**自然言語処理を用いたDocumentDB　 schema構成ツールの提案**

**Abstract**

NoSQLデータベースの一種であるドキュメント志向データベースは扱うデータの柔軟性と大規模データ分析に利用できる性能面で利点があり、広く使われるようになってきている。このドキュメント志向データベースの特徴としてあらかじめ明示的にスキーマを定義しておかなくても不定形のデータを格納することができる。つまりはスキーマレスで動的なスキーマをもつ。しかしながらこの特徴をもつことでデータ構造が複雑になりやすく、アプリケーションや分析に活用する場合に困難をともなう場合がある。そのため開発者の作業を容易にし、分析をしやすくするには適切で明示的なスキーマを作成する必要がある。しかし、適切なスキーマを構成するには格納するデータの内容およびデータベースの特性を理解する必要があり、初学者には困難である。

本研究では代表的なドキュメント志向データベースであるMongoDBスキーマにおいて、適切なスキーマ構成の生成をサポートする手法について提案する。このスキーマを生成するために、ドキュメントをグループ化し、適切な階層的データ構造を提案するツールを作成する。自然言語処理を用いて複雑化した構造に対して適切な構造を提案するアプローチを提案する。

Intro　はじめに

コンピューターシステムで扱うデータの複雑化、大規模化にともなって既存のRDBでは固定したテーブルスキーマでは格納できない、また格納してもパフォーマンスを十分確保できないまたはできても困難なケースが発生している。 コンピューターシステムで扱うデータのサイズ、種類、生成速度の増加、すなわちビッグデータ化に伴い、従来のデータストアは、構造的な硬直性や高いアクセスレイテンシーによる応答性の悪さなど、多くの理由で性能が低下している。一方で性能と可用性が最優先される新世代のアプリケーションにおいて、このようなデータをいかに効率的に管理するかが重要になっている。その結果、スキーマの柔軟性、スケーラビリティ、高性能、パーティション耐性などの新しい概念で、従来のリレーショナルデータベースの機能を拡張するNoSQL（Not Only SQL）データベースの利用が増えている。

MongoDBはドキュメント指向のNoSQLデータベースの一つである。MongoDBはJSONドキュメント構造でデータを格納し、スキーマレスなので柔軟なデータモデリングが可能である。また、分散環境でのスケーラビリティや高いパフォーマンスを実現するための設計に基づいています。MongoDBはオープンソースソフトウェアであり、企業やスタートアップなど様々な業界で幅広く利用されている。

MongoDBを含むNoSQLでも、データベース構造を規定するスキーマは存在する。しかし、そのようなスキーマがなくても自由にデータを取り扱うことができるのがNoSQＬの特徴であり、実際スキーマなしで利用されることが多い。

参照（<https://www.mongodb.com/developer/products/mongodb/mongodb-schema-design-best-practices/>）

MongoDBでは、データをどのようにモデル化するかは、特定のアプリケーションのデータアクセスパターンに完全に依存する。アプリケーションのクエリや更新の方法に合わせて、データを構造化したいという要求があり、そのため、データベース管理者側ではなく、クライアント側のアプリケーション開発者によって作成されることが多い。 NoSQLデータベースではアプリケーション側でのユースケースがスキーマの構成、パフォーマンスに影響し、アプリケーション開発者が作成するほうが都合がよい。しかし、扱うデータが大規模で複雑であることに加え、設計者のデータモデル技術未成熟であること 、モデリングガイドラインが不十分であることは、NoSQL スキーマの品質を保つのが困難となっている。その結果、誤ったデータベースのモデリングや設計が増え、低パフォーマンスで非セキュアで耐久性の低いシステムが製造されることになる。

一方、関係データベースで指摘されているように長期間利用されるにつれ構造が悪化し、“匂い(smell)”をもつとしている。Ambler [1] は，悪化したデータベースは次のような“匂い(smell)” を持つとしている．

*•* 複数目的で使用するテーブルやカラム

*•* 重複のあるデータ

*•* 多すぎるカラム数

*•* 多すぎる行数

MongoDBにはスキーマはあるが、RDBのテーブルに相当するDocumentは異なる構造を持っていいというフレキシブリティを有する[MongoDBスキーマの文献]。

https://www.mongodb.com/docs/manual/core/data-modeling-introduction/

このために、開発段階から構造悪化が発生し、匂いが生じると考える。著者らは以下の匂いに着目した。理由はこれらの匂いが頻繁に発生すると考えたからである（本当？　文献あるか）

1. MongoDBのデータを格納した階層構造において、適切でないところにデータが置かれる。
2. MongoDBにおいて、階層構造を持たずに、データ階層が2階層で格納される。これはCSV形式のデータをそのままMongoDBに取り込んだときに発生しうる。

共に、MongoDBの再利用する際に保守工数の増大、またP2に関してはアクセスに時間がかかるという問題がある。

以上の点より，本論文ではドキュメントデータベースの，影響範囲の調査に着目し，スキーマの作成を支援するツールについて述べる．本ツールは，データベースを管理 するデータの目的に合わせた設計をしている場合に，JSONデータの見出しの上下関係は密接に関連しているという仮説に基づき，自然言語処理の語間の距離を計測する手法を用いて， ドキュメント名間の距離から再構成対象とする手法を提案する．

本論文の構成を次に示す．まず，本研究のベースになっているcollection名とfield名の問題とこの問題を解決するために必要な機能を 2 節で述べる．3 節で提案手法の概要を，4 節で本ツールの機能概要を述べる．5 節で，本ツールを実際のJSONデータおよび構成するCSVに適用して上記の仮設を検証し，ツールの有効性を示す．関連研究を 6 節で述べ，7 節でまとめを 述べる．

2要求定義

2.1用語の定義

MongoDBはドキュメント指向データベースであり、JSON形式でデータを格納される。データはコレクションに格納され、各ドキュメントはコレクション内にJSON形式で格納される。

MongoDBのデータ構造は以下の要素で説明される

（https://www.mongodb.com/docs/manual/introduction/）

コレクション（Collection）：MongoDB内に格納されるデータの集合体で、関連するドキュメントをまとめることができる。RDBMSでいうところのテーブルに相当する。

ドキュメント（Document）：MongoDBで格納される最小単位のデータで、JSON形式で格納されます。RDBMSでいうところのレコードに相当します。”{“,”}”で囲われた複数のフィールドから構成される。

フィールド（Field）：ドキュメント内のデータの要素であり、“：”で区切られた名前と値のペアで表現されます。RDBMSでいうところの列に相当します。

ネストされたドキュメント（Nested Document）：ドキュメント内に別のドキュメントを埋め込んで格納することができます。具体的にはフィールドの値にドキュメントを置くことができる。RDBMSでいうところの親子関係を持つテーブルに相当します。

配列（Array）：ドキュメント内に配列を格納することができます。RDBMSでいうところの複数の値を持つ列に相当します。

ドキュメントは、一つのテーブルの行に相当します。ドキュメントは、複数のフィールドで構成され、キーと値のペアで表現されます。ドキュメントには、必ず一つの"\_id"フィールドが含まれ、MongoDBにおいては、このフィールドによってユニークな識別子が自動的に割り当てられます。

ドキュメントは、JSON形式で表現されるため、データの入れ子構造を持つことができます。例えば、以下のようなドキュメントが考えられます。



このドキュメントは、"name"、"age"、"address"、"phone"の4つのフィールドから構成されています。"address"フィールドは、入れ子になったドキュメントであり、"phone"フィールドは、配列の形式で複数の値を持っています。

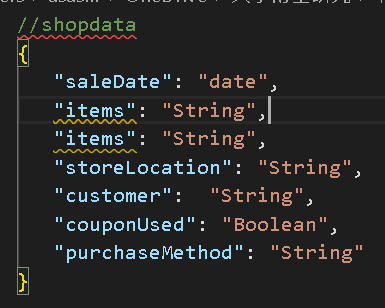
2.2 ドキュメントDBのスキーマ設計に対する問題点

イントロダクションで述べたmongoDBの有する匂いの引き起こす問題を述べる。

1. 不適切なフィールドの配置

MongoDBのデータを格納した階層構造において、適切でないところにデータフィールドが置かれる。これには以下のような場合がある。

1. 一つのドキュメント下の同じキーをもった複数のフィールドがある場合。具体例を示す。



★ここに例を入れる

データベース設計の原則である「One Fact in One Place」に反するケースである。これらが同じフィールドであるのか、異なるフィールドであるのかが判別できず、データベースの保守性を大きく損なうことになる。

1. 異なるドキュメント下のあるフィールドが異なる箇所にある場合。具体例を示す。



★ここに例を入れる

プログラムでこれらの2つのデータにアクセスする場合、データの所在を指定してアクセスするため、以下のようなクエリAPIをかき分けることになる。

★ここにプログラム例を入れる。

どういう規則で複数のフィールドがあるかわからないとプログラムを書くことは不可能である。

1. ドキュメント内のフィールドの上下関係は適切でない。これには以下のような場合がある。

★ここに例を入れる

このような例はデータ構造の可読性を下げ、保守性の低下に繋がる。

（2）階層構造を持たずに、データ階層が2階層で格納される。これはCSV形式のデータをそのままMongoDBに取り込んだときに発生しうる。例を示す。



★ここに例を入れる。

このようなフラットなデータ構造のフィールドは、指定したフィールドアクセス時に全てのフィールドをフィールドが見つかるまで読み込みことになり、フィールド数が膨大な場合には時間を要することになる。

★ここにプログラム例を入れる。

2.2　必要とされる機能

★問題を解決するための要求定義を書く。3節への橋渡し

本ツールで必要とされる機能について述べる。本ドキュメントDBスキーマ構成ツール

3自然言語処理を用いたスキーマ構成の検出

本研究ではJSON構造をもつスキーマの構成をデータのフィールド名をもとにスキーマのドキュメント間の意味的な距離および上下関係の指標を明示することでスキーマ設計を支援することを目的とする。

★スキーマの上下関係が全体→部分(has-a-relationship)になっていることを述べる。全体→部分はテーブルに対応したものだが、

JSONはこれだけでいいか。例えば、抽象→具体など。クラス定義の仕方を調べるか？

提案手法についての概要を現した図をいかに示す。提案手法はテキストコーパスから単語間の上下関係を評価できる環境を構築し、自然言語処理を使用してスキーマの上下関係の指標★名前はある？を算出★こちらの算出の仕方を少し書く？また上下関係とともに語の類似度を測定してツリー構造となるクラスター構成となる単語群を作成し、ドキュメント間の意味的な距離を算出する。両指標をベース修正対象の構造を検出し、候補となるドキュメントを推薦、評価する。この類似度の算出にはドキュメントのフィールド名に使用している単語を自然言語処理を行ってベクトル変換を行い、そのコサイン類似度を算出することで類似度としている。この算出にはFastText(https://fasttext.cc/)

を使用した。得られたコサイン類似度を用いてクラスタリングを行い、その結果によって語間の意味的な距離を表示する。上下関係についてはsketch engine2

を使用した。単語間の共起関係から上下関係を検出し、関係が成立するか判定する。

3.1 語の上下関係の算出

Has a part of の関係抽出

語の上下関係★語の上下関係とHas-a-relationshipは微妙に違うことを意識のことの算出にはsketch engineを使用した

sketch Engineは、自然言語処理のための強力なコーパス構築および解析ツールです。 それは、膨大な言語データから有用な情報を抽出するための様々なアルゴリズムを提供し、言語学習者、研究者、翻訳者などに役立ちます。 Sketch Engineは、英語、スペイン語、ドイツ語、フランス語、イタリア語などの言語のコーパスを持っており、さらに多くの言語にも対応しています。 それは、単語の出現頻度、単語の共起、グラマー、スキャニングなどの機能を提供し、文法的なパターンや語彙の使用を研究するために使用される。

このsketch engineにはユーザー独自のコーパスを作成する機能、コーパスに対して柔軟な複雑な語彙のパターンを検索できる特別なクエリ言語であるCQL(Corpus Query Language)を使用した。

CQLでは下位語から上位語

★どこかでHas-a-relationship抽出のためのCQL定義を記載のこと

-----------------------------------------------------------------------

Table 3. Meronym patterns

Meronym Patterns Example

NPx PPy

—PPy starts with of, inside door of the car, Walls inside the building

NPx PPy

—PPx starts with above they ambute his leg above the knee.

NPys NPx buiding’s basement

NPy verb NPx

—verb have car has an engine

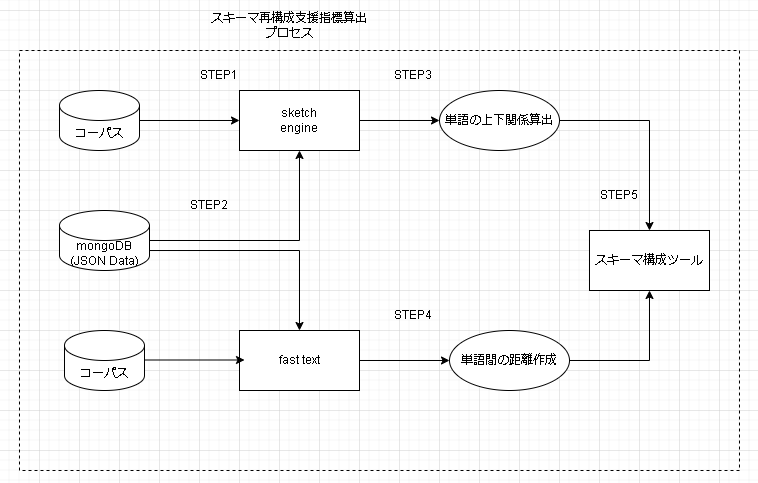
NPx verb NPz PPy

—PPy starts with of

—NPz is a part or is a member finger is a part of hand, Iceland is a member of NATO

-------------------------------------------------------------------------

3.2 階層のクラスタリングおよび凝集度算出



Mongodbスキーマ構成プロセス

★ここは前回論文の3.2に合わせる

上記のプロセスを実行するため、スキーマのドキュメントの上下関係を算定し、スキーマの構成に使用する

STEP1

Sketch engineで使用するコーパスを設定する。

Sketch engineではTenTenを使用した。

TenTenコーパス（TenTen corpora）

URL(https://www.sketchengine.eu/documentation/tenten-corpora/)

は、ウェブから作成されたテキストコーパスである。このコーパスは、以下の言語的に価値のあるWebコンテンツのみを収集することに特化した技術によって構築されている。Sketch engine上で最大規模のコーパスであり、分析するドメインの幅が広い場合に有効である

STEP2

Mongodbのスキーマのドキュメント見出し単語のペアを抽出する

STEP3

Sketch engine上で単語の上下関係の統計情報を抽出する

単語の上下関係の強度についてはMIscore(相互情報量スコア)★(文献コーパス研究が切り開く新しい日本語教育)を使用した

MIscoreは 、単語が別々に出現する回数と比較して、どの程度共起しているかを表現するものです。MIスコアは、 頻度に強く影響され 、低頻度の単語は、誤解を招く可能性のある高いMIスコアに達する傾向があります。そのため、Sketch Engineでは、頻度の低い単語を計算から除外するために、頻度制限を設定することができる。

MIスコア（または相互情報量）は次式で手議される値である。

MI=log2共起頻度×総語数中心語頻度×共起語頻度

MIスコアが2以上になると有意な組み合わせであるとされる。 MIスコアは、頻度は低いが特殊な結びつきをしているコロケーションがうまく検出できるとされている（低頻度の語を強調する傾向がある）。

STEP4

ステップ 3 で得られたカラム名のベク トル表現に対して，クラスタリングを行い，カラ ム名をクラスタ化する．クラスタ化は DBSCAN (文献　以前の4番)を用いた．DBSCAN は高密度領域をクラス タとして固め，距離のある語をクラスタ外という形でクラスタ化を行う．このため，距離を持っ た要素を近傍の距離を上限をパラメタ (eps) としている． スキーマの再構成方針としては，以下が考えら れる．1)DBSCAN のクラスタリングにより類似 度が高いドキュメント名をクラスタとして分離する．2) それ以外のドキュメント名は元のツリーに残す．元の ツリーよりも役割が明確になったツリーを分離したことになる。

上記新しく作成するツリーについて分岐するオブジェクトを作成する必要が発生する。この中間オブジェクトについて名前を別途指定する必要がある。ユーザの指定により名前は決めるが、その候補としてツール上で推薦する。この候補はツリー下部のドキュメントに含まれる単語、および下部のツリーより共起語として検出できる単語を候補として表示する。

STEP5

利用者はクラスタ化されたドキュメント名間の類似度をもとに再構成対象のカ ラムを選択し再構成を行う．先の述べたツリーの複雑さの指標としてクラスター内誤差平方和 (SSE) を用いる．これは以 下の式で表される．

SSE = ΣK j=1Σxi∈Cj ||xi − µj ||2

ここで，µj = 1 nj Σxi∈Cj xi はクラスタ Cj の平 均を，nj はクラスタ Cj 内の要素数を表す．再構成効果をスキーマの再構成前後の凝集度を使って評価する．評価結果が妥当であれば，再構成を検討する。

4 支援ツールの概要

3 節で述べた機能をスキーマ構成支援ツールとして以下のように実現した．本ツールは３つの表示部で構成さ れている．それぞれ，JSONスキーマ構成の表示，ドキュメントの類似度関係表示，ドキュメント間上下関係の関係表示を行う．表示したグラフを操作してスキーマ の構成を変化させる．例えば，ノードをあるツリー から別のツリーに移動させる，ツリーを分離させて 別のツリーを生成するなどの再構成を行う．その結果，変化する凝集度および上下関係の指標を計算することにより，再構成後 のスキーマの評価を行う．

また構成を変更した場合の指標を計算した結果から候補ドキュメントを表示する。

・対象データベースのスキーマの構成表示

　対象となるmongodbのスキーマ構成の概要を表示する。実際のデータおよび設定されていればJSONスキーマを表示する。再構成時には再構成後の構成および変化箇所を表示する

・各ツリーのドキュメントの類似度関係表示

　上記で得られたドキュメント間のコサイン類似度を元にドキュメントのクラスタリング結果を表示する。ドキュメント間の類似度を表示することにより、JSONスキーマの再構成時にツリー構成の候補を提示する。

・各ツリーのドキュメント上下関係値の表示

各ドキュメント間の上限関係統計値をツリー上の有効グラフで表示する。関係が強く、上下関係が成立するほどエッジの線を太く表示し、表示する閾値を変更することができる。またエッジの線の接続先を変更することで再構成後の値を確認することができる。　これにより再構成候補を検出できる。

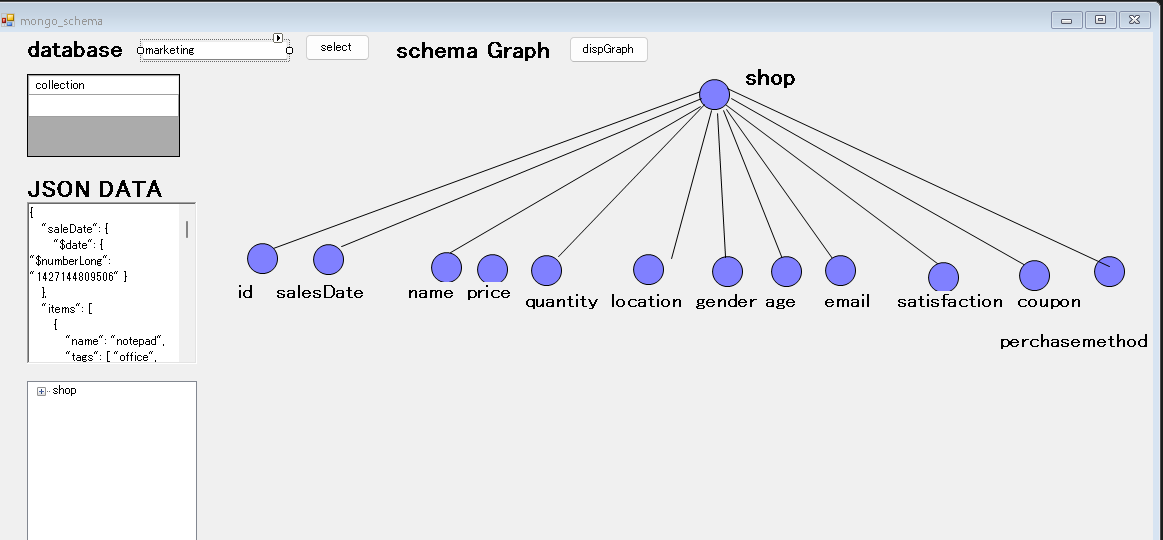
・スキーマの再構成: 再構成対象のノードが検出できると，そのノードが属するツリーと再構成 先のツリーを同時に表示し，再構成対象のドキュメントを選択し再構成先としたツリーに移動させ， スキーマの再構成を行う．新たにツリーを作る場合は，右のテーブルでコンボボック スから新規作成を選んで新しいテーブルを作る．

・再構成の評価：

上記情報をもとにユーザは既存スキーマに対して画面より変更を加え、加えた場合の変更結果を表示する。この際、上限関係の指標および凝集度を再計算して表示する。またその際のクラスタリングを再実行し、グループ分けした結果を表示する。この結果をもとにユーザはスキーマの構成および修正を行う。

ユーザーインターフェースはマイクロソフト.NET Framework4.6 で windows フォームアプリケーショ ンで実装した．また DBSCAN の計算環境は AWS の EC2 インスタンス上に Linux（ubuntu）を インストールしてバックエンドとして実装した．

以下画面の詳細および操作について述べる



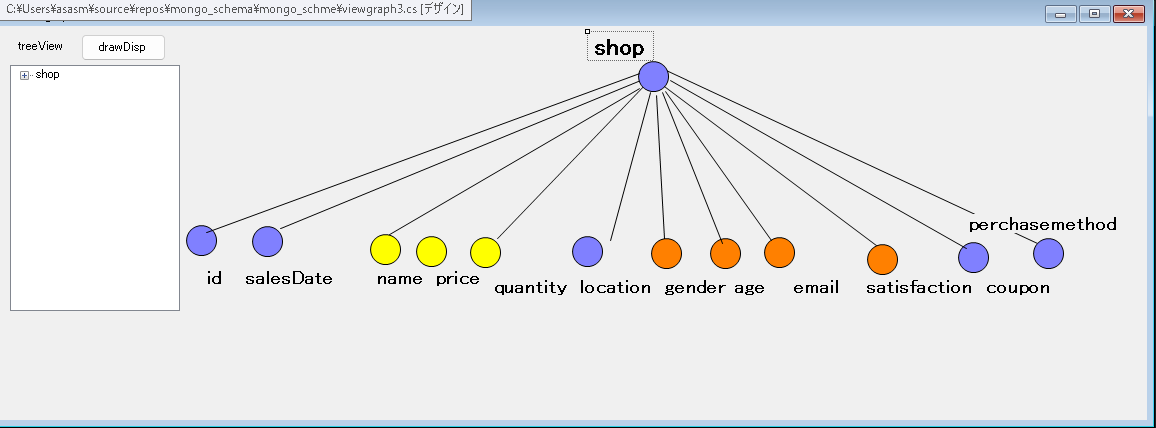
画面１

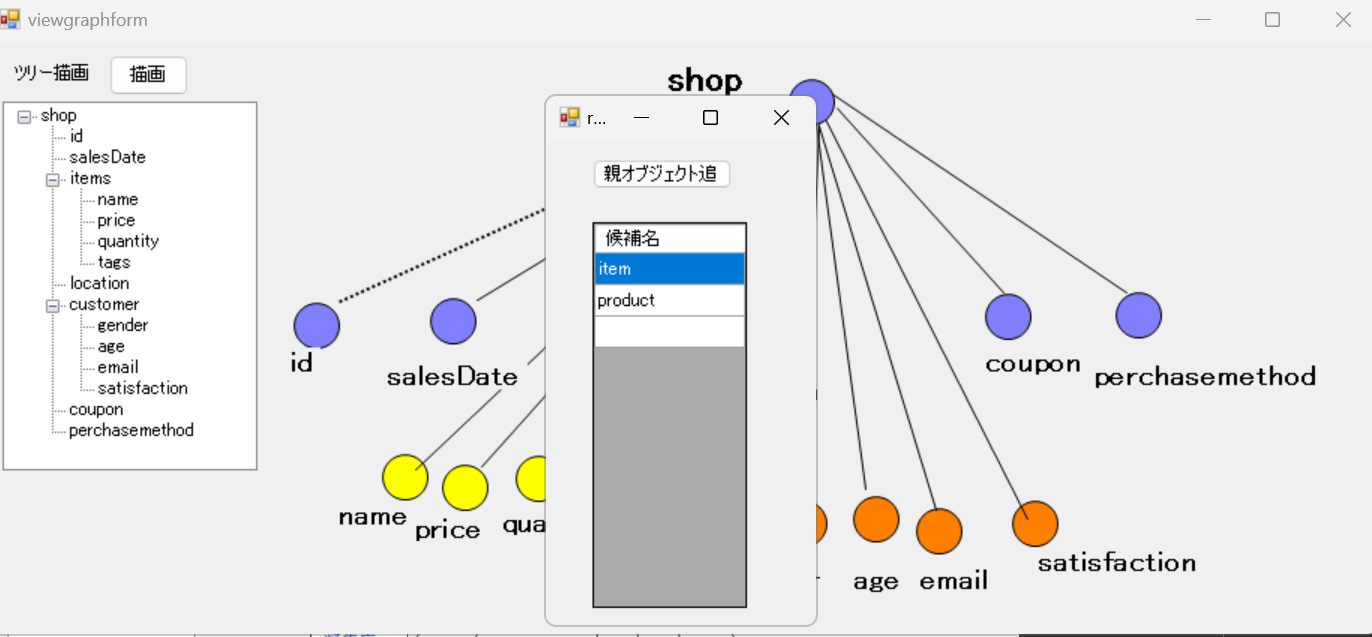
画面１は対象データベースの基本情報を表示する。最上部には接続したmongodb名を表示している。Collection一覧では接続したmongodbより格納しているcollectionを一覧表示し、ユーザはこの一覧より表示および操作対象とするcollectionを選択する。

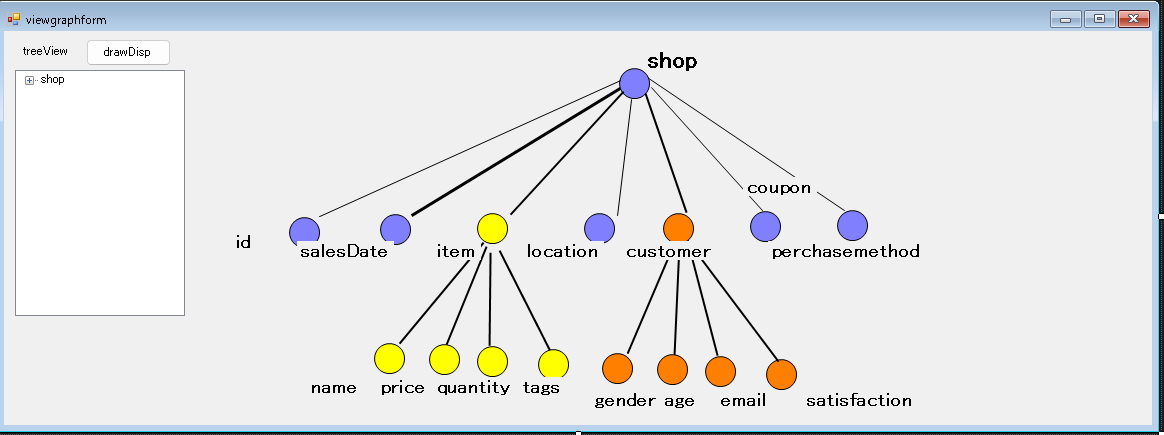
JSON DATA表示部では実際のcollectionの生データを表示確認できる。

画面中央のグラフ表示部では現在のJSON DATAからスキーマ部分を抽出し、ツリー構造として表示している。

これらによりデータベースの構造の概略をユーザー自身が把握できる。







画面２　スキーマ変更部

画面２はスキーマ変更を推薦および変更した際の結果を予測表示する部分である。

画面中央のグラフ表示部ではクラスタリング結果を表示し、グループごとにノードを色分けして表示している。またノード間の上下関係の近さは線の太さと吹き出しによる指標の表示で表している。ユーザーはこの結果を参考にノード間に新たなオブジェクトの追加およびオブジェクトを別のノードに移動させることができる。移動した後、再度再計算結果を表示、再構成前後のスキーマの完成度を比較することができる

5 検証実験

★2つの評価値はうまく提示されている？

MIscoreと凝集度からツリーを分離できたことを説明修正

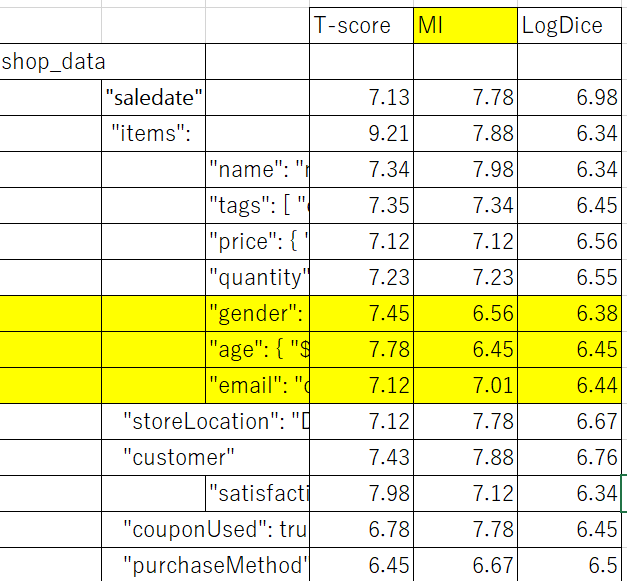
本研究では提案するスキーマ作成支援ツールの有効性を評価するため、オープンデータから評価用テーブルを抽出し、JSONスキーマを構成できるか検証実験を行った。評価法としてはオープンデータサイトDatahubよりEコマースの購買データ(shop.json)を使用してJSONスキーマの上下関係を再現できるかを検証した。またhotellist.csvを取得してJSONスキーマを構成できるか検証をおこなった

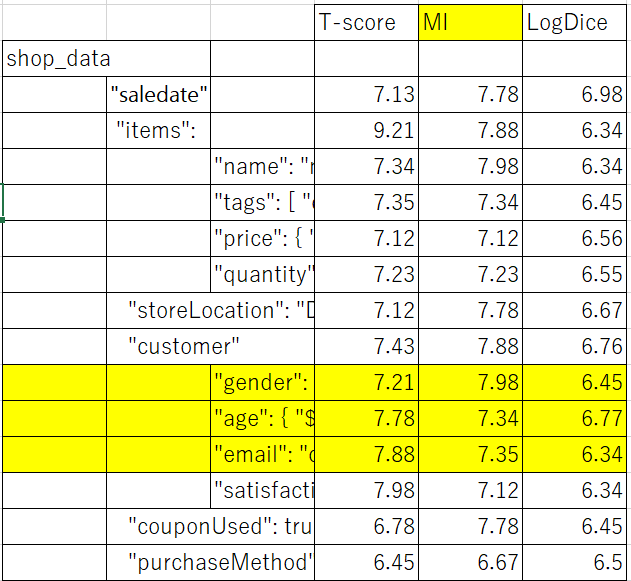
TenTenコーパス（TenTen corpora）は、証ウェブから作成されたテキストコーパスである。このコーパスは、以下の言語的に価値のあるWebコンテンツのみを収集することに特化した技術によって構築されている。Skecth engine上で最大規模のコーパスであり、分析するドメインの幅が広い場合に有効である

参照（<https://www.sketchengine.eu/blog/build-a-corpus-from-the-web/>）

表はsketch engineで単語とshopデータスキーマの各ドキュメントの上位のラベルとのM1scoreを算出したものである。 ★この表でご利益が分かるだろうか？

M1scoreを８.0付近としてみるとgender,age,emailは別のグループとわけることができる。これによりこの３つのデータは別の階層にいれることを検討したほうがよいとなる。これはもともとのスキーマと一致する。★もとのスキーマとは



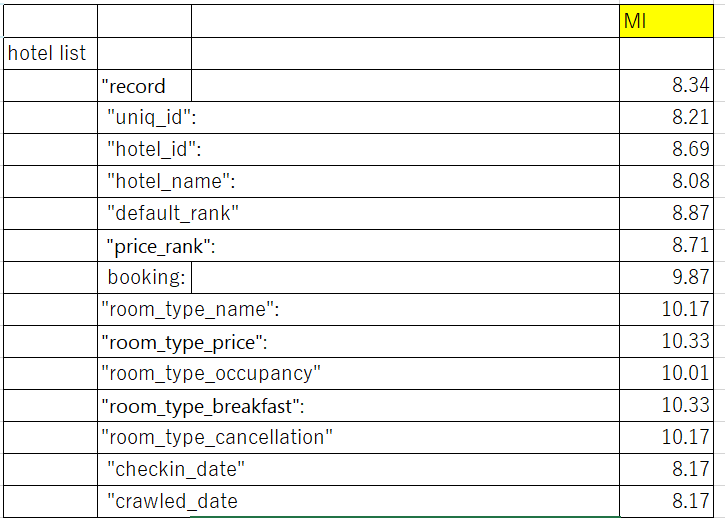


インターネット上に公開しているオープンなCSVデータでJSONの階層構造が作成できるか検証を行った。こうしたオープ ンデータで csv 形式が多く，かつ必ずしもJSONスキーマで扱うことを前提にはしていないが，JSON形式で公開することも検討されることが多い。上記はホテル情報をあつめたCSVデータである。

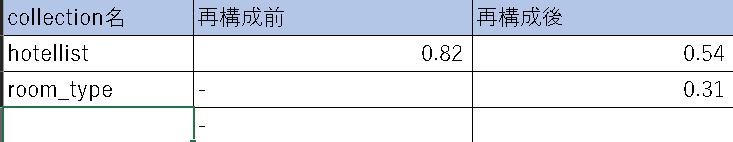
MIscoreが10ポイントのところで切り分けるとroom\_type\_name~room\_type\_cansellation

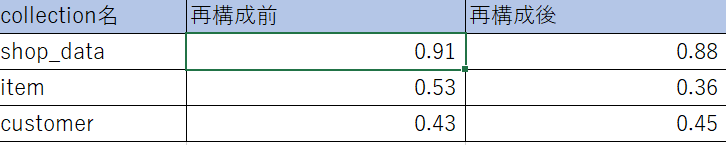
で別のグループと判断することができる。ここからJSON形式に変換すること想定すると

room\_typeからroom\_type\_cansellationは別階層を検討したほうがよいことを示している



★2つの凝集度の計算、再構成後の値





上記、クラスタリング結果に基づき、再構成後の凝集度を測定した。クラスタの凝集度は 1 に近いほどクラスタ内の要素が分散していることを表す．再構成前は 1 つのツリーであるhotelistは 0.82 であったが，再 構成後はhotellistは 0.54，新たに作成されたツリーはそれぞれ 0.31のように小さくなっ ており，ドキュメント名の凝集が増し，再構成が有効に行わ れたことを示している．

6. 関連研究

MongoDBはスキーマを規定しなくても利用できることから様々な問題を引き起こしており、その問題を解決するためにツールが開発されている。

まず、データの可視化ツールを紹介するMongoDB Compass Communityでは、LIST構造とTable構造で可視化するツールを提供している(文献CompassのURL)。前者はJSONと同じような形式で、後者は空カラムの多いものとなっている。RobomongoとMngoVUEは木構造でデータを表示し、データを扱う手段を提供している（文献：URL）。(文献tree)では、これらのツールはJSONの知識が必要でありエンドユーザ向けには使いにくいとしている。JSONの知識を知らないエンドユーザでもデータを木構造で表示、操作する機能を提供している。

mongoDBのデータ構造を解析するものとして、jHoundがある(文献：jHound)。これはmongoDBのデータをツリーの深さ、フィールドが多く存在すうるレベル、フィールド数を表示することでユーザは当該mongoDBの構造の良さ、悪さを知ることができる。

スキーマを生成するアプローチが提案されている。

(文献：SchemaManagemnt)はスキーマのないJSONのデータ構造からスキーマを抽出する。

(文献：A model-drivin)

1. Bojanowski, P., Grave, E., Joulin, A. & Mikolov, T. Enriching word vectors with subword information. *Transactions of the association for computational linguistics* **5**, 135–146 (2017).

2. Kilgarriff, A. *et al.* The Sketch Engine: ten years on. *Lexicography* **1**, 7–36 (2014).

[Dataset] Bizer, C., Eckert, K., Faralli, S., Meusel, R., Paulheim, H., and Ponzetto, S. P. (2016). Web

Data Commons - WebIsA Database

Chien, O. (2013). Modeling and Querying Data in MongoDB. *International Journal of Scientific &*

*Engineering Research* 4, 141–144

Chillon, A. H., Ruiz, D. S., Molina, J. G., and Morales, S. F. (2019). A Model-Driven Approach

to Generate Schemas for Object-Document Mappers. *IEEE Access* 7, 59126–59144. doi:10.1109/

ACCESS.2019.2915201

KEY: Chillon2019

ANNOTATION: schema などのNoSQL のデータを組織的にアクセスするためにOR Mapping

するメカニズムを作った

Chodorow, K. (2013). *MongoDB: The Definitive Guide* (O’Reilly), 2nd edn.

Faralli, S., Bizer, C., Eckert, K., Meusel, R., and Ponzetto, S. P. (2016). A web application to search

a large repository of taxonomic relations from the web. In *Proc. 15th International Semantic Web*

*Conference*. vol. 1690, 2–5

Jia, T., Zhao, X., Wang, Z., Gong, D., and Ding, G. (2016). Model Transformation and Data Migration

from Relational Database to MongoDB. In *Proc. 2016 IEEE International Congress on Big Data*

*Model*. 60–67. doi:10.1109/BigDataCongress.2016.16

Klettke, M., St¨orl, U., and Scherzinger, S. (2015). Schema extraction and structural outlier detection

for JSON-based NoSQL Data Stores. In *International Conference on Database Systems for Business,*

*Technology and Web*. 425–444

KEY: Klettke2015

ANNOTATION: XML のDTD を用いてXML データのスキーマを作るというアイデアを使っ

て、JSON についたラベルを元に構造化してスキーマとしている。

Sheena, N., Jasmine, S. M., and Joseph, S. (2016). Automatic Extraction of Hypernym & Meronym

Relations in English Sentences Using Dependency Parser. In *Proc. 6th International Conference On*

*Advances In Computing & Communications* (The Author(s)), vol. 93, 539–546. doi:10.1016/j.procs.

2016.07.269

Wang, L., Zhang, S., Shi, J., Jiao, L., Hassanzadeh, O., Zou, J., et al. (2015). Schema management for

document stores. In *Proc. VLDB Endowment*. vol. 8, 922–933. doi:10.14778/2777598.2777601

KEY: Wang2015

ANNOTATION: 要は作り方の規則のないMongoDB のcollecton(SQL のtable に相当) からい

かに、一つのcollection のschema を作るかという論文。collection の定義ありき。こちらは

collection をどう作るかというところになる。

[Dataset] Wright, A., Andrews, H., Hutton, B., and Dennis, G. (2022). JSON Schema: A Media Type for

Describing JSON Documents