag{4}$$

$$= \frac{\sum(E_{e_i} \cap E_{e_j})}{\sum E_{e_i}} \tag{5}$$

すなわち，求める類似度 $similarity_{ij}$ は， ある 2 つのエンティティノード $v_{e_i}$，$v_{e_j}$ に対し，それぞれのエンティティノー

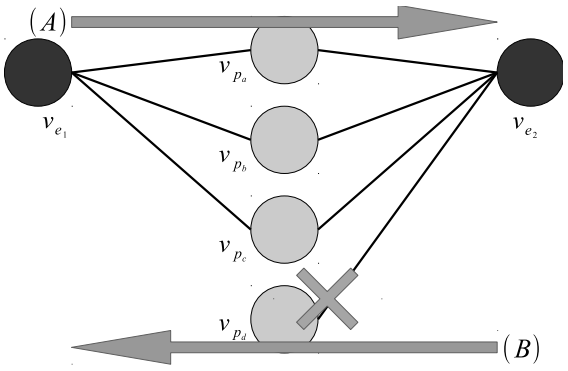


図 2　2 エンティティノード間の到達効率の違い
ノード $v_{e_1}$ とノード $v_{e_2}$ を繋ぐプロパティノードの数が出発ノードによって異なる

ドに共通して連結しているプロパティノードのエッジ総和を，$v_{e_i}$ の持つ全てのプロパティノードへのエッジ総和で割ったものである．類似度は 0 から 1 までの実数を取り，1 に近いほど大きい．また，$similarity_{ij}$ と $similarity_{ji}$ は同じ値になるとは限らない（非対称である）．

非対称類似度については，比較するデータの特徴の個数が大きく異なる場合に度々用いられる [6], [7]．本研究が想定する状況では，データの詳細度，質はそれを保有するサービスによって異なるため，非対称類似度の導入が適していると考えた．

本研究での類似度は，あるエンティティノードから別のエンティティノードへ到達するための有効エッジの割合（到達効率）を表していると考える．2 エンティティノード間の到達効率が異なる例を以下に示す．

**ある 2 エンティティノード間の類似度 = 到達効率が異なる例.**

エンティティノード $v_{e_1}$ と $v_{e_2}$ があり，以下のエッジがあるとする．

$$E_{e_1} = v_{e_1} \longrightarrow v_{p_a}, v_{p_b}, v_{p_c}$$
$$E_{e_2} = v_{e_2} \longrightarrow v_{p_a}, v_{p_b}, v_{p_c}, v_{p_d}$$

$w$ を全て 1 としたとき，

$$sim(v_{e_1}, v_{e_2}) = 3/3 = 1.0$$
$$sim(v_{e_2}, v_{e_1}) = 3/4 = 0.75$$

となる．

これは，$v_{e_1}$ から出発し $v_{e_2}$ へ到達するためには 3 つ全てのエッジが有効であるため到達効率が良く（図 *2* (*A*) 方向），一方で，$v_{e_2}$ から出発し $v_{e_1}$ へ到達するためには，4 つのエッジの内 $v_{e_2} \rightarrow v_{p_d}$ が利用できないため到達効率が悪くなる（図 *2* (*B*) 方向）ことを表している．

## 4. 分散環境上でのグラフ構造データの非対称類似度の算出

データの規模が大きくなるに従い，各エンティティノード組について類似度を求める計算時間は大きくなる．本章では，分散環境上でグラフ構造データに対する計算を行うためのアルゴリズムの 1 つである Pregel を導入し，第 3. 章で述べた非対称類似度算出の手法を適用する．

### 4.1 Pregel による実装

Malewicz らによる Pregel [8] は，分散環境上での実行を想定した，大規模なグラフ構造データの処理に適したアルゴリズムである．本研究では，大規模なデータを対象にした類似度算出の時間の増大を抑えるために，類似度算出アルゴリズムを Pregel を採用した．

Pregel における計算で定義する必要がある処理は，ノードが持つ値の形式，ノード間に流すメッセージの形式，各ノードから別ノードへメッセージを転送しノードへメッセージの処理を依頼する（*superstep* と呼ばれる）回数，各ノードが *superstep* で受け取ったメッセージの処理内容である．本研究ではそれぞれ以下のように定義した．

**ノードが持つ値の形式.**
*Map[ID(Entity)* $\rightarrow$ *Number]*

**ノード間エッジに流すメッセージの形式.**
*Map[ID(Entity)* $\rightarrow$ *Number]*

**superstep の回数.**
*3*

**superstep.**
*Algorithm 1* に示す.

### 4.2 実装の詳細

この実装では，$similarity_{ij}$ を求めるためにまず，エンティティノード $v_{e_j}$ からメッセージを送信する．この際には，エッジの重みは考慮しない（Algorithm 1: $sendMessagesFromEntityToProperties$）．

このメッセージは $v_{e_j}$ に接続しているプロパティノード全てに送られるが，プロパティノードは，複数のエンティティノードと接続することができるため，どのエンティティノードからメッセージを受け取ったかを *score_map* に記録する．プロパティノード毎に記録された *score_map* は，そのプロパティノードに接続しているエンティティノード宛にメッセージとして送られる．その際，エッジの重みが考慮される（Algorithm 1: $forwardMessagesFromPropertyToEntities$）．

プロパティノードからメッセージを受け取ったエンティティノードは $v_{e_i}$ となり，式 5 に対応する処理を行う．受け取ったメッセージを，メッセージに含まれるエンティティ毎に集計し，エッジの重み総数で割る．この値が，$v_{e_i}$ から見た，他のエンティティノードとの類似度である．従って，2 エンティティノードが 1 つ以上の同じプロパティノードにエッジを持たなければ，互いの類似度は得られない（この場合の類似度は 0 である）（Algorithm 1: *aggregateScores*）．

## 5. 実　　験

Pregel の実装として Hadoop 上で動作する Giraph を採用し

```
Procedure superstep(vertex, messages:Iterable[Map]):
    switch counter_of_superstep do
        case 0
            vertex.initializeValue() ;
            sendMessagesFromEntityToProperties(vertex) ;
        end
        case 1
            forwardMessagesFromPropertyToEntities(vertex,
            messages) ;
        end
        case 2
            aggregateSocres(vertex, messages) ;
        end
        otherwise
            vertex.halt() ;
        end
    endsw
end

/* superstep 0 → 1                                          */
Procedure
sendMessagesFromEntityToProperties(entity_vertex):
    foreach edge ← entity_vertex.getEdges() do
        entity_vertex.sendMessage(edge.target_vertex,
        Map(entity_vertex, 1.0)) ;
    end
end

/* superstep 1 → 2                                          */
Procedure
forwardMessagesFromPropertyToEntities(property_vertex,
messages):
    foreach message ← messages do
        score_map.put(message.value.getKey(),
        message.value.getValue()) ;
    end
    foreach edge ← property_vertex.getEdges() do
        property_vertex.sendMessage(edge.target_vertex,
        score_map * edge.weight) ;
    end
end

/* superstep 2 → 3                                          */
Procedure aggregateSocres(entity_vertex, messages):
    foreach message ← messages do
        aggregated_socres.putAll(message.value.getKey(),
        reduceByKey(plus, message.value.getValue())) ;
    end
    entity_vertex.setValue(aggregated_socres /
    sum(entity_vertex.getEdges())) ;
end
```

**Algorithm 1:**
2 エンティティノード間の類似度を求める superstep

た．Giraph は，Hadoop Distributed File Sytem（HDFS）を介さず，メモリを介した Hadoop RPC を用いグラフ内のノード間でメッセージを伝播させる．本実験で用いた Hadoop クラスタの環境を以下に示す．

- Hadoop バージョン: 2.0.0-cdh4.4.0（Cloudera's Distribution Including Apache Hadoop 4.4 に含まれる）
- Giraph バージョン: 1.1.0-SNAPSHOT（2013/10/11 ビルド）
- Hadoop TaskTracker 台数: 3 台（同一スペック）
- OS: CentOS 6.4 / 64bit
- Memory: 4GB
- HDD: 500GB

### 5.1 類似度算出時間の検証

ダミーデータを用い，エンティティノード間の類似度を算出するために必要な時間を計測した．ダミーデータを生成した条件は以下のとおりである．

- エンティティノード数: 1000 から 100000
- プロパティノード数: エンティティノード数と同じ
- 1 つのエンティティノードに接続するプロパティノード数: 10 から 60 のランダムな整数

Giraph は，処理の並列数（並列スレッド数の全てのマシンでの合計）を制御できる．そこで，並列数を 5 から始め，60 秒を超えたジョブが現れた段階で並列数を 5 ずつ増やす制御を行い，計算時間の増加しやすさを観察した．計算を始め全ての *superstep* が完了する時間をプロットしたものを図 3 に示す．

図 3 から，並列数が限定されている場合，ノード数が増えるにつれ計算時間は指数関数的に増大することが分かる．しかし，並列数を適切に設定することで，計算時間を抑えることができていることも分かる．

また，計算を，グラフ構造データファイル読み込み，各 *superstep*，計算結果のファイルへの書き込み，のフェーズに分割しそれぞれかかる時間をプロットしたものを図 4 に示す．

図 4 によって，ノード数の増加に伴い顕著に計算時間が増加するフェーズは，Shutdown フェーズ，すなわち，計算結果のファイルへの書き込みであることが分かった．これはファイルを保持するディスクデバイスの性質に依存するものである．また，*superstep*3，すなわち接続しているプロパティノードからメッセージを受け取ったエンティティノードがメッセージをまとめるフェーズでわずかにノード数増加の影響を受けていることも分かった．

以上の結果から，提案したアルゴリズムは，計算に十分な並列数を確保できる場合，大規模なデータの場合でも実用的な時間に計算時間が収めることが可能であると判断できる．

### 5.2 小規模データを用いた検証

次に，Movie Data Set [注5] の内 "casts.html" の HTML テキストを解析し，エンティティノードを映画タイトル（ノード識別子は映画タイトル名），プロパティノードをキャスト（ノード識別子はキャスト名）として抽出，グラフ構造データに変換

(注5)：http://archive.ics.uci.edu/ml/datasets/Movie

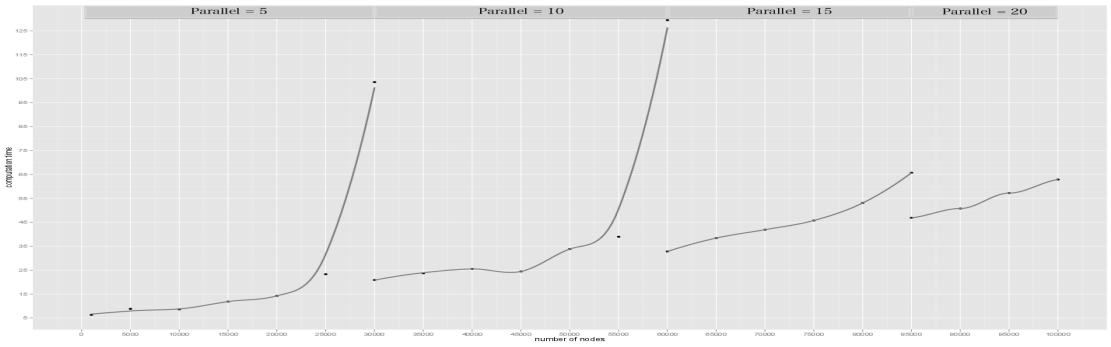


図 3　ノード数の増加に伴う計算時間の変化
並列数を変更しない場合，指数関数的に計算時間が増大する

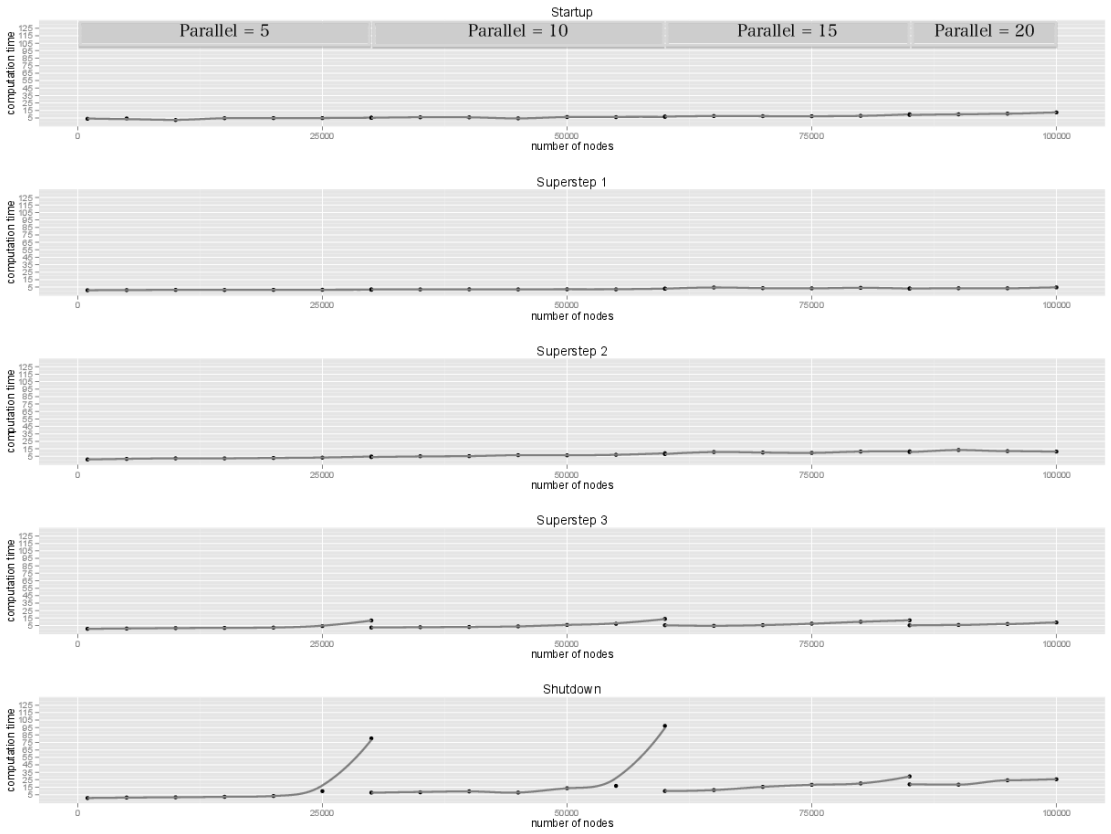


図 4　ノード数の増加に伴う各フェーズの計算時間の変化
指数関数的に計算時間が増大するフェーズは Shutdown フェーズ（計算結果の書き込み）であり，その他のフェーズは影響をほとんど受けない

し，類似度の算出を行った．データの統計は以下のとおりである．

- 総エンティティノード（映画タイトル）数: 8,635
- 総プロパティノード（キャスト数）: 16,597
- 1 エンティティノードに接続しているプロパティノード数の中央値，平均値: 4，5.2
- エンティティノードとプロパティノード間の重み: 1.0

全エンティティノード中，0 より大きい類似度で対応するエンティティノードが得られたものは，8,032 件であった．その内，類似度 1.0 で対応するエンティティノードを持つエンティティノードが，1,179 件あり，それらに接続するプロパティノード数の中央値は 1.0，平均値は 1.29 であった．つまり，これらの映画タイトルは十分なプロパティを持っておらず，約 1 件のキャストで繋がるだけで他の映画タイトルに対し大きい類似度を示していることになる．すなわち，この結果は，本研究で得られる類似度が，エンティティノードに対し主観的であり，接続するプロパティノードが不十分な場合，強い意味を持たないことを表している．逆にプロパティノードが豊富なエンティティノードから見た場合，この問題は見られない．従って，この問題に対しては，接続するプロパティノードを豊かにする（追加する）などの準備，推薦などのアプリケーションのフィードバックを活用しノード間の重みを変更するなどの加工を行う

ことで，類似度を改善し有意性を高めることが出来ると考える．

一方で，5 つ以上のプロパティノードと繋がっているエンティティノードの内，類似度 1.0 の対応するエンティティノードを得たものがあった．それらを，その映画タイトルが属するシリーズ名，逆のエンティティノードからの類似度と共に，表 1 に示す．

同じシリーズに属する映画タイトルについて大きい類似度を示している一方で，出演者や監督など，同じシリーズでもキャストが異なれば類似度が大きく変わることが分かった．例えば，本実験で用いたデータには “*Star Wars*” シリーズの映画: “*Episode 1, The Phantom Menace*” が含まれるが，キャストが大きく異なるため，同シリーズの他の映画と大きい類似度を得られていない．この場合，同シリーズであるという知見から，これらの映画タイトル間は大きい類似度としたい．

この問題の解決についても，プロパティノードとして「シリーズ名」を加え他のエンティティノードと結び付け重みを大きくするなど，プロパティノードの充実が有効であると考える．

### 5.3 大規模実サービスデータを用いた検証

より大きなサイズでのデータを用いた処理を検証するため，Yahoo! Japan が提供している Web サービスである，Yahoo! 映画データと GyaO!の動画データ（1902 年から 2014 年に公開された著名な映画および動画データが含まれる）を用い実験を行った．データの統計は以下のとおりである．

- Y!映画
  - タイトル数: 約 50,000
  - 抽出人物数: 約 130,000
    （1 タイトル当たり平均 11，中央値 10）
  - その他特徴要素数: 約 150,000
- GyaO!
  - タイトル数: 約 10,000
  - 抽出人物数: 約 32,000
    （1 タイトル当たり平均 14，中央値 10）
  - その他特徴要素数: 約 62,000

類似度を算出するにあたり，エンティティノードの識別子はタイトルとし，プロパティノードの識別子はキャストである人物名とその他タイトルを特徴付ける文字列とした．従って，類似度はタイトル間で求められる．

前処理として，テレビ番組など，同一タイトルを持ち同一キャストからなるシリーズものとして配信されているタイトルについてはあらかじめまとめ，1 つのタイトルとした．また，サービスで用いている非構造化データからのプロパティ抽出には，形態素解析と特徴語抽出を用いた．また，外国人名のカタカナ表記揺れに対応するため，Unidic 2.1.2 (注6) に含まれる英語表記-カタカナ表記の辞書を用い，統一した．

その他の特徴要素として，本実験ではタイトルの 3-gram 文字列を用いた．例えば，“ムービー” というタイトルは，“ムービ” と “ービー” という 2 つの文字列をそれぞれ識別子とした 2 つのプロパティノードを生成し，エンティティノードと結び付けられる．同一の識別子を持つプロパティノードは 1 つとなる．

(注6)：http://www.ninjal.ac.jp/corpus_center/unidic/

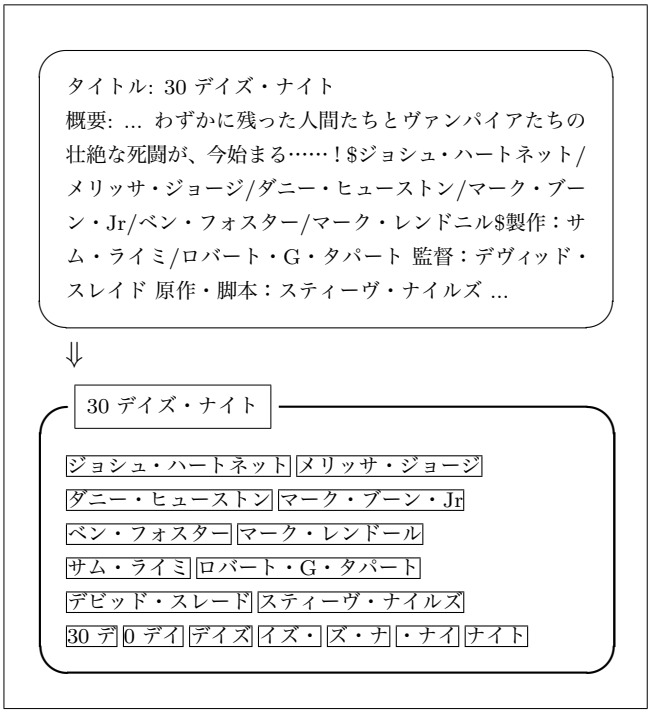


図 5　本実験で用いたデータ例
“*30* デイズ・ナイト” というエンティティノードに対して結び付いているプロパティノードを列挙している

表 2　評 価 基 準

| 判定項目（スコア） | 説明 | 例 |
|---|---|---|
| similar (4) | 同一タイトルである | エンティティとして一致する他，プロモーション映像 |
| excellent (3) | かなり関連がある | 同一シリーズのタイトル |
| good (2) | 関連がある | 番外編やリメイク，実写版とアニメ版などテーマが同一のタイトル |
| fair (1) | 関連が少しある | キャスト，原作者などが同一のタイトル |
| bad (0) | 関連が無い | 関連が明らかに無いタイトル |

図 5 に，本実験で用いたデータの例を示す．

類似度算出の結果に対し，ランダムにタイトル 708 件を選出し，各タイトルから類似度が付与されており，かつ，値の大きい順に上位最大 3 件のタイトルを取得し，複数の評価者により，表 2 の基準を用い評価を行った．あるタイトルに対応する類似度が最も大きなタイトルを rank 1，以下順に rank 2，rank 3 とし，各ランクに含まれる評価の割合を結果とし，表 3 および図 6 に示す．

本実験で用いたデータでは，全てのタイトルについて，必ずしも関連が強いタイトルが 1 つ以上存在するとは限らず，また，全体の約 7%のタイトルはキャスト情報が無いか不十分であった．

fair を有効閾値とした場合，あるタイトルに対し類似度が大

表 1　類似度 1.0 を得たタイトル組み合わせのリスト

| シリーズ名 | タイトル | | タイトル | 類似度 |
|---|---|---|---|---|
| *Andy Hardy* シリーズ | *Andy Hardy Comes Home* | ⟶ | *You're Only Young Once* | 1.0 |
| | | ⟵ | | 0.83 |
| | | ⟶ | *Love Laughs at Andy Hardy* | 1.0 |
| | | ⟵ | | 0.71 |
| | *You're Only Young Once* | ⟶ | *Love Laughs at Andy Hardy* | 1.0 |
| | | ⟵ | | 0.86 |
| *Die Nibelungen* シリーズ | *Krimhild's Revenge* | ⟶ | *Sigfried* | 1.0 |
| | | ⟵ | | 1.0 |
| *Four Daughters* シリーズ | *Daughters Courageous* | ⟶ | *Four Wives* | 1.0 |
| | | ⟵ | | 0.71 |
| | *Four Daughters* | ⟶ | *Daughters Courageous* | 1.0 |
| | | ⟵ | | 0.8 |
| | | ⟶ | *Four Wives* | 1.0 |
| | | ⟵ | | 0.57 |
| *Monty Python* シリーズ | *Monty Python and the Holy Grail* | ⟶ | *Monty Python's Life of Brian* | 1.0 |
| | | ⟵ | | 0.83 |
| | | ⟶ | *Monty Python's The Meaning of Life* | 1.0 |
| | | ⟵ | | 0.83 |
| | *Monty Python's Life of Brian* | ⟶ | *Monty Python's The Meaning of Life* | 1.0 |
| | | ⟵ | | 1.0 |
| *Rocky* シリーズ | *Rocky* | ⟶ | *Rocky III* | 1.0 |
| | | ⟵ | | 0.71 |
| *Smokey and the Bandit* シリーズ | *Smokey and the Bandit III* | ⟶ | *Smokey and the Bandit II* | 1.0 |
| | | ⟵ | | 0.71 |
| *Star Wars* シリーズ | *Return of The Jedi* | ⟶ | *The Empire Strikes Back* | 1.0 |
| | | ⟵ | | 0.63 |
| *Tarzan* シリーズ | *Tarzan and the Trappers* | ⟶ | *Tarzan's Fight for Life* | 1.0 |
| | | ⟵ | | 1.0 |
| *The Three Musketeers* シリーズ | *Four Musketeers* | ⟶ | *The Three Musketeers* | 1.0 |
| | | ⟵ | | 0.66 |

表 3　評価結果の割合

| スコア | rank 1 | rank 2 | rank 3 | rank 1 ～ 3 における割合 |
|---|---|---|---|---|
| similar | 27.2 % | 15.47 % | 3.83 % | 19.98 % |
| excellent | 3.27 % | 8.56 % | 9.36 % | 5.84 % |
| good | 2.98 % | 4.70 % | 5.96 % | 4.00 % |
| fair | 24.01 % | 32.32 % | 42.55 % | 26.97 % |
| bad | 42.05 % | 38.95 % | 38.30 % | 40.51 % |
| 計 | 100 % | 100 % | 100 % | 100 % |

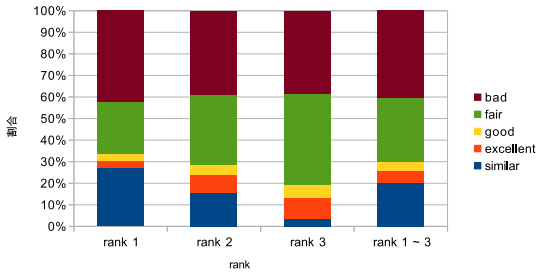


図 6　評価結果の割合

きいとされたタイトル群の約 60%が有効であることが分かった．いずれのランクにおいても 40%程度含まれる bad に該当するタイトルは，タイトルの 3-gram 文字列を共通プロパティとして持つタイトル同士であった．映画のエンティティが多く含まれるというデータの特性上，“劇場版” や “*THE MOVIE*” など複数のタイトルで現れやすい文字列が存在し，これにより，関連の無いタイトル間で類似度が得られている．これを解決するには，IDF (Inverse Document Frequency) 等を応用し，多くのエンティティに結び付いている，一般的なプロパティを発見し，プロパティノードあるいはエッジに対するフィルタの導入，またはエッジ重みの調整などの手法が有効であると考える．

また，bad を除いた場合の NDCG@3 (Normalized Discounted Cumulative Gain) は 0.97 であった．従って，類似度によるタイトルのランキングは妥当である．今後の課題として，bad に該当するタイトル排除のために上記の手法の他，類似度の有効下限閾値の導入，タイトルの 3-gram 文字列ではなくより適切なプロパティノードの採用を検討する．

## 6. ま と め

本研究では，実サービスで利用されている非構造化データか

ら自然言語処理を用い特徴となる要素を抜き出し，データが指すエンティティとそれに属するプロパティに分類，グラフ構造データに変換し，データが指すエンティティ間の非対称類似度を求める手法を提案した．

評価実験として，まずダミーデータを用い，データ規模の変化に対する類似度算出計算時間の推移の検証により，計算が実用的な時間に収まることを確認した．次に，規模の異なる 2 種類の実世界のデータを用いた検証と複数評価者の目視による評価により，あるエンティティから関連のあるエンティティとされたものの内約 60%が有効であることを確認した．

今後は，結果に対する定性評価から，類似度算出時のエッジ重みを更新し，類似度を改善する．さらに，Wikipedia データ等から多種多様なエンティティを導入し，本研究成果による類似度を用いたアプリケーションの多様性を拡大することを目指す．

## 文　献


[1] Christen, P.: *Data Matching: Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection*, Springer Publishing Company, Incorporated (2012).

[2] Zhou, Y., Cheng, H. and Yu, J. X.: Graph clustering based on structural/attribute similarities, *Proc. VLDB Endow.*, Vol. 2, No. 1, pp. 718–729 (2009).

[3] Ma, S., Cao, Y., Huai, J. and Wo, T.: Distributed graph pattern matching, in *Proceedings of the 21st international conference on World Wide Web*, WWW '12, pp. 949–958, New York, NY, USA (2012), ACM.

[4] Herzig, D., Mika, P., Blanco, R. and Tran, T.: Federated Entity Search Using On-the-Fly Consolidation, in Alani, H., Kagal, L., Fokoue, A., Groth, P., Biemann, C., Parreira, J., Aroyo, L., Noy, N., Welty, C. and Janowicz, K. eds., *The Semantic Web ISWC 2013*, Vol. 8218 of *Lecture Notes in Computer Science*, pp. 167–183, Springer Berlin Heidelberg (2013).

[5] Blanco, R., Cambazoglu, B., Mika, P. and Torzec, N.: Entity Recommendations in Web Search, in Alani, H., Kagal, L., Fokoue, A., Groth, P., Biemann, C., Parreira, J., Aroyo, L., Noy, N., Welty, C. and Janowicz, K. eds., *The Semantic Web ISWC 2013*, Vol. 8219 of *Lecture Notes in Computer Science*, pp. 33–48, Springer Berlin Heidelberg (2013).

[6] Rodríguez, M. A. and Egenhofer, M. J.: Comparing Geospatial Entity Classes: An Asymmetric and Context-Dependent Similarity Measure, *International Journal of Geographical Information Science*, Vol. 18, pp. 229–256 (2004).

[7] 岡部正幸, 梅村恭司：頻度差が著しい場合における一対多関係を推定する類似尺度 (2005 年情報学シンポジウム講演論文集–社会システムを支える情報学) – (セッション 4:情報学の実践), 情報学シンポジウム講演論文集, Vol. 2005, pp. 129–136 (2005).

[8] Malewicz, G., Austern, M. H., Bik, A. J., Dehnert, J. C., Horn, I., Leiser, N. and Czajkowski, G.: Pregel: A System for Large-scale Graph Processing, in *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data*, SIGMOD '10, pp. 135–146, New York, NY, USA (2010), ACM.