オープンデータの連携に関する研究
(Study on the cooperation of open data)

2020年3月

久永忠範

概要

近年，多くの行政・団体がオープンデータの公開，活用に取り組んでいるが政府の推進する「オープンデータは，機械判読可能で人手を多くかけずにデータの 2 次利用が可能である」というデータ活用までには至っていない．本研究では，地方自治体のオープンデータの可能性を探るために開示されている CSV データの項目名，列データをベクトル化することによりデータの連携度を測る述語ベクトル法を提案する．またその述語ベクトル法を用いて，各データ間の連携可能性について実験を行い，オープンデータ間における項目名のみの連携度と列データのみの連携度を比較した．結果として列データのみから算出した連携度が高いオープンデータは特定の列の類似度が高くなり，データ連携を取りやすいことが示された．

キーワード：オープンデータ，CSV，ベクトル化，類似度，述語ベクトル法，連携度

Summery

In recent years, many administrative organizations and local governments are engaged in opening and utilizing open data but until the data utilization has not reached where the government promotes "Open data can be secondarily used without much manpower being machine-readable". In this research, we propose a predicate vector method that measures the similarity of data by vectorizing item names and item values of CSV data disclosed to explore the possibility of open data of a local government. In addition, using the predicate vector method, we experimented on the possibility of cooperation between each data, and compared the degree of cooperation of each item name and the degree of cooperation of each column data among open data. As a result, it was shown that open data with a high degree of cooperation calculated from column data has a high degree of similarity of a specific column, making it easy to establish data cooperation.

Keywords: Open Data, CSV, Vectorization, Degree of similarity, Predicate vector method, Degree of cooperation

目次

図目次

表目次

第１章　序論

IT（情報通信技術）の進展に伴い，私たちの社会生活や経済の成長，そして国の政策や行政活動にも大いに影響を与えている．アメリカ合衆国のオバマ政権では，2009 年からオープンガバメント政策[1]が積極的に推進され，①透明性，②市民参加，③官民連携の 3 つの基本原則が示された．日本政府においてもオープンガバメントの取り組みが，各分野において広がってきている．それにともない近年，ビッグデータ[2 やオープンデータ[3 の活用が推進され，2011 年 3 月の東日本大震災[4 を契機にオープンデータ活用が高まり，国や地方公共団体をはじめ多くの団体がオープンデータの公開，活用に取り組んでいる．

2000 年に内閣官房情報通信技術（IT）総合戦略室[5]設置され，これは高度情報通信ネットワーク社会推進戦略本部（IT 総合戦略本部）[6]事務局の役割を果たすと共に，IT の利活用による国民の利便性の向上及び行政運営の改善に係る総合調整などを行い，国や地方公共団体のオープンデータ利活用の推進役も担っている．

ただこのようなオープンデータを活用したサービスをどのように利用していくかは，これまでに国や地方公共団体が積上げたデータはどのようなものがあるか，ホームページやデータカタログサイドで開示されているオープンデータのデータ形式やデータの組み合わせをどのように行うかなど，まだ不明な点などもあり，多くは利活用まで至っていない．また利用者側の一般市民や民間企業からすると，オープンデータを活用しやすい環境が整っているとは言い難く，オープンデータの存在すら知らない人も多いのが現状である．これらを鑑みて，2009 年以降オープンガバメントの方向性の明示やいろいろな施策を決定してきている．図 1 は，これまでのオープンデータに関する国の主な取組・施策を示したものである．

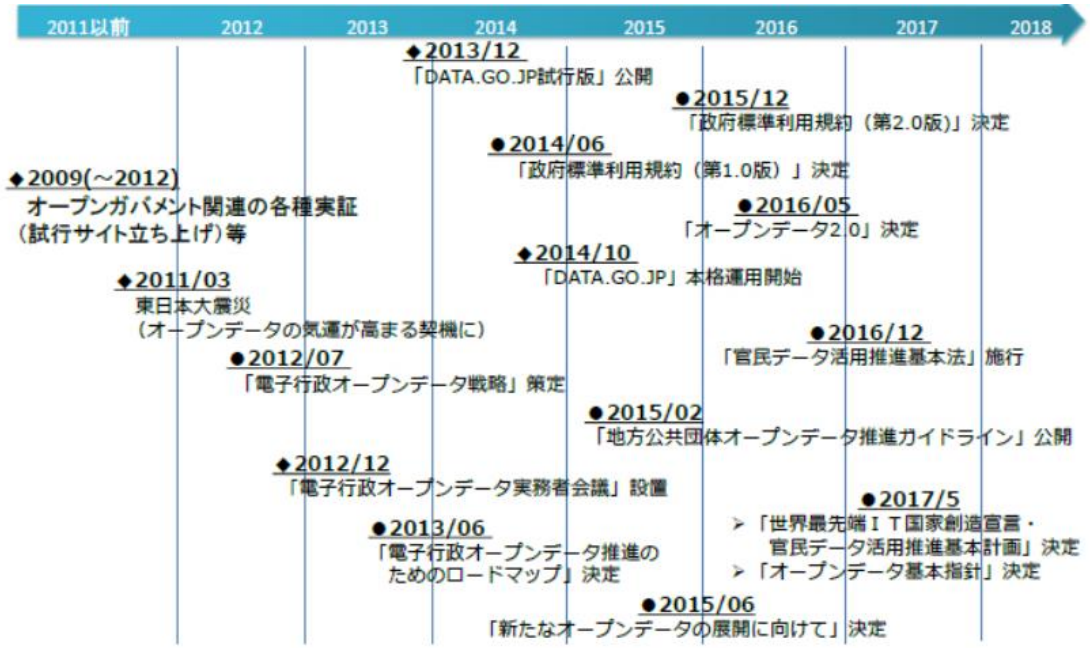


図 1　オープンデータを始めよう～地方公共団体のための最初の手引書[7]

国の取組として，オープンガバメントの推進に当たっては，オープンデータは国民共有の財産であるという認識の下，オープンデータの活用を促進するための取組に速やかに着手し，それを広く展開することにより，国民生活の向上，企業活動の活性化等を図り，我が国の社会経済全体の発展に寄与することが重要であるため，オープンデータの活用促進のための基本戦略として，「電子行政オープンデータ戦略」[8]された．

電子行政オープンデータ戦略における主な内容は，オープンデータの活用を促進する意義・目的，これまでのオープンガバメントの取組，海外の動向と我が国の現状，オープンデータ活用の取組を進めるための基本的な方向性，またその基本的な施策と推進体制等が記されている．

このオープンデータの活用を促進する意義・目的には，次のような 3 つのことが掲げられた．

1.　透明性・信頼性の向上

公共データが二次利用可能な形で提供されることにより，国民が自ら又は民間

のサービスを通じて，政府の政策等に関して十分な分析，判断を行うことが可能になる．それにより，行政の透明性が高まり，行政への国民からの信頼を高めることができる．

2. 国民参加・官民協働の推進

広範な主体による公共データの活用が進展し，官民の情報共有が図られることにより，官民の協働による公共サービスの提供，さらには行政が提供した情報による民間サービスの創出が促進される．これにより，創意工夫を活かした多様な公共サービスが迅速かつ効率的に提供され，厳しい財政状況，諸活動におけるニーズや価値観の多様化，情報通信技術の高度化等我が国を取り巻く諸状況にも適切に対応することができる．

3. 経済の活性化・行政の効率化

公共データを二次利用可能な形で提供することにより，市場における編集，加工，分析等の各段階を通じて，様々な新ビジネスの創出や企業活動の効率化等が促され，我が国全体の経済活性化が図られる．

また，国や地方公共団体においても，政策決定等において公共データを用いて分析等を行うことで，業務の効率化，高度化が図られる．

これらは，オープンデータの価値の創造とどのようにその価値を活用していくか，またその活用により，国民にとってどのようなメリットがあるかを示している．

オープンデータ活用の取組を進めるための基本的な方向性において，次の 4 つの基本原則[9]も定められている．

① 政府自ら積極的に公共データを公開すること

② 機械判読可能な形式で公開すること

③ 営利目的，非営利目的を問わず活用を促進すること

④ 取組可能な公共データがから速やかに公開等の具体的な取組に着手し，成果を確

実に蓄積していくこと

① の政府自ら積極的に公共データを公開することに対しては，政府は 2014 年 10 月にデータカタログサイト DATA.GO.JP[10]の本格運用を開始して，国の保有する省庁を跨いだオープンデータの開示を促進した（図 2）．近年では，国のデータだけでなく，地方公共団体，独立行政法人等，その他民間団体等のもつオープンデータのデータカタログサイトとなっており，各データサイトの名称・組織名・ライセンス・ＡＰＩの有無・主な分類・概要・更新日等が掲示され，オープンデータの活用を推進している．

図 2　Data.go.jp のトップページ

②の機械判読可能な形式で公開すること，と③の営利目的，非営利目的に問わず活用を促進することを要約すると「オープンデータは機械判読に適したデータ形式で二次利

用が可能な利用ルールで公開されたデータである必要があり，それによって人手を多くかけずにデータの二次利用を可能にする」と考えられる．

オープンデータは，機械判読の容易性，著作権等の扱いにより，その開放性の程度が

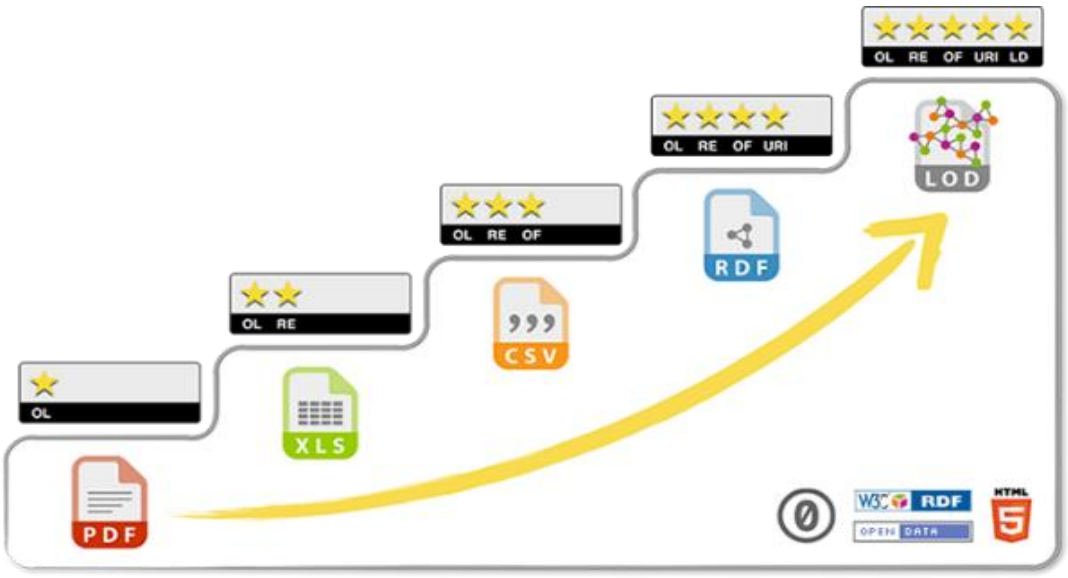


図 3　5つ星オープンデータ

異なっている．これらのオープンデータの形式を Tim Berners-Lee[11]が示す「5 つ星オープンデータ」図 3 の 5 つの段階とそれに値するデータ形式を提言した．[12]

1 つ星の第 1 段階は，データ形式が PDF，JPG 等のオープンライセンスでのデータ公開であり，編集が不可能であるので機械判読には適さない．

2 つ星の第 2 段階は，データ形式が XLS，DOC 等のコンピュータ処理可能な特定のアプリケーションに依存しているいるので，それらのアプリケーションがなければ編集ができないので，一般の機械判読には適さない．

3 つ星の第 3 段階は，データ形式が XML，CSV 等のオープンに利用できるフォーマットでデータが公開されているので，データフォーマットの規則性が確立されていれば，コンピュータによる編集と機械判読は可能である．

4 つ星の第 4 段階は，データ形式が RDF（Resource Description Framework）[13]と呼ばれ，情報についてのメタデータを表記するための汎用的な手法を定めたデータ形

式の一つである．これは Web 標準のフォーマットでデータが公開されているので機械判読は可能である．

5 つ星の第 5 段階は，LOD（Linked Open Data）[14]と呼ばれ，RDF 形式を含め，物事の識別に URI を利用たり，他へのリンクを入れたデータ形式であるため，機械判読可能である．

しかし第 4，5 段階の RDF 形式や LOD 形式のデータ作成には，ある程度の IT やプログラムの知識が必要であり，一般の自治体職員がこの形式のデータを作成することは少々難しいと思われる．

④の取組可能な公共データがから速やかに公開等の具体的な取組に着手し，成果を確実に蓄積していくことについては，地方公共団体においては，地域独自のオープンデータポータルサイトを立ち上げたり，地方公共団体のホームページにオープンデータをアップするページを設けたりしているところが増えてきている．

このようにオープンデータの施策は，政府や地方自治自体等が推進してきているが，より具体的な社会課題の解決や利用促進を行うために 2016 年 5 月に「オープンデータ 2.0」[15]を情報通信ネットワーク社会推進戦略本部（IT 総合戦略本部）が決定した．この「オープンデータ 2.0」は，官民一体となったデータ流通の促進を行い，課題解決のためのオープンデータの「実現」を目指すものである．また 2020 年までにオープンデータの更なる深化を図る施策として，①政策課題を踏まえた強化分野の設定，②国及び地方公共団体におけるオープンデータの取り組みの推進，③地方公共団体における防災等の地域を跨いだ共通的な分野における取組の推進が掲げられた．特に③においては，政府 CIO 補佐官[16]による地方公共団体への訪問や，オープンデータ伝道師[17]の制度を活用した地方公共団体への人材派遣，オープンデータパッケージ及びオープンデータ 100[18]の横展開のみならず，地方の特性に応じた課題解決に向けた取り組みを支援することで，地方公共団体への普及啓発や利活用に向けた取り組みを推進することが謳わ

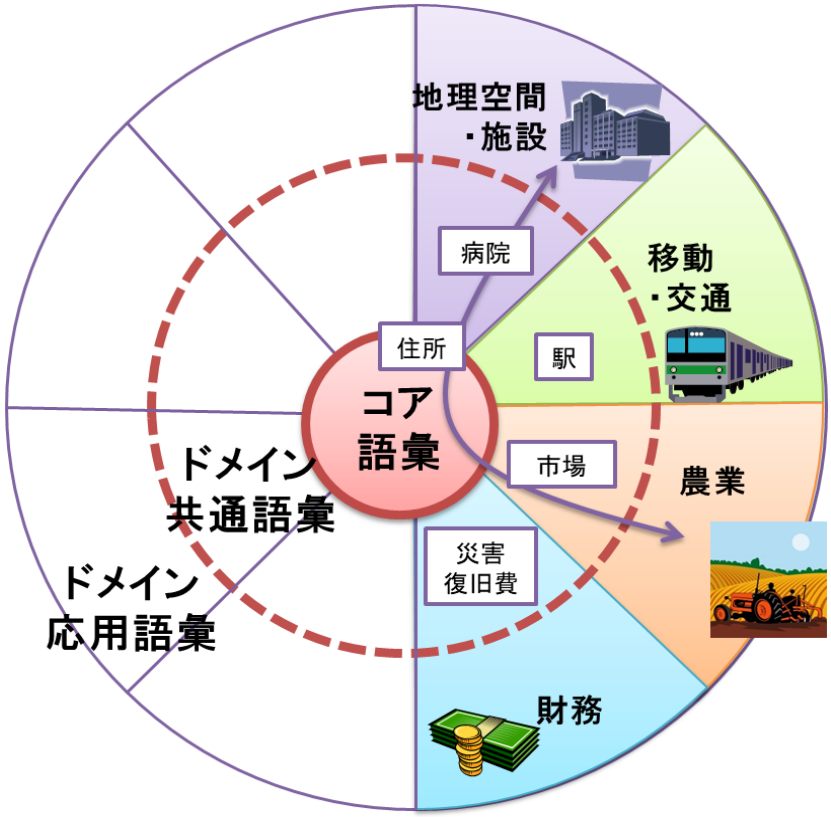


図 4　ＩＰＡ共通語彙基盤コア語彙解説より

れている．

しかし国の機関，全国の自治体のオープンデータの開示状況，データ形式を調査してみても，データ連携，機械判読に値する状況は未だに少なく，既存のデータをそのまま掲示してたり，画像や PDF を張り付けてあるオープンデータも見受けられる状況が多く見受けられる．また機械判読可能な CSV 形式であっても，項目名としてデータを作成する担当者の主観であったり，当該行の固有の表現が用いられており，語彙の統一性に欠けている状況も多くみられる．

オープンデータとして行政が保有するデータの利活用を推進していくためには，データで用いる様々な用語の表記や意味，データ形式等の構造を統一することが重要である．

情報連携に不可欠な基本情報やツールを提供する IMI 共通語彙基盤サイト[19]を独立行政法人情報処理推進機構（IPA）が運営している．ここでは，IMI 共通語彙基盤の中核をなす共通語彙，IMI 共通語彙基盤を基にしたデータモデル記述の DMD，共通語

彙を利活用するためのツール，IMI 共通語彙基盤活用の各種取組，事例紹介やさまざまな意見も募集するコミュニティも設置されている．共通語彙に関しては，語彙の階層構造をからなり，図 4 に示すようにコア語彙，ドメイン共通語彙，ドメイン応用語彙の 3 つに分類される．特にコア語彙においては，約 60 のクラス語彙と約 250 のプロパティ語彙からなり，語彙の共通化を図っている．ドメイン共通語彙は，個別のドメインのための語彙であり，ドメイン応用語彙は個別の利用ケースのための語彙である．これらはコア語彙の拡張として定義され，コア語彙と一緒に使われることによって，個別のドメインや利用ケースの語彙としての働きをもっている．

またこれらの語彙の共通化においても約 6 万の文字のフォントや画数の統一も必要であり，それらの文字の情報を統一するために 2010 年から内閣官房，総務省，法務省，経済産業省，文化庁が協力して文字情報基盤構築[20]に着手し，2017 年に完了している（図 5）．そしてこの文字情報基盤をベースにして共通語彙基盤の活用がなされ，公共データをアプリケーションなどを利用することにより新たなデータの利活用されることになる．

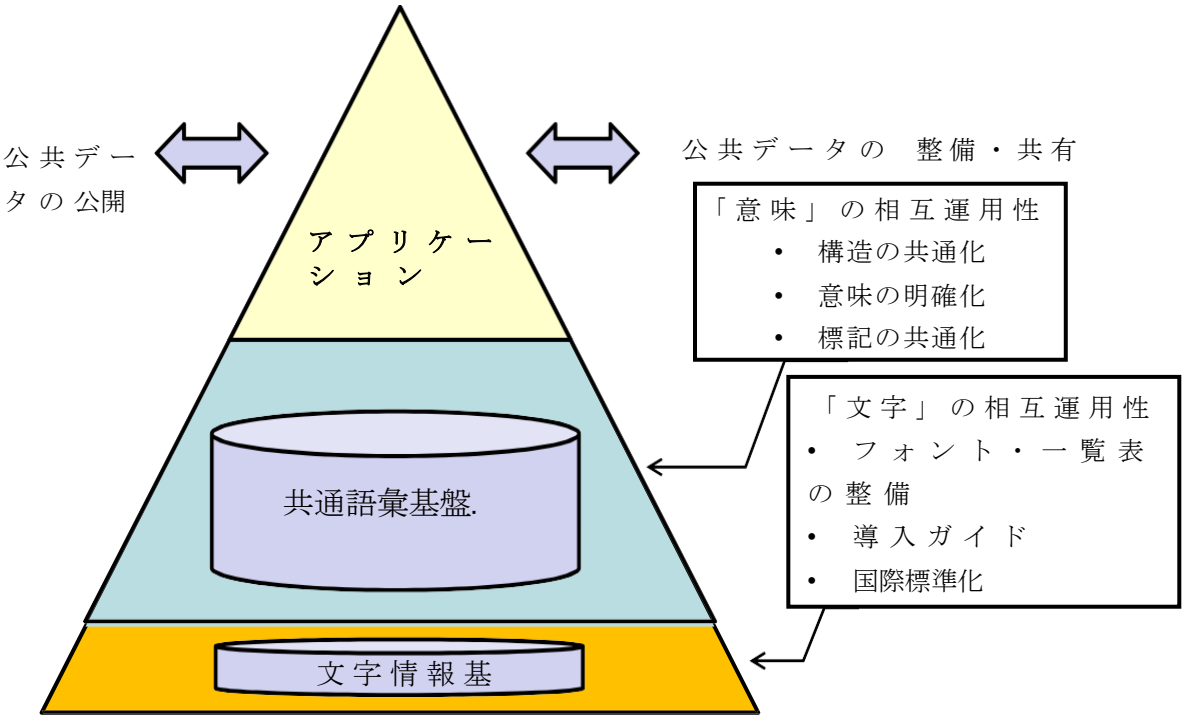


図 5　IPA 文字情報基盤と共通語彙基盤

2018 年に総務省が，オープンデータを推進する地方公共団体職員の人材育成のためにオープンデータ研修ポータルサイト[21]を開設した（図 6）.

図 6　総務省オープンデータ　研修ポータル

このサイトは，オープンデータを推進する地方公共団体職員を育成するため，必要な知見・技術を体系的に習得できる研修環境を整備し，オープンデータの取組に結びつけるところまで継続的に支援を行うこととしている. オープンデータリーダ育成研修やオープンデータ化支援研修等があり，データ作成の基本から e-learning による研修も盛り込まれている．このような支援研修により，今後はより多くの自治体職員が機械判読可能なデータ作成を行うことが可能になる.

本研究においては, 一般的なデータ形式として開示され易い 5 つの星オープンデータの第 3 段階における CSV データの連携に着目して，データの連携について調査した. 人による視覚的なデータ連携ではなく，機械判読による連携を行うために，本研究では

統計処理を用いた述語ベクトル法により，オープンデータ間の連携度の算出方法を提案する．

以降，第２章では，当研究における関連研究の紹介，第３章では提案手法である述語ベクトル法の説明，第４章ではこの述語ベクトル法による地方自治体のオープンデータの連携度の実験と考察，そして第５章では実験結果のまとめと今後の課題について記載する．

第 2 章　関連研究

2013 年 6 月に G8 サミットで合意された「オープンデータ憲章」[22]により，世界中で政府の持つデータの積極的なオープン化が進展してきている．日本政府もこれに伴い「世界最先端 IT 国家創造宣言」[23]を閣議決定し，官民にわたる多くの組織がオープンデータを活用し，分野を超えた情報交換を行うために，独立行政法人情報処理推進機構を中心に共通の単語の表記・意味・データ構造を統一するために「共通語彙基盤」の構築を進めている．ここでは，情報連携に不可欠な基本情報やツールを提供する共通語彙基盤サイト IMI（Infrastructure for Multi-layer　Interoperability）を開設している．データ連携に欠かせない共通語彙や IMI を基にしたデータモデル記述（DMD）も提供している．またデータ提供者が利用する機能で，表計算ソフトや CSV などのデータを DMD に定義された構造化データ（JSON－LD, Tutle, RDF/XML, XML）への変換も行うツールも用意されているが，一般の地方公共団体の職員が活用するにはある程度の知識が必要である．

また最近では，地方自治体のオープンデータの防災におけるデータ活用の研究[24][25][26]も進んでおり，特に 2011 年の東日本大震災や 2016 年の熊本地震などでは，SNS による情報発信や情報共有による災害時の対応に多くのデータが活用された．浦田ら[28]は，自治体におけるオープンデータ推進への受容性向上を目的として，地域防災情報におけるオープンデータ推進を目指した実践研究に取り組んでいる．防災情報のオープンデータにおける課題を論じた後に，熊本地震におけるニーズ調査を踏まえ，災害時生活情報のオープンデータ化の提案を行っている．この有用性を確認するために，愛知県尾張旭市と日進市を対象に，防災啓発アプリの企画・開発と実証実験を実施し，災害時生活情報のオープンデータ化の有用性の確認と地域課題であった自助意識向上

への寄与を達成している．これらを通じて，実際に自治体の災害時生活情報のオープンデータが公開され，またオープンデータ推進に対する庁内理解の促進に繋がったことから，自治体におけるオープンデータ推進への受容性向上が達成できたと考えられた．そのことから防災情報のオープンデータ化を推進するために，自治体職員の負担軽減や費用対効果を考え，自治体におけるオープンデータ推進への受容性向上を目的とした研究を進めている．

吉賀ら[29]は，多くの被災地では自治体から避難勧告が発令される等，避難行動を促す情報が出されたものの，自宅に留まる等により，多くの方が亡くなる結果となっていると述べている．このことから住民の立場に立てば，降雨の状況や住民が影響を受ける恐れのある河川の水位情報は，自らの行動を決定する重要な材料となり得ると考えられる．例え自治体から避難勧告発令以前であっても，避難行動に時間を要する年少者や高齢者のいる場合でも，避難勧告を待たずして避難の準備や避難行動を行うことで災害を未然に防げることに着目している．これらのことを解決するために，水防情報を構成する河川の水位，雨量，潮位および風向風速情報を Linked Data として蓄積し，機械可読な標準的ウェブ API により配信提供するシステムについて検討している．これは，国土交通省や都道府県が提供する防災情報や水位情報が個別の表として提供されており，表同士を結び付けるリンクが付されてはいるものの．河川と水系は別サイトを参照しなければならないなど，閲覧者が直接見ることを想定しており，機械可読なデータとして公開されいないことが原因であると述べている．

菱田[30]は，地方自治体のオープンデータに着目し，データのばらつきや不足情報による公開者及び利用者の負担を無くすことで，データ活用を円滑にする手法について検討を行っている．そして，オープンデータの現状を調査し整理したデータを元に，公開されるデータ項目を統一し，必要項目が含まれているかのチェックシステムを構築している．また不足しているデータに対しては，情報収集支援システムを構築することで，

不足データの収集を行い，オープンデータの補強を行っている．その構築したシステム用いて実地検証を行い，実際のオープンデータを利用して動作する確認ができたと述べている．AED 設置情報のオープンデータを利用して，データの活用や補足が可能であると検証をしているが，今後は他の種類のオープンデータへの活用においても補足する機能を補完して有用性があるとも述べている．結論として地方自治体のオープンデータには，同じ内容の項目であっても，いろいろな呼称の項目名が用いられて，データの内容から項目名の意味を推測する必要があるものもあると述べている．

またオープンデータ活用を推進する要因として，2014 年に地域の課題解決に向けたコミュニティづくりと作品コンテストとしてアーバンデータチャレンジ[31] が設立された．このアーバンデータチャレンジ（UDC）は，地域課題の課題解決を目的に，主に地方自治体を中心とするオープンデータを活用したデータ活用型コミュニティづくりと一般参加を伴う作品コンテストの 2 つのパートで構成されている．前者においては，2014 年から 2019 年の 6 年間に「地域拠点」と称して，各都道府県単位から 1 つずつ活動の核となる場を作りながら，地理空間情報の流通や利活用を日本各地で促進する活動を行ってきている．後者についても，毎年広く募集を行い，地域課題解決に資する優良な作品を表彰する取り組みを行っている．私自身もこのアーバンデータチャレンジの鹿児島のコーディネータとして 2017 年から参加し，鹿児島のオープンデータ活用について，鹿児島市情報システム課と渕田研究室との連携により鹿児島市のオープンデータの推進に係ってきた．

これらの施策や関連研究においては，オープンデータの利活用の推進はなされているが，実際のオープンデータの連携や国の推進する機械判読可能なデータ活用にまでは至っていないことがわかる．これらの関連研究[32][33][34][35][36]をもとに，本研究においては，現在のオープンデータの連携度について検証を行った．

第 3 章　提案手法

本章ではオープンデータ間の連携度を計算するために述語ベクトル法を提案し，それを用いて具体的な連携度の算出方法について説明する．述語ベクトルとは，オープンデータの 1 つの列の内容を数値化したベクトルのことである．述語ベクトルを計算するためにまず全国の地方公共団体が公開している CSV 形式のオープンデータを収集し，あらかじめ定義した約 15,000 の項目判定関数に基づき述語ベクトルを生成する．項目判定関数については後述する．得られた述語ベクトルからオープンデータの各列間類似度を計算し，オープンデータ間の連携度合いを測る．

3.1　オープンデータの収集

地方公共団体の保有するオープンデータをデータカタログサイト「DATA.GO.JP」の中のデータベースサイト，地方公共団体の 359 のリンクサイト[37]から 3 万件のオープンデータを収集した．そのデータの内，CSV 形式である 24,913 ファイルを抽出した．

3.2　項目名と列データの関連について

抽出した CSV 形式のファイルを調べてみると，最初の行に項目名ではなく，データの内容の説明書きがあったり，ファイルの作成年月日等が記載されているものもあった．また各地方公共団体の施設情報を比較してみると，1 行目が項目名，2 行目から項目値のファイルであっても，項目名が地方公共団体によって表記仕方が違ったり，記号や英語表記のものも多くみられた．

表 1 は，広島市オープンデータの区民文化センターの施設情報[38]である．

表 1　広島市のオープンデータ＞文化施設＞区民文化センター

名称	X座標	Y座標	所在地	TEL	FAX	URL	開館時間	休館日
広島市まちづくり市民交流プ	132.4582215	34.39130456	広島市中区袋町6番36号	082-545-3911	082-545-3838	http://www.cf.city.	9時30分～22時	月の第3月曜日、年末年
アステールプラザ（文化創造	132.4480662	34.38781836	広島市中区加古町4番17号	082-244-8000	082-246-5808	http://www.cf.city.	9時～21時	年末年始（12月29日～
文化創造センター	132.4480662	34.38781836	広島市中区加古町4番17号	082-244-8000	082-246-5808	http://www.cf.city.	9時～21時	年末年始（12月29日～
中区民文化センター	132.4480662	34.38781836	広島市中区加古町4番17号	082-244-8000	082-246-5808	http://www.cf.city.	9時～21時	月曜日（祝日及び8月
国際青年会館	132.4480662	34.38781836	広島市中区加古町4番17号	082-244-8700	082-246-5808	http://hiyh.pr.arena	9時～21時	年末年始（12月29日～
東区民文化センター	132.4831885	34.39568779	広島市東区東蟹屋町10番31号	082-264-5551	082-264-5774	http://www.cf.city.	9時～21時	月曜日（祝日及び8月
南区民文化センター	132.468576	34.38102384	広島市南区比治山本町16番27号	082-251-4120	082-256-8811	http://www.cf.city.	9時～21時	月曜日（祝日及び8月
西区民文化センター	132.4486264	34.40876247	広島市西区横川新町6番1号	082-234-1960	082-293-1860	http://www.cf.city.	9時～21時	月曜日（祝日及び8月
安佐南区民文化センター	132.4744285	34.45171263	広島市安佐南区中筋一丁目22番17号	082-879-3060	082-879-3061	http://www.cf.city.	9時～21時	月曜日（祝日及び8月
安佐北区民文化センター	132.5077138	34.5260475	広島市安佐北区可部七丁目28番25号	082-814-0370	082-814-0770	http://www.cf.city.	9時～21時	月曜日（祝日及び8月
安芸区民文化センター	132.5266214	34.37217711	広島市安芸区船越南三丁目2番16号	082-824-1330	082-824-1337	http://www.cf.city.	9時～21時	月曜日（祝日及び8月
佐伯区民文化センター	132.3595789	34.37964654	広島市佐伯区五日市中央六丁目1番10号	082-921-7550	082-921-9898	http://www.cf.city.	9時～21時	月曜日（祝日及び8月

項目名は，名称，X 座標，Y 座標，所在地，TEL，FAX，URL，閉館時間，休館日の９つの要素から成り立っている．これに準ずる他の地方公共団体の施設情報の類似した項目名を調べてみた結果を表 2 に示す．

表 2　広島市の施設情報項目名と他地方公共団体の類似した項目名

広島市施設情報の項目名	類似した項目名
名称	施設名，建物名，名前，物名，呼称
X 座標	緯度，北緯，位置
Y 座標	経度，東経，位置
所在地	住所，場所名，位置
TEL	電話，電話番号，ナンバー
FAX	Fax 番号，Fax ナンバー
URL	ホームページ，ホームページアドレス，ウェブサイト
開館時間	営業時間，運営時間，Open
休館日	休日，休み，Closed

「名称」という項目名に対しても「施設名」「建物名」「名前」「物名」「呼称」等，数多くの表現が用いられていることがわかった．したがって，各オープンデータの列間の項目名だけをみても，各列間の類似度を比較することは難しく，項目名よりは項目名の

下に位置する列データを比較することが類似度を測る尺度になるのではないかと考えられる．

3.3　述語ベクトルの生成

述語ベクトルを生成するために，ある CSV ファイルの項目名以下にある列データひとつひとつに判定条件を照らし合わせ，それと一致すれば 1 を，一致しなければ 0 を返し，その合計数を列データ数で割り平均を取る項目判定関数を準備する．図 7 に CSV ファイルにおける項目名と列データを示す．

図 7　CSV ファイルにおける項目名と列データ

この関数から得られた数値の連続がその項目名の述語ベクトルとなる．図 8 に述語ベクトルの生成方法を視覚的に示す．

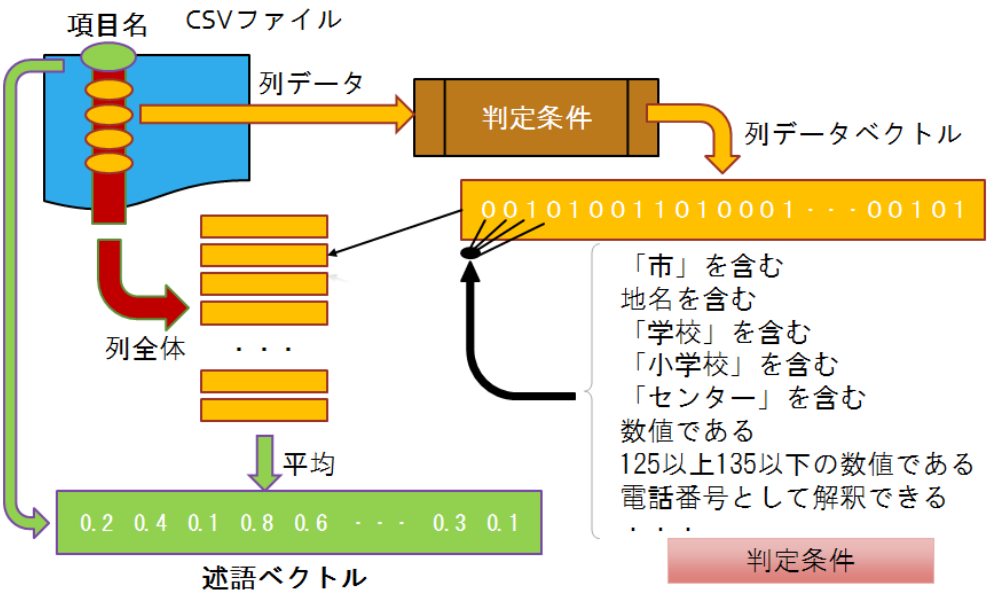


図 8　述語ベクトルの生成方法

また述語ベクトルの生成と同時に，ある項目名のすべての列データが数値，または空のデータである場合はその項目名は述語ベクトルを生成しないようにしている．なぜならそのような列データから取得される値が数量なのか，あるいは面積なのか何のデータなのか把握するのは難しいからである．今回の実験では不明な数値のみのデータは除外している．電話番号や FAX 番号の-(ハイフン)が含まれているものや緯度・経度等の一般的な表現形式に限り，正規表現を用いて判定されるようにしている．

次にすべての判定条件にヒットせず，述語ベクトルがすべて 0 である項目名の除去を行う．最後に項目名と述語ベクトルに判定条件を追加したものを CSV ファイルとして出力して生成完了である．

3.3.1　行数に応じた重み付け

オープンデータの各データの行数を調べてみると，少ないものは 1 行から多いものは数千行に及ぶものもあった．述語ベクトルは行数の平均なので，行数の少ないものは 1 つの行の影響が大きくなるので，述語ベクトルの各要素を行数に応じて式（1）に示すシグモイド関数[39]で重み付けする．ゲインaの値は，行数の中央値が約 38 であったこ

とから，38 行程度のデータの場合に 1 になるように$a = 0.2$とした．図 9 に$a = 0.2$の場合のグラフを示す．

$$f(x) = \frac{2}{1 + e^{-ax}} - 1 \qquad (1)$$

ここで，xはオープンデータの行数である．

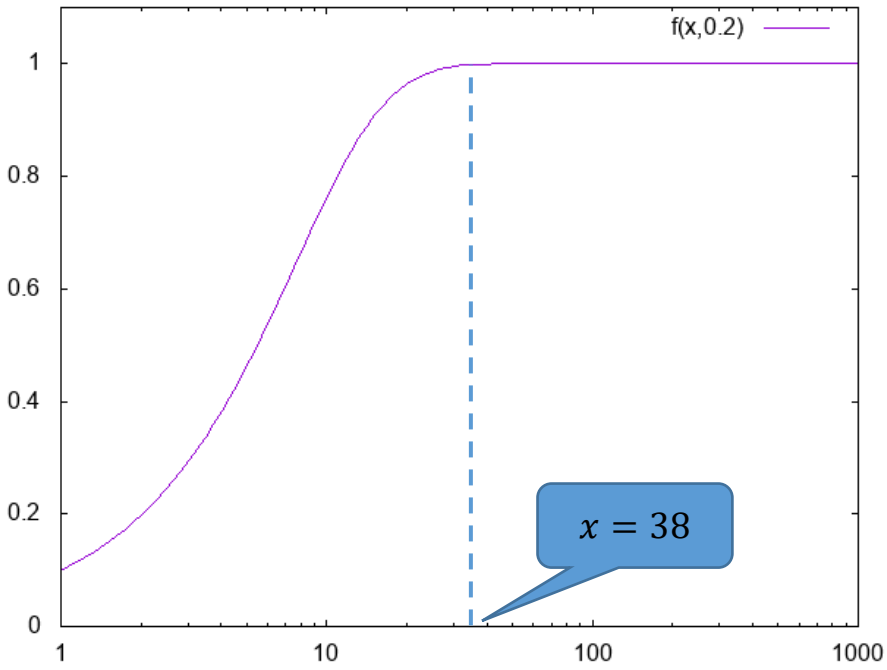


図 9　行数に応じた重み付けしたシグモイド関数

3.4　項目判定関数

述語ベクトルの生成における各項目値を判定するものを項目判定関数と呼ぶ．項目判定関数は，先に抽出した地方自治体の 13,000 ファイルの中の語彙を抽出して，MeCab(IPA 辞書)[40]を用いて分かち書きを行い，その分かち書きを行った 333,220 個の語彙から頻度 30 個以上の語彙 15,000 個の単語を抽出した．

3.4.1　項目判定方法

項目の判定方法には，Judges1,2,3 の 3 つの判定方法を用いた．

Judges1 は，該当する単語を含むか含まないかの判断を行う．例えば「施設」と言う単語を含む場合は 1 を返し，含まない場合は 0 を返す．このような単語を約 15,000 個

準備した．

Judges2 は，1 つの軸とする各グループ内の単語を一つでも含むかどうかの判断を行う．例えば「教育施設」という軸に対して「保育園，幼稚園，こども園，小学校，中学校，高校，大学」のいずれかを含めば 1 を返し，含まなければ 0 を返す．

Judges3 は，正規表現を用いて語彙の種類，性質に応じて判断を行う．例えば経度，緯度を判断するためには以下のような正規表現を用いた．

```
^(1(2[2-9]|[34][0-9]|5[0-3]))¥..*$
^([23][0-9]|4[0-5])¥..*$
```

これは，日本の経度は 122 度～153 度の範囲に入っており，緯度は 20 度～45 度の範囲に入っていると判断している．

述語ベクトル法の具体的な例を表 3 に示す．例えば，項目の列に施設名称のデータが入っていた場合に判定条件となる単語「鹿児島，駅，病院，公園，市立，県立，小学校，中学校，高校」に合致していれば 1 を返し，合致していなければ 0 を返すことになる．その項目列の平均が述語ベクトルとなる．

表 3　項目判定関数

項目名	施設名	項目判定関数(軸)								
		鹿児島	駅	病院	公園	市立	県立	小学校	中学校	高校
項目値	鹿児島市立病院	1	0	1	0	1	0	0	0	0
	鹿児島中央駅	1	1	0	0	0	0	0	0	0
	天文館公園	0	0	0	1	0	0	0	0	0
	鹿児島県立吉野公園	1	0	0	1	0	1	0	0	0
	鹿児島市立山下小学校	1	0	0	0	1	0	1	0	0
	維新ふるさと館	0	0	0	0	0	0	0	0	0
	鹿児島市立健康の森公園	1	0	0	1	1	0	0	0	0
	鹿児島県立鹿児島中央高校	1	0	0	0	0	1	0	0	1
	鹿児島駅	1	1	0	0	0	0	0	0	0
	平田公園	0	0	0	1	0	0	0	0	0
	述語ベクトル	0.7	0.2	0.1	0.4	0.3	0.2	0.1	0	0.1

3.5　列間類似度の計算

すべてのオープンデータについて各列の述語ベクトルを計算できたら，それを使ってすべての列の組み合わせに対して列間の類似度を計算する．オープンデータiの第k列の述語ベクトルをv_{ik}として，以下の式で列間類似度s_{ikjl}を計算した．

$$s_{ikjl} = v_{ik} \cdot v_{jl}$$

ここで「·」はベクトルの内積である．列間類似度をコサイン類似度によって計算する方法も考えられるが，述語ベクトルではベクトルの大きさに意味があるため，大きさを1に正規化する方法は適さないと判断し，内積を使用することとした．

3.6　オープンデータ間の連携度の計算

オープンデータ間の連携度とは，類似度とは異なり，どれだけ似ているかの尺度ではない．2つのオープンデータを連携させてアプリケーションを作成することを考える場合，それら2つのオープンデータのすべての列が似ている必要はなく，いくつかの特定の列に類似性があればよい．単純な例では，2つのオープンデータがどちらも緯度と経度の列を持っていれば，それらを両方とも地図上に表示するアプリを作成することが可能となる．

したがって，2つのオープンデータの連携度を計算するために，すべての列の類似度の平均を用いるのは適切ではなく，何らかの重みを付けて計算する必要があると考えられる．本節では，どのような重み付けが適切かを考察するため，まず基本的な計算式を示した後，いくつかの重み付けを用いた連携の計算式を示し，それぞれの連携度のうちどの式が適切かについて考察する．

3.6.1　基本的な計算式

2 つのオープンデータi, j間の連携度をそれぞれのオープンデータの列間類似度の平均として定義する．連携度$C(i, j)$の基本的な計算式を式(2)に示す．

$$C(i,j) = \frac{\sum_{k=1}^{n_i} \sum_{l=1}^{n_j} s_{ikjl}}{n_i n_j} \qquad (2)$$

ここで，i, jはオープンデータの番号，s_{ijkl}はオープンデータiのk列とオープンデータjのl列との類似度，n_i, n_jはそれぞれオープンデータi, jの列数である．

3.6.2 連携度の重み付け

オープンデータ間の連携度として，以下の 4 種類の重み付き連携度を定義した．

連携度 A

連携度 A（C_A）は，重みを付けない単純な列間類似度の平均として式(3)で定義する．

$$C_A(i,j) = \frac{\sum_{k=1}^{n_i}\sum_{l=1}^{n_j} s_{ikjl}}{n_i n_j} \qquad (3)$$

連携度 B

連携度 B（C_B）は，列間類似度の順位に応じて正規分布で重みを付けた類似度の平均として式(4)で定義する．

$$C_B(i,j) = \frac{\sum_{k=1}^{n_i}\sum_{l=1}^{n_j} G(o_{kl}) s_{ikjl}}{\sum_{k=1}^{n_i}\sum_{l=1}^{n_j} G(o_{kl})} \qquad (4)$$

ここで，o_{kl}は 2 つのオープンデータの列間類似度の順位であり，最も類似度が高いものを 0 番，次を 1 番の順で数える．$G(o_{kl})$は平均 0 の正規分布であり，標準偏差σは式(5)で計算した．

$$\sigma = 0.1 c_i c_j \qquad (5)$$

ここで，c_i, c_jはそれぞれオープンデータi, jの列数である．

連携度 C

連携度 C（C_C）は，列間類似度の順位に応じて減衰関数で重みを付けた類似度の平均として式(6)で定義する．

$$C_C(i,j) = \frac{\sum_{k=1}^{n_i}\sum_{l=1}^{n_j} e^{-o_{kl}/\tau} \cdot s_{ikjl}}{\sum_{k=1}^{n_i}\sum_{l=1}^{n_j} e^{-o_{kl}/\tau}} \qquad (6)$$

ここで，o_{kl}は 2 つのオープンデータの列間類似度の順位であり，τは以下の式で定義する減衰の時の定数である．

$$\tau = 0.5 c_i c_j$$

ここで，c_i, c_jはそれぞれオープンデータi, jの列数である．

連携度 D

連携度 D（C_D）は，閾値によって重み付けした類似度の平均として式(7)で定義する．

$$C_D(i,j) = \frac{\sum_{k=1}^{n_i} \sum_{l=1}^{n_j} h(s_{ikjl})}{n_i n_j} \qquad (7)$$

ここで，$h(x)$は閾値関数であり，閾値を超えた場合はxを返し，そうでなければ 0 を返す．ここでは閾値として 0.5 を用いた．

3.7 連携度 A～D の比較

連携度 A～D と行数による重み付けでどの列データ間のデータ同士が類似しているかの実験を全国の地方公共団体の施設情報を用いて，データ各列の述語ベクトルを比較する実験を行った．

3.7.1 施設情報を基にした連携度の比較

全国の地方自治体 626 個の施設情報の CSV ファイルに対して 302 次元（軸）の項目判定関数を用いて列データ間類似度を計算した．連携度 A(重みなし)と連携度 C（減衰関数）は，項目名が違っても同じような列の内容を抽出する傾向にあったが，一致する項目名が少ないことがわかった．連携度 B（正規分布）は，項目名の違いがあっても同じデータ内容のものが抽出され，多くの項目名と合致する傾向があることがわかった．連携度 D（閾値 0.5）は，多くの列同士の項目名が合致するが，同じ地方公共団体の同じような項目名と合致する傾向があることがわかった．図 10 は連携度 B の順位を降順に並べた時の連携度別グラフである．このグラフから連携度 A～B のは同じような傾向があることがわかる．上記の実験結果から類似度 B(正規分布)の計算式がより良い連

携度を表す傾向があることがわかった.

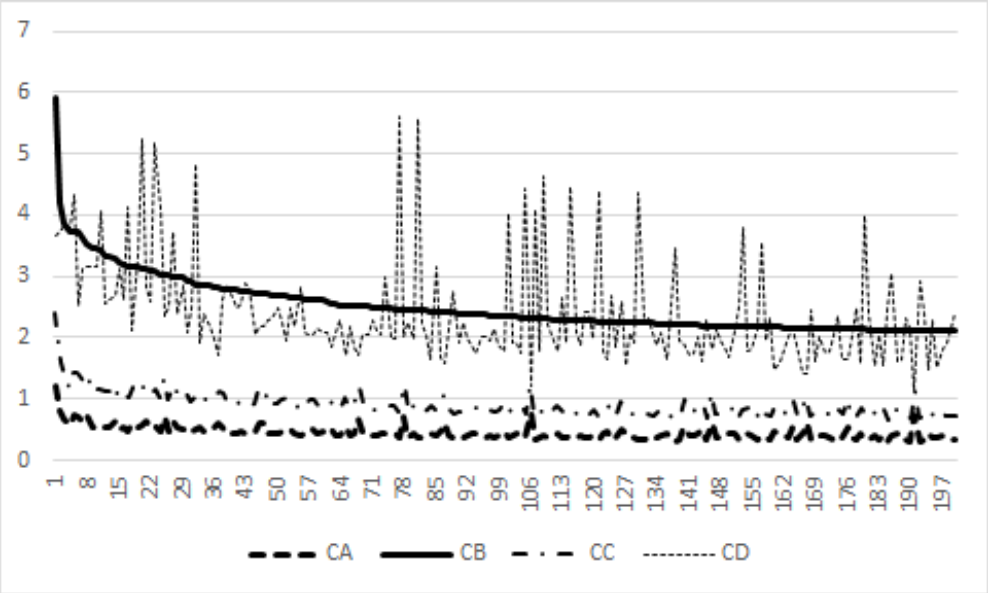


図 10　連携度 B の降順の他連携度

第 4 章　実験と考察

本章では，提案したオープンデータ間の連携度の有効性を検証するため，実際に地方公共団体が公開しているオープンデータを用いて連携度を計算する実験を行った. 比較のために，列データではなく項目名だけを対象とした述語ベクトルによる実験も行い，結果を考察した.

4.1　実験データの準備

政府のポータルサイト DATA.GO.JP から 142 の地方公共団体のオープンデータ 25,000 ファイルを抽出した．そのオープンデータの各地方公共団体の占める割合は，新潟市が 2,392 フィイル，横浜市が 2,302 ファイル，徳島市が 1,945 ファイルとオープンデータファイルの多い都市もあるが，越前市，光市，喜多方市，大津市，新宿区，草加市，天理市，敦賀市，宇部市，調布市，愛媛県など 1 つの CSV ファイルしかない都市もあった．そこでファイル数の多い都市に偏りがないようにランダムに各都市 10 ファイル以内で 300 のファイルを抽出した.

抽出した 300 のファイルのデータの内容がどのような分野に属するかを調べるために，各データの内容のチェックを行い，施設，防災，統計，環境，経済，行政，交通，医療福祉，文化，行事，生活，観光の 12 の分野に分類をおこなった．またそれぞれの分類でも表 4 に示すように他に分類の詳細な内容がわかるように付帯情報も追記した.

表 4　オープンデータの分類

オープンデータファイル名	オープンデータの分類（付帯情報）
www.city.tosu.lg.jp/Material/52444.csv	鳥栖市の統計情報（年齢別人口）
www.city.matsuyama.ehime.jp/shisei/opendata/metadata/syu_kizitsu.files/h26syusen2_ki	松山市の行政情報（投票者数）
www.city.hofu.yamaguchi.jp/uploaded/attachment/64440.csv	防府市の行政情報（予算額）
www.city.kagoshima.lg.jp/jousys/documents/4-2_jisinnjinotaihibasyo.csv	鹿児島市の施設情報（公園）
www.city.nagaoka.niigata.jp/shisei/cate10/toukei/file/toukei_27.csv	長岡市の行政情報（予算額）
www.city.maebashi.gunma.jp/sisei/499/509/p012146_d/fil/siminservicecenter.csv	前橋市の施設情報（サービスセンター）
opendata.pref.miyazaki.lg.jp/dataset/908/resource/2805/08_ibento.csv	宮崎県の行事情報（担当課別）
www.city.niigata.lg.jp/shisei/seisaku/it/open-data/opendata-kenkoiryo/od-kenkohigai.files	新潟市の福祉情報（検診受診者数）
www.pref.gunma.jp/contents/000363787.csv	群馬県の行政情報（適合施設）
www.city.nisshin.lg.jp/dbps_data/_material_/_files/000/000/021/926/23230_kyouikukikan.	日進市の施設情報（小学校）
www.city.nara.lg.jp/www/contents/1147846435189/files/bunkan_csv.csv	奈良市の施設情報（公民館分館）
www.city.yamato.lg.jp/web/content/000122731.csv	大和市の行政情報（犬の登録等）
www.city.yamagata-yamagata.lg.jp/kakuka/kankyo/gomigen/sogo/gazoufile/gomishusekijo	山形市の環境情報（ごみ収集場所）
www.city.sumida.lg.jp/kuseijoho/sumida_info/opendata/opendata_ichiran/gyoseikisosiryo/	墨田区の統計情報（教育機関障害者数）
www.city.aizuwakamatsu.fukushima.jp/docs/2007091400823/files/odei0823.csv	会津若松市の環境情報（浄水場調査）
www.city.ikoma.lg.jp/cmsfiles/contents/0000013/13459/292095_population_20180401.csv	生駒市の統計情報（人口）
www.city.utsunomiya.tochigi.jp/_res/projects/default_project/_page_/001/010/185/syokuty	宇都宮市の福祉情報（食中毒）
www.city.toyohashi.lg.jp/secure/16259/21daichou.csv	豊橋市の統計情報（年月日別世帯人口）
www.city.shima.mie.jp/ikkrwebBrowse/material/files/group/2/2016ago_gomicalendar.csv	志摩市の環境情報（ごみ処理）
www.city.sumida.lg.jp/kuseijoho/sumida_info/opendata/opendata_ichiran/gyoseikisosiryo/	墨田区の統計情報（施設利用数）

4.2　実験の流れ

オープンデータ間の連携度を算出する流れは以下のとおりである．

① オープンデータのすべての列についての述語ベクトルを計算する．
② 2 つのオープンデータのすべての列の組み合わせに対して列間類似度を計算する．
③ 連携度 B を用いてすべてのオープンデータ間の連携度を計算する．

ただし，今回は 2 列以上の列を持つデータを対象とし，列数が多いと計算にかなりの時間を要するので，列数が 2 以上 20 以下のファイルを対象とし，44850 ペアの連携度の結果を得た（表 5）．

表 5　列データを用いたオープンデータ間連携度の上位 20 位まで

順位	オープンデータ1	オープンデータ2	連携度B
1	品川区の防災情報（防災無線）	江戸川区の防災情報（無線設置場所）	5.911649945
2	三条市の施設情報（小学校）	半田市の統計情報（小学世帯数）	4.207914452
3	宇部市の施設情報（ふれあいセンター）	府中市の施設情報（介護施設）	3.86827203
4	江戸川区の施設情報（子育てひろば）	江戸川区の施設情報（公園場所）	3.736977013
5	墨田区の統計情報（施設利用数）	宇部市の施設情報（ふれあいセンター）	3.73477836
6	永平寺町の施設情報（教育機関）	永平寺町の施設情報（避難所）	3.712774825
7	日進市の施設情報（小学校）	三条市の施設情報（小学校）	3.622607361
8	日進市の施設情報（小学校）	半田市の統計情報（小学世帯数）	3.537816723
9	江戸川区の施設情報（小学校住所）	江戸川区の施設情報（公園場所）	3.454594508
10	板橋区の施設情報（地域センター）	宇部市の施設情報（ふれあいセンター）	3.446024132
11	豊川市の施設情報（公園）	豊川市の施設情報（公園）	3.426980983
12	日進市の施設情報（小学校）	秦野市の生活情報（小学校情報）	3.342527594
13	江戸川区の施設情報（子育てひろば）	江戸川区の施設情報（小学校住所）	3.337788509
14	日進市の施設情報（小学校）	長久手市の施設情報（教育機関）	3.289943842
15	三条市の施設情報（小学校）	秦野市の生活情報（小学校情報）	3.24152996
16	宇部市の施設情報（ふれあいセンター）	福島県の施設情報（子育て支援センター）	3.192532406
17	宇部市の施設情報（ふれあいセンター）	浦添市の環境情報（ごみ収集日）	3.164679583
18	板橋区の施設情報（地域センター）	府中市の施設情報（介護施設）	3.157339014
19	綾瀬市の行政情報（各施設紹介）	坂井市の観光情報（施設紹介）	3.151520694
20	半田市の統計情報（小学世帯数）	笠間市の統計情報（小学校学年別人数）	3.131830284

4.3　列データ間類似度による連携度

列データ間の類似度をもとに各オープンデータ間の連携度の上位 1 位から 7 位と 1000 位の列の項目名同士類似度の内容を調べてピボットテーブルを用いて，列データ間の類似度をグラフ化した．

4.3.1　連携度の第 1 位（連携度 5.91）

連携度の第 1 位は品川区の防災情報（防災無線設置場所）と江戸川区の防災情報（防災無線設置場所）となり，防災情報が一致している（図 11）．項目名においては品川区の種別，緯度，経度に対して，江戸川区は名称，緯度，経度に対応している．

品川区の項目名：「種別」の項目値は，「防災行政無線固定系」であり，江戸川区の項目名：「名称」の項目値は，「行政防災無線」と内容が似通っていることがわかる．項目名が違っても，項目値が同じ意味を示す単語であれば類似度も高くなっていることがわかる．

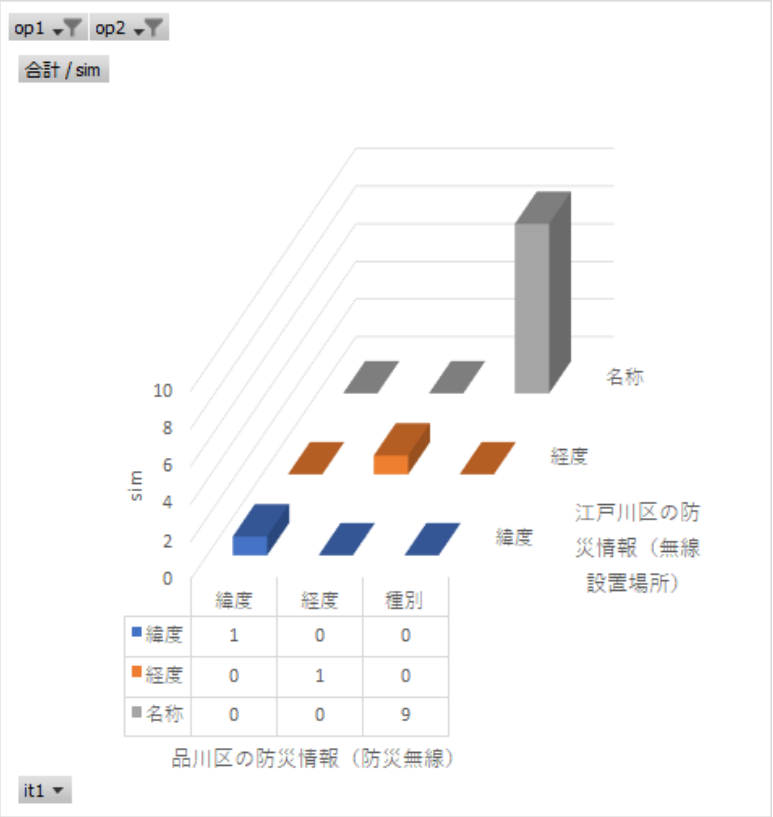


図 11　品川区の防災情報と江戸川区の防災情報

4.3.2　連携度の第 2 位（連携度 4.21）

連携度の第 2 位は半田市の統計情報（小学校の世帯数）と三条市の施設情報（小学校）となった（図 12）. 各々の項目名は，半田市が小学校区名称，年月日であり，三条市は，address, category, latitude, longitude, name, phone_number, uri という英語表記であった. 半田市の項目名：「小学校区名称」対して，三条市の項目名が英語表記の「category」「name」の列データ間類似度が高いのは，どちらにも「小学校」という単語が含まれていた．また半田市の項目名「年月日」と三条市の項目名「phone_number」の列データ間類似度が高いの半田市の「年月日」の列データ値が 2017-04-30 の様にハイフンで 3 つに区切ってあることが，三条市の「phone_number」のハイフンで 3 つに区切ってあるという正規表現と合致し，列データ間類似度が高い数値になっていることが原因である．この類似度は，イレギュラーな結果となっている．

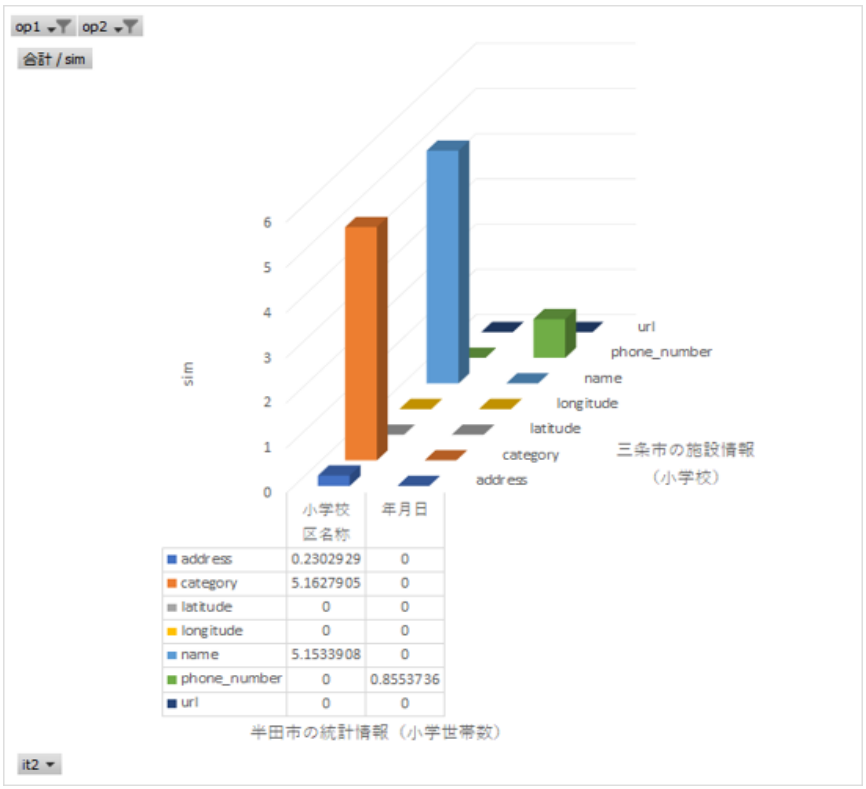


図 12　半田市の統計情報と三条市の施設情報

4.3.3　連携度の第 3 位（連携度 3.87）

連携度の第 3 位は宇部市の施設情報（ふれあいセンター）と府中市の施設情報（介護施設）となった（図 13）．宇部市の項目名は，サマリ，タイトル，メモタイトル 1，メモ内容 1，大字名，番地 2 で，府中市の項目名は，名称，郵便番号，所在地，電話番号，FAX である．府中市の項目名の「名称」に対して，宇部市のサマリ，タイトルの列データ間類似度が高いのは，どちらともセンターという語彙が多く含まれていた．また府中市の項目名「所在地」と宇部市の項目名「メモ内容 1」，「大字名」の列データ間類似度が高いのは，「メモ内容 1」と「大字名」に住所が含めれていることが大きな類似度の高い要因と考えられる．府中市の項目名「電話番号」「郵便番号」に対しては，宇部市の項目名に該当するものがなく，類似度は 0 になっている．

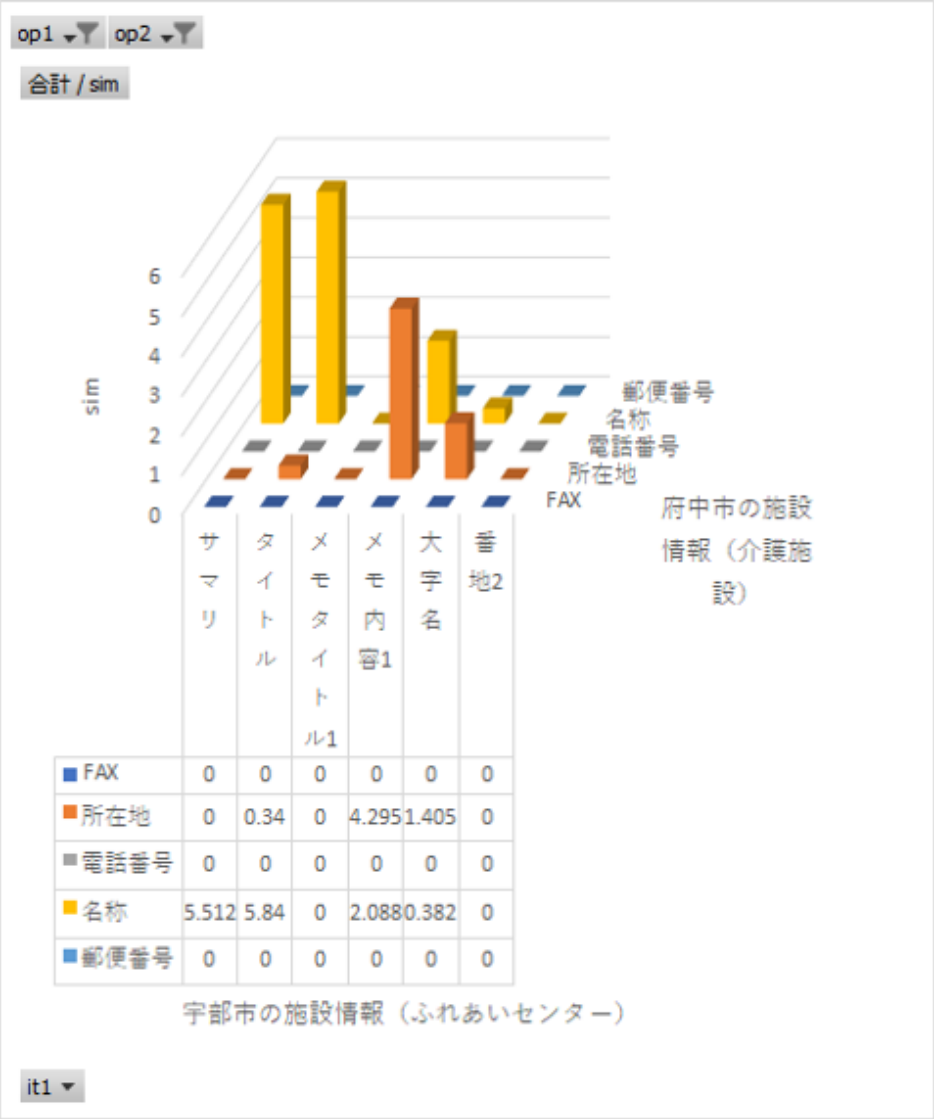


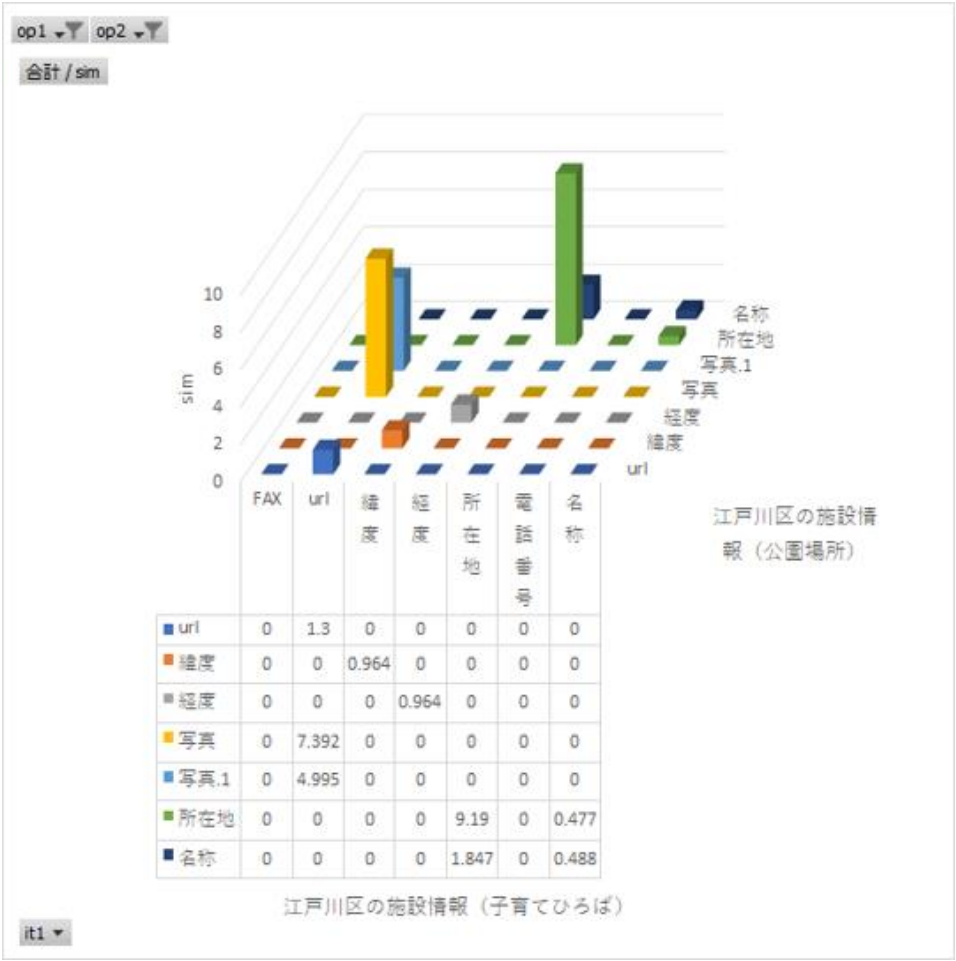
図 13　府中市の施設情報と宇部市の施設情報

4.3.4　連携度の第 4 位（連携度 3.74）

連携度の第 4 位は，江戸川区 1 の施設情報（子育てひろば）と江戸川区 2 の施設情報（公園場所）となった（図 14）．江戸川区 1 の項目名は，名称，電話番号，所在地，緯度，経度，url, FAX であり，江戸川区 2 の項目名は，名称，所在地，写真 1，写真，緯度，経度，url である．同じ地方公共団体の施設情報なので，ある程度項目名のフォーマットも似通っているが，江戸川区 1 の項目名「url」に対して江戸川区 2 の項目名「写真」，「写真 1」の項目名が違うのに列データ間類似度が高いのは，どちらとも写真の保

存先の url が表示されていることが要因である．

江戸川区 1 の項目名「所在地」と江戸川区 2 の項目名「所在地」の類似度の高いのは，同じ地区の住所が多く合致しているからである．ここでは，お互いの各々の「所在地」「名称」の類似度も高く出ているのも同じ要因と考えれらる．



	FAX	url	緯度	経度	所在地	電話番号	名称
url	0	1.3	0	0	0	0	0
緯度	0	0	0.964	0	0	0	0
経度	0	0	0	0.964	0	0	0
写真	0	7.392	0	0	0	0	0
写真.1	0	4.995	0	0	0	0	0
所在地	0	0	0	0	9.19	0	0.477
名称	0	0	0	0	1.847	0	0.488

図 14　江戸川区１の施設情報と江戸川区２の施設情報

4.3.5　連携度の第 5 位（連携度 3.73）

連携度の第 5 位は，宇部市の施設情報（ふれあいセンター）と墨田区の統計情報（施設利用数）となった（図 15）．宇部市の項目名は，サマリ，タイトル，メモタイトル，メモ内容，大字名，番地２で，墨田区は，緑と花の学習園，緑化相談件数である．墨田区のこの CSV データは，1 行目に項目名がなく，1 列目，2 列目に項目名が表記されている．これは表 6 を見るとわかるように，1 列目と 2 列目の列データ値を合わせて，項

目名として緑と花の学習園の緑化相談件数とするべきところを 2 つの列データ値として表示されているので，宇部市の項目名「サマリ」,「タイトル」の列データ値である「センター」という単語と多く合致して列データ間類似度が高くなっていることがわかる．

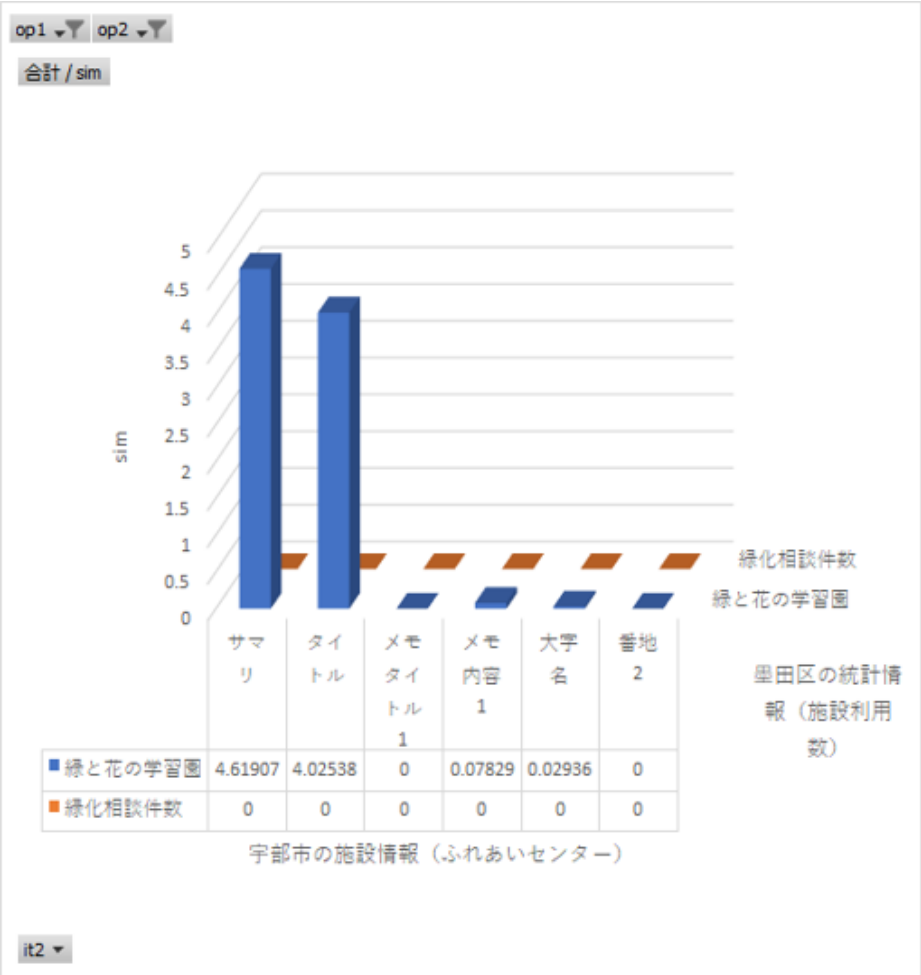


図 15　宇部市の施設情報と墨田区の統計情報

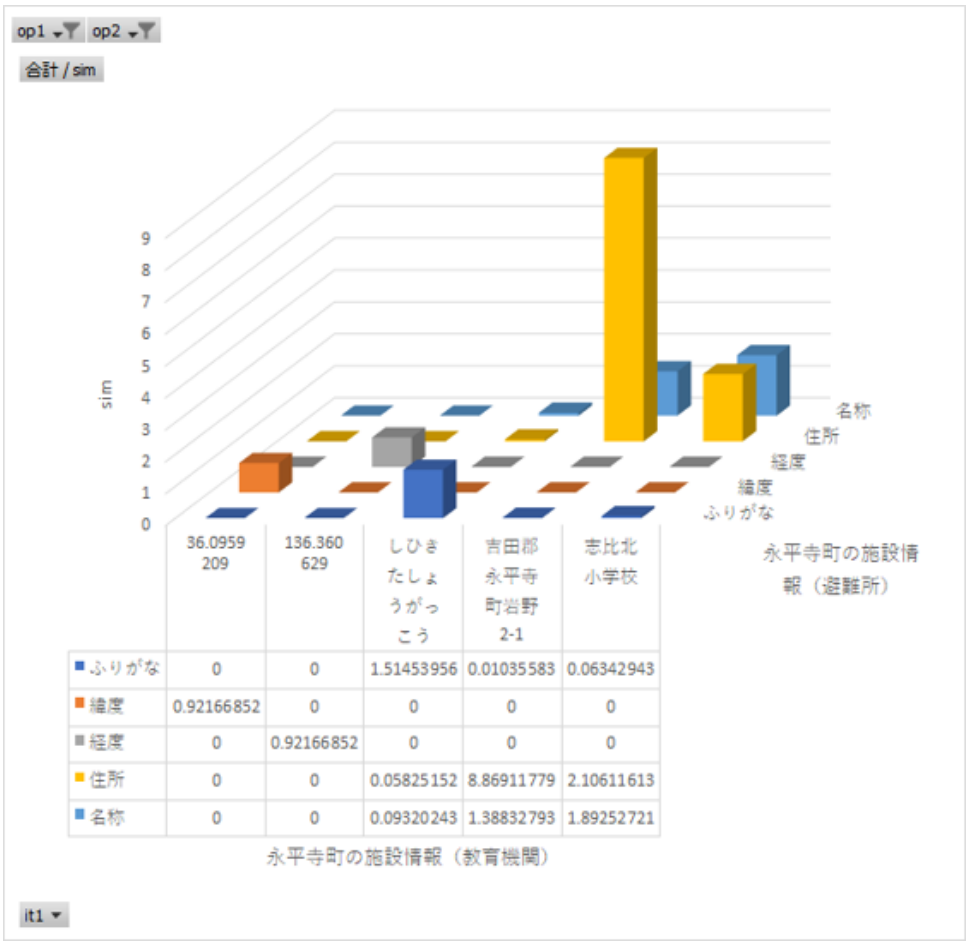
表 6　墨田区の統計情報

	A	B	C	D	E	F	G
1	緑と花の学習園	緑化相談件数	427	423	442	148	180
2	緑と花の学習園	来園者数	10544	12779	11841	10559	9126
3	すみだ環境ふれあい館	来館者数	7476	7085	8264	8002	5610
4	すみだリサイクル活動センター	講習会開催回数	9	9	8	8	8
5	すみだリサイクル活動センター	講習会参加者数	158	156	160	178	136
6	すみだリサイクル活動センター	リサイクルショップ登録者数	5223	5431	5607	5786	5976
7	すみだリサイクル活動センター	リサイクルショップ売買点数	23382	24483	23103	21540	21960
8	すみだリサイクルセンター	来館者数	9451	8965	8372	7971	7383
9	すみだリサイクルセンター	展示品数	740	740	720	740	748
10	すみだリサイクルセンター	申込数	7115	7058	7306	7440	7185

このオープンデータ同士の連携度が高いのは，墨田区の統計情報のオープンデータ作成時に間違った CSV 形式のデータを入力したものが原因である．

4.3.6　連携度の第 6 位（連携度 3.71）

連携度の第 6 位は，永平寺町 1 の施設情報（教育機関）と永平寺町 2 の施設情報（避難所）となった（図 16）．永平寺町 1 は，1 行目に項目名が記載されておらず 1 行目から列データ値になっていて，永平寺町 2 の項目名は名称，住所，経度，緯度，ふりがなである．列データ間類似度を見ると永平寺町 1 の 1 列目，2 列目には緯度，経度の数値となっており，永平寺町 2 の項目名の「緯度」,「経度」と合致している．また永平寺町 1 のふりがなで表示してある学校名が，永平寺町 2 の項目名の「ふりがな」とひらがな表記で合致している．また永平寺 1 の各行の住所表記が，永平寺 2 の項目名「住所」の



	36.0959209	136.360629	しひきたしょうがっこう	吉田郡永平寺町岩野2-1	志比北小学校
ふりがな	0	0	1.51453956	0.01035583	0.06342943
緯度	0.92166852	0	0	0	0
経度	0	0.92166852	0	0	0
住所	0	0	0.05825152	8.86911779	2.10611613
名称	0	0	0.09320243	1.38832793	1.89252721

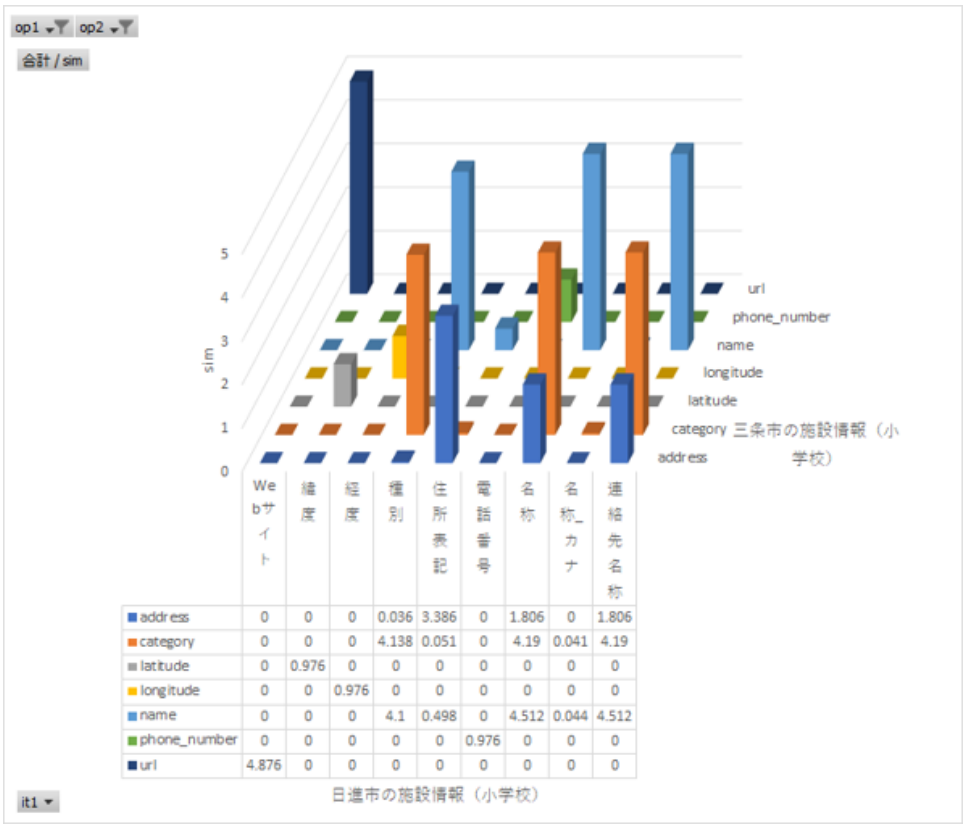
永平寺町の施設情報（教育機関）

図 16　永平寺町 1 の施設情報と永平寺町 2 の施設情報

列データと合致し，高い類似度を示している．

4.3.7　連携度の第 7 位（連携度 3.62）

連携度の第 7 位は，日進市の施設情報（小学校）と三条市の施設情報（小学校）となった（図 17）．日進市の項目名は，連絡先名称，名称-カナ，名称，　電話番号，住所表記, 種別, 経度, 緯度, Web サイトであり, 三条市の項目名は, addres, category, latitude, longitude, name, phone_number, uri の英語表記である．三条市の項目名の「name」と「category」に対して，日進市の項目名の「種別」,「名称」,「連絡先名称」の列データ間類似度が高いのは，それぞれのデータ値に「小学校」「中学校」「高校」や学校名，地名がお互いに含まれているのが要因だと考えられる．日進市の項目名の「Web サイト」と三条市の項目名の「uri」の類似度が高いのは，各小学校のホームページのアドレスが記載されているのが要因である．また日進市の項目名の「緯度」, 経度」に対して，三条市の項目名「latitude」,「longitude」のデータ値は，正規表現により数値の範囲が



	Webサイト	緯度	経度	種別	住所表記	電話番号	名称	名称_カナ	連絡先名称
address	0	0	0	0.036	3.386	0	1.806	0	1.806
category	0	0	0	4.138	0.051	0	4.19	0.041	4.19
latitude	0	0.976	0	0	0	0	0	0	0
longitude	0	0	0.976	0	0	0	0	0	0
name	0	0	0	4.1	0.498	0	4.512	0.044	4.512
phone_number	0	0	0	0	0	0.976	0	0	0
url	4.876	0	0	0	0	0	0	0	0

図 17　日進市の施設情報と三条市の施設情報

合致しているので，項目名が違っても類似度が高くなっている．

4.3.8　連携度の第 1000 位（連携度 3.54）

連携度の第 1000 位は，東久留米市の防災情報（避難所）と千葉市の施設情報(防災情報)となった（図 18）．東久留米市の項目名は，電話番号，中項目，大項目，所在地，施設名，経度，緯度であり，千葉市の項目名は，緯度（世界測地系），経度（世界測地系），施設・場所・イベントの名称，住所，説明文である．東久留米市の項目名の「緯度」，「経度」に対して，千葉市の項目名の「緯度（世界測地系）」，「経度（世界測地系）」は，正規表現により数値の範囲が合致しているので，項目名が違っても類似度が高くなっている．東久留米市の項目名の「大項目」と千葉市の項目名「場所・イベントの名称」，「説明文」との項目間類似度が高いのはお互いに防災，防犯という共通の語彙が含まれているのが要因である．東久留米市の項目名「所在地」と千葉市の「住所」は，同じ内容の語彙が含まれているので，類似度が高い．

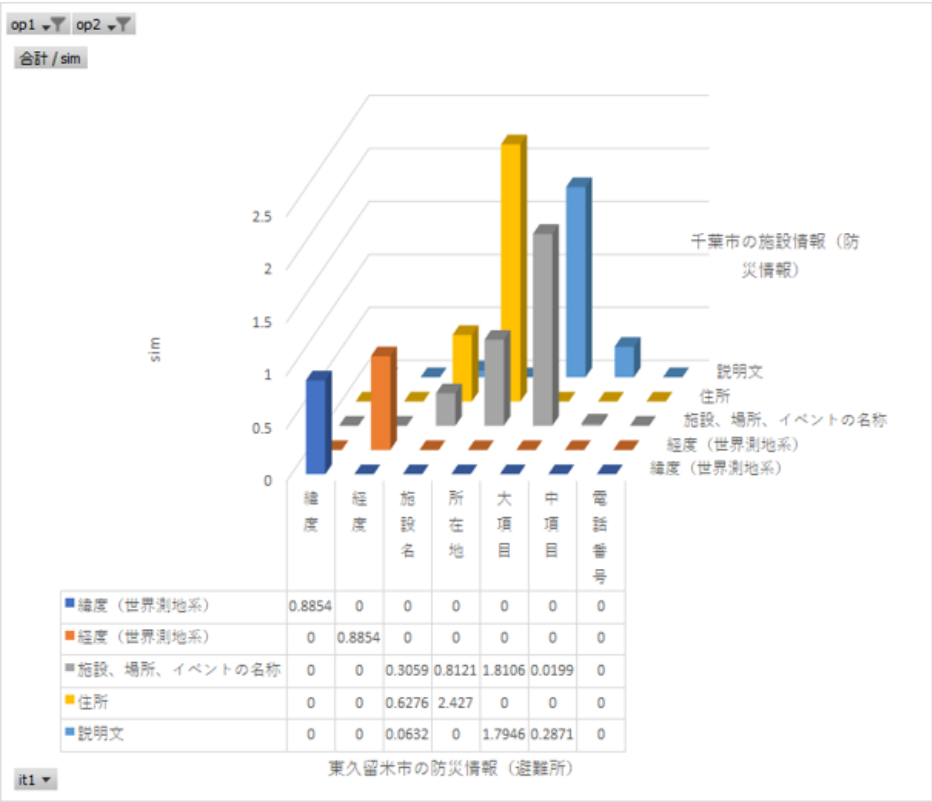


図 18　東久留米市の防災情報と千葉市の施設情報

4.3.9　列データ間類似度を用いた実験のまとめ

列データ間類似度を用いて計算したオープンデータの連携度においては，連携度の高いものについては，片方のオープンデータの一つの列について，もう片方のオープンデータの 1 ないし数個の列に高い類似度があるという特徴が見られる．これは，2 つのオープンデータを連携させるときに，それらの類似度の高い列をキーとして新たなデータを作成することができる有用な特徴であると考えられる．例えば，「緯度」「経度」をお互いのキーとして避難所と AED の場所を合致することができる．

また，項目名が日本語のみならず英語の単語や記号のようなもので，お互いが合致しなくても，列データ値が類似していれば類似度が高い結果が得られる．

一方で，数値のみの列のデータについては，電話番号，緯度・経度等のように正規分布で判読されるもの以外は，類似度を測ることが難しいという問題もある．数値データに関しては，データの特徴をとらえて正規表現で判断する手法が必要となる．

4.4　項目名間類似度による連携度

項目名間類似度も列データ間類似度の計算の同様の 300 のオープンデータ間の各々の項目名間類似度を 15000 次元の軸を用いて連携度の計算を行った．ここでも列データ間類似度と同様に列数が 2 以上 20 以下のファイルの計算を行い，44850 ペアの類似度の結果を得た（表 7）．

表 7　項目名のみを用いたオープンデータ間連携度の上位 20 位まで

順位	オープンデータ1	オープンデータ2	連携度B
1	防府市の行政情報（予算額）	防府市の行政情報（予算額）	7.644626
2	川崎市の施設情報（美容室）	奈良市の施設情報（旅館）	7.34621
3	三重県の行政情報（業種別認可）	三重県の行政情報（業種別認可）	6.829889
4	川崎市の施設情報（美容室）	三重県の行政情報（業種別認可）	6.316383
5	川崎市の施設情報（美容室）	三重県の行政情報（業種別認可）	6.245795
6	川崎市の施設情報（美容室）	奈良市の行政情報（食品店認可）	6.001161
7	さいたま市の統計情報（地区別人口）	さいたま市の統計情報（地区別人口）	5.855432
8	長岡市の行政情報（投票者数）	知多市の行政情報（選挙投票者数内訳）	5.815247
9	川崎市の施設情報（美容室）	宇都宮市の施設情報（クリーニング店）	5.710058
10	奈良市の行政情報（食品店認可）	奈良市の施設情報（旅館）	5.437359
11	三重県の行政情報（業種別認可）	奈良市の施設情報（旅館）	5.324943
12	三重県の行政情報（業種別認可）	奈良市の施設情報（旅館）	5.252509
13	奈良市の施設情報（クリーニング店）	奈良市の施設情報（旅館）	5.171285
14	徳島市の統計情報（産業別雇用者数）	函館市の統計情報（労働者雇用）	5.098786
15	福島県の行政情報（議会情報）	福島県の行政情報（三役）	5.047107
16	千葉市の施設情報（イベント名称）	旭川市の施設情報（介護施設）	4.990488
17	千葉市の施設情報（イベント名称）	豊田市の施設情報（こども園）	4.950444
18	松山市の行政情報（投票者数）	知多市の行政情報（選挙投票者数内訳）	4.945077
19	奈良市の行政情報（食品店認可）	三重県の行政情報（業種別認可）	4.751153
20	川崎市の施設情報（美容室）	奈良市の施設情報（クリーニング店）	4.726888

項目名列間類似度の上位 1 位から 4 位の 1000 位のオープンデータのペアの項目名同士の類似度を調べてピボットテーブルを用いて，項目名列間の類似度をグラフ化した．

4.4.1　連携度の第 1 位（連携度 7.64）

防府市の行政情報（26 年と 27 年度予算額）と防府市の行政情報（27 年と 28 年度予算額）となった（図 19）．同じ自治体の掲載年度だけが違う同じ内容のオープンデータであり，項目名もすべて合致していることから語彙が一緒で数値のみ違うので，項目名間の類似度が高くなっている．

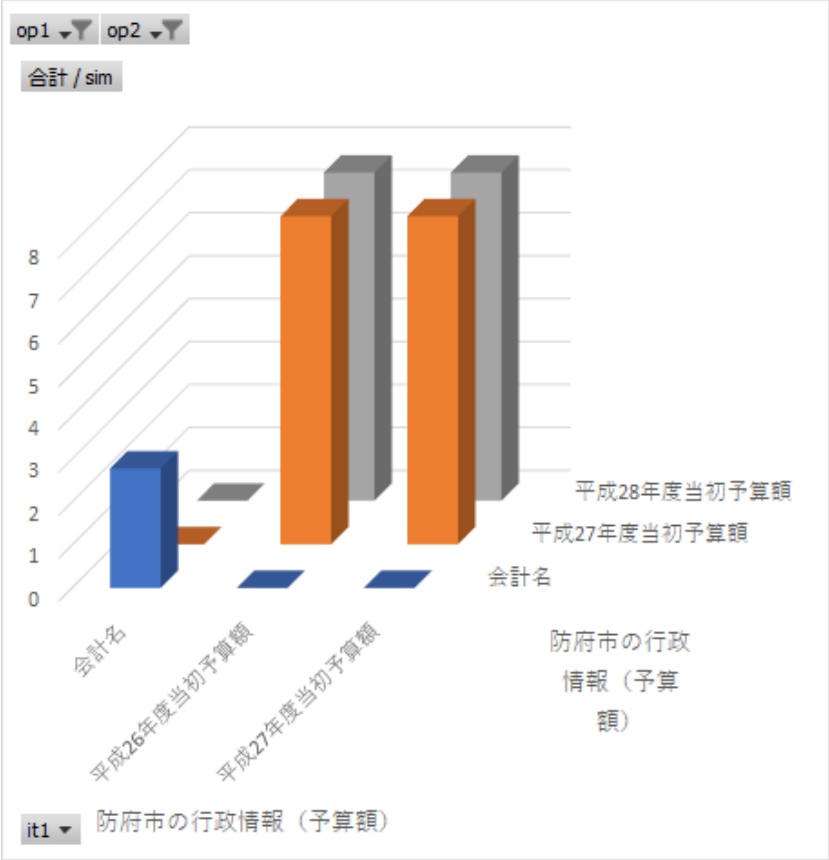


図 19　防府市１の行政情報と防府市２の行政情報

4.4.2　連携度の第２位（連携度 7.35）

川崎市の施設情報（美容室）と奈良市の施設情報（旅館）となった（図 20）．お互いのオープンデータの内容は，行政による施設の許認可のデータである．項目名に共通の単語（営業，所在地，氏名，名，番号など）が多く含まれているので列間類似度が高い結果になっている．項目名が長かったり，同じ単語が多いと項目名間の類似度が高くな

ることがわかる．

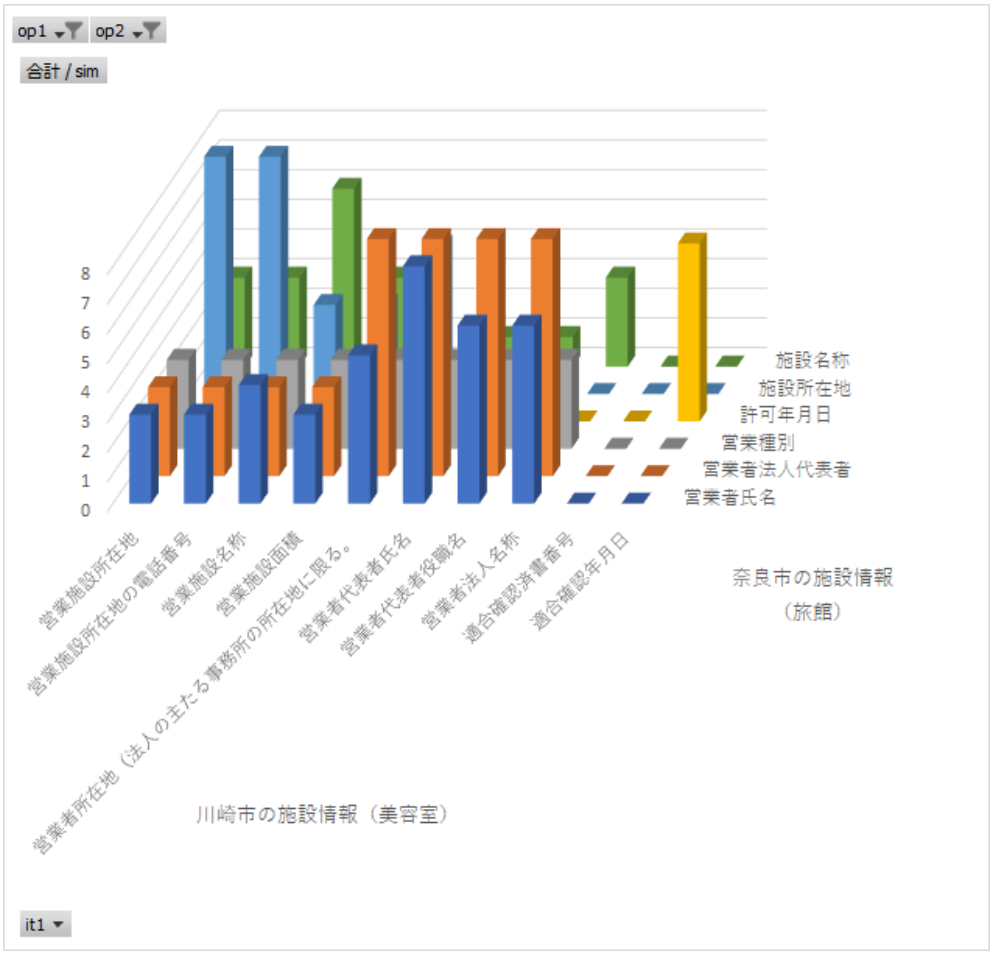


図 20　川崎市の施設情報と奈良市の施設情報

4.4.3　連携度の第 3 位(連携度 6.83)

三重県 1 の行政情報と三重県 2 の行政情報となった（図 21）．三重県 1 は，営業所の許認可の施設情報であり，三重県 2 は営業所の屋号の施設情報で，項目名に営業，氏名，番号，号，所，業，と言う語彙が多く含まれることで項目名間類似度が高くなっている．このように共通の語彙が多く含まれると類似度が高くなる．

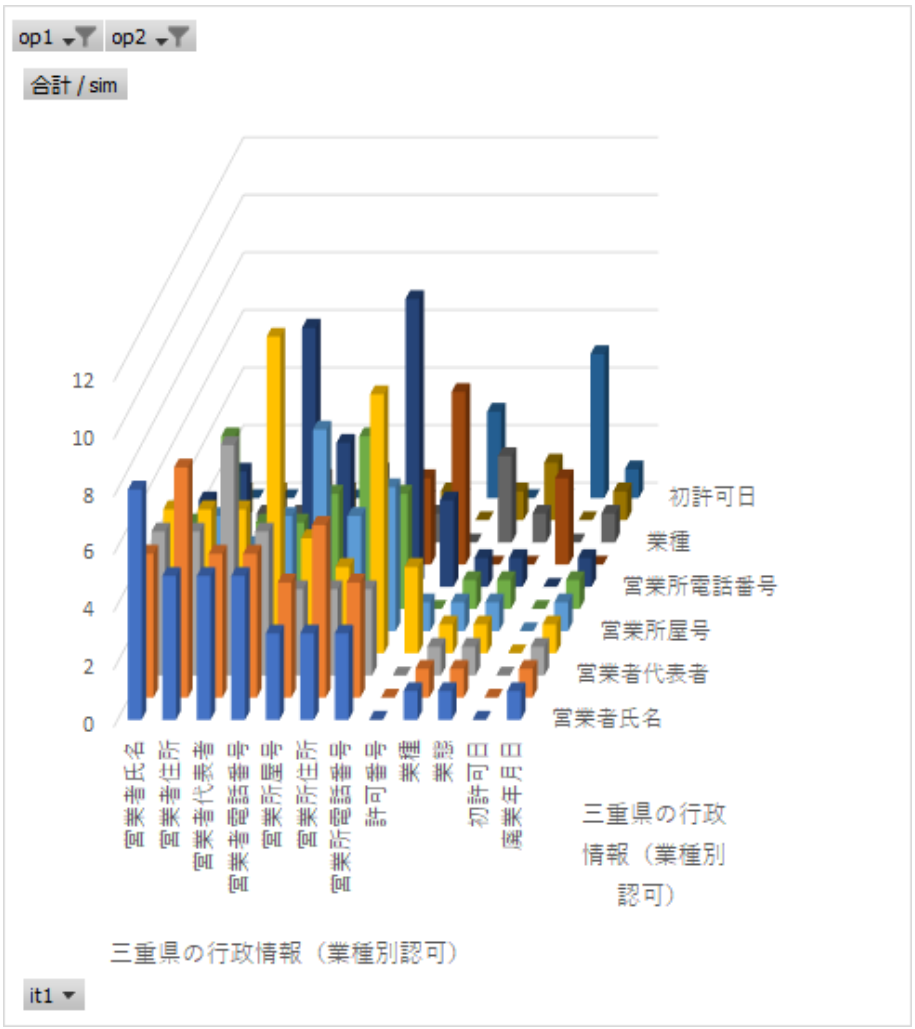


図 21　三重県 1 の行政情報と三重県 2 の行政情報

4.4.4　連携度の第 4 位（連携度 6.32）

川崎市の施設情報（美容室）と三重県の行政情報（業種別認可）となった（図 22）．お互いの項目名に営業，所，番号，号，名，業など共通の語彙が多く含まれるので，項目名間類似度が高くなっている．お互いのデータは，許認可情報として共通している内容でもある．

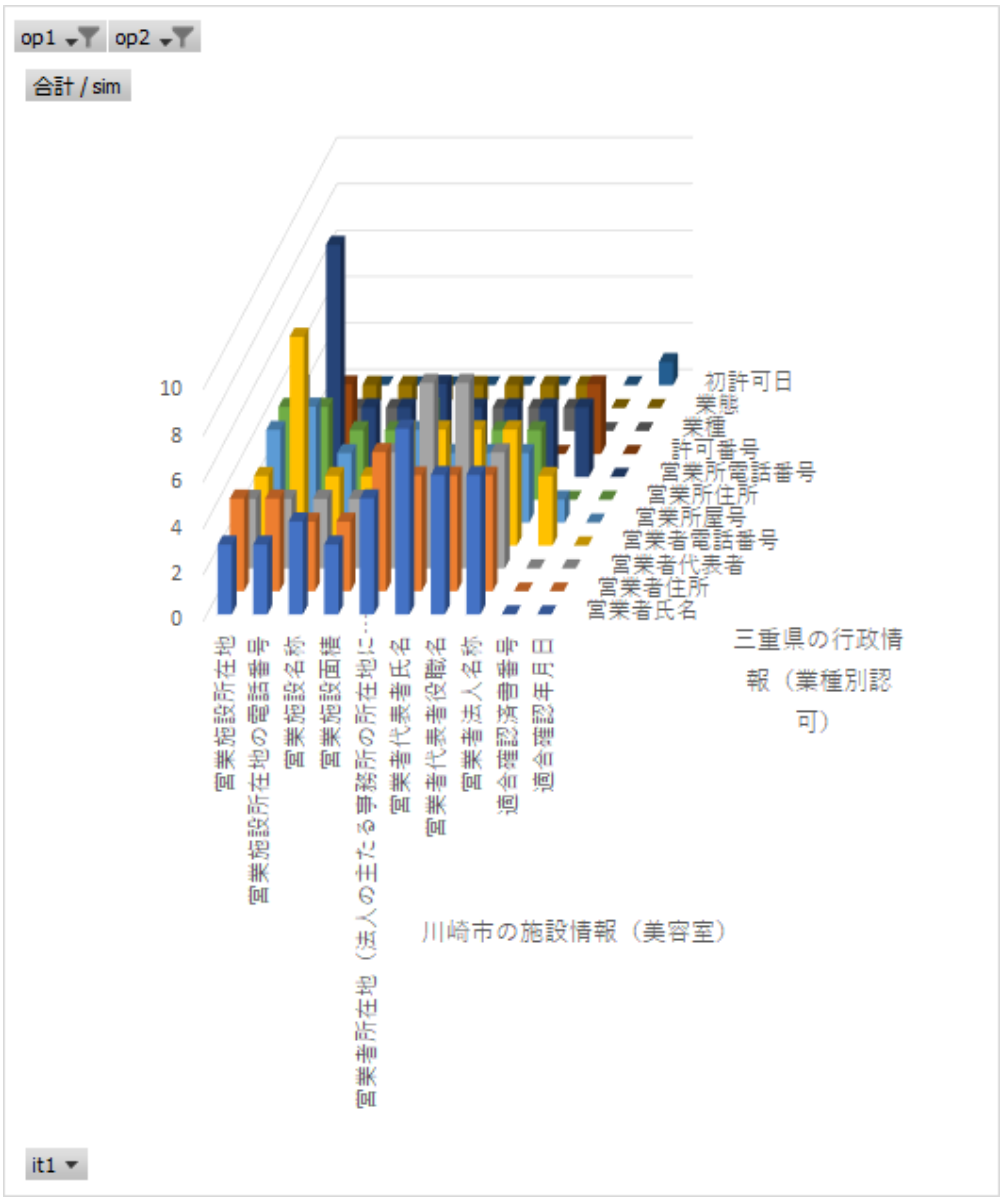


図 22　川崎市の施設情報と三重県の行政情報

4.4.5　連携度の第 1000 位（連携度 2.40）

川崎市の施設情報(美容室)と志摩市の防災情報(避難種別施設)でとなった(図 23).

項目名の各単語意味はお互いに全く違うものであるが, 部分的に同じ単語や漢字が使われているので類似度が高いものがある. お互いのデータの内容としては, 許認可情報と避難所施設という違った内容になっている.

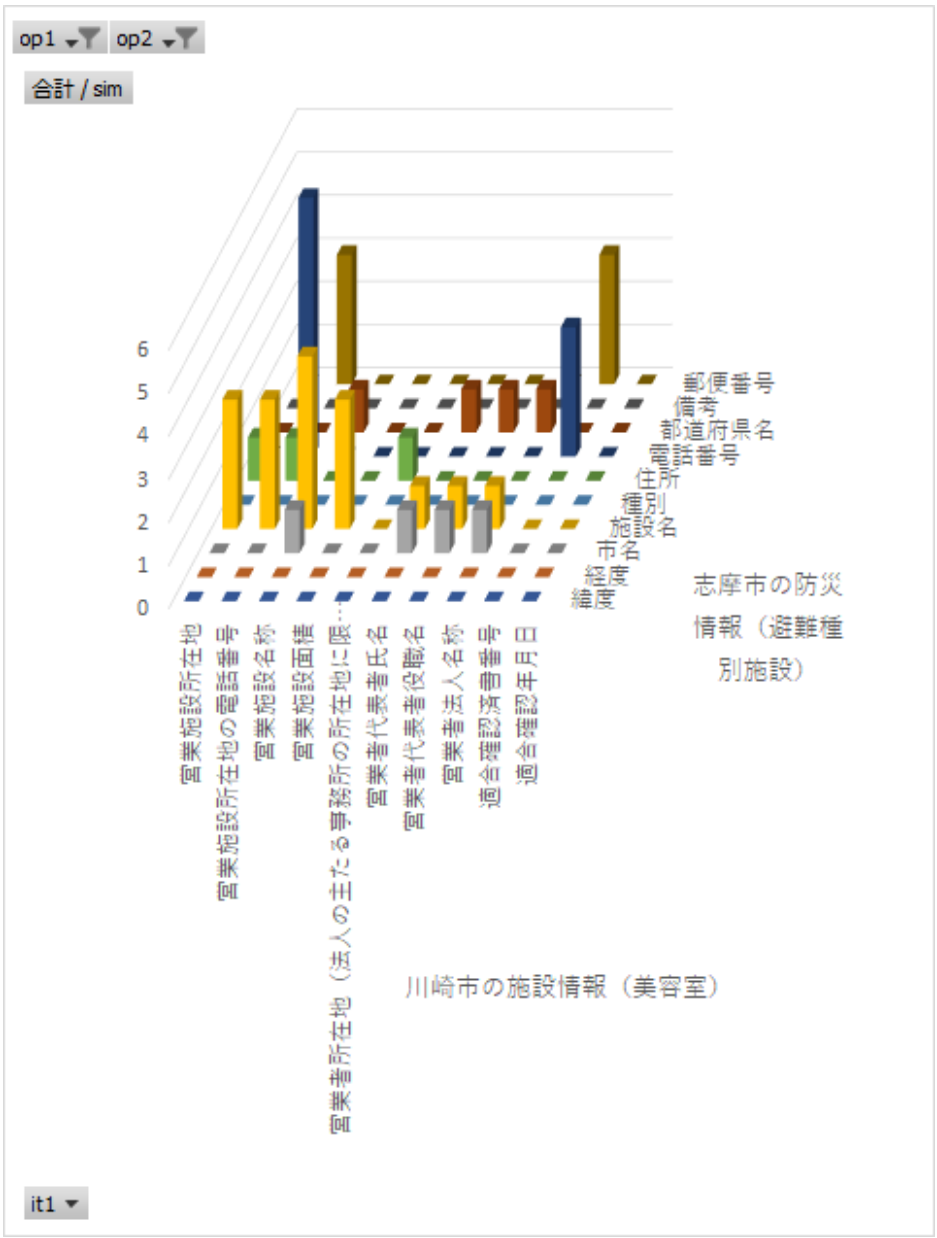


図 23　川崎市の施設情報と志摩市の防災情報

4.4.6　項目名間類似度を用いた実験のまとめ

項目名間類似度を用いて計算した連携度においては，片方のオープンデータの一つの列に対して，もう片方のオープンデータの多く列の類似度が高くなる傾向が見られた．これは，項目名として同じ単語が複数の列に使用されているケースが多くあることが原因である．例えば，連携度第 2 位の川崎市の施設情報（美容室）と奈良市の施設情報（旅館）の場合は，「営業～」という項目名が多く表れた結果，ほとんどの列に対して類似度が高くなっている．このような場合は，2 つのオープンデータを連携させることが難しいと考えられる．

一方，データが数字のみの場合でも，項目名がそのデータの意味を表していれば，正しく連携されるという特徴もある．ただし，現状で公開されているオープンデータの項目名は列の内容を正しく表していないケースも多く，項目名に頼って内容を判断するだけではよい結果は得られなかった．

4.5　考察

列データ間類似度を用いて計算した連携度（以下連携度 I）と，項目名間類似度を用いて計算した連携度（以下連携度 II）にどのような違いがあるかについて考察する．

連携度 I の第 1 位の組み合わせである品川区の防災情報（防災無線）と江戸川区の防災情報（無線設置場所）は，連携度 II では第 748 位となった．連携度 I のグラフを図 8 に，連携度 II のグラフを図 24 に示す．これらを比較すると，緯度・経度についてはいずれも類似度が高い結果となっているが，種別と名称の間の類似度が，連携度 I では高いが連携度 II では 0 になっている．これは，連携度 II では項目名の「種別」と「名称」に間に共通する単語がないため，類似度を算出できていないためである．一方，連携度 I では，「種別」と「名称」の各データの中身を見て，それが類似していることを正しく示すことができている．

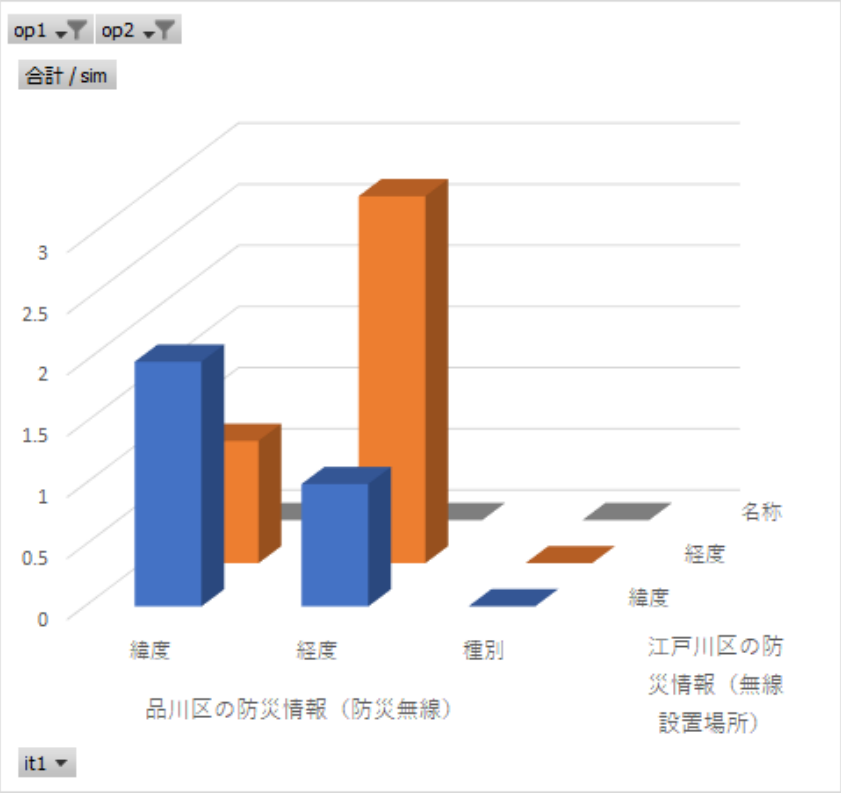


図 24　連携度 II の第 748 位

連携度 I の第 2 位である三条市の施設情報（小学校）ローマ字表記と半田市の統計情報（小学校世帯数）は，連携度 II では第 27088 位となった．この場合の連携度 I のグラフを図 9 に，連携度 II のグラフを図 25 に示す．これらのオープンデータはいずれも小学校に関するデータであるが，片方の項目名は日本語であり，もう片方は英語となっているため，連携度 II の類似度は全く一致せず，すべて 0 になっている．一方，連携度 I の類似度は列データの中身を見て判断しているため，それらが似たデータであることを正しく判定できていることがわかる．

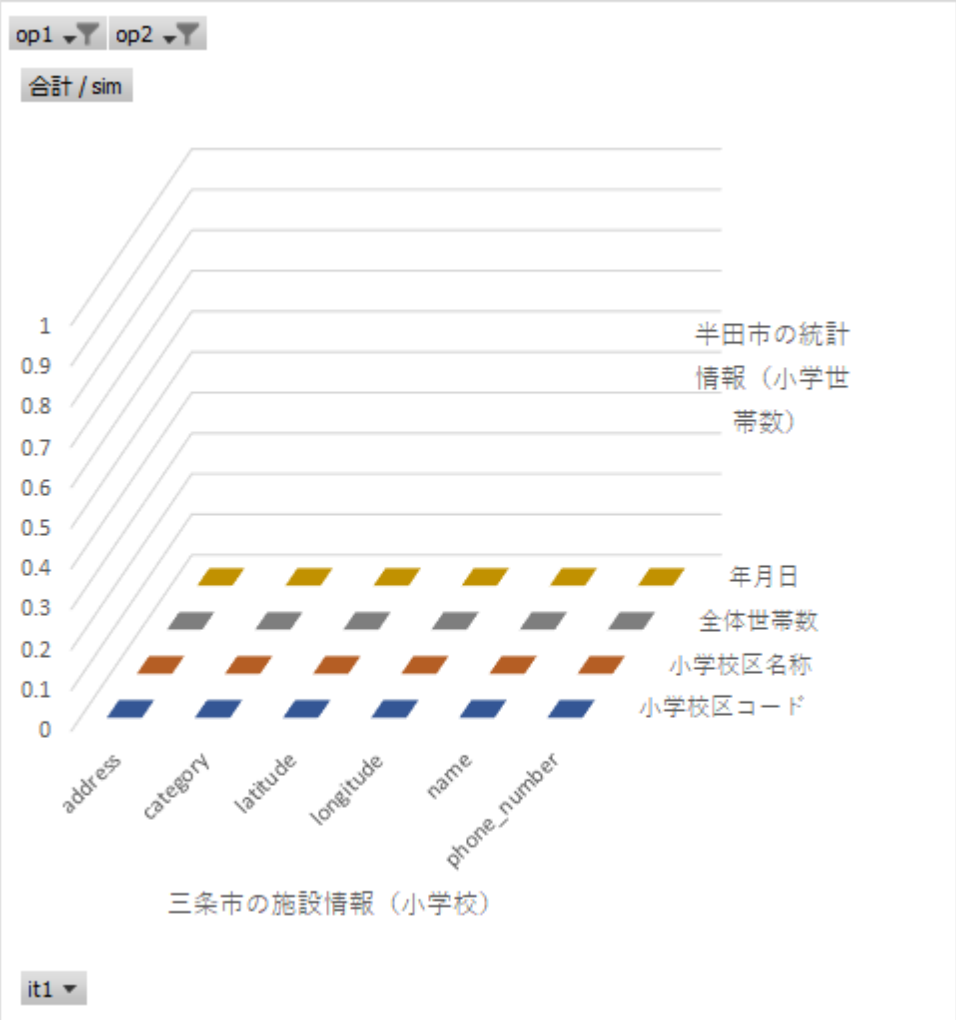


図 25　連携度 II の第 27088 位

このような結果から連携度 II において，各自治体によって項目名の表現，表記の仕方が違うので，連携度が高くても項目名でのデータ連携は難しいと思われる．一方，連携 I においては，項目名は違っても，列データの内容が似通っていればお互いのデータの連携を可能にする列が見られ，データの連携を可能にしやすいことがわかった．

第５章　まとめと今後の課題

最後に，本研究の内容を総括し，今後の課題について述べる．

5.1　まとめ

本研究では，全国の地方公共団体のオープンデータの現状はどのような状況なのか，またどのような活用がなされているのかを最初に調査した．現状としては，徐々に各地方公共団体においてオープンデータのポータルサイトが立ち上がる状況にあるものの，開示されるデータ形式が不揃いであったり，機械判読のできないデータが多いことがわかった．特に国政調査等で収集される統計情報が，PDF や XLS 形式で添付されているサイトも多く，人間の目で見て判断しなければならいオープンデータが数多くみられた．

オープンデータの利活用を行うためには，機械判読が出来て，二次利用が可能でなければならないが，まだそこに至っていなオープンデータの状況である．このことから，現状のオープンデータを活用する手法として，現在のデータ形式が CSV のオープンデータに着目して，CSV ファイルの列データから連携度の高いファイルを抽出する手法を提案した．政府の推進するオープンデータの利活用を行うためには，データ間の連携が重要であり，かつ大量のデータから連携を図るためには，機械判読で連携可能なデータを抽出できることが重要である．

各地方公共団体のオープンデータの CSV ファイルの内容をみると，人間の目で見やすい形式にしてあるものや，項目名と項目値だけでなくメタデータとしてファイルの説明文が１行目にあるもの，さらに１列目に項目名があるものもあった．また各地方公共団体によって同じ意味であっても項目名として異なった単語を用いられていることが多く，統一されていないことがわかった．したがってデータの連携を図るために項目名だけを利用したのでは不十分であり，列データ同士に共通する単語をチェックすること

が必要である．これらの項目名，列データの内容の類似性を調べるために全国の地方公共団体の CSV 形式のオープンデータを収集し，そのデータに含まれる単語を抽出し，それらのデータに含まれる頻度をベクトル化し，類似度を測る述語ベクトル法を考案した．抽出した CSV 形式のオープンデータも列データの値が，数値が多く比較が難しいデータも多くあったが，正規表現等を用いてある程度の数値データも類似度の計算を行った．また列データだけでは，データ同士の連携度を測ることが難しいことがわかった．

本研究ではこれらの課題を解決するために述語ベクトル法を用いて列間類似度を計算し，さらにこれを用いてオープンデータの連携度を算出した．

連携度が大きければ特定の列の類似度が高いものが含まれているが，必ずしも連携がとれるとは限らない．なぜなら現状のオープンデータの CSV ファイルの中には項目名が縦に書いてあるなど，フォーマットが不適切なものも含まれているからである．そこで，連携度の値について以下に考察してみる．

もし同じような内容を表す二つのデータがあった場合，それらの対応する列同士の類似度が高くなり，それ以外の列同士の類似度は低くなる．今，代表的なデータとして列数が 10 である二つのデータを考えると，列の組み合わせの総数は 100 であり，そのうち類似度の高い列の数は 10 となる．このときの二つのデータの連携度は，式（4）より，

$$C(i,j)=\frac{\sum_{k=0}^{9}\sum_{l=0}^{9}G(o_{kl})s_{ikjl}}{\sum_{k=0}^{9}\sum_{l=0}^{9}G(o_{kl})}$$

となる．ここで，正規分布の標準偏差は式（5）より 10.0 となるので，高い類似度を 1.0，低い類似度を 0.0 と仮定すると，

$$C(i,j)=\frac{G(0)+G(1)+\cdots+G(9)+0+\cdots+0}{G(0)+G(1)+\cdots+G(99)}$$

$$=\frac{0.071+0.070+\cdots+0.047}{0.071+0.070+\cdots+3.69\times10^{-23}}$$

$$\approx 0.67$$

となる．したがって，今回の実験においては連携度が 0.67 程度以上が，二つのオープンデータの連携の目安となると考えられる．連携度 0.67 となる二つのオープンデータは 9205 位であった．

図 26 に今回，実験に使用した 300 個のオープンデータの連携度の順位と，人間が二つのデータの中身を見て判断した分類の一致数のグラフを示す．分類は施設，防災，統計など 12 種類を使用した．表 4 にその一部を示す．

このグラフを見ると，連携度による順位と分類との間には相関があり，順位が 10000 前後で一致数の傾きに変化がみられることがわかる．10000 位の前後で分類の一致数が増えていることから，前述した連携度 0.67 程度がデータ連携の目安になるのではないかと考えられる．

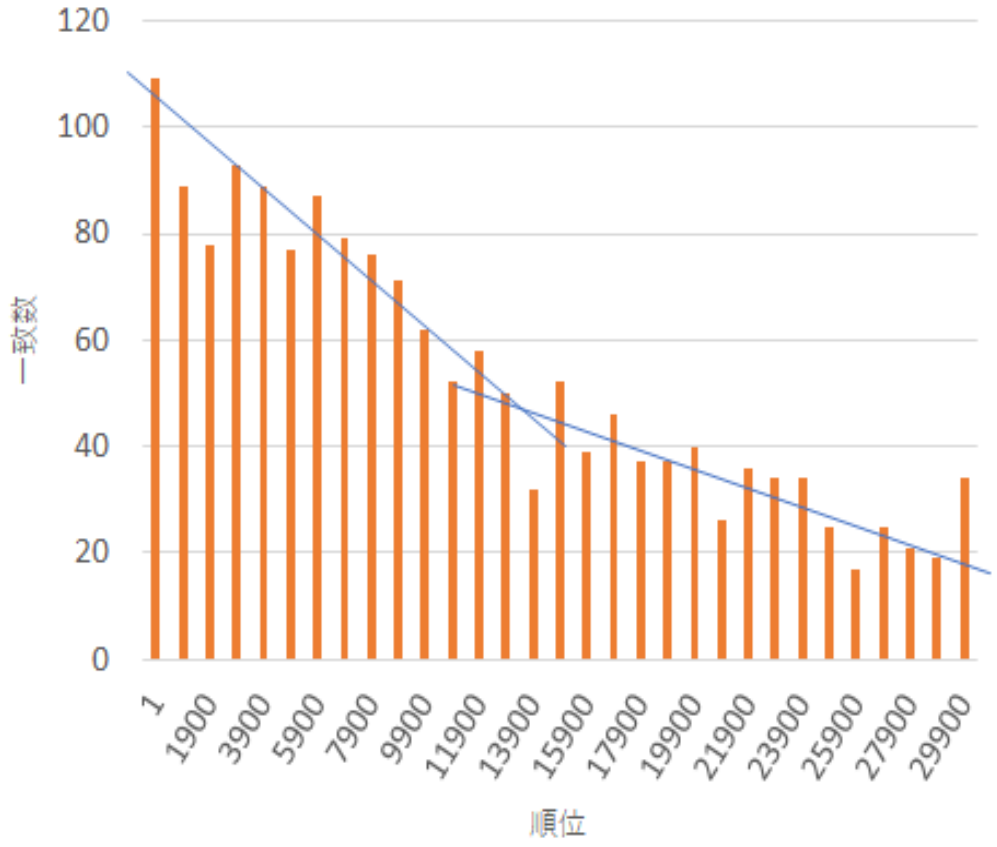


図 26　オープンデータの順位による分類の一致数

連携度Ⅰの結果より，2つのオープンデータ間の列データの類似度の特徴は，項目名が違っても列データの内容が似通っていれば類似度が高くなり，それらの列を共通化することにより，それらのオープンデータの連携が可能になる．一方で連携度Ⅱの結果より，項目名だけの述語ベクトルを用いた場合は，項目名の似通ったもの同士が抽出されてデータの連携にはあまり適していないことがわかった．

2つのオープンデータ間の列データの類似度の特徴は，項目名が違っても列データの内容が似通っていれば類似度が高くなり，それらの列を共通化することにより，それらのオープンデータの連携が可能になる．一方で，項目名だけの述語ベクトルを用いた場合は，項目名の似通ったもの同士が抽出されてデータの連携にはあまり適していないことがわかった．

5.2　今後の課題

述語ベクトル法における項目データの判定に項目判定関数を利用するが，今回の実験では約 15,000 個の軸を利用した．軸が多いと述語ベクトルの計算に時間がかかるようになる．今回は，ランダムに抽出した地方公共団体の 300 ファイルの 44,850 ペアの連携度を計算するのに約 30 時間を要したので，オープンデータ間の連携度を高めつつ，この項目判定関数の軸の精査し，削減を行う必要がある．

今後はオープンデータ間のより良い連携を図るために，類似度や連携度の計算式に用いるパラメータの調整も必要である．．

列間類似度の計算を行う前に項目名や列データの分かち書きをすれば，列間類似度の精度が高まるものと思われる．

今回の実験では，列データが数値のみの場合は，郵便番号，電話番号，緯度・経度のデータしか計算をしなかったが，項目名と数値を照らし合わせて列間の類似度を計算できる仕組みを考えたい．

またオープンデータ間の連携度の高いものについては，そのデータを利用したアプリ

ケーションの作成も視野にいれていきたい.

謝辞

本研究，本論文をまとめるに当たり，長年に渡り，あらゆる面でご指導，ご鞭撻を賜りました主査，鹿児島大学理工学研究科の渕田孝康准教授に感謝いたします．本研究の副査として，数々の有益なご教示，激励を賜りました小野智司准教授，重井徳貴准教授，佐藤公則教授，湯ノ口万友教授に感謝いたします．また，社会人学生として大学院博士課程への研究へ導いていただいた森邦彦教授に感謝いたします．

本研究は，2014 年から 2019 年まで鹿児島大学理工学研究科渕田研究室で行ってまいりました．渕田研究室の私の研究を支えてくれた学生の皆様にも感謝いたします．

社会人として制約のある中で，たくさんの学会に参加させていただき，また渕田准教授には，一緒に参道いただき，いろいろな研究者との出会いや多くの研究について学ばせていただきました．

6 年間の学生と社会人という両立の中で，株式会社フォーエバーの社員には多大な苦労と迷惑をおかけしたと思いますが，何一つ私の学究に対しての不満を述べることもなく，会社を支え、私を見守ってくれたことに感謝いたします．

最後に，本研究は家族の理解と協力なしでは成し得なかった，妻　由起子，子供達　愛，岳，伶央に感謝の意を記す．

本研究は JSPS 科研費 JP16K00421 の助成を受けたものです．

参考文献

[1] 新しい公共のかたちオープンガバメント（20091209）
東京大学公共政策大学院　奥村雄一
https://www.rieti.go.jp/jp/events/bbl/09120901_okumura.pdf(2019)

[2] ビッグデータ（2019）
ウィキペディア（Wikipedia）
https://ja.wikipedia.org/wiki/ビッグデータ#政府（2019）

[3] オープンデータ(2019)
政府 CIO ポータル　内閣官房情報通信（IT）総合戦略室
https://cio.go.jp/policy-opendata(2020)

[4] 東日本大震災(2011)
特集東日本大震災　内閣府防災情報のページ
http://www.bousai.go.jp/kohou/kouhoubousai/h23/63/special_01.html(2019)

[5] 内閣官房情報通信（IT）総合戦略室(2000)
https://www.kantei.go.jp/jp/singi/it2/kanbou_it.html (2019)

[6] 高度情報通信ネットワーク社会推進本部(2001)
内閣官房情報通信（IT）総合戦略室
https://www.kantei.go.jp/jp/singi/it2/(2019)

[7] オープンデータをはじめよう～地方公共団体のための最初の手引書～(2015)
DATA.GO.JP 内閣官房
https://www.data.go.jp/data/dataset/cas_20150305_0002(2019)

[8] 電子行政オープンデータ戦略(2012)
高度情報通信ネットワーク社会推進戦略本部
https://www.kantei.go.jp/jp/singi/it2/pdf/120704_siryou2.pdf (2019)

[9] 二次利用の促進のための府省のデータ公開に関する基本的考え方(2013/6/25)
各府省情報化統括責任者（ＣＩＯ）連絡会議決定(ガイドライン)
www.kantei.go.jp/singi/densi/kettei/gl_honbun(2019)

[10] DATA.GO.JP サイト(2014)
内閣官房情報通信技術(IT)総合戦略室
http://www.data.go.jp/ (2019)

[11] Tim Berners-Lee
https://www.w3.org/People/Berners-Lee/ (2019)

[12] 5 つの星オープンデータ
Open Data のための 5 つ段階（2012） EC FP7 Support Action LOD-Around-The-

Clock （LATC）
https://5stardata.info/en/ (2019)
[13] Resource Description Framework
IT 用語辞典 e-Words
http://e-words.jp/w/RDF.html(2019)
[14] Linked Open Data
ウィキペディア（Wikipedia）
https://ja.wikipedia.org/wiki/Linked_Open_Data(2019)
[15] オープンデータ 2.0(2016)
官民一体となったデータ流通の促進～課題解決のためのオープンデータの「実現」～
高度情報通信ネットワーク社会推進戦略本部
https://www.kantei.go.jp/jp/singi/it2/densi/opendata2/data_sokushin.pdf (2019)
[16] 政府 CIO 補佐官(2019)
内閣官房情報通信（IT）総合戦略室
https://cio.go.jp/hosakan(2019)
[17] オープンデータ伝道師(2019)
内閣官房情報通信（IT）総合戦略室
https://cio.go.jp/policy-opendata(2019)
[18] オープンデータ 100(2019)
内閣官房情報通信（IT）総合戦略室
https://cio.go.jp/opendata100(2019)
[19] 共通語彙基盤サイト IMI(2013)
独立行政法人情報処理推進機構（IPA）
https://imi.go.jp/goi/ (2019)
[20] 文字情報基盤(2010)
独立行政法人情報処理推進機構（IPA）
https://mojikiban.ipa.go.jp/(2019)
[21] オープンデータ研修ポータル(2018)
総務省情報流通行政局地域通信振興課地方情報化推進室
https://www.opendata-training.org/ (2019)
[22] オープンデータ憲章(2013)
外務省
https://www.kantei.go.jp/jp/singi/it2/densi/dai4/sankou8.pdf (2019)
[23] 世界最先端 IT 国家創造宣言(2013)
高度情報通信ネットワーク社会推進戦略本部
https://www.kantei.go.jp/jp/singi/it2/kettei/pdf/20130614/siryou5.pdf (2019)

[24] 杉本直也：GIS による防災情報の発信とオープンデータへの取組　2014 年　日本地理学会発表集　2014s 巻　S1040
[25] 鈴木比奈子，田口仁，佐野浩彬，堀田弥生，臼田裕一郎：「水害地形分類図デジタルアーカイブ」の公開　2016 年　日本地理学会集　2016s 巻　607
[26] 光原弘幸：ICT 利活用型防災教育システムの現状と展望　2018 年　教育システム情報学会誌　2018 年 35 巻 2 号 66-80
[27] 熊本地震(2016)
内閣府防災
http://www.bousai.go.jp/updates/h280414jishin/h28kumamoto/pdf/h280729sanko01.pdf(2019)
[28] 吉賀 夏子,堀 良彰,牛島 清豪： 防災のための Linked Data を用いた河川水位情報の蓄積配信システムの検討　2019 年度　電気・情報関係学会九州支部連合大会発表論文誌　11-2P-03 P591
[29] 菱田隆彰：（公財）電気通信普及財団　研究調査助成報告書 No.31 2016
[30] 浦田真由，荻島和真，中條裕基，遠藤守，安田孝美：地域防災情報における自治体オープンデータ推進の実践　社会情報学会　2018 年 7 巻 1 号 p. 1-17
[31] アーバンデータチャレンジ(UDC)(2013)
一般社団法人社会基盤情報流通推進協議会（AIGID）
https://urbandata-challenge.jp/(2019)
[32] 荻島和馬，福安真奈，浦田真由，遠藤守，安田孝美：観光イベント情報を活用したオープンデータ化の試行と実践　社会情報学　2016 年 4 巻 2 号 1-16
[33] 中村英人，石野洋子：地方公共団体のオープンデータの取組：統計データ公開のあり方の検討　社会情報学　2019 年 8 巻 2 号　79-94
[34] 楽奕平：官民データのオープン化政策の変遷とデータ活用の社会的意義に関する一考察　イノベーション・マネジメント　2019 年 16 巻 157-169
[35] 渡邊雄貴，内田拓馬，秋本泰良，太田幸司，梶克彦，内藤克浩，水野忠則，菱田隆彰：地方自治体の持つオープンデータに関する活用支援手法の提案　情報地理学会研究報告　Vol.2015-MBL-77 No.2 Vol.2015-ITS-No.2
[36] 西田亮介，小野塚亮：なぜ鯖江市は公共データの公開に積極的なのか―協働推進と創造的な行政経営，地域産業構造の変化の視点から　Why Does Sabae City Promote “Open Data”?：A study of “Data City Sabae” 情報社会学会誌　2013 年　Vol.8 No.1 51-63
[37] 地方公共団体データベースサイト
DATA.GO.JP データカタログサイト　内閣官房情報通信技術(IT)総合戦略室
data.go.jp/list-of-database/local-government/(2019)
[38] 広島市オープンデータ　文化施設　区民文化センター等

http://www.city.hiroshima.lg.jp/www/opendatamain/contents/1455760193536/index.html (2019)

[39] シグモイド関数(2019/9/12)
Qitta Python 機械学習
https://qiita.com/SabanoMizuni/items/ab4b73cd9b8e733da11a(2019)

[40] MeCab
京都大学情報学研究科–日本電信電話株式会社コミュニケーション科学基礎研究所
共同研究ユニットプロジェクト
http://taku910.github.io/mecab/(2019)

Appendix 1 実験環境と使用したプログラムについて

ここでは，オープンデータ間連携度を算出するために使用した計算機実験環境と使用した Python のプログラム等について述べる．

A1-1 計算機環境

実験に使用した計算機のスペックは以下のとおりである．

- CPU: Intel® Core™ i7-7700 CPU @ 3.60GHz
- RAM: 32.0GB
- OS: Windows10 Pro ver.1903 ビルド 18362.592
- GPU: GeForce GTX 1080Ti

A1-2 使用言語環境

実験には主に Python 言語を使用し，データ処理にはマイクロソフト Excel を用いた．またスクリプトの実行には bash シェルを用いた．使用したプログラミング言語環境は以下のとおりである．

- Python 3.6.4 :: Anaconda Inc.
- Pip 18.1
- numpy 1.15.4
- scipy 1.0.0
- pandas 0.22.0
- matplotlib 2.1.2
- Microsoft© Excel© for Office 365 MSO 32 ビット
- bash version 4.4.12(3)-release

A1-3 実験スクリプト exp.sh

実験にはリスト A1-1 に示す bash スクリプト exp.sh を使用した．
このスクリプトは, コマンドライン引数で指定した出力対象フォルダ dir に計算結果のすべてのファイルを書き出す. 実験に使用するオープンデータのリストは, スクリプト内の 2 つの変数 list_A と list_B に指定する．また使用する述語ベクトルファイルを変数 pv_file に，述語ベクトル内のオープンデータファイル名のリストを変数 pv_list に指定する．もし実験に使用するオープンデータが pv_list 内にない場合，このスクリプトはプログラム make_pv.py を用いて述語ベクトルを生成し，それを pv_file に追加する．同時にそのファイル名を pv_list に追加する．述語ベクトルの生成には多大な時間を要するので，この処理により同じオープンデータについての述語ベクトルを再度計算する手間を省いている．
変数 C1 と C2 は，述語ベクトルの次元数が少ないデータと多いデータを処理対象から外す

ために使用している．このスクリプトでは C1=2,C2=20 としている．これは，1 列しか要素のない述語ベクトルと 21 列以上の要素を持つ述語ベクトルを，連携度の計算対象から外すことを意味している．この処理は，列数が数百にもおよぶ膨大な列を持つオープンデータがまれに存在し，それによって計算が異常に遅くなることを回避するために設けた．
このスクリプトの処理は大きく以下の 3 つの部分からなる．

1. 述語ベクトルの生成
2. 列間類似度の計算
3. オープンデータ間連携度の計算

1.の述語ベクトルの生成では，make_pv.py というプログラムを使用し，まだ計算されていないオープンデータの述語ベクトルがあれば，それを計算して pv_file と pv_list を更新する．プログラム make_pv.py については A1-4 で詳説する．
2.の列間類似度の計算では，calc_item_sim.py というプログラムを使用し，list_A と list_B に含まれるすべてのオープンデータの組み合わせについて，さらにそれらのオープンデータ内のすべての列の組み合わせに関する列間類似度を計算し，dir/item_sim.csv ファイルに出力する．プログラム calc_item_sim.py については A1-5 で詳説する．
3.のオープンデータ間連携度の計算では，calc_op_sim.py というプログラムを使用し，8 種類のオープンデータ間連携度を計算し，それらを dir/op_sim-[A,B,C,D,E,F,G,H].csv ファイルに出力する．さらに，それらの 8 つのファイルをプログラム combine-op_sim.py を使用して 1 つにまとめた dir/op_sim.csv ファイルを生成する．プログラム calc_op_sim.py については A1-6 で，プログラム combine-op_sim.py については A1-7 で詳説する．
また，実験結果は dir/log.txt に出力される．その抜粋をリスト A1-8 に示す．

A1-4 プログラム make_pv.py

述語ベクトルを生成するための Python プログラムである make_pv.py をリスト A1-2 に示す．また，このプログラムの中で使用している項目判定関数 judge()で使用する 3 つの判定 Judges1,Judges2,Judges3 のうち Judges1 に含める単語を格納したテキストファイル judges1.txt の一部をリスト A1-3 に示す．このファイルは 15,934 個の単語を含むテキストファイルであり，この紙面にすべてを収容することは不可能であるため，一部を示すにとどめる．このプログラムは次のように使用する．

```
$ python make_pv.py pv_file pv_list op_list op_dir
```

ここで，

pv_file：述語ベクトルファイル
pv_list：述語ベクトルを生成済みのオープンデータのリスト
op_list：述語ベクトルを生成する対象のオープンデータのリスト
op_dir：オープンデータが置かれているディレクトリへのパス

である．
このプログラムは, op_list で指定されたすべてのオープンデータのうち pv_list にないデータに対して述語ベクトルを生成し，pv_file に追加するとともに，オープンデータのファイル名を pv_list に追加する．
1 つのオープンデータ op についての述語ベクトルの生成は calcPV(op)関数で行っている．op の各々の列について述語ベクトルを計算する．ただし，行数が少ないオープンデータの場合，1 つの行の影響が大きく出てしまうため，行数による重みづけを行っている．この重みは関数 row_weight(r)で計算している．重みづけの詳細は論文の本文を参照されたい．
1 つの列についての述語ベクトルは関数 judge()で計算する．ここでは 3 つの判定関数 Judges1,Judges2,Judges3 をそれぞれ適用し，その列が各判定関数にどれだけ一致したかを[0,1]の数値に変換して述語ベクトルの列要素とする. 現在の判定関数は Judge1 が 15,934 個，Judges2 が 12 個，Judges3 が 6 個であり，全部で 15,952 次元ベクトルになる．

A1-5 プログラム calc_item_sim.py

列間類似度を計算するための Python プログラムである calc_item_sim.py をリスト A1-4 に示す．このプログラムは次のように使用する．

```
$ python calc_item_sim.py pv_file list_A list_B item_sim C1 C2
```

ここで,

pv_file：述語ベクトルファイル
list_A：列間類似度を求めるオープンデータのリスト A
list_B：列間類似度を求めるオープンデータのリスト B
item_sim：列間類似度を出力するファイル名
C1：最小列数
C2：最大列数

である．
このプログラムは，list_A と list_B の中にあるオープンデータのすべての 2 つの組み合わせ op1,op2 に含まれるすべての列の組み合わせ it1,it2 について列間類似度 sim を計算しファイル item_sim に CSV 形式で出力する．その出力形式は次の通りである．

```
op1,op2,it1,it2,sim
```

例えば，list_A に 3 個，list_B に 5 個のオープンデータがあり，それぞれのオープンデータに 7 列のデータが含まれていたとすると，出力される類似度データ数は$3 \times 5 \times 7 \times 7 = 735$行となる．ただし，list_A と list_B に同じオープンデータが含まれていた場合は処理されない．C1 と C2 は処理するオープンデータの述語ベクトルの列数を制限するために使用されている．詳細は A1-3 を参照.

A1-6 プログラム calc_op_sim.py

オープンデータ間連携度を計算するための Python プログラムである calc_op_sim.py をリスト A1-5 に示す．このプログラムは以下のように使用する．

```
$ python calc_op_sim.py path list_A list_B it_sim_file method op_sim_file
```

ここで,

path：オープンデータフォルダへのパス（未使用）

list_A：オープンデータ間連携度を求めるオープンデータのリスト A

list_B：オープンデータ間連携度を求めるオープンデータのリスト B

it_sim_file：項目間類似度ファイル

method：連携度の計算方法（A,B,C,D,E,F,G,H のいずれか）

op_sim_file：オープンデータ間連携度を出力するファイル名

である．

このプログラムは，list_A と list_B の中にあるオープンデータのすべての 2 つの組み合わせ op1,op2 の間の連携度を計算し，ファイル op_sim_file に CSV 形式で出力する．その出力形式は次の通りである．

```
op1,op2,sim
```

もし list_A と list_B に同じオープンデータが含まれていた場合は処理されない．

計算方法には A,B,C,D,E,F,G,H があり，それぞれ以下のようになっている．

A：重みなし．連携度は単純な類似度の平均値.

B：ガウス分布による重み付き．ガウス分布の標準偏差は 2 つのオープンデータの列数の積の 0.1 倍にしている．

C：減衰関数による重み付き．減衰関数の時定数は 2 つのオープンデータの列数の積の 0.5 倍にしている．

D,E,F,G,H：しきい値による重み付き．それぞれのしきい値は 0.5,0.6,0.7,0.8,0.9 である．

A1-7 プログラム combine-op_sim.py

オープンデータの関連度 A～H の 8 つのファイルをまとめて 1 つのファイルにするための Python プログラムである combine-op_sim.py をリスト A1-6 に示す．このプログラムは以下のように使用する．

```
$ python combine-op_sim.py path
```

ここで,

path：関連度ファイルへのパス

である．

このプログラムは，8 つの連携度ファイルをマージして 1 つの連携度ファイルにまとめる．

Python の Pandas が持つ merge()関数を使用している.

リスト A1-1　スクリプト exp.sh

```
#!/usr/bin/bash
# 2 つのオープンデータリスト list_A,list_B を読み,
# その中に列挙してあるオープンデータ間の類似度を計算する.
# ---
# Usage: bash exp.sh dir
#   dir: 出力対象フォルダ名
# bash によるコマンド例:
#   $ dir=20200105-1; (time bash exp.sh $dir) 2>&1 | tee log.txt ; mv log.txt
$dir ; cp exp.sh $dir

echo -n start...
date

body=$1
python=/cygdrive/c/ProgramData/Anaconda3/python   # for windows
#python=python   # for linux
path=../opendata_l_5-prep2
list_A=Kagoshima_list_AED.txt
list_B=Kagoshima_list_AED.txt
pv_file=pred_vec.csv
pv_list=pred_vec.txt
C1=2                    # C1 以上の列をもつオープンデータのみが処理対象となる
C2=20                   # C2 以下の列をもつオープンデータのみが処理対象となる

# 結果を入れるフォルダ作成
if [ ! -e $body ]; then
  mkdir $body
fi

# オープンデータリストを対象フォルダへコピーする
cp $list_A $body/$list_A
cp $list_B $body/$list_B

# 述語ベクトルの生成
# 述語ベクトルファイル$pv_file を読み込み,
# その中にすでに対象のオープンデータの述語ベクトルが
# あれば，再計算はしない.
# なかった場合は，$pv_file に追加する.
echo -n making predicate vector ...
date
echo for list_A:
$python make_pv.py $pv_file $pv_list $list_A $path
echo for list_B:
$python make_pv.py $pv_file $pv_list $list_B $path

# 列間類似度の計算
echo -n calculating item similarities...
date
$python     calc_item_sim.py     $pv_file     $body/$list_A     $body/$list_B
$body/item_sim.csv $C1 $C2

# オープンデータ間類似度の計算
echo -n calculating opendata similarities...
date
$python calc_op_sim.py $path $body/$list_A $body/$list_B $body/item_sim.csv A
$body/op_sim-A.csv
$python calc_op_sim.py $path $body/$list_A $body/$list_B $body/item_sim.csv B
$body/op_sim-B.csv
$python calc_op_sim.py $path $body/$list_A $body/$list_B $body/item_sim.csv C
$body/op_sim-C.csv
```

```
$python calc_op_sim.py $path $body/$list_A $body/$list_B $body/item_sim.csv D
$body/op_sim-D.csv
$python calc_op_sim.py $path $body/$list_A $body/$list_B $body/item_sim.csv E
$body/op_sim-E.csv
$python calc_op_sim.py $path $body/$list_A $body/$list_B $body/item_sim.csv F
$body/op_sim-F.csv
$python calc_op_sim.py $path $body/$list_A $body/$list_B $body/item_sim.csv G
$body/op_sim-G.csv
$python calc_op_sim.py $path $body/$list_A $body/$list_B $body/item_sim.csv H
$body/op_sim-H.csv

# オープンデータ間類似度の結合
$python combine-op_sim.py $body

echo -n done...
date
```

リスト A1-2 プログラム make_pv.py

```
# -*- coding:utf-8 -*-
# 述語ベクトルの生成
# ---
# 述語ベクトルファイルとオープンデータリストを読み込み,
# 述語ベクトルファイルにないオープンデータについてのみ,
# 述語ベクトルを生成する.
# オープンデータは utf-8-BOM とする.
# 結果を新しい述語ベクトルファイルに出力する.
# 全部が 0 のベクトルは出力しない.
# ---
# judges1 は「文字列を含む」の条件（文字列自体が条件名）
# judges2 は「どれかを含む」の条件（judges2[0]が条件名）
# judges3 は正規表現にマッチするかの条件
# ---
# Usage: python make_pv.py pv_file op_list op_dir
#       pv_file: 述語ベクトルファイル
#       op_list: オープンデータリスト
#       op_dir : オープンデータが格納されているフォルダ
import sys
import re
import os
import math
import pandas as pd
import numpy as np

# 判定
# 単独で含む単語の判定用（ファイル"judges1.txt"から読む）
judges1=[]
# 複数の単語を含む判定用
judges2=[
 ["人名 2",["佐藤","鈴木","高橋","田中","伊藤","渡辺","山本","中村","小林","加藤"]],
 ["地名 2",["札幌","函館","小樽","旭川","室蘭","釧路","帯広","北見","夕張","岩見沢","
網走","留萌","苫小牧","稚内","美唄","芦別","江別","赤平","紋別","士別",
  "名寄","三笠","根室","千歳","滝川","砂川","歌志内","深川","富良野","登別","恵庭","
伊達","北広島","石狩","北斗","当別","新篠津","松前","福島","知内","木古内",
  "七飯","鹿部","森","八雲","長万部","江差","上ノ国","厚沢部","乙部","奥尻","今金","
せたな","島牧","寿都","黒松内","蘭越","ニセコ","真狩","留寿都","喜茂別",
  "京極","倶知安","共和","岩内","泊","神恵内","積丹","古平","仁木","余市","赤井川","
南幌","奈井江","上砂川","由仁","長沼","栗山","月形","浦臼","新十津川","妹背牛",
  "秩父別","雨竜","北竜","沼田","鷹栖","東神楽","当麻","比布","愛別","上川","東川","
美瑛","上富良野","中富良野","南富良野","占冠","和寒","剣淵","下川","美深",
  "音威子府","中川","幌加内","増毛","小平","苫前","羽幌","初山別","遠別","天塩","猿払
","浜頓別","中頓別","枝幸","豊富","礼文","利尻","利尻富士","幌延","美幌",
  "津別","斜里","清里","小清水","訓子府","置戸","佐呂間","遠軽","湧別","滝上","興部
","西興部","雄武","大空","豊浦","壮瞥","白老","厚真","洞爺湖","安平","むかわ",
  "日高","平取","新冠","浦河","様似","えりも","新ひだか","音更","士幌","上士幌","鹿追
","新得","清水","芽室","中札内","更別","大樹","広尾","幕別","池田","豊頃",
  "本別","足寄","陸別","浦幌","釧路","厚岸","浜中","標茶","弟子屈","鶴居","白糠","別
海","中標津","標津","羅臼","青森","弘前","八戸","黒石","五所川原","十和田",
  "三沢","むつ","つがる","平川","平内","今別","蓬田","外ヶ浜","鰺ヶ沢","深浦","西目屋
","藤崎","大鰐","田舎館","板柳","鶴田","中泊","野辺地","七戸","六戸","横浜",
  "東北","六ヶ所","おいらせ","大間","東通","風間浦","佐井","三戸","五戸","田子","南部
","階上","新郷","盛岡","宮古","大船渡","花巻","北上","久慈","遠野","一関",
  "陸前高田","釜石","二戸","八幡平","奥州","滝沢","雫石","葛巻","岩手","紫波","矢巾
","西和賀","金ケ崎","平泉","住田","大槌","山田","岩泉","田野畑","普代","軽米",
  "野田","九戸","洋野","一戸","仙台","石巻","塩竈","気仙沼","白石","名取","角田","多
```

賀城","岩沼","登米","栗原","東松島","大崎","富谷","蔵王","七ヶ宿","大河原",
"村田","柴田","川崎","丸森","亘理","山元","松島","七ヶ浜","利府","大和","大郷","大衡","色麻","加美","涌谷","美里","女川","南三陸","秋田","能代","横手","大館",
"男鹿","湯沢","鹿角","由利本荘","潟上","大仙","北秋田","にかほ","仙北","小坂","上小阿仁","藤里","三種","八峰","五城目","八郎潟","井川","大潟","美郷","羽後","東成瀬",
"山形","米沢","鶴岡","酒田","新庄","寒河江","上山","村山","長井","天童","東根","尾花沢","南陽","山辺","中山","河北","西川","朝日","大江","大石田","金山","最上",
"舟形","真室川","大蔵","鮭川","戸沢","高畠","川西","小国","白鷹","飯豊","三川","庄内","遊佐","福島","会津若松","郡山","いわき","白河","須賀川","喜多方","相馬","二本松",
"田村","南相馬","伊達","本宮","桑折","国見","川俣","大玉","鏡石","天栄","下郷","檜枝岐","只見","南会津","北塩原","西会津","磐梯","猪苗代","会津坂下","湯川","柳津",
"三島","金山","昭和","会津美里","西郷","泉崎","中島","矢吹","棚倉","矢祭","塙","鮫川","石川","玉川","平田","浅川","古殿","三春","小野","広野","楢葉","富岡","川内",
"大熊","双葉","浪江","葛尾","新地","飯舘","水戸","日立","土浦","古河","石岡","結城","龍ケ崎","下妻","常総","常陸太田","高萩","北茨城","笠間","取手","牛久","つくば",
"ひたちなか","鹿嶋","潮来","守谷","常陸大宮","那珂","筑西","坂東","稲敷","かすみがうら","桜川","神栖","行方","鉾田","つくばみらい","小美玉","茨城","大洗","城里","東海",
"大子","美浦","阿見","河内","八千代","五霞","境","利根","宇都宮","足利","栃木","佐野","鹿沼","日光","小山","真岡","大田原","矢板","那須塩原","さくら","那須烏山","下野",
"上三川","益子","茂木","市貝","芳賀","壬生","野木","塩谷","高根沢","那須","那珂川","前橋","高崎","桐生","伊勢崎","太田","沼田","館林","渋川","藤岡","富岡","安中","みどり",
"榛東","吉岡","上野","神流","下仁田","南牧","甘楽","中之条","長野原","嬬恋","草津","高山","東吾妻","片品","川場","昭和","みなかみ","玉村","板倉","明和","千代田","大泉",
"邑楽","さいたま","川越","熊谷","川口","行田","秩父","所沢","飯能","加須","本庄","東松山","春日部","狭山","羽生","鴻巣","深谷","上尾","草加","越谷","蕨","戸田","入間",
"朝霞","志木","和光","新座","桶川","久喜","北本","八潮","富士見","三郷","蓮田","坂戸","幸手","鶴ヶ島","日高","吉川","ふじみ野","白岡","伊奈","三芳","毛呂山","越生","滑川",
"嵐山","小川","川島","吉見","鳩山","ときがわ","横瀬","皆野","長瀞","小鹿野","東秩父","美里","神川","上里","寄居","宮代","杉戸","松伏","千葉","銚子","市川","船橋","館山",
"木更津","松戸","野田","茂原","成田","佐倉","東金","旭","習志野","柏","勝浦","市原","流山","八千代","我孫子","鴨川","鎌ケ谷","君津","富津","浦安","四街道","袖ケ浦","八街",
"印西","白井","富里","南房総","匝瑳","香取","山武","いすみ","大網白里","酒々井","栄","神崎","多古","東庄","九十九里","芝山","横芝光","一宮","睦沢","長生","白子","長柄",
"長南","大多喜","御宿","鋸南","千代田","中央","港","新宿","文京","台東","墨田","江東","品川","目黒","大田","世田谷","渋谷","中野","杉並","豊島","北","荒川","板橋","練馬",
"足立","葛飾","江戸川","八王子","立川","武蔵野","三鷹","青梅","府中","昭島","調布","町田","小金井","小平","日野","東村山","国分寺","国立","福生","狛江","東大和","清瀬",
"東久留米","武蔵村山","多摩","稲城","羽村","あきる野","西東京","瑞穂","日の出","檜原","奥多摩","大島","利島","新島","神津島","三宅","御蔵島","八丈","青ヶ島","小笠原","横浜",
"川崎","相模原","横須賀","平塚","鎌倉","藤沢","小田原","茅ヶ崎","逗子","三浦","秦野","厚木","大和","伊勢原","海老名","座間","南足柄","綾瀬","葉山","寒川","大磯","二宮",
"中井","大井","松田","山北","開成","箱根","真鶴","湯河原","愛川","清川","新潟","長岡","三条","柏崎","新発田","小千谷","加茂","十日町","見附","村上","燕","糸魚川","妙高

",
"五泉","上越","阿賀野","佐渡","魚沼","南魚沼","胎内","聖籠","弥彦","田上","阿賀","出雲崎","湯沢","津南","刈羽","関川","粟島浦","富山","高岡","魚津","氷見","滑川","黒部",
"砺波","小矢部","南砺","射水","舟橋","上市","立山","入善","朝日","金沢","七尾","小松","輪島","珠洲","加賀","羽咋","かほく","白山","能美","野々市","川北","津幡","内灘",
"志賀","宝達志水","中能登","穴水","能登","福井","敦賀","小浜","大野","勝山","鯖江","あわら","越前","坂井","永平寺","池田","南越前","越前","美浜","高浜","おおい","若狭",
"甲府","富士吉田","都留","山梨","大月","韮崎","南アルプス","北杜","甲斐","笛吹","上野原","甲州","中央","市川三郷","早川","身延","南部","富士川","昭和","道志","西桂",
"忍野","山中湖","鳴沢","富士河口湖","小菅","丹波山","長野","松本","上田","岡谷","飯田","諏訪","須坂","小諸","伊那","駒ヶ根","中野","大町","飯山","茅野","塩尻","佐久",
"千曲","東御","安曇野","小海","川上","南牧","南相木","北相木","佐久穂","軽井沢","御代田","立科","青木","長和","下諏訪","富士見","原","辰野","箕輪","飯島","南箕輪","中川",
"宮田","松川","高森","阿南","阿智","平谷","根羽","下條","売木","天龍","泰阜","喬木","豊丘","大鹿","上松","南木曽","木祖","王滝","大桑","木曽","麻績","生坂","山形","朝日",
"筑北","池田","松川","白馬","小谷","坂城","小布施","高山","山ノ内","木島平","野沢温泉","信濃","小川","飯綱","栄","岐阜","大垣","高山","多治見","関","中津川","美濃","瑞浪",
"羽島","恵那","美濃加茂","土岐","各務原","可児","山県","瑞穂","飛騨","本巣","郡上","下呂","海津","岐南","笠松","養老","垂井","関ケ原","神戸","輪之内","安八","揖斐川","大野",
"池田","北方","坂祝","富加","川辺","七宗","八百津","白川","東白川","御嵩","白川","静岡","浜松","沼津","熱海","三島","富士宮","伊東","島田","富士","磐田","焼津","掛川","藤枝",
"御殿場","袋井","下田","裾野","湖西","伊豆","御前崎","菊川","伊豆の国","牧之原","東伊豆","河津","南伊豆","松崎","西伊豆","函南","清水","長泉","小山","吉田","川根本","森",
"名古屋","豊橋","岡崎","一宮","瀬戸","半田","春日井","豊川","津島","碧南","刈谷","豊田","安城","西尾","蒲郡","犬山","常滑","江南","小牧","稲沢","新城","東海","大府","知多",
"知立","尾張旭","高浜","岩倉","豊明","日進","田原","愛西","清須","北名古屋","弥富","みよし","あま","長久手","東郷","豊山","大口","扶桑","大治","蟹江","飛島","阿久比","東浦",
"南知多","美浜","武豊","幸田","設楽","東栄","豊根","津","四日市","伊勢","松阪","桑名","鈴鹿","名張","尾鷲","亀山","鳥羽","熊野","いなべ","志摩","伊賀","木曽岬","東員","菰野",
"朝日","川越","多気","明和","大台","玉城","度会","大紀","南伊勢","紀北","御浜","紀宝","大津","彦根","長浜","近江八幡","草津","守山","栗東","甲賀","野洲","湖南","高島","東近江",
"米原","日野","竜王","愛荘","豊郷","甲良","多賀","京都","福知山","舞鶴","綾部","宇治","宮津","亀岡","城陽","向日","長岡京","八幡","京田辺","京丹後","南丹","木津川","大山崎",
"久御山","井手","宇治田原","笠置","和束","精華","南山城","京丹波","伊根","与謝野","大阪","堺","岸和田","豊中","池田","吹田","泉大津","高槻","貝塚","守口","枚方","茨木","八尾",
"泉佐野","富田林","寝屋川","河内長野","松原","大東","和泉","箕面","柏原","羽曳野","門真","摂津","高石","藤井寺","東大阪","泉南","四條畷","交野","大阪狭山","阪南","島本","豊能",
"能勢","忠岡","熊取","田尻","岬","太子","河南","千早赤阪","神戸","姫路","尼崎","明石","西宮","洲本","芦屋","伊丹","相生","豊岡","加古川","赤穂","西脇","宝塚","三木","高砂",

"川西","小野","三田","加西","篠山","養父","丹波","南あわじ","朝来","淡路","宍粟","加東","たつの","猪名川","多可","稲美","播磨","市川","福崎","神河","太子","上郡","佐用","香美",
"新温泉","奈良","大和高田","大和郡山","天理","橿原","桜井","五條","御所","生駒","香芝","葛城","宇陀","山添","平群","三郷","斑鳩","安堵","川西","三宅","田原本","曽爾","御杖",
"高取","明日香","上牧","王寺","広陵","河合","吉野","大淀","下市","黒滝","天川","野迫川","十津川","下北山","上北山","川上","東吉野","和歌山","海南","橋本","有田","御坊","田辺",
"新宮","紀の川","岩出","紀美野","かつらぎ","九度山","高野","湯浅","広川","有田川","美浜","日高","由良","印南","みなべ","日高川","白浜","上富田","すさみ","那智勝浦","太地",
"古座川","北山","串本","鳥取","米子","倉吉","境港","岩美","若桜","智頭","八頭","三朝","湯梨浜","琴浦","北栄","日吉津","大山","南部","伯耆","日南","日野","江府","松江","浜田",
"出雲","益田","大田","安来","江津","雲南","奥出雲","飯南","川本","美郷","邑南","津和野","吉賀","海士","西ノ島","知夫","隠岐の島","岡山","倉敷","津山","玉野","笠岡","井原",
"総社","高梁","新見","備前","瀬戸内","赤磐","真庭","美作","浅口","和気","早島","里庄","矢掛","新庄","鏡野","勝央","奈義","西粟倉","久米南","美咲","吉備中央","広島","呉","竹原",
"三原","尾道","福山","府中","三次","庄原","大竹","東広島","廿日市","安芸高田","江田島","府中","海田","熊野","坂","安芸太田","北広島","大崎上島","世羅","神石高原","下関","宇部",
"山口","萩","防府","下松","岩国","光","長門","柳井","美祢","周南","山陽小野田","周防大島","和木","上関","田布施","平生","阿武","徳島","鳴門","小松島","阿南","吉野川","阿波",
"美馬","三好","勝浦","上勝","佐那河内","石井","神山","那賀","牟岐","美波","海陽","松茂","北島","藍住","板野","上板","つるぎ","東みよし","高松","丸亀","坂出","善通寺","観音寺",
"さぬき","東かがわ","三豊","土庄","小豆島","三木","直島","宇多津","綾川","琴平","多度津","まんのう","松山","今治","宇和島","八幡浜","新居浜","西条","大洲","伊予","四国中央",
"西予","東温","上島","久万高原","松前","砥部","内子","伊方","松野","鬼北","愛南","高知","室戸","安芸","南国","土佐","須崎","宿毛","土佐清水","四万十","香南","香美","東洋",
"奈半利","田野","安田","北川","馬路","芸西","本山","大豊","土佐","大川","いの","仁淀川","中土佐","佐川","越知","梼原","日高","津野","四万十","大月","三原","黒潮","北九州",
"福岡","大牟田","久留米","直方","飯塚","田川","柳川","八女","筑後","大川","行橋","豊前","中間","小郡","筑紫野","春日","大野城","宗像","太宰府","古賀","福津","うきは","宮若",
"嘉麻","朝倉","みやま","糸島","那珂川","宇美","篠栗","志免","須恵","新宮","久山","粕屋","芦屋","水巻","岡垣","遠賀","小竹","鞍手","桂川","筑前","東峰","大刀洗","大木","広川",
"香春","添田","糸田","川崎","大任","赤","福智","苅田","みやこ","吉富","上毛","築上","佐賀","唐津","鳥栖","多久","伊万里","武雄","鹿島","小城","嬉野","神埼","吉野ヶ里","基山",
"上峰","みやき","玄海","有田","大町","江北","白石","太良","長崎","佐世保","島原","諫早","大村","平戸","松浦","対馬","壱岐","五島","西海","雲仙","南島原","長与","時津","東彼杵",
"川棚","波佐見","小値賀","佐々","新上五島","熊本","八代","人吉","荒尾","水俣","玉名","山鹿","菊池","宇土","上天草","宇城","阿蘇","天草","合志","美里","玉東","南関","長洲",
"和水","大津","菊陽","南小国","小国","産山","高森","西原","南阿蘇","御船","嘉島","

```
益城","甲佐","山都","氷川","芦北","津奈木","錦","多良木","湯前","水上","相良","五木
","山江",
  "球磨","あさぎり","苓北","大分","別府","中津","日田","佐伯","臼杵","津久見","竹田
","豊後高田","杵築","宇佐","豊後大野","由布","国東","姫島","日出","九重","玖珠","宮崎
","都城",
  "延岡","日南","小林","日向","串間","西都","えびの","三股","高原","国富","綾","高鍋
","新富","西米良","木城","川南","都農","門川","諸塚","椎葉","美郷","高千穂","日之影
","五ヶ瀬",
  "鹿児島","鹿屋","枕崎","阿久根","出水","指宿","西之表","垂水","薩摩川内","日置","曽
於","霧島","いちき串木野","南さつま","志布志","奄美","南九州","伊佐","姶良","三島","
十島",
  "さつま","長島","湧水","大崎","東串良","錦江","南大隅","肝付","中種子","南種子","屋
久島","大和","宇検","瀬戸内","龍郷","喜界","徳之島","天城","伊仙","和泊","知名","与論
",
  "那覇","宜野湾","石垣","浦添","名護","糸満","沖縄","豊見城","うるま","宮古島","南城
","国頭","大宜味","東","今帰仁","本部","恩納","宜野座","金武","伊江","読谷","嘉手納
",
  "北谷","北中城","中城","西原","与那原","南風原","渡嘉敷","座間味","粟国","渡名喜","
南大東","北大東","伊平屋","伊是名","久米島","八重瀬","多良間","竹富","与那国"]],
  ["診療科目 2",["内科","循環器科","呼吸器科","腎臓科","内分泌科","リウマチ科","アレル
ギー科","血液内科","神経内科","心療内科",
              "感染症科","腫瘍科","外科","一般外科","胸部外科","乳腺外科","甲状腺外科
","小児科","肛門科","整形外科","形成外科",
              "脳神経外科","小児外科","産婦人科","皮膚科","泌尿器科","眼科","耳鼻咽喉
科","リハビリテーション科","放射線科","精神科",
              "麻酔科","臨床検査科","病理診断科","歯科","小児歯科","矯正歯科","歯科口
腔外科"]],
  ["教育文化施設 2",["講堂","ホール","センター","動物園","図書","体育館","球場","自然の
家","学校","神社","寺"]],
  ["福祉医療施設 2",["保育園","老人ホーム","総合病院","病院","保健"]],
  ["産業施設 2",["ビル","観測","気象","旅館","ホテル","デパート","量販店","娯楽","喫茶
","食堂","放送局","空港","流通センター","養殖場"]],
  ["行政施設 2",["裁判所","大使館","郵便","税務","警察","交番","消防","自衛隊","浄水
","下水処理","清掃","リサイクル","火葬","刑務","拘置"]],
  ["web 関連 2",["http","jp","com","@","www"]],
  ["データ関連 2",["pdf","csv","png","html","bmp","jpeg"]],
  ["                                                                    月
2",["Jan","Feb","Mar","Apr","May","Jun","Jul","Aug","Sep","Oct","Nov","Dec"]
],
  ["文章 2",[". ",", ","です","ます"]],
  ["災害 2",["暴風","竜巻","豪雨","豪雪","洪水","崖崩れ","土石流","高潮","地震","津波
","噴火","地滑り"]]
 ]
# 正規表現にマッチするかどうかの判定用
judges3=[
  ["7 桁番号 3","(¥d{3}|[０－９]{3})(-|‐)(¥d{4}|[０－９]{4})"],
  ["携帯電話 3","(0[7|8|9]0|０[７－９]０)(-|‐)(¥d{4}|[０－９]{4})(-|‐)(¥d{4}|[０
－９]{4})"],
  ["電話・FAX 番号 3","(¥d{3}|[０－９]{3})(-|‐)(¥d{3}|[０－９]{3})(-|‐)(¥d{4}|[０
－９]{4})"],
  ["固定電話番号 3","(0¥d{1,4}|０[０－９]{1,4})(-|‐)(¥d{1,4}|[０－９]{1,4})(-
|‐)(¥d{1,4}|[０－９]{1,4})|^(¥d{2}|[０－９]{2})(-|‐)(¥d{4}|[０－９]{4})+$"],
  ["経度 3","^(1(2[2-9]|[34][0-9]|5[0-3]))¥..*$"],          # 日本の経度は 122.5 -
153.6
  ["緯度 3","^([23][0-9]|4[0-5])¥..*$"],                    # 日本の緯度は 20.25 -
45.33
```

```
#    ["時刻 3","([0-9]|[０-９])(:|：)([0-5][0-9]|1[0-9])|([０-５][０-９]|１[０-９])(:|：)([0-5][0-9]|[０-５][０-９])|(2[0-4]|２[０－４])(:|：)([0-5][0-9]||[０-５][０-９])|([0-9]|[０-９])時([0-5][0-9]|[０-５][０-９])分|(1[0-9]|１[０-９])時([0-5][0-9]|[０-５][０-９])分|(2[0-4]|２[０－４])時([0-5][0-9]|[０-５][０-９])分"],
#  ["年月日(元号表記)3","¥A[H|S][0-9][0-9][.|/]¥d{1,2}[.|/]¥d{1,2}"],
#  ["年月日(西暦表示)3","¥A[1|2]¥d{3}[.|/]¥d{1,2}[.|/]¥d{1,2}"],
#  ["台数 3","¥d+台"], #追加
#  ["人数 3","¥d+(人|名)"], #追加
#  ["金額 3","¥d+円"], #追加
#  ["半角数字列のみ 3","^¥d+$"],
#  ["半角英文字列のみ 3","^[a-zA-Z]+$"],
#  ["全角数字列のみ 3","^[０-９]+$"],
#  ["全角英文字列のみ 3","^[ａ-ｚＡ-Ｚ]+$"],
#  ["ひらがなのみ 3","^[ぁ-ん][ぁ-ん一・＝]+$"],
#  ["カタカナのみ 3","^[ァ-ヴ][ァ-ヴ一・＝]+$"],
#  ["半角ｶﾀｶﾅのみ 3","^[ｦ-ﾟ][ｦ-ﾟ-]+$"],
#  ["有無 3","^有$|^無$|^有り$|^無し$"],
#  ["可能 3","^可$|^可能$|^不可$|^不可能$"],
#  ["○×3","^o$|^×$"],
#  ["時間 3","¥d+時|¥d+分|¥d+秒"],
#  ["曜日 3","(月|火|水|木|金|土|日)曜"],
#  ["年齢 3","¥d+歳"],
#  ["月日 3","([1-9]|1[0-2])月([1-9]|1[0-9]|2[0-9|3[0-1])日"],
#  ["年号 3","^(明治|大正|昭和|平成)(¥d+|[一二三四五六七八九十]+)年"]
 ]

## 条件判定
## items: 判定したい列のデータが入っているリスト
def judge(items):
  # 全部数値の列データは処理しない
  #all_num=True
  #for i in items:
  #  if type(i)==str and i!="":
  #    all_num=False
  #if all_num:
  #  return None
  # 空リストを生成する
  pv=[]
  # judges1 の述語ベクトルの生成
  for j in range(jn1):
    s=0.0;
    for it in items:
      if   type(it)==int   or   type(it)==float   or   type(it)==np.int64   or type(it)==np.float64 or type(it)==bool: continue
      if it.find(judges1[j])!=-1:
        s+=1
    pv.append(s/len(items))
  # judges2 の述語ベクトルの生成
  for j in range(jn2):
    s=0.0
    for it in items:
      if   type(it)==int   or   type(it)==float   or   type(it)==np.int64   or type(it)==np.float64 or type(it)==bool: continue
      for js in judges2[j][1]:
        if it.find(js)!=-1:
          s+=1
          break
    pv.append(s/len(items))
```

```
  # judges3 の述語ベクトルの生成
  for j in range(jn3):
    s=0.0
    for it in items:
      if re.search(judges3[j][1],str(it)):
        s+=1
    pv.append(s/len(items))
  return pv

## 行数に対する重み
a=0.2
def rows_weight(r):
  return 2/(1+math.exp(-a*r))-1

## オープンデータ op の述語ベクトルを計算する
def calcPV(op):
  # オープンデータの読み込み
  filename=(op_dir+"/"+op).strip()
  data=pd.read_csv(filename,encoding="utf-8-sig")  # UTF-8 BOM付
  data=data.fillna("")
  rows=len(data)              # 行数
  names=data.columns
  items=data.values.T          # 列を行に転置
  cn=len(names)               # 項目名数＝列数
  # 述語ベクトルの作成
  # 項目名に改行が含まれる場合があるので，replace()で改行を消している．
  pv=[0 for x in range(cn)]
  for i in range(cn):
    # ↓項目名も判定して述語ベクトルを作成する場合
    #a = np.insert(items[i],0,names[i])
    #pv=judge(a)
    #print(pv)
    pv=judge(items[i])
    if pv!=None and any(pv):
      global pv_data
      pv=np.array(pv)
      w=rows_weight(rows)
      pv*=w
      it=names[i].replace("¥n","")
      it=it.replace("¥x0a","")
      it=it.replace("¥x0d","")
      se=pd.Series([op,it]+pv.tolist(),index=pv_data.columns)
      pv_data=pv_data.append(se,ignore_index=True)

### メイン ###

# 引数処理
pv_file=sys.argv[1]
pv_list_file=sys.argv[2]
op_list=sys.argv[3]
op_dir=sys.argv[4]

# Judges1 の読み込み
with open("judges1.txt","r",encoding="utf-8-sig") as f:
  judges1=f.readlines()
  judges1=list(map(lambda x:x.strip(),judges1))

# 各判定関数の個数を得る
jn1=len(judges1)
jn2=len(judges2)
jn3=len(judges3)
```

```
# 述語ベクトルリストの読み込み
if os.path.exists(pv_list_file):
  with open(pv_list_file,"r",encoding="utf-8-sig") as f:
    pv_list=f.readlines()
    pv_list=list(map(lambda x:x.strip(),pv_list))
else:
  pv_list=[]

# 述語ベクトルファイルの読み込み
# もし無い場合は，項目名だけの述語ベクトル CSV ファイルを作成して読み込む.
if not os.path.exists(pv_file):
  with open(pv_file,mode="w",encoding="utf-8-sig") as f:
    f.write("op,it,"+
            ",".join(judges1)+","+
            ",".join([j[0] for j in judges2])+","+
            ",".join([j[0] for j in judges3]))
print("reading PV-Vector file. It often takes long time...",flush=True)
pv_data=pd.read_csv(pv_file,encoding="utf-8-sig")
print("Done",flush=True)

# オープンデータリストの読み込み
with open(op_list,"r",encoding="utf-8-sig") as f:
  op_list=f.readlines()
  op_list=list(map(lambda x:x.strip(),op_list))

# 全オープンデータの述語ベクトルの計算
# ただし，すでにあるオープンデータについては何もしない.
flag=False
for op in op_list:
  if op not in list(pv_list):
    print("  processing...",op,flush=True)
    pv_list.append(op)
    try:
      calcPV(op)
    except AttributeError as e:
      print("Error! : ",e)
    flag=True
  else:
    print("  skipped...",op,flush=True)
  print("flag=",flag)

# 新しい述語ベクトルがあれば述語ベクトルを書き出す
print("flag=",flag)
if flag:
  print("writing  PV_Vectors  to  the  file.  It  often  takes  long
time...",flush=True)
  pv_data.to_csv(pv_file,encoding="utf-8-sig",index=False)
  with open(pv_list_file,"w",encoding="utf-8-sig") as f:
    f.write("¥n".join(pv_list))
  print("Done",flush=True)
```

リスト A1-3　項目判定関数用項目データ Judges1.txt（抜粋）

髙野 髙木 髙島 髙田 髙村 髙松 髙山 髙原 髙見 髙橋 髙丸 髙岡 髙井 髙 﨑幸 﨑 塚区 塚 熙書 鼾 凧池 黎明 鶺鴒 鷄頭 鶚 鵯 鶫 鵙第 鷲 鶯 鴉 鳰 鰡 鰆 鯒 鮓 鬚 騏 饅頭 餡 餃子 颯子 颪 顔 頸城 頸 頌徳 霰 雉野 雉山 雉 隠 閘門 鍼灸 鈑金 邊 轆轤 輻射熱 躾 跛 贅沢 賤 諌見 蟷螂 蟲 蓁 蟋蟀 螢 蠅 蝙蝠 蝸牛 蜩 蜻蛉 蜆 蛛 蚯蚓 蘆 藝 藏 蕾 薔薇 薊 葬 蓼 萬 萍 菫 菴 莉莎 莟 茗荷 茗 茉莢 苺 芒 艘 膾 隋 肛門 聲 聊斎 翡翠 罠 繪 緞帳 綺麗 絲瓜 絆 糯 糀 籘 籔下 籔 簀 築 簣 籠 篋印 箒 筴 筧 筵 笥 竝 稲妻 稲 禮 齋 祠 礫 砒素 盡 癸 瘤 甕 珈琲 獏 狢 焼 炬燵 瀝青 濾過 濱 澁伝 灌水 渣 湮滅 涵養 涅槃会 涅槃 浚渫 沐浴 氣 殘 歸 欅 櫻 櫃 檻 樅 樋 楡 橡 椹 楮 椚 梵鐘 梛 梟 檜 枡 杼 杤 杣山 杣 朧夜 朧月 朧 晝 旱 拉致 拂 戉 戈 戀 徘徊 并 幟 帷子 巖 嶌郷 嵜 屶 屏風 寐 子孑 娵田 梦 壽 壼出 壺 址 圦裏 圦樋 圦添 圦池 圦 圓 國 囃子 嘔吐 喘息 喀痰 哺乳 咄 曼荼羅 曼珠沙華 參 勁文 剪定 凰社 凩 處 冑 兒 兀 偕楽 會田 會 俟 儘 來 佛 亞以 井 腕時計 腕 湾 椀 蕨 藁 鰐口 鰐 鷲 枠 惑星 脇 賄 話題 話し合い 話 和洋 和本 和風 和装 和人 和食 和尚 和室 和式 和字 和紙 和合 和敬 和牛 和気 和楽 和菓子 和歌 和 倭文 倭 論語 論 録音

（・・・中略・・・）

飴 虻 宛名 宛 扱い 圧力 圧縮 圧 梓 芦 葦 旭 悪魔 悪性 悪臭 悪化 悪く 悪い 悪 茜 葵 逢瀬 逢 挨拶 愛着 愛蔵 愛情 愛称 愛児 愛好 愛護 愛犬 愛育 愛 阿弥陀如来 阿弥陀 阿弥 亜熱帯 亜鉛 ワールド ワープロ ワード ワークブック ワークス ワークショップ ワーク わ ろ過 ロング ロマン ロボット ロボッツ ロフティング ロビー ろば ロバ ロツダ ロッダ ロッジ ロック ロッキング ロッカー ロジャー・デュボアザン ロジスティクス ロケット ログハウス ロクザ ロク ローン ロール ローリング ローラースケート ローラー ローラ ローヤル ロープ ロード ロータリー ロータリ ローソンストア ロース ロー ろうそく ろう ロウ ロイヤルパレスビル ロイヤル ろ ロ レンチ レンタル レンタサイクル レンタカー レンズ レンジ レンサ れんが レンガ れん レン レモン レポート レポレベル レフ レトルト レディス レディース レディ

（・・・中略・・・）

アイスクリーム アイス あいさつ アイクラフト あいあい あい アイ アールイーナンバー アール アーム アーバンホーム アーバンビル アーバンハイム アーバンエナジー アートチャイルドケア アート・ポスタ アート アーテック アーティスト アーチ アース アーケード アークフーズ アーク ア ZERO YYYY ｙｏｕ YMCA Yes X線 xxURLxx xls www ｗｔ WSW WOMEN WNW width Wi WHO WEST wed Web Ｗｅｂ WC WAVE WAKO W vol Ver V USEN URL URL URUPS UP UFJ Tシャツ Tシャツ TV TV TUE TSP TOWN total top TOHO TL TLTK TIS thu The terminal TEL TEL target t

（・・・中略・・・）

FKD Feb FDC FC FB FAX FAX Factory FACE FA F Express Excel EV ETC ETC ESE English ENE Elementary edu eBook ｅａｇｅｒｉｓｅ DVD DVD DV drug ｄｒｕｇ DR Dr dpi DO DM DK Dining den Dec Dear DE Day DATE data DAISEI d CVM CU csv CS CROWN COMPANY Community COFFEE COD CoCo CO CO CMS cm ｃｍ Club CLUB class CL city CHOZAIYAKKYOKU CHOME ｃｈｏｉｃｅ Children CH Ceramiales Center CD CCW CCBY Care Cafe ｃａｆｅ c BS BRT BOX BOX BOP books ｂｏｏｋｓ Book ｂｏｏｋ Ｂｌｕｅ blank BL Big Best Berry Beauty Be BCG BCG BASE BAR Ｂａｋｅｒｙ Aug au ATM ATM AT ASA Apr AOKI ａｎｄ ANA amp AM ALSOK All AED AED aBFsOEAK aBFNbnClUAJ ABC x

リスト A1-4　プログラム calc_item_sim.py

```
# -*- coding: utf-8 -*-
# 述語ベクトルファイルと 2 つのオープンデータのリスト A,B を読み込み,
# リスト A とリスト B のオープンデータのすべての列間の類似度を計算し,
# 標準出力に出力する.
# ---
# python calc_item_sim.py pv_file list_A list_B item_sim C1 C2
#   pv_file : 述語ベクトルファイル
#   list_A  : オープンデータリスト A
#   list_B  : オープンデータリスト B
#   op_dir  : オープンデータへのパス
#   item_sim: 列間類似度の出力ファイル名
#   C1      : 最小列数（C1 より少ない列数のオープンデータは処理対象としない）
#   C2      : 最大列数（C1 より大きい列数のオープンデータは処理対象としない）
#
import pandas as pd
import numpy as np
import sys

# 類似度の計算
def calc_sim(r1,r2):
  v1=np.array(r1)
  v2=np.array(r2)
  #return np.dot(v1,v2)/np.linalg.norm(v1)/np.linalg.norm(v2)   # コサイン類似度
  return np.dot(v1,v2)   # 内積

### メイン ###

# 引数処理
pv_file=sys.argv[1]
list_A=sys.argv[2]
list_B=sys.argv[3]
item_sim=sys.argv[4]
C1=int(sys.argv[5])
C2=int(sys.argv[6])

# 述語ベクトルファイルの読み込み
print("reading PV-Vector file. It often takes long time...",flush=True)
pv_data=pd.read_csv(pv_file,encoding="utf-8-sig")
op_list=pv_data['op'].tolist()    # オープンデータファイル名のリストを作っておく
pv_data=pv_data.set_index('op')   # op 列をインデックス化する
#print(pv_data)
print("Done",flush=True)

# オープンデータのリストを読み込む
with open(list_A,"r",encoding="utf-8-sig") as f:
  names=f.readlines()
  names_A=list(map(lambda x:x.strip(),names))
with open(list_B,"r",encoding="utf-8-sig") as f:
  names=f.readlines()
  names_B=list(map(lambda x:x.strip(),names))

# オープンデータの述語ベクトルがあるかどうかを確認し,
# もし無い場合はリストから削除する
# これは全部ゼロベクトルだった場合など，オープンデータの
# 述語ベクトルが作成されない場合があるから
print("removing op for non-exist pred_vec...")
rm_list=[]
for op in names_A:
  if op not in op_list:
```

```
    rm_list.append(op)
for op in rm_list:
  names_A.remove(op)
  print(" pred_vec not found for "+op)
rm_list=[]
for op in names_B:
  if op not in op_list:
    rm_list.append(op)
for op in rm_list:
  names_B.remove(op)
  print(" pred_vec not found for "+op);
print("done.")

# 列間類似度の計算
cols=['op1','op2','it1','it2','sim']
df=pd.DataFrame(columns=cols)
for i,op1 in enumerate(names_A):
  print("  i=",i,end="",flush=True)
  op1_pv=pv_data.loc[[op1]].values
  c1=len(op1_pv)     # op1 の列数
  if c1<C1 or c1>C2: # op1 の列数が C1 より小さい，または C2 より大きいなら,
    print(" skipped: c1=",c1,flush=True)
    continue        # 処理しない
  else:
    print(" processing: c1=",c1,flush=True)
  processed_list=[]             # 既に処理済みのオープンデータのリスト
  if i>0:
    processed_list=names_A[:i]
  for j,op2 in enumerate(names_B):
    print("    j=",j,end="",flush=True)
    if op1==op2:       # 同じオープンデータは
      print(" skipped: same opendata.",flush=True)
      continue        # 処理しない
    if op2 in processed_list:  # 既に外側のループで処理されていれば,
      print(" skipped: processed opendata.",flush=True)
      continue        # 処理しない
    op2_pv=pv_data.loc[[op2]].values
    c2=len(op2_pv)     # op2 の列数
    if c2<C1 or c2>C2: # op2 の列数が C1 より小さい，または C2 より大きいなら,
      print(" skipped: c2=",c2,flush=True)
      continue        # 処理しない
    else:
      print(" processing: c2=",c2,flush=True)
    for pv1 in op1_pv:
      for pv2 in op2_pv:
        s=calc_sim(pv1[1:],pv2[1:])
        se=pd.Series([op1,op2,pv1[0],pv2[0],s],index=df.columns)
        df=df.append(se,ignore_index=True)

# 結果をファイルに出力する
df.to_csv(item_sim,encoding="utf-8-sig",index=False)
```

リスト A1-5　プログラム calc_op_sim.py

```
# -*- coding:utf-8 -*-
# 2 つのオープンデータリスト A,B と列間類似度ファイルを読み込み,
#  リスト A とリスト B の間のすべての組み合わせについてオープンデータ間類似度を標準出力に出力する.
# ---
# python calc_op_sim.py path list_A list_B it_sim_file method op_sim_file
#   path       : オープンデータへのパス
#   list_A     : オープンデータのリスト A
#   list_B     : オープンデータのリスト B
#   it_sim_file : 列間類似度ファイル
#   method     : A:重み無し B:ガウス重み C:減衰関数重み D:閾値
#   op_sim_file : 出力ファイル

import numpy as np
import pandas as pd
import math
import sys

# 正規分布
def ND(x,sig):
  return math.exp(-x*x/sig/sig/2)

# 減衰関数
def AF(x,tau):
  return math.exp(-x/tau)

# op1 と op2 との類似度の計算（A:重みなし）
def calcSimA(op1,op2):
  a=it_sim[it_sim['op1'].isin([op1]) & it_sim['op2'].isin([op2])]
  if a.empty:
    return 0
  std = a['sim'].std()
  return np.sum(a['sim'].as_matrix())/len(a)

# op1 と op2 との類似度の計算（B:ガウス重み付き）
def calcSimB(op1,op2):
  a1=it_sim[it_sim['op1'].isin([op1]) & it_sim['op2'].isin([op2])]
  if a1.empty:
    return 0
  sig=len(a1)/10
  a2=a1.sort_values("sim",ascending=False)
  i=0
  sum1=0
  sum2=0
  for s in a2['sim']:
    w=ND(i,sig)
    sum1+=w*s
    sum2+=w
    i+=1
  return sum1/sum2

# op1 と op2 との類似度の計算（C:減衰重み付き）
def calcSimC(op1,op2):
  a1=it_sim[it_sim['op1'].isin([op1]) & it_sim['op2'].isin([op2])]
  if a1.empty:
    return 0
  tau=len(a1)/2
  a2=a1.sort_values("sim",ascending=False)
  i=0
  sum1=0
```

```
  sum2=0
  for s in a2['sim']:
    w=AF(i,tau)
    sum1+=w*s
    sum2+=w
    i+=1
  return sum1/sum2

# op1 と op2 との類似度の計算（D:閾値）
def calcSimD(op1,op2,th):
  a1=it_sim[it_sim['op1'].isin([op1]) & it_sim['op2'].isin([op2])]
  if a1.empty:
    return 0
  a2=a1.sort_values("sim",ascending=False)
  sum=0
  num=0
  for s in a2['sim']:
    if s>=th:
      sum+=s
      num+=1
  if num==0:
    return 0
  return sum/num

### メイン ###

# 引数処理
path=sys.argv[1]
list_A=sys.argv[2]
list_B=sys.argv[3]
it_sim_file=sys.argv[4]
method=sys.argv[5]
op_sim_file=sys.argv[6]

# オープンデータのリストを読み込む
path=sys.argv[1]
with open(list_A,"r",encoding="utf-8-sig") as f:
  names=f.readlines()
  names_A=list(map(lambda x:x.strip(),names))
with open(list_B,"r",encoding="utf-8-sig") as f:
  names=f.readlines()
  names_B=list(map(lambda x:x.strip(),names))

# 項目間類似度ファイルを読み込む
it_sim=pd.read_csv(it_sim_file,encoding="utf-8-sig")   # UTF-8 BOM付

# すべてのオープンデータ間の類似度を計算する
# ただし，同じオープンデータは処理しない
# さらに，names_A の中ですでに処理されているオープンデータは処理しない
n_A=len(names_A)
n_B=len(names_B)
op_sim=open(op_sim_file,"w",encoding="utf-8-sig")
op_sim.write("op1,op2,sim-"+method+"¥n")
for i in range(n_A):
  processed_list=[]
  if i>0:
    processed_list=names_A[:i]
  for j in range(n_B):
    s=0
    if names_A[i]==names_B[j]:
      continue
    if names_B[j] in processed_list:
      continue
```

```
    if(method=="A"): s=calcSimA(names_A[i],names_B[j])
    if(method=="B"): s=calcSimB(names_A[i],names_B[j])
    if(method=="C"): s=calcSimC(names_A[i],names_B[j])
    if(method=="D"): s=calcSimD(names_A[i],names_B[j],0.5)
    if(method=="E"): s=calcSimD(names_A[i],names_B[j],0.6)
    if(method=="F"): s=calcSimD(names_A[i],names_B[j],0.7)
    if(method=="G"): s=calcSimD(names_A[i],names_B[j],0.8)
    if(method=="H"): s=calcSimD(names_A[i],names_B[j],0.9)
    op_sim.write("¥""+names_A[i]+"¥",¥""+names_B[j]+"¥","+str(s)+"¥n")
```

リスト A1-6　プログラム combine-op_sim.py

```
# -*- coding:utf-8 -*-
# オープンデータ類似度ファイルを結合する
# ---
# Usage: python combine-op_sim.py path

import pandas as pd
import sys

path=sys.argv[1]

simA=pd.read_csv(path+"/op_sim-A.csv",encoding="utf-8-sig")    # UTF-8 BOM付
simB=pd.read_csv(path+"/op_sim-B.csv",encoding="utf-8-sig")    # UTF-8 BOM付
simC=pd.read_csv(path+"/op_sim-C.csv",encoding="utf-8-sig")    # UTF-8 BOM付
simD=pd.read_csv(path+"/op_sim-D.csv",encoding="utf-8-sig")    # UTF-8 BOM付
simE=pd.read_csv(path+"/op_sim-E.csv",encoding="utf-8-sig")    # UTF-8 BOM付
simF=pd.read_csv(path+"/op_sim-F.csv",encoding="utf-8-sig")    # UTF-8 BOM付
simG=pd.read_csv(path+"/op_sim-G.csv",encoding="utf-8-sig")    # UTF-8 BOM付
simH=pd.read_csv(path+"/op_sim-H.csv",encoding="utf-8-sig")    # UTF-8 BOM付

sim=pd.merge(simA,simB)
sim=pd.merge(sim,simC)
sim=pd.merge(sim,simD)
sim=pd.merge(sim,simE)
sim=pd.merge(sim,simF)
sim=pd.merge(sim,simG)
sim=pd.merge(sim,simH)

sim.to_csv(path+"/op_sim.csv",encoding="utf-8-sig",index=False)      #  UTF-8
BOM付
```