LOD データセット生成の自動化のための宣言的記述による半構造化データの抽出とその統合手法

筑波大学

図書館情報メディア研究科

2019 年 3月

豊田 将平

目次

図目次

表目次

1. はじめに

構造化されたデータを Web 上で公開，共有するための仕組みとして Linked Open Data(LOD)がある．LOD は Resource Description Framework（RDF）と呼ばれる標準に従ってデータを記述し，すべてのリソースを International Resource Identifier（IRI）を用いて識別する．Web 上でデータ同士を参照可能にできるため，データセットの公開方法として LOD データセットの利用が望まれる．

様々な分野で LOD に基づいて作成された LOD データセットが公開されている．LOD データセットは人手によるデータ入力で作成されることもあるが，大規模な LOD データセットの多くは既存のデータセットを RDF データに変換することで生成される．リレーショナルデータベースなどの構造化されたデータを LOD データセットとして公開するためのマッピング言語やシステムが提案されている．

Web 上には HTML 文書のような Web ページに記述された情報が大半を占める．HTML は木構造を持つマークアップ言語であるが，HTML 文書の中に記述された情報に対して意味的な構造定義を持つわけではない．HTML 文書は，段落やリストなどの文書構造を要素として定義しているが，その段落やリストなどに何が記述されているかは定義していないからである．このような Web ページに記述された情報から LOD データセットを生成する試みが存在し，代表的な事例には DBpedia[2]がある．DBpedia は，Wikipedia の記事の一部から半構造化情報を抽出し RDF データへ変換することで，LOD データセットを生成している．

Web ページから情報を抽出し RDF データへ変換するアプローチは，他の Web サイトに対しても適用可能であると考えられる．DBpedia は Wikipedia という特定の Web サイトを対象としているが，Web 上には共通の対象に関する情報が複数の Web サイトに分散して記述されている．そこで，ある LOD データセットを作ろうと考えたときに，複数の Web サイトに記述された情報を抽出し，集約することでデータセットを生成するという発想がある．複数の Web サイトを参照し LOD データセットを生成することで，単一の Web サイトからは得られない情報を補完あるいは検証したデータセットを提供することができると考える．このとき同一の実体に関する記述が複数の Web サイトに含まれる可能性があるため，記述対象の同一性を解決しなければならない．また，Web サイトのコンテンツは更新される可能性があり，その更新を反映するためには LOD データセット生成を反復して実行しなければならない．そのため，この LOD データセットを生成する一連の手続きは，可能な限り自動化することが望ましい．

そこで本研究では，複数の Web サイト上の記述を対象に LOD データセットを生成す

ることを目的として，Web ページからの半構造化データの抽出，抽出されたデータの記述対象の同一性の判定，RDF データへの変換，といった一連の LOD データセット生成の処理内容を宣言的に記述する規則を提案する．また，それらに基づいて自動的に LOD データセットを生成する手法を提案する．本手法は，LOD データセット生成におけるデータ抽出やその統合処理を規則として宣言的に記述し，それらの規則に基づいて LOD データセットの生成処理を実行する．

2. 既存データセットからの LOD データセット生成

2.1. Linked Open Data データセット

Linked Open Data(LOD) とは，Tim Berners-Lee によって提唱された 4 つの原則[14]に従い，構造化データを Web 上で公開するための仕組みである．LOD の記述には Resource Description Framework(RDF)[20]が用いられる．RDF のデータモデルは，主語，述語，目的語で構成される組（トリプル）をリソース記述の基本単位とする．トリプルは，主語と目的語をノードとし，述語をそれらの間を結ぶエッジとする連結グラフで表現することができる．また，RDF ではすべてのリソースを International Resource Identifier（IRI）を用いて識別する．例えば，<http://example.com/res/Leonardo_da_Vinci>を主語，<http://example.com/ns/creatorOf>を述語，<http://example.com/res/the_Mona_Lisa>を目的語とするトリプルがあるとする．主語の IRI が参照する実体は画家レオナルド・ダ・ヴィンチであり，目的語の IRI が参照する実体は絵画モナ・リザであるとする．述語の IRI は，主語は目的語の作者であるという関係を示す．従って，このトリプルは「レオナルド・ダ・ヴィンチは『モナ・リザ』の作者である」ということを記述している．

本研究では，LOD に基づいて作成されたデータセットを LOD データセットと呼ぶことにする．LOD を用いてデータセットを公開することで，Web 上であらゆるリソースを相互に参照することができる．CSV ファイルやデータベースダンプなどの機械処理可能な構造化されたデータを公開するだけは，参照可能な識別子が与えられていないためデータ同士の関係を示すことはできない．一方，LOD データセットであれば，単に構造化された形式でデータを公開するだけでなく，他のデータセットに含まれるデータとの関係をリンク付けして利用できる．現在では，既に Web 上には様々な LOD データセットが公開されている．LOD データセット間の関係を調査した LOD cloud diagram[24]によれば，2018 年 6 月時点で 1,231 件のデータセットおよび 16,132 個のデータセット間のリンクが存在している．

2.2. 既存のデータセットからの抽出と変換に基づく LOD データセット生成

LOD データセットを作成するためには，まず，リソースに与える IRI の設計や RDF を用いて記述するためのスキーマを設計する必要がある．RDF では，あらゆるリソースを IRI で識別するため，どのようにリソース対して IRI を与えるか定義しておく必要がある．例えば DBpedia Japanese の場合，<http://ja.dbpedia.org/resource/{リソースのラベル}>のような設計になっており，「筑波大学」には<http://ja.dbpedia.org/resource/筑波大学>という IRI が与えられる．また，スキーマの定義とは，より具体的にはクラスとプロパティの集まりである語彙を定義することが基本であり，RDF Schema[21]などを用いて記述する．加えて，実際に LOD データセットを公開するためには，Web 上に公開する方法などを含めて考える必要があるが，本研究における LOD データセット生成とは，与えられた IRI の設計やスキーマに従った RDF データを作成する工程を中心にデータセット生成に関する研究を進める．

LOD データセットは，人手によるデータ入力で作成されることもあるが，典型的には作成の効率の良さから既存の LOD でないデータセットを機械的に変換することで生成される．これは，既存のシステムやアプリケーションによって管理，生成されたデータセットが存在し，それを LOD として公開したい場合に有用である．例えば，CSV 形式やリレーショナルデータベースなどの構造化されたデータから RDF データへ変換する． CSV 形式やリレーショナルデータベースは表形式で表現されるデータ構造であるが，データ構造に従って意味が適切に表現されているデータであれば，LOD への変換は難しくない．表形式であれば，各列に対応する RDF プロパティを決定し主語の IRI の生成方法を定めることで，行毎に各値を目的語とする単純な RDF トリプルへの変換が実現できる．具体的な変換例を表 1 に示す(学籍番号, 氏名, 年齢)という 3 つの列を持つ表で説明する．主語の IRI は<http://example.com/student/{学籍番号}>というテンプレートに従い生成し，列「氏名」には<http://example.com/ns/name>，列「年齢」には<http://example.com/ns/age>という RDF プロパティを割り当てる．このとき，(1234, 筑波太郎, 20)という行を変換すると，図 1 に示すような 2 つの RDF トリプルが得られる．

表 1　表形式のデータ例

学籍番号	氏名	年齢
1234	筑波太郎	20

```
<http://example.com/student/1234> <http://example.com/ns/name> "筑波太郎" .
<http://example.com/student1234> <http://example.com/ns/age> 20 .
```

図 1　表形式のデータから RDF トリプルへの変換例 (N-Triples 形式)

一方で，Web 上には構造化されていない HTML 文書のような Web ページで記述される情報が多く存在する．Web サイトをデータセットとして見て LOD データセットを生成するには，何らかの方法で LOD データセットとして提供したい実体を Web ページの記述から抽出し，IRI を与えて RDF データへ変換する必要がある．代表的な事例としては， DBpedia[2]がある．DBpedia とは，Wikipedia から構造化情報を抽出し LOD データセットを生成するコミュニティプロジェクトである． DBpedia では Wikipedia の記事中の一部の情報を抽出し，それらを RDF データへ変換している．いくつかの抽出対象が存在するが，主な抽出対象は Infobox である．Infobox とは，記述項目やそのフォーマットを定義したテンプレートであり，記事中に埋め込んで使用される．図 2 は，TV アニメに関する記事で使用される Infobox animanga/TVAnime の使用例であり，そのテンプレートは図 3 のように定義される．

true tears	
ジャンル	恋愛、青春
アニメ	
原作	La'cryma
監督	西村純二
シリーズ構成	岡田麿里
キャラクターデザイン	関口可奈味
音楽	菊地創
アニメーション制作	P.A.WORKS
製作	true tears製作委員会
放送局	放送局参照
放送期間	2008年1月 - 2008年3月
話数	全13話
その他	ハイビジョン制作
	テンプレート - ノート
プロジェクト	アニメ
ポータル	アニメ

図 2　Infobox animanga/TVAnime の使用例[1]

引数 ◆	
タイトル	タイトル title
原作	原作
総監督	総監督
監督	監督 director
シリーズディレクター	シリーズディレクター
シリーズ構成	シリーズ構成
脚本	脚本 writer
キャラクターデザイン	キャラクターデザイン
メカニックデザイン	メカニックデザイン
音楽	音楽 music
アニメーション制作	アニメーション制作 アニメ制作 制作 studio
製作	製作
放送局	放送局 network
放送開始	放送開始 開始 first
放送終了	放送終了 終了 last
話数	話数 episodes
その他	その他
インターネット	インターネット

図 3　Infobox animanga/TVAnime のテンプレート定義[2]

[1] https://ja.wikipedia.org/wiki/True_tears_(%E3%82%A2%E3%83%8B%E3%83%A1)
[2] https://ja.wikipedia.org/wiki/Template:Infobox_animanga/TVAnime

DBpedia は Infobox から抽出したデータを RDF データへ変換する方法として，Infobox の各項目名をそのまま RDF トリプルの述語に割り当てる方法と，各項目に対応する述語を手作業で定義したマッピングを利用する方法を採用している．マッピング定義には DBpedia Mapping Language[10]が用いられ，DBpedia Mappings Wiki[3]上で編集できるようになっている．例えば，図 2 の「監督」は，<http://ja.dbpedia.org/property/監督> という項目名をそのまま用いた述語に変換される．一方，手作業で定義された Infobox animanga/TVAnime のマッピングは，図 4 のように定義されており，「監督」に対するマッピングは，DBpedia オントロジーのプロパティ <http://dbpedia.org/ontology/director> へ対応する．

```
{{TemplateMapping
|mapToClass = Anime
|mappings =
  {{ PropertyMapping | templateProperty = 製作 | ontologyProperty = creator }}
  {{ PropertyMapping | templateProperty = 制作 | ontologyProperty = creator }}
  {{ PropertyMapping | templateProperty = キャラクターデザイン | ontologyProperty =
creator }}
  {{ PropertyMapping | templateProperty = メカニックデザイン | ontologyProperty =
creator }}
  {{ PropertyMapping | templateProperty = 放送局 | ontologyProperty = publisher  }}
  {{ PropertyMapping | templateProperty = アニメーション制作  | ontologyProperty =
productionCompany  }}
  {{ PropertyMapping | templateProperty = 監督 | ontologyProperty = director }}
  {{ PropertyMapping | templateProperty = 総監督 | ontologyProperty = director }}
  {{ PropertyMapping | templateProperty = シリーズディレクター | ontologyProperty =
director }}
  {{ PropertyMapping | templateProperty = 原作  | ontologyProperty = writer  }}
  {{ PropertyMapping | templateProperty = 脚本 | ontologyProperty = writer }}
  {{ PropertyMapping | templateProperty = タイトル | ontologyProperty = foaf:name  }}
  {{ PropertyMapping | templateProperty = 音楽 | ontologyProperty = musicComposer }}
}}
```

図 4　Infobox animanga/TVAnime の DBpedia マッピング定義[4]

DBpedia の Wikipedia 記事から Infobox を抽出するようなアプローチは，Wikipedia だけでなく，共通のテンプレートを用いて記述された情報を持つ他の Web サイトに対しても適用できると考えられる．DBpedia が対象とする Wikipedia は，幅広い領域に関する情報を総合的に扱った百科事典サイトであるが，Web 上には共通の対象に関する情報が複数の Web サイトに分散して記述されている．

[3] http://mappings.dbpedia.org
[4] http://mappings.dbpedia.org/index.php/Mapping_ja:Infobox_animanga/TVAnime

ある領域に関する LOD データセットを作ろうと考えたときに，その領域に関する記述を持つ Web サイトから LOD データセットを生成するという考えがある．さらに，分散して記述されることを生かし，複数の Web サイトから共通の対象に関する記述を抽出し寄せ集めることでデータセットを生成するという考えがある．なぜならば，同じ実体に関する記述であっても，Web サイトによって異なる記述を持っている可能性があり，それらを統合することでより豊富な情報を持つデータセットが生成されることが期待できるからである．あるいは，同じ実体に関する記述を比較することで，情報の確度を検証しながらデータセットを生成することにも利用できる．個々の Web サイトの記述が不十分であっても，それらを統合して生成された実体に IRI を与え RDF データに変換することで，LOD データセットを提供したいというシナリオにおいて，特に有用であると考える．本研究では，このような複数の Web サイトに記述された情報の抽出および統合することによる LOD データセット生成に取り組む．

複数の Web サイトから抽出したデータを統合するためには，解決すべき課題があり，次節で詳しく述べる．

2.3. データセットの統合におけるインスタンスの同一性の解決

単一の Web サイトから LOD データセットを生成するのではなく，異なる複数の Web サイトから LOD データセットを生成する場合には，単純な RDF データへの変換だけでは解決できない問題がある．それは，本来同一である記述対象に対して異なる識別子が与えられてしまう問題である．単に個別の Web サイトから抽出したデータを RDF データへ変換し，それらを併合するだけでは，本来同じ記述対象を示す記述であっても異なる IRI が与えられてしまうからである．つまり，複数の Web サイトに含まれる同一の実体に関する記述を識別し，それらの記述を LOD データセットの 1 つのインスタンスとして生成する必要がある．なお，この処理を抽出データの統合と呼ぶことにする．

異なるデータセットの間で同一の実体を示すインスタンスを発見する問題は，インスタンスマッチング(instance matching)[5]と呼ばれる．また，entity resolution[1]や duplicate detection[8]としても知られている．同じ実体に関する記述であっても，異なるデータセットでは表記のゆれや記述の部分的な誤りなどの差異が存在する場合が考えられる．共通した識別子が与えられていない限りヒューリスティックな方法で識別する必要があり，同一の対象に対する記述の差異を考慮した様々な手法が提案されている．代表的な手法としては，インスタンス間の類似度を算出し，マッチングを行う手法である．

インスタンスマッチングは，LOD データセット間のリンクを生成する場合でもよく

用いられる．典型的には，異なる複数のデータセット間で同一の実体を示すインスタンスに対し，同一であることを意味する owl:sameAs の関係を発見し，リンクを生成するために用いられる．このようなリンク生成のためのインスタンスマッチングを実行するツールも存在する．

一方で，本研究ではリンク生成が目的ではなく，同一の実体に関するデータを統合した結果を LOD データセットの 1 つのインスタンスとして生成するというケースを扱う．これを実現するには，インスタンスマッチングを実行して識別されたデータの集合を統合し，RDF インスタンスへ変換する必要がある．この RDF データへの変換においては，異なる複数の項目から一つの述語へ対応させるような多対一のマッピングと，それらが持つ目的語となる値をどのように扱うか考えねばならない．

例えば，映画作品に関する情報を提供する 2 つの Web サイトから抽出されたデータセット A, B があるとする．映画作品が公開された日付を，データセット A は「公開日」，データセット B では「上映開始日」という項目に記述していたとする．これらを一つの RDF プロパティ <http://example.com/terms/publishedAt> にマッピングするような場合である．このとき，データセット A，B が持つ同一の映画作品のデータにおいて，映画が公開された日付が異なって記述されていたとき，どのように値を目的語にマッピングするかを指定できる必要がある．

3. 関連研究

3.1. RDF マッピング記述言語

Web ページから抽出されたデータを RDF データへ変換するにあたり，DBpedia の事例では DBpedia Mapping Language というマッピング記述言語が用いられている．より汎用の RDF マッピング記述言語も数多く提案されている．

D2RQ[3]は，リレーショナルデータベースから RDF への動的な変換を実現するプラットフォームである．D2RQ では，リレーショナルデータベースと RDF 間のマッピングを宣言的に記述するための言語 D2RQ Mapping Language を定義している．R2RML[19]も同様に，リレーショナルデータベースから RDF へのマッピングを記述するための言語であり，W3C 勧告である．RML[7]は，R2RML を拡張して提案されたマッピング言語である．RML は，入力フォーマットの定義を一般化することで，リレーショナルデータベースだけでなく CSV や XML などの多様なフォーマットから RDF へのマッピングを記述できる．

また，宣言的な言語ではなく，問い合わせ言語を拡張して RDF へのマッピングとして用いる例もある．TARQL[12]や SPARQL-Generate[11]は，RDF 問い合わせ言語である SPARQL[23]を拡張した言語である．TARQL は CSV 形式のデータへ問合せし，問合せ結果を SPARQL の COUNTRUCT 句を用いて RDF トリプルを生成することができる．SPARQL-Generate は，XML，JSON，CSV，HTML など多様な入力フォーマットに対応し，問い合わせ結果を RDF トリプルとして得られる．

3.2. インスタンスマッチング

LOD データセットに対してインスタンスマッチングを行うツールとして Silk Linking Framework[9, 15]がある．SPARQL エンドポイントでアクセス可能なデータセットに対して，異なるデータセットに含まれるインスタンス同士で類似度を算出し，同一と考えられるインスタンスを発見する．また，類似度を算出する規則を宣言的に記述できる Silk Link Specification Language(Silk-LSL)[17]を定義している．図 5 は Silk-LSL によるインスタンスマッチングの規則の記述例である．規則は，マッチングの対象となるデータセットの SPARQL エンドポイントの指定，マッチングに用いる類似度を算出する方法の指定，出力形式の指定，から構成される．類似度の算出においては，同一のインスタンスと判定するためのしきい値をユーザが定義する必要があり，マッチングの精度はユ

ーザの作成する類似度算出の規則へ大きく依存する.

```
<Silk>
  <Prefixes>
    <Prefix id="rdfs" namespace="http://www.w3.org/2000/01/rdf-schema#" />
    <Prefix id="dbpedia" namespace="http://dbpedia.org/ontology/" />
    <Prefix id="gn" namespace="http://www.geonames.org/ontology#" />
  </Prefixes>
  <DataSources>
    <DataSource id="dbpedia">
      <Param name="endpointURI" value="http://demo_sparql_server1/sparql" />
      <Param name="graph" value="http://dbpedia.org" />
    </DataSource>

    <DataSource id="geonames">
      <Param name="endpointURI" value="http://demo_sparql_server2/sparql" />
      <Param name="graph" value="http://sws.geonames.org/" />
    </DataSource>
  </DataSources>
  [<Blocking blocks="100" />]
  <Interlinks>
    <Interlink id="cities">
      <LinkType>owl:sameAs</LinkType>

      <SourceDataset dataSource="dbpedia" var="a">
        <RestrictTo>
          ?a rdf:type dbpedia:City
        </RestrictTo>
      </SourceDataset>

      <TargetDataset dataSource="geonames" var="b">
        <RestrictTo>
          ?b rdf:type gn:P
        </RestrictTo>
      </TargetDataset>

      <LinkageRule>
        <Aggregate type="average">
          <Compare metric="levenshteinDistance" threshold="1">
            <Input path="?a/rdfs:label" />
            <Input path="?b/gn:name" />
          </Compare>
          <Compare metric="num" threshold="1000" >
            <Input path="?a/dbpedia:populationTotal" />
            <Input path="?b/gn:population" />
          </Compare>
        </Aggregate>
      </LinkageRule>
      <Filter limit="1" />
      <Outputs>
        <Output type="file" minConfidence="0.95">
          <Param name="file" value="accepted_links.nt" />
          <Param name="format" value="ntriples" />
        </Output>
        <Output type="file" maxConfidence="0.95">
          <Param name="file" value="verify_links.nt" />
          <Param name="format" value="alignment" />
        </Output>
      </Outputs>
    </Interlink>
  </Interlinks>
</Silk>
```

入力データセットの指定

類似度算出の指定

出力形式の指定

図 5　Silk-LSL によるインスタンスマッチング規則の記述例

3.3. LOD データセット作成のためのフレームワーク

本研究では，Web ページを対象としたデータ抽出から RDF データへの変換までの一連の処理を担うツールチェーンが求められる．関連するツールとして OpenRefine[18]がある．Web ページのスクレイピングやデータの整形，クレンジング，RDF への変換などの一連の作業を行うためのツールである．OpenRefine は，外部のデータセットとテキストベースのマッチングを実行して識別子を照合する，一種のインスタンスマッチングを機能として提供している（OpenRefine ではこれを reconciliation と呼んでいる）．しかし，Reconciliation Service API[5]に対応しているデータセットに対してしか実行することができない．また，OpenRefine はユーザが GUI を通じてデータを操作することを基本としており，定型的な処理を記述しておいて実行するような仕組みは十分でない．

[5] https://github.com/OpenRefine/OpenRefine/wiki/Reconciliation-Service-API

4. 宣言的記述を用いた複数のWebサイトからのLODデータセット生成

本研究では，第 2 章で述べたような，複数の Web サイトの記述からデータを抽出および統合し，LOD データセットを生成するケースを想定し，次の要件を定める．

1. ユーザが指定した Web ページ中の半構造化データを抽出できること
2. 抽出データの集合から記述対象が同一であると考えられる抽出データを識別できること
3. 同一であると識別された抽出データの集合から RDF データへ変換できること

要件 1 は，ユーザによる指定に従って，LOD データセットへ変換する対象となる半構造化データを Web ページから抽出することである．ここで述べる「半構造化データ」とは，抽出対象となる複数の Web ページで共通する文書構造を持つ領域（テンプレート）に記述されたデータを指す．典型的には HTML 文書における表(table 要素)やリスト(ol, li 要素)などの要素で記述される．

要件 2 は，複数の Web サイトの記述から抽出されたデータの集合に対して，記述対象が同一であるデータを識別できるようにすることである．共通の識別子が与えられていない Web ページ中の記述に対し，同一性を判定する必要がある．

要件 3 は，記述対象が同一であると判定された抽出データの集合に対し，ユーザが定義したマッピング定義に従って RDF データへ変換できるようにすることである．具体的には，異なるデータソース（Web サイト）から抽出された複数の抽出データを，RDF トリプルに割り当てる処理が必要になる．

また，Web サイトのコンテンツは更新される可能性があり，更新されたデータを生成 LOD データセットに反映するためには，この生成手続きを一度きりでなく反復して実行することが求められる．生成を静的に実行するのではなく，LOD データセットが参照されたときに動的に生成するラッパーとして実現する方法も考えられるが，データソースが Web ページであると処理時間やデータの履歴の保証の問題，対象 Web ページが自身の管理下にない場合はサーバ負荷などの問題が存在し，実現が容易ではない．従って，一連の生成手続きは可能な限り自動化することが望ましい．

本研究では，これらの要件を満たすように，LOD データセット生成を自動化するための宣言的に記述する規則を提案する．また，その規則を利用することで LOD データセットを生成する手法を提案する．宣言的な規則を導入することによって，個別のデータソースに依存する処理内容と共通する生成手続きを分離することができる．データソース毎に個別の生成プログラムを記述および管理する必要がなくなり，生成処理の自動

化に寄与すると考えられる．また，Web ページからのデータ抽出やインスタンスマッチングの具体的な手法について知識のないユーザであっても，規則を記述することで生成処理が実行可能である．

提案手法による LOD データセットの生成手続きを図 6 に示す．生成手続きは次の 3 つの処理に分けられ，ユーザは各処理で用いる 3 種類の規則を作成する．それぞれの規則の詳細は次章で述べる．

1. Web ページからの半構造化データの抽出
2. 抽出データに対するインスタンスマッチング
3. 抽出データの統合および RDF データへの変換

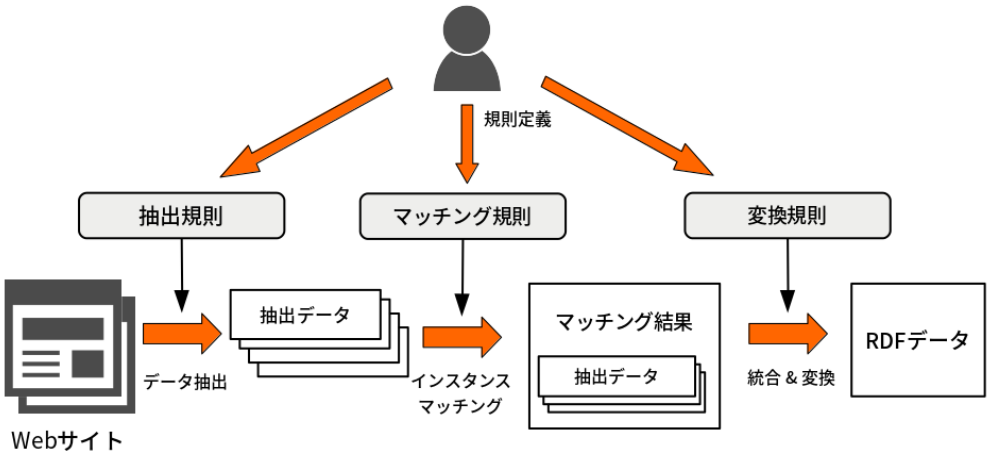

図 6　データセット生成手続きの概要

LOD データセットの生成手続きは，まず，LOD データセットを生成するためのデータソースとなる Web サイトを入力とし，抽出規則に従って各 Web ページから半構造化データを抽出する（要件 1）．抽出されたデータに対し，マッチング規則に基づいてインスタンスマッチングを実行する（要件 2）．インスタンスマッチングを実行した結果，識別された抽出データを変換規則に従って RDF データへ統合および変換する（要件 3）．この一連の手続きによって Web サイトからの LOD データセット生成が実現される．

5. LOD データセット生成の自動化のための宣言的記述の提案

5.1. 半構造化データの抽出規則

4 章で述べた要件 1 を満たすために，HTML で記述された Web ページから半構造化データを抽出するための規則を定義する．本抽出規則は，Web ページが与えられたときに，その Web ページ中に Key-Value 型の構造でデータが記述されていると仮定し，それらを抽出するための規則を定める．

データ抽出の規則として，抽出対象のデータが記述された Web ページ中の要素を指定するための XPath[16]式や CSS selector[22]を Key-Value の組で定義する．また，抽出処理は 1 ページに対してだけではなく，複数の Web ページにまたがる場合が考えられる．例えば，検索結果ページが 1 ページ目，2 ページ目，…と複数ページに分割されている場合（ページネーション）である．このような構造を持つページに対しては，「次のページ」を示すリンクを抽出するための規則を別途定義しておき，そのリンク先のページに対して再帰的に抽出処理を実行する．

抽出規則は JSON 形式で記述され，記述例を図 7 に示す．この記述例では，elements プロパティで 3 組の Key-Value 組の抽出を定義している．elements プロパティの配列の 1 番目の要素は，CSS selector「`thead > tr > .fst`」で指定される要素を key として，CSS selector「`tbody > tr > .fst`」で指定される要素を value として抽出することを意味している．また，recursiveCrawling プロパティでは，次のページを示すリンクを抽出する規則を定義している．CSS Selector「`#new_asf > ul > li.pageSkip > ul > li.nxt > a`」で指定される要素の「href」属性を次のページのリンクとすることを意味している．

```
{
   "ruleID":"MADB-AnimeEpisodes",
   "elements":[
      {
         "property":"thead > tr > .fst",
         "value":"tbody > tr > .fst",
         "literalProperty":false,
         "valueAttribute":null
      },
      {
         "property":"thead > tr > :nth-child(2)",
         "value":"tbody > tr > :nth-child(2)",
         "literalProperty":false,
         "valueAttribute":null
      },
      {
         "property":"thead > tr > :nth-child(3)",
         "value":"tbody > tr > :nth-child(3)",
         "literalProperty":false,
         "valueAttribute":null
      }
   ],
   "multiEntity":true,
   "recursiveCrawling":{
      "selector":"#new_asf > ul > li.pageSkip > ul > li.nxt >
a",
      "attribute":"href"
   }
}
```

図 7　Web ページ中の半構造化データ(table)に対する抽出規則の記述例

5.2. 抽出データのインスタンスマッチング規則

4 章で述べた要件 2 を満たすために，抽出規則に基づいて抽出されたデータに対してインスタンスマッチングを実行するための規則を定義する．

インスタンスマッチングには様々な手法が存在する．Silk[9, 15]はインスタンス間の値を比較して類似度を算出するアプローチでインスタンスマッチングを行うツールであり，マッチングの規則を宣言的に記述するための言語 Silk-LSL を提案していた．

本提案手法では，インスタンスマッチングの手法には，Shu らによる機械学習を用いたインスタンスマッチングの研究[13]を参考に，決定木学習を用いた手法を採用する．2 つのインスタンスの組の類似度ベクトルが入力されたときに，それらが同一であるか

否かを 0/1 で出力する分類木をインスタンスマッチングに用いる．これによりユーザは手動でしきい値を設定する代わりに，訓練用データとして正解となるインスタンスの組を少数作成すればよい．インスタンスマッチング実行時にはそれらをもとに決定木を訓練する．また，類似度ベクトルの生成にあたっては，Silk-LSL をより簡略化したマッチング規則を定義し，類似度の算出方法を規則として記述できるようにする．

マッチング規則は大きく 3 つの要素で構成される．マッチング対象の抽出データセットの指定，類似度を算出するための比較項目およびその評価尺度，決定木学習に用いる訓練用データの 3 つである．

インスタンスマッチング規則は JSON 形式で記述され，図 8 に記述例を示す．datasets プロパティでは，マッチング対象の 2 つのデータセットを定義している．trainingData プロパティは訓練用データを定義し，dataset プロパティで指定したデータセット間の正しいマッチングの組を記述している．matchingElements プロパティでは，類似度を算出するための比較項目とその尺度を定義している．例えば matchingElements の 1 番目の要素は，抽出データセット MADB-AnimeEpisode の「各話タイトル」プロパティと抽出データセット Syobo-Episodes の「SubTitle」プロパティの値を評価尺度「levenshtein」で評価した結果を類似度とする，という定義である．なお，評価尺度「levenshtein」とはレーベンシュタイン距離を意味し，より長い方の文字列の文字数で割り，正規化した値を類似度とする．matchingElements の 2 番目の要素は，抽出データセット MADB-AnimeEpisode の「各話表記」プロパティと抽出データセット Syobo-Episode の「Number」プロパティの値を評価尺度「inclusion」で評価した結果を類似度とする，という定義である．評価尺度「inclusion」とは，比較対象の 2 つの値を文字列として見たときに，包含関係があるか否かを 0/1 とする尺度である．

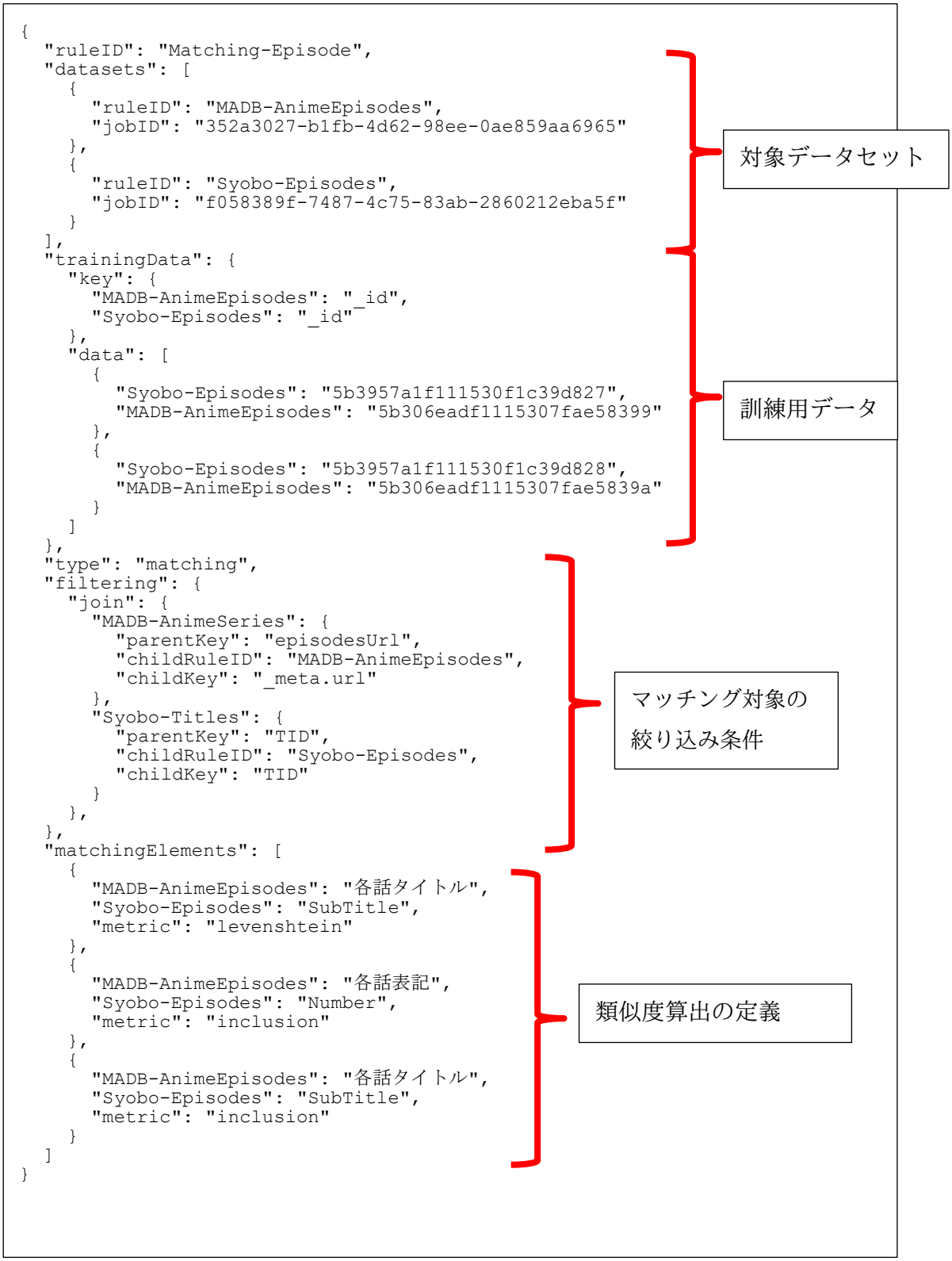

図 8　インスタンスマッチング規則の記述例

5.3. 抽出データの統合および RDF データへの変換規則

4 章で述べた要件 3 を満たすために，インスタンスマッチングにより同一であると識別されたインスタンスの集合から，RDF データへの変換を定義するための変換規則を定義する．本変換規則は RDF マッピング言語 RML を拡張することで定義する．拡張の要点は，複数のデータソースを変換対象として定義できること，および，複数のデータソースから値を参照する際の条件を定義できることである．

以下では，表 2 に示す名前空間と接頭辞の定義に従い説明する．接頭辞 rr:は，R2RML，接頭辞 rml:は RML において定義される語彙の名前空間である．本提案で新たに定義する語彙は，名前空間 https://mdlab.slis.tsukuba.ac.jp/ns/rmle/#，接頭辞 rmle:を用いることにする．

表 2　名前空間と接頭辞の定義

接頭辞	名前空間 IRI
rr:	http://www.w3.org/ns/r2rml#
rml:	http://semweb.mmlab.be/ns/rml#
rmle:	https://mdlab.slis.tsukuba.ac.jp/ns/rmle/#

RML において，変換対象となるデータソースは rml:LogicalSource クラスのインスタンスとして定義される．rml:LogicalSource は，データソースの IRI を示す rml:source プロパティ，データへアクセスする形式を示す rml:referenceFormulation プロパティ，イテレータを示す rml:iterator プロパティを持つ．

インスタンスマッチングによって識別されたデータを変換対象のデータソースとして扱うために，rml:LogicalSource クラスのサブクラスとして rmle:IdentifiedSource クラスを新たに定義する．rmle:IdentifiedSource は 0 または 1 個の rmle:priorityValue プロパティを持つ．rmle:priorityValue プロパティは整数の値を持ち，大きいほどデータソースの優先度が高いことを意味する．図 9 は，rmle:IdentifiedSource のインスタンスの定義例であり，#MADB と#Syobo という 2 つのデータソースが定義されている．

RML において，rml:LogicalSource は， RDF トリプルへの変換規則を定義するための rr:TripleMap クラスが持つ 1 個の rml:logicalSource プロパティから参照される．インスタンスマッチングが実行されるデータセットは必ず 2 つ以上であるので，rmle:IdentifiedSource のインスタンスを参照するための rmle:indentifiedSource プロパティを新たに定義する．また，rr:TripleMap は，0 個または少なくとも 2 個の

rmle:indentifiedSource プロパティを持つように定義を拡張する．

rr:TripleMap によって定義された変換規則が実行される際には，一回の変換処理の反復で，rml:logicalSource に指定されたデータソースから，変換対象となる 1 個のデータが取得される．新たに定義した rmle:identifiedSource を用いる場合は，1 回の変換処理の反復で，rmle:identifiedSource に指定された複数のデータソースから，同一であると識別された 1 個以上のデータが取得されることになる．

```
<#MADB>
    a rmle:IdentifiedSource ;
    rdfs:label "Media Arts Database" ;
    rml:source <file://data/MADB-AnimeSeries.json> ;
    rml:referenceFormulation ql:JSONPath ;
    rmle:priorityValue 0
    .
<#Syobo>
    a rmle:IdentifiedSource ;
    rdfs:label "Syoboi Calendar" ;
    rml:source <file://data/Syobo-Titles.json> ;
    rml:referenceFormulation ql:JSONPath ;
    rmle:priorityValue 1
    .
```

図 9　Identified source の定義の記述例

次に，主語や目的語の変換規則において，特定の rmle:identifiedSource から取得した値を変換対象と指定するためのプロパティを定義する．RML では，主語の変換規則を rr:SubjectMap クラス，目的語の変換規則を rr:ObjectMap クラスを用いて記述する．そこで，変換元のデータソースを指定する rmle:mappedSource プロパティを新たに定義し，rr:SubjectMap および rr:ObjectMap クラスは 0 個または 1 個の rmle:mappedSource プロパティを持つように定義を拡張する．rmle:mappedSource プロパティで指定された rmle:IdentifiedSource が値の参照に用いられることを意味する．

また，目的語の変換規則では，rmle:mappedSource プロパティで特定のデータソースを指定せずに，複数の rmle:identifiedSource から取得した値をある条件（マージ条件）に基づいて変換する方法も定義する．本提案においては，次の条件を定義した．

- rmle:all:
  全てのデータソースから取得した値を目的語に変換する
- rmle:priority:
  優先度の高いデータソースから取得した値を目的語に変換する
  (優先度は rmle:priorityValue を参照)

- rmle:equality:
  全てのデータソースから取得した値が等しければ,目的語へ変換する
- rmle:majority:
  データソースから取得した値から多数決ととり,目的語へ変換する.

このマージ条件は,述語-目的語の組の変換規則を定義する rr:PredicateObjectMap クラスにおいて,rmle:condition プロパティを新たに定義することにより指定する.

図 10 は，データソースに rmle:IdentifiedSource を用い，インスタンスマッチング結果のマージ条件を含む変換規則の例である．この変換規則では，インスタンスマッチングにより識別された 2 つのデータソース#MADB と#Syobo を参照している. 主語の変換規則を定義した rr:subjectMap プロパティを見ると，rmle:mappedSource プロパティで指定されたデータソース#MADB から値を取得し，IRI のテンプレート「http://example.moe/anime/{アニメシリーズ ID}」に基づいて生成された IRI が主語となる. 具体的には, データソース#MADB の項目「アニメシリーズ ID」の値が「ANS0001」であったとすると，「http://example.moe/anime/ANS0001」が主語の IRI となる.

述語-目的語の変換規則を定義した rr:predicateObjectMap プロパティを見ると, 1 つの述語に対して 2 つのデータソースに対する目的語の変換規則が定義されている．1 つ目の目的語の変換規則は，データソース#MADB の JSONPath「$.タイトル」を評価して得られた値を言語タグ ja 付きのリテラルとする変換規則である. 2 つ目の目的語の変換規則は，データソース#Syobo の JSONPath「$.TItleEN」を評価して得られた値を言語タグ en 付きのリテラルとする変換規則である. さらに rmle:condition プロパティでマージ条件 rmle:all が指定されているので，これら 2 つとも述語 dcterms:title の目的語として変換される.

```
<#AnimeMapping>
    a rr:TriplesMap;

    rmle:identifiedSource <#MADB>;
    rmle:identifiedSource <#Syobo>;

    rr:subjectMap [
        rr:template "http://example.moe/anime/{アニメシリーズID}";
        rr:class pcmm:Anime;
        rmle:mappedSource <#MADB>;
    ];

    rr:predicateObjectMap [
        rr:predicate dcterms:title;
        rr:objectMap [
            rmle:mappedSource <#MADB>;
            rml:reference "$.タイトル";
            rr:language "ja"
        ];
        rr:objectMap [
            rmle:mappedSource <#Syobo>;
            rml:reference "$.TitleEN";
            rr:language "en"
        ];
        rmle:condition rmle:all
    ]
    .
```

図 10　マージ条件を含む変換規則の記述例

6. LOD データセット生成の自動化システムの実装

6.1. システムの構成

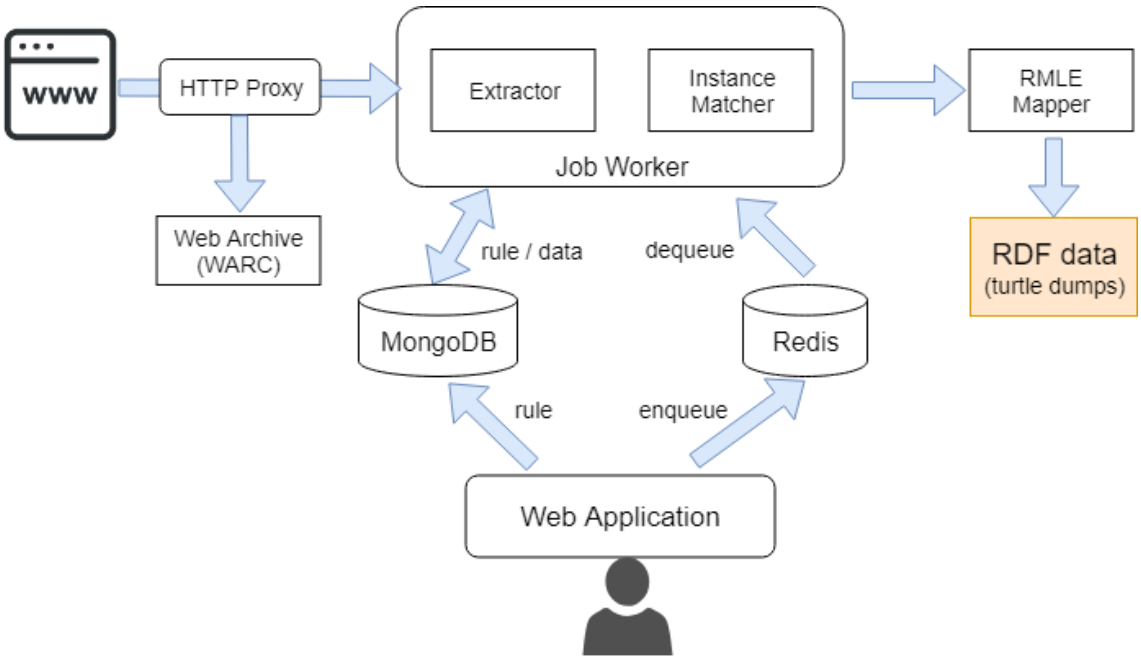

図 11　システム構成図

前章で述べた各規則を処理し，LOD データセットを生成するシステムについて説明する．図 11 は，提案手法を実装したシステムの構成図である．

システムのユーザは，Web アプリケーションを通じてデータ抽出規則やインスタンスマッチング規則を作成および登録する．Web ページからのデータ抽出処理や抽出データのインスタンスマッチング処理は，対象とする数が多いと処理に時間がかかるため，データ構造サーバ Redis[6]を用いてジョブキューで管理する．ユーザは Web アプリケーションを通じて各種処理をキューに登録し，ジョブワーカーによって処理が実行される．ユーザにより登録された規則定義や，処理の過程で生成されるデータは MongoDB[7]へ保存される．

[6] https://redis.io/
[7] https://www.mongodb.com/

6.2. 抽出規則の作成と抽出の実行

本システムでは，ユーザによる抽出規則の作成を容易にするためのツールを提供している．本ツールは Web ブラウザの拡張機能として実行される．ユーザは抽出対象の Web ページを表示しながら，Key-Value の順で抽出したい箇所をクリックすることで，当該箇所に対する抽出規則が自動生成される．このツールは，ユーザがクリックした HTML 要素を指し示す CSS Selector を算出し，抽出規則を作成する．

抽出処理プログラムは，抽出対象の HTML 文書と抽出規則を入力とし，抽出規則を実行することで得られた Key-Value 型の抽出データを JSON 形式で出力する．

図 12　Web ページに対する抽出規則作成ツールの使用例

6.3. インスタンスマッチングの実行

インスタンスマッチングの実行に必要なマッチング規則は，Web インタフェースを用いて作成できる．ユーザがマッチング対象とする抽出データセットを選択すると，類似度算出のための項目が自動的に取得される．項目のリストから比較対象としたい項目と，その評価尺度を選択し，一つの類似度を算出する規則を定義する．

図 13 はインスタンスマッチング規則の作成画面の例である．MADB-AnimeSeris と Syobo-Titles という名前の 2 つのデータセット間のインスタンスマッチング規則を定義しており，6 つの項目間の類似度を算出する規則が定義されている．

図 14 はインスタンスマッチングに用いる決定木の訓練用データの入力画面である．ユーザは正しいマッチングのデータの組を手動で作成し，そのデータ ID の組をシステムへ登録する．インスタンスマッチング実行時には，これらのデータを元に決定木が構成され，インスタンスマッチングに用いられる．

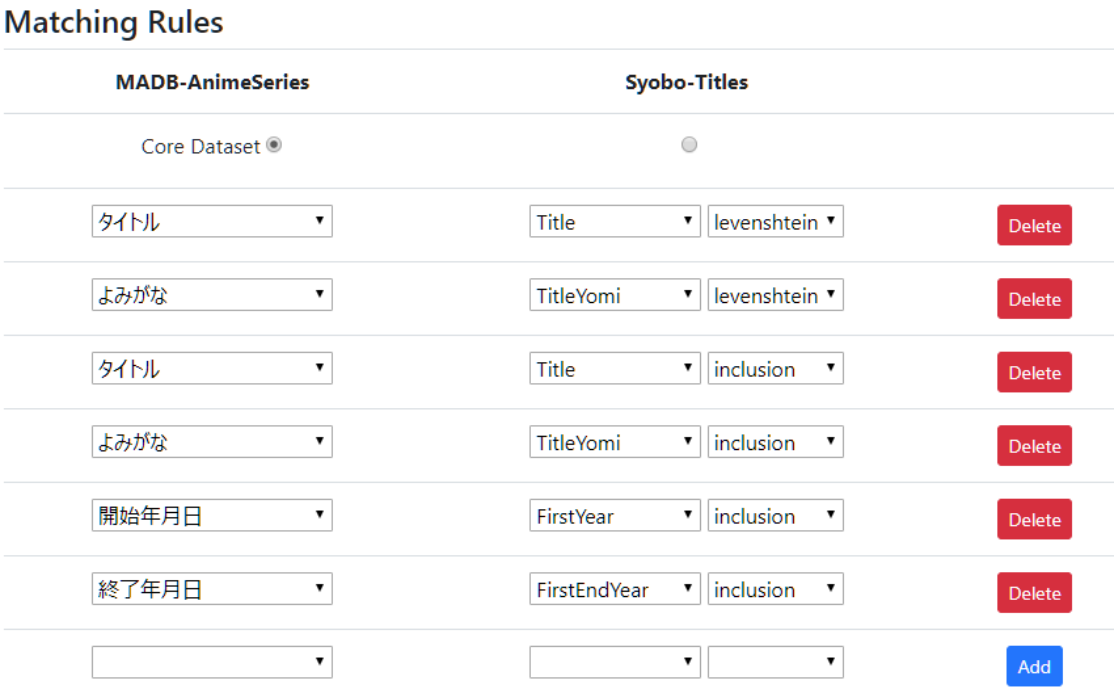

図 13　インスタンスマッチング規則の作成画面

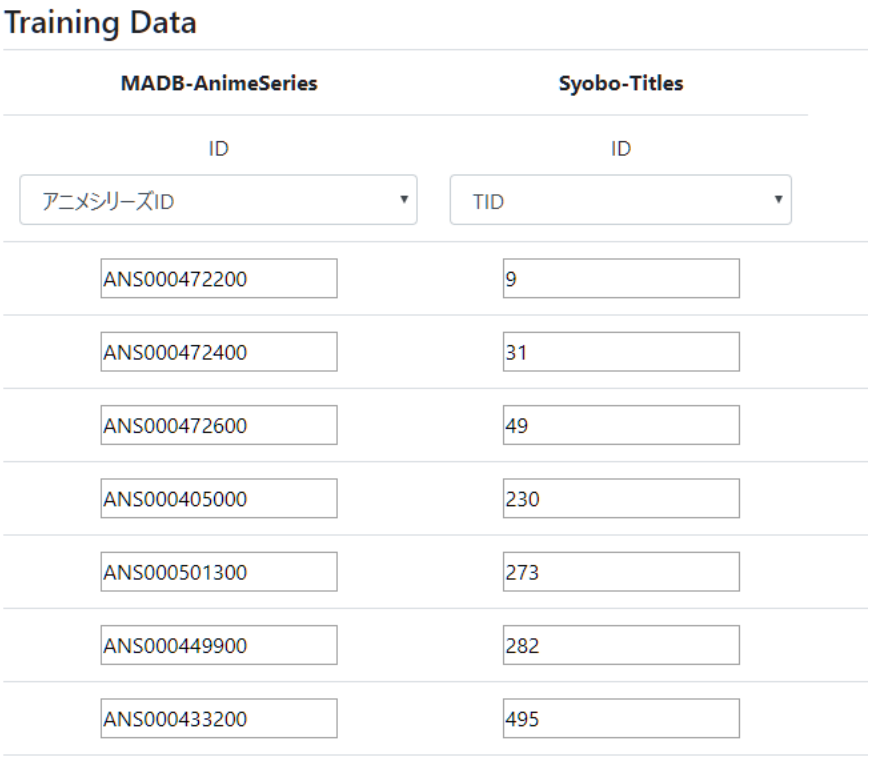

図 14　インスタンスマッチングの訓練用データの入力画面

6.4. 統合および RDF への変換

RML はマッピングプロセッサの実装の一つである RML-Mapper[8]が公開されているが，前章で述べた仕様を拡張するためにプログラミング言語 Python を用いて独自にマッピングプロセッサを実装した．RDF データの操作には，Python の RDF ライブラリである RDFLib[9]を利用した．

本マッピングプロセッサは，Turtle 形式で記述した変換規則と JSON 形式のデータファイルを入力し，変換規則を実行することで得られた RDF データを Turtle 形式で出力するプログラムである．

6.5. データセット生成の履歴管理

一連のデータセット生成において，生成されるデータセットの来歴を保証するために，

[8] https://github.com/RMLio/RML-Mapper
[9] https://github.com/RDFLib/rdflib

各処理結果や各規則をシステム上で保存している．Web ページからデータ抽出する段階では，透過的に通信内容を WARC(Web ARChive)形式[25]で保存できる HTTP プロキシ Warcprox[10]を用いて，抽出処理時点の Web ページが保存される．抽出結果やインスタンスマッチングの結果は，実行したジョブ毎に MongoDB に保存される．

システムの仕様に変更がないと仮定すれば，保存しておいた WARC ファイルを再生し，同じ規則を用いて生成処理を実行すれば，生成される LOD データセットの再現性は保証される．

[10] https://github.com/internetarchive/warcprox

7. 実際の Web サイトを用いた LOD データセット生成による評価実験

提案システムを用いて実際の Web サイトを対象とした LOD データセットの生成を実行し，提案手法を評価する．本実験は 3 つの異なる Web サイトを情報源とする．これらを対象に各種規則を作成し，規則に基づいて LOD データセットを生成する．

7.1. 対象 Web サイトと生成するデータセットの概要

本実験では，LOD データセットの生成事例として，アニメ作品に関する LOD データセットを作成した．データソースとなる Web サイトは，メディア芸術データベース[11]，しょぼいカレンダー[12]，アニメハック[13]の 3 つを選定した．

メディア芸術データベースは，文化庁メディア芸術デジタルアーカイブ事業によって提供されている Web サイトである．マンガ・アニメ・ビデオゲーム・メディアアートの 4 分野の作品の制作・流通などに関する基本情報を提供している．なお，本実験ではアニメ作品の情報だけを用いる．しょぼいカレンダーは，アニメや特撮などの TV 番組に関する情報を提供している Web サイトである．ユーザによってデータが作成・整備されている．アニメハックは，株式会社エイガ・ドット・コムが運営する Web サイトで，アニメ作品のスタッフやキャストなどの基礎的情報からアニメや声優に関連するイベント情報などを提供している．

これらのサイトが提供する情報の中から，図 15 に示す 4 種の実体を定義し，各 Web サイトから抽出する対象とした．実体間を結ぶ実線は，それらの実体が記述されている Web ページがリンクで結ばれていることを意味する．メディア芸術データベースからは，AnimeEdition，AnimeEpisode を抽出し，しょぼいカレンダーはメディア芸術データベースの持つ実体に加えて Broadcast を抽出する．アニメハックからは AnimeEdition，Event を抽出する．

[11] https://mediaarts-db.bunka.go.jp/
[12] http://cal.syoboi.jp/
[13] https://anime.eiga.com/

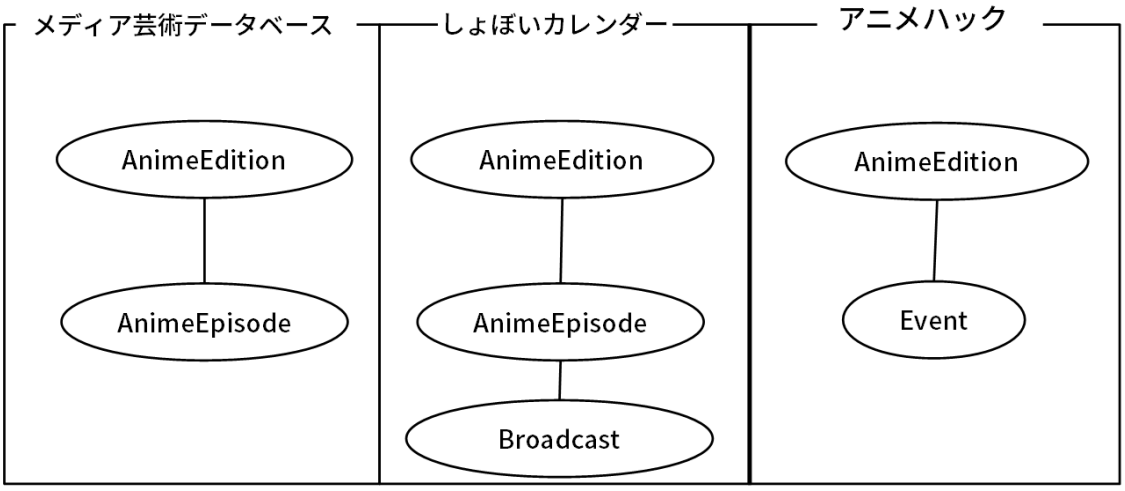

図 15　各 Web サイトから取得対象となるデータ実体

7.2. データセット生成のためのルール作成

7.2.1. 抽出規則の作成

各サイトから，前節で述べた各実体を抽出するために作成した規則について説明する．メディア芸術データベースに対する抽出規則は，以下の 3 つ作成した．

i. 作品情報（シリーズ）ページのリストを取得する
ii. (i)の抽出結果を利用して,作品情報（シリーズ）ページから AnimeEdition を取得する
iii. (ii)の抽出結果を利用して,各話情報一覧ページから AnimeEpisode を取得する

(i)の規則は，図 16 に示すような検索結果ページに表示される表から，作品情報（シリーズ）ページへのリンクを抽出するために作成した．(ii)の規則は，(i)で得られたリンク先 URL を利用して，図 17 に示すような作品情報（シリーズ）ページ内の表（赤枠内）に記述される情報を AnimeEdition とし，抽出する．(iii)の規則は，(ii)から得られた各話情報一覧ページへのリンク（図 17 青枠内）先のページを対象とし，図 18 のようなページ内の表（赤枠内）に記述された情報を AnimeEpisode として抽出する．

また，(i)と(iii)の抽出対象ページは，複数ページに分割されている場合があるため，「次のページ」へのリンクを取得し，再帰的に抽出を実行するための規則を含む．

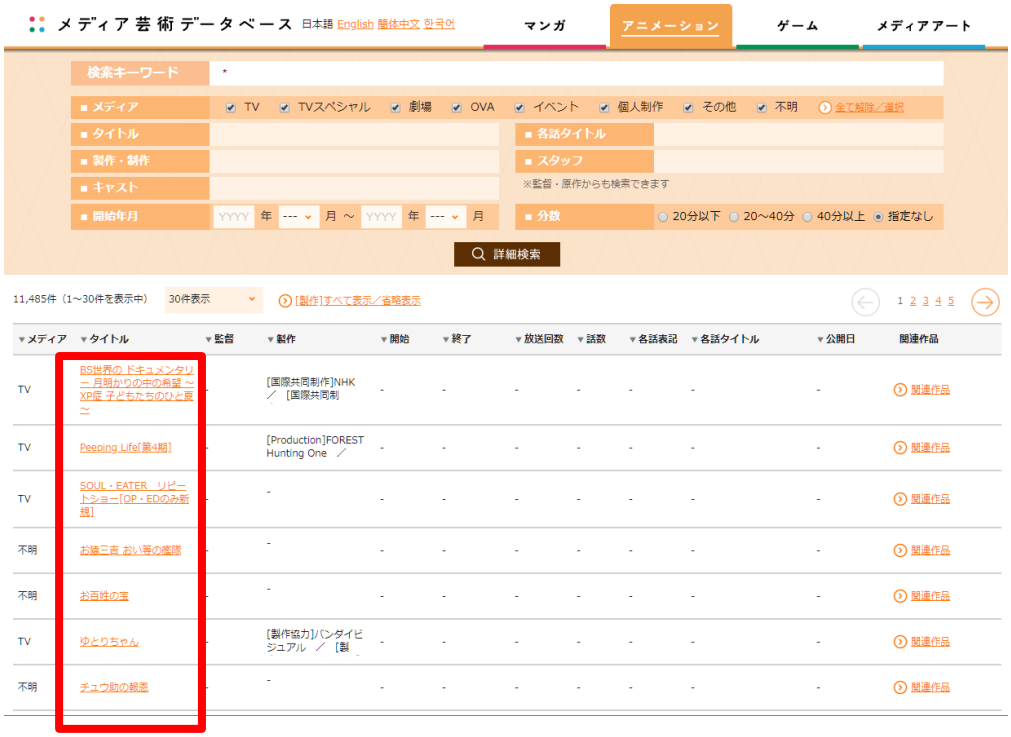

図 16　メディア芸術データベース アニメ分野検索結果ページ

図 17　メディア芸術データベース アニメ作品情報（シリーズ）ページ

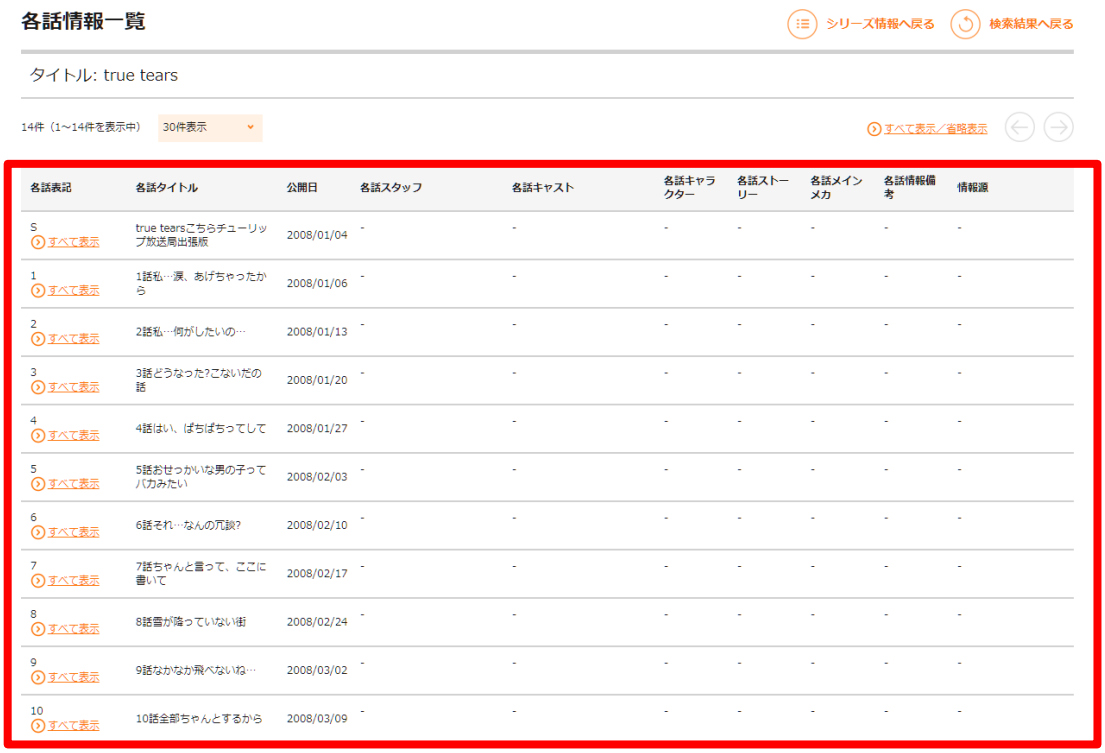

図 18　メディア芸術データベース　アニメ各話情報一覧ページ

メディア芸術データベースの(i)～(iii)の抽出規則の具体的な記述は付録に添付しておく．抽出規則を実行してメディア芸術データベースから抽出された AnimeEdition と AnimeEpisode の抽出データ（JSON 形式）の例を図 19 と図 20 に示す．

```
[
  {
    "episodesUrl": "https://mediaarts-db.bunka.go.jp/an/anime_series/10406/anime_episodes",
    "よみがな": "トゥルーティアーズ",
    "アニメシリーズ ID": "ANS000606500",
    "アニメ作品 ID": "ANT000606500",
    "キャスト": "【仲上眞一郎】石井真　／　【石動乃絵】高垣彩陽　／　【湯浅比呂美】名塚佳織　／　【安藤愛子】井口裕香　／　【野伏三代吉】吉野裕行　／　【石動純】増田裕生　／　【黒部朋与】渡辺智美　／　【眞一郎の父】藤原啓治　／　【眞一郎の母】高橋理恵子　／　【酒蔵の少年】土倉有貴　／　【高岡ルミ】ミルノ純　／　【あさみ】下田麻美　／　【美紀子】渡部恵子",
    "タイトル": "true tears",
    "メインスタッフ": "[原作]La'cryma　／　[シリーズ構成]岡田麿里　／　[監督]西村純二　／　[キャラクター原案]上田夢人　／　[キャラクターデザイン]関口可奈味　／　[メイン美術設定]高畠聡　／　[美術監督]竹田悠介　／　[美術監督]篠原理子　／　[音楽]菊地創",
    "メディア": "TV",
    "主題歌情報": "OP・ED2「リフレクティア」　／　IS「そのままの僕で」歌 eufonius　／　ED1「セカイノナミダ」歌結城アイラ",
    "原作": "-",
    "放送回数": "-",
    "放送局／劇場／販売元": "-",
    "放送期間など": "-",
    "放送枠時間／収録時間": "-",
    "監督": "-",
    "終了年月日": "2008/03/30",
    "製作・制作": "[アニメーション制作]P.A. WORKS[ロゴ]　／　[製作]true tears 製作委員会　／　[製作]バンダイビジュアル　／　[製作]ランティス",
    "話数": "-",
    "開始年月日": "2008/01/06"
  }
]
```


図 19　メディア芸術データベース AnimeEdition の抽出例


```
[
  {
    "公開日": "2008/01/04",
    "各話キャスト": "-",
    "各話キャラクター": "-",
    "各話スタッフ": "-",
    "各話ストーリー": "-",
    "各話タイトル": "true tears こちらチューリップ放送局出張版",
    "各話メインメカ": "-",
    "各話情報備考": "-",
    "各話表記": "S\n すべて表示省略表示",
    "情報源": "-"
  },
  …
  <省略>
]
```


図 20　メディア芸術データベース AnimeEpisode の抽出例

しょぼいカレンダーについては Web API[14]が提供されているため，Web API にアクセスすることによって AnimeEdition，AnimeEpisode，Broadcast に相当するデータをそれぞれ取得した．

アニメハックに対する抽出規則は，以下の 4 つ作成した．

i. イベント情報ページのリストを取得する
ii. (i)の抽出結果を利用して,イベント情報ページから Event を取得する
iii. (ii)の抽出結果を利用して,作品情報ページから AnimeEdition を取得する

アニメハックは，カレンダー型のイベント情報のリストを持つページが存在し，(i)の抽出規則でそれらからイベント情報ページへのリンクを取得する．(ii)の抽出規則では，(i)で得られたリンク先 URL を利用し，イベント情報ページから Event に相当する情報を抽出する．(iii)の抽出規則では，(ii)で得られた作品情報ページへのリンク先 URL を利用し，作品情報ページから AnimeEdition に相当する情報を抽出する．

7.2.2. インスタンスマッチング規則の作成

抽出されたデータに対して，異なる抽出データセット間の同じ種類の実体についてインスタンスマッチングを実行する．図 21 に示すマッチング A, B, C の計 3 通りのインスタンスマッチングを実行した（マッチング C はメディア芸術データベースとアニメハック間の AimeEdition に対するインスタンスマッチングである）．

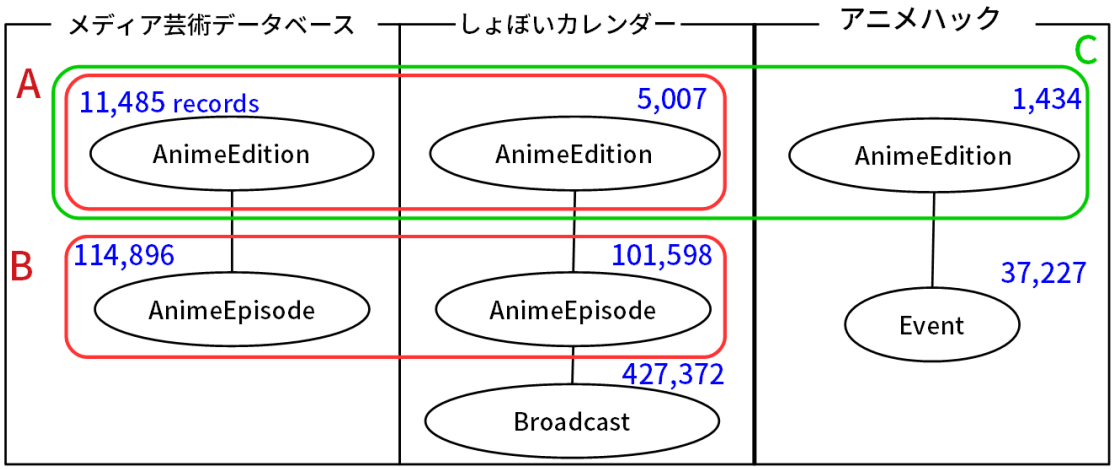

図 21　インスタンスマッチングの対象となる抽出データ

[14] https://sites.google.com/site/syobocal/spec

マッチング A, B, C に対するインスタンスマッチング規則をそれぞれ表 3，表 4，表 5 に示した．これらの各規則によって生成された類似度の集合を類似ベクトルとし，インスタンスマッチングの決定木に入力され，インスタンスマッチングが実行される．

表 3　メディア芸術データベースとしょぼいカレンダー間の AnimeEdition のマッチング規則

メディア芸術 DB の項目	しょぼいカレンダーの項目	比較尺度
タイトル	Title	levenshtein
よみがな	TitleYomi	levenshtein
タイトル	Title	inclusion
よみがな	TitleYomi	inclusion
開始年月日	FirstYear	inclusion
終了年月日	FirstEndYear	inclusion

表 4　メディア芸術データベースとしょぼいカレンダー間の AnimeEpisode のマッチング規則

メディア芸術 DB の項目	しょぼいカレンダーの項目	比較尺度
各話タイトル	SubTitle	levenshtein
各話表記	Number	inclusion
各話タイトル	SubTitle	inclusion

表 5　メディア芸術データベースとアニメハック間の AnimeEdition のマッチング規則

メディア芸術 DB の項目	アニメハックの項目	比較尺度
タイトル	title	levenshtein
製作・制作	制作会社	inclusion
開始年月日	broadcastSeason	levenshtein

インスタンスマッチングに用いる決定木を訓練するための訓練用データは，マッチング A では 30 組，マッチング B では 24 組，マッチング C では 14 組を作成した．決定木の学習アルゴリズムには，CART(classification and regression trees)アルゴリズム[4]を用い，木の最大の深さを 5 に設定した．

また，マッチング B においては，実行時間を削減するために，マッチング A によっ

て識別された AnimeEdition とリンクを持つ AnimeEpisode のインスタンス同士でのみマッチングを実行した．

7.3. 変換規則

変換規則は，RML を拡張したマッピング言語を用いて，4 クラス 25 プロパティの変換規則を定義した．4 クラスは， AnimeEdition，AnimeEpisode，Broadcast，Event である．25 プロパティのうち 3 プロパティは，インスタンス間の関係を示すリンクである．AnimeEdition クラスの変換規則は，メディア芸術データベース，しょぼいカレンダー，アニメハックの抽出データを統合する変換規則である．AnimeEpisode の変換規則は，メディア芸術データベースとしょぼいカレンダーの抽出データを統合する変換規則である．変換規則の具体的な記述は付録に添付しておく．

7.4. データセット生成結果

抽出データの数およびインスタンスマッチングの結果を表 6 に示す．

データ抽出は，メディア芸術データベースおよびしょぼいカレンダーについては 2018 年 7 月時点，アニメハックについては 2018 年 11 月時点の実行結果である．

表 6　データセット生成結果

<table>
<tr><th>クラス</th><th>データソース</th><th>抽出データ数</th><th colspan="2">マッチした組の数</th></tr>
<tr><td rowspan="3">AnimeEdition</td><td>メディア芸術データベース</td><td>11,485</td><td rowspan="2">A:3,636</td><td rowspan="3">C: 961</td></tr>
<tr><td>しょぼいカレンダー</td><td>5,007</td></tr>
<tr><td>アニメハック</td><td>1,434</td><td>N/A</td></tr>
<tr><td rowspan="2">AnimeEpisodes</td><td>メディア芸術データベース</td><td>114,896</td><td colspan="2" rowspan="2">B: 21,980</td></tr>
<tr><td>しょぼいカレンダー</td><td>101,598</td></tr>
<tr><td>Broadcast</td><td>しょぼいカレンダー</td><td>427,372</td><td>N/A</td><td>N/A</td></tr>
<tr><td>Event</td><td>アニメハック</td><td>37,227</td><td>N/A</td><td>N/A</td></tr>
</table>

インスタンスマッチングの実行結果に対しては，マッチした組をランダムにサンプリングし，人手で正しい結果であるか評価した．マッピング A,B,C の正解率はそれぞれ次の通りである．括弧内の数字は，サンプリングした件数を示す．

A) 78% (120 件)
B) 82% (120 件)
C) 61% (100 件)

変換規則に従い RDF データへ変換した結果，計 3,279,292 トリプルの LOD データセットが生成された.

8. 考察

7 章の評価実験について考察する．3 つの Web サイトをデータソースとして選定し，生成処理に必要な各規則を作成した．それらに基づいてデータセットの生成処理を実行し，意図した LOD データセットを生成できることが検証できた．

抽出データのインスタンスマッチング結果は，メディア芸術データベースとしょぼいカレンダー間の AnimeEdition については，しょぼいカレンダーから抽出した AnimeEdition の約 73%にあたる 3,636 件がマッチした．メディア芸術データベースとアニメハック間の AnimeEdition については，アニメハックの AnimeEdition の約 67%にあたる 961 件がマッチした．これらのマッチング結果を分析すると，典型的な誤検出として，タイトルが類似しており異なるメディアで公開されたアニメ作品が同一と判定されている結果が多く見られた． 表 7 は，誤ってマッチした例であり，タイトルの前方と開始年が一致しているが，一方は DVD パッケージで出版された作品で，他方は TV アニメ作品である．

表 7　AnimeEdition の誤ったマッチング結果の例

メディア芸術データベース		しょぼいカレンダー	
項目名	**値**	**項目名**	**値**
タイトル	宙のまにまに DVD オリジナルアニメーション	Title	宙のまにまに
よみがな	ソラノマニマニ　ディーブイディーオリジナルアニメーション	TitleYomi	そらのまにまに
開始年月日	2009/12/16	FirstYear	2009
終了年月日	-	FirstEndYear	2009

マッチング結果を改善するには，マッチングアルゴリズムの変更や訓練用データを追加する方法も考えられるが，類似度ベースのマッチングアルゴリズムを使う以上は，マッチング規則での類似度算出をより適切に設定することが求められる．しかしながら，より適切に値を比較するには，現在のマッチング規則の定義では十分に記述できない例が見られた．例えば，評価実験では，メディア芸術データベースの「開始年月日」に対する比較項目はしょぼいカレンダーの「FirstYear」と設定しており，メディア芸術データベースは公開年月日の情報を持ちながら実際には公開月しか評価されていない．しょぼいカレンダーは，「FirstMonth」という項目で公開月にあたる値を持っているが，これ

をメディア芸術データベースの公開年月日の「月」と比較するには，「月」の部分を抽出するような値の整形規則，あるいはこの文字列を日付型と解釈するような規則を記述できる必要がある．

このように実際の抽出データには，データクレンジングが必要なデータが多分に含まれており，前処理として値の整形などを規則として定義できる必要があると考える．しかしながら，規則の記述能力をむやみに高めるだけでは，規則を記述するユーザへの負担の増大や規則定義の保守性の低下を招く恐れがある．規則記述の支援や自動生成など手法の検討が求められる．

生成された LOD データセットの利用者の観点から考察すると，データセットの中のあるインスタンスが，複数の抽出データが統合されることによって生成されたかどうかは陽に判別できない．つまり，誤ったマッチング結果が生じた場合は，本来異なった複数のインスタンスが 1 つの実体としてユーザに提示されてしまう．従って，ユーザに誤ったデータを提供しないという意味においては，インスタンスマッチングの誤検出を改善することが重要であると考えられる．

データセット生成の自動化という点においては，一連の生成処理の内容を宣言的な規則として記述することで，反復した実行においても手動でデータ操作をする作業を低減でき，一定程度達成できたと言える．しかし，生成されたデータセットの来歴の保証に関しては，データソースや生成結果，規則定義を単に記録するだけに留まっており，来歴に関するメタデータ付与までは考慮されていない．なお，LOD データセット生成や公開における来歴情報の生成についての議論は，Anastasia らの研究[6]が詳しい．また，抽出規則の定義は，データソースとなる Web ページの文書構造に依存しており，Web ページが更新され文書構造が変化した場合は，規則の修正あるいは新規作成が適宜必要である．データソースが更新された際の，規則修正のためのワークフローの整備が今後の課題として挙げられる．

9. おわりに

本研究では，LOD データセット生成の自動化を目的に，生成処理の内容を宣言的に記述する規則を提案した．また，その規則に基づいて，Web ページから半構造化データの抽出およびその統合し LOD データセットを生成する手法を提案し，それを実現するシステムを実装した．

本提案手法は，LOD データセット生成における Web ページからのデータ抽出やその統合処理を規則として宣言的に記述し，それらの規則に基づいて LOD データセットの生成処理を実行する手法である．Web ページから抽出対象となる要素を指定し，Key-Value 型のデータとして抽出する規則を定義した．抽出されたデータに対して，同一の実体に対する記述であると考えられるデータを識別するインスタンスマッチングを実行するための規則を定義した．インスタンスマッチングには決定木学習を用いた手法によって，類似度を算出する規則とともに訓練用データを与えることで，インスタンスマッチングを実行した．RDF データへの変換にあたっては，インスタンスマッチングの結果，識別された抽出データ群から RDF トリプルへの変換を記述するための規則を，RDF マッピング言語 RML を拡張することで定義した．

提案手法を実装したシステムを用いて，実際の Web サイトを対象に LOD データセットの生成実験を行った．各規則を作成し，規則に基づいて各 Web サイトの記述を統合した LOD データセットが生成されたことを検証した．実際のデータの多くは適切なデータクレンジングを要し，インスタンスマッチングの結果の改善やより柔軟な RDF データへの変換を実現するには，抽出したデータの整形を定義する規則が必要であると考える．

謝辞

2 年間の研究活動において，ご指導いただいた杉本重雄先生，永森光晴先生，三原鉄也先生に深く感謝申し上げます．また，一緒に議論をして研究を進めてきた研究室の皆さん，ありがとうございました．

参考文献

1. Bhattacharya, I., Getoor, L.: Entity Resolution in Graphs. In: Mining Graph Data. pp. 311–344 John Wiley & Sons, Inc., Hoboken, NJ, USA (2006).
2. Bizer, C., Lehmann, J., Kobilarov, G., et al.: DBpedia - A crystallization point for the Web of Data. J. Web Semant. 7, 3, 154–165 (2009).
3. Bizer, C., Bizer, C.: D2RQ - treating non-RDF databases as virtual RDF graphs. Proc. 3RD Int. Semant. WEB Conf. (ISWC2004. (2004).
4. Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J.: Classification and Regression Trees. (1984).
5. Daskalaki, E., Flouris, G., Fundulaki, I., Saveta, T.: Instance matching benchmarks in the era of Linked Data. J. Web Semant. 39, 1–14 (2016).
6. Dimou, A., De Nies, T., Verborgh, R., Mannens, E., Van De Walle, R.: Automated Metadata Generation for Linked Data Generation and Publishing Workflows. In: LDOW@WWW. (2016).
7. Dimou, A., Sande, M. Vander, Colpaert, P., et al.: RML: A Generic Language for Integrated RDF Mappings of Heterogeneous Data. In: Bizer, C. et al. (eds.) Proceedings of the 7th Workshop on Linked Data on the Web. (2014).
8. Elmagarmid, A.K., Ipeirotis, P.G., Verykios, V.S.: Duplicate Record Detection: A Survey. IEEE Trans. Knowl. Data Eng. 19, 1, 1–16 (2007).
9. Isele, R., Jentzsch, A., Bizer, C.: Silk server - Adding missing links while consuming linked data. In: Proceedings of the First International Conference on Consuming Linked Data. pp. 85–96 CEUR-WS.org (2010).
10. Jentzsch, A.A., Sahnwaldt, C., Isele, S.R., Bizer, C.: DBpedia Mapping Language, https://github.com/dbpedia/extraction-framework/raw/master/core/doc/mapping_language/DBpedia_Mapping_Language.pdf, (accessed 2018-12-4).
11. Lefrançois, M., Zimmermann, A., Bakerally, N.: A SPARQL extension for generating RDF from heterogeneous formats. In: Proc. Extended Semantic Web Conference (ESWC'17). , Portoroz, Slovenia (2017).
12. Richard Cyganiak: Tarql: SPARQL for Tables – Tarql – SPARQL for Tables: Turn CSV into RDF using SPARQL syntax, http://tarql.github.io/, (accessed 2018-7-18).
13. Rong, S., Niu, X., Xiang, E., Wang, H.: A Machine Learning Approach for Instance Matching Based on Similarity Metrics. In: Proceedings of the 11th

international conference on The Semantic Web - Volume Part I. pp. 460–475 Springer, Berlin, Heidelberg (2012).
14. Tim Berners-Lee: Linked Data - Design Issues, https://www.w3.org/DesignIssues/LinkedData, (accessed 2018-12-1).
15. Volz, J., Bizer, C., Gaedke, M., Kobilarov, G.: Discovering and Maintaining Links on the Web of Data. In: Proceedings of the 8th International Semantic Web Conference. pp. 650–665 Springer-Verlag (2009).
16. Document Object Model (DOM) Level 3 XPath Specification, https://www.w3.org/TR/DOM-Level-3-XPath/, (accessed 2019-1-2).
17. Link Specification Language | Silk Link Discovery Framework Project | Assembla, https://app.assembla.com/wiki/show/silk/Link_Specification_Language, (accessed 2018-12-18).
18. OpenRefine, http://openrefine.org/, (accessed 2018-12-18).
19. R2RML: RDB to RDF Mapping Language, https://www.w3.org/TR/r2rml/, (accessed 2018-12-18).
20. RDF - Semantic Web Standards, https://www.w3.org/RDF/, (accessed 2018-12-3).
21. RDF Schema 1.1, https://www.w3.org/TR/rdf-schema/, (accessed 2019-1-8).
22. Selectors Level 3, https://www.w3.org/TR/selectors-3/, (accessed 2019-1-2).
23. SPARQL 1.1 Query Language, https://www.w3.org/TR/sparql11-query/, (accessed 2018-12-18).
24. The Linked Open Data Cloud, https://lod-cloud.net/, (accessed 2018-12-1).
25. WARC, Web ARChive file format - Library of Congress, https://www.loc.gov/preservation/digital/formats/fdd/fdd000236.shtml, (accessed 2019-1-5).

付録

付録 1　メディア芸術データベースの抽出規則(i)

```
{
    "ruleID": "MADB-AnimeSeriesList",
    "elements": [
        {
            "property": "url",
            "value": "tbody > tr > td:nth-child(2) > a",
            "literalProperty": true,
            "valueAttribute": "href"
        }
    ],
    "multiEntity": true,
    "type": "extraction",
    "targetURL": "https://mediaarts-
db.bunka.go.jp/an/anime_series?locale=ja&asf%5Bkeyword%5D=*&asf%5Bper%5D=1000\nhttps://mediaarts-
db.bunka.go.jp/an/anime_series?asf%5Bkeyword%5D=%2A&asf%5Bper%5D=1000&locale=ja&page=2\nhttps://mediaarts-
db.bunka.go.jp/an/anime_series?asf%5Bkeyword%5D=%2A&asf%5Bper%5D=1000&locale=ja&page=3\nhttps://mediaarts-
db.bunka.go.jp/an/anime_series?asf%5Bkeyword%5D=%2A&asf%5Bper%5D=1000&locale=ja&page=4\nhttps://mediaarts-
db.bunka.go.jp/an/anime_series?asf%5Bkeyword%5D=%2A&asf%5Bper%5D=1000&locale=ja&page=5\nhttps://mediaarts-
db.bunka.go.jp/an/anime_series?asf%5Bkeyword%5D=%2A&asf%5Bper%5D=1000&locale=ja&page=6\nhttps://mediaarts-
db.bunka.go.jp/an/anime_series?asf%5Bkeyword%5D=%2A&asf%5Bper%5D=1000&locale=ja&page=7\nhttps://mediaarts-
db.bunka.go.jp/an/anime_series?asf%5Bkeyword%5D=%2A&asf%5Bper%5D=1000&locale=ja&page=8\nhttps://mediaarts-
db.bunka.go.jp/an/anime_series?asf%5Bkeyword%5D=%2A&asf%5Bper%5D=1000&locale=ja&page=9\nhttps://mediaarts-
db.bunka.go.jp/an/anime_series?asf%5Bkeyword%5D=%2A&asf%5Bper%5D=1000&locale=ja&page=10\nhttps://mediaarts-
db.bunka.go.jp/an/anime_series?asf%5Bkeyword%5D=%2A&asf%5Bper%5D=1000&locale=ja&page=11\nhttps://mediaarts-
db.bunka.go.jp/an/anime_series?asf%5Bkeyword%5D=%2A&asf%5Bper%5D=1000&locale=ja&page=12",
}
```

付録 2　メディア芸術データベースの抽出規則(ii)

```
{
    "ruleID": "MADB-AnimeSeries",
    "elements": [
        {
            "property": ":nth-child(1) > :nth-child(1) > tbody > :nth-child(1) > .t1",
            "value": ":nth-child(1) > :nth-child(1) > tbody > :nth-child(1) > .bd",
            "literalProperty": false,
            "valueAttribute": null
        },
        {
            "property": ":nth-child(1) > :nth-child(1) > tbody > :nth-child(1) > .t3",
            "value": ":nth-child(1) > :nth-child(1) > tbody > :nth-child(1) > .t4",
            "literalProperty": false,
            "valueAttribute": null
        },
        {
            "property": ":nth-child(1) > :nth-child(1) > tbody > :nth-child(2) > th",
            "value": ":nth-child(1) > :nth-child(1) > tbody > :nth-child(2) > td",
            "literalProperty": false,
            "valueAttribute": null
        },
        {
            "property": ":nth-child(1) > :nth-child(1) > tbody > :nth-child(4) > th",
            "value": ":nth-child(1) > :nth-child(1) > tbody > :nth-child(4) > td",
            "literalProperty": false,
            "valueAttribute": null
        },
        {
            "property": ":nth-child(1) > :nth-child(1) > tbody > :nth-child(5) > th",
            "value": ":nth-child(1) > :nth-child(1) > tbody > :nth-child(5) > td",
            "literalProperty": false,
            "valueAttribute": null
        },
        {
            "property": ":nth-child(1) > :nth-child(2) > tbody > :nth-child(1) > .t1",
            "value": ":nth-child(1) > :nth-child(2) > tbody > :nth-child(1) > .t2",
            "literalProperty": false,
            "valueAttribute": null
        },
        {
            "property": ":nth-child(1) > :nth-child(2) > tbody > :nth-child(1) > .t3",
            "value": ":nth-child(1) > :nth-child(2) > tbody > :nth-child(1) > .t4",
            "literalProperty": false,
            "valueAttribute": null
        },
```

```
        {
            "property": ":nth-child(1) > :nth-child(2) > tbody > :nth-child(2) > th",
            "value": ".main > :nth-child(1) > :nth-child(2) > tbody > :nth-child(2) > td",
            "literalProperty": false,
            "valueAttribute": null
        },
        {
            "property": ":nth-child(1) > :nth-child(2) > tbody > :nth-child(3) > th",
            "value": ".main > :nth-child(1) > :nth-child(2) > tbody > :nth-child(3) > td",
            "literalProperty": false,
            "valueAttribute": null
        },
        {
            "property": ":nth-child(1) > :nth-child(2) > tbody > :nth-child(4) > th",
            "value": ".main > :nth-child(1) > :nth-child(2) > tbody > :nth-child(4) > td",
            "literalProperty": false,
            "valueAttribute": null
        },
        {
            "property": ":nth-child(1) > :nth-child(2) > tbody > :nth-child(5) > th",
            "value": ".main > :nth-child(1) > :nth-child(2) > tbody > :nth-child(5) > td",
            "literalProperty": false,
            "valueAttribute": null
        },
        {
            "property": ":nth-child(1) > :nth-child(2) > tbody > :nth-child(6) > th",
            "value": ".main > :nth-child(1) > :nth-child(2) > tbody > :nth-child(6) > td",
            "literalProperty": false,
            "valueAttribute": null
        },
        {
            "property": ":nth-child(1) > :nth-child(2) > tbody > .nb > th",
            "value": ":nth-child(1) > :nth-child(2) > tbody > .nb > td > p",
            "literalProperty": false,
            "valueAttribute": null
        },
        {
            "property": ".main > :nth-child(2) > table > tbody > :nth-child(2) > th",
            "value": ".main > :nth-child(2) > table > tbody > :nth-child(2) > td",
            "literalProperty": false,
            "valueAttribute": null
        },
        {
            "property": ".main > :nth-child(2) > table > tbody > :nth-child(1) > th",
            "value": ":nth-child(2) > table > tbody > :nth-child(1) > td",
            "literalProperty": false,
            "valueAttribute": null
        },
        {
            "property": ".main > :nth-child(2) > table > tbody > :nth-child(3) > th",
            "value": ".main > :nth-child(2) > table > tbody > :nth-child(3) > td > p",
            "literalProperty": false,
            "valueAttribute": null
        },
        {
            "property": ".main > :nth-child(2) > table > tbody > :nth-child(4) > th",
            "value": ".main > :nth-child(2) > table > tbody > :nth-child(4) > td > p",
            "literalProperty": false,
            "valueAttribute": null
        },
        {
            "property": ".main > :nth-child(2) > table > tbody > :nth-child(5) > th",
            "value": ".main > :nth-child(2) > table > tbody > :nth-child(5) > td",
            "literalProperty": false,
            "valueAttribute": null
        },
        {
            "property": "episodesUrl",
            "value": ":nth-child(1) > p > a",
            "literalProperty": true,
            "valueAttribute": "href"
        }
    ],
    "multiEntity": false,
    "type": "extraction",
    "inputFrom": {
        "rule": "MADB-AnimeSeriesList",
        "dataset": "9d8c9e3d-e708-492f-8bb7-0e54602a5187",
        "property": "url"
    },
}
```

付録 3　メディア芸術データベースの抽出規則(iii)


```
{
    "ruleID": "MADB-AnimeEpisodes",
    "elements": [
        {
            "property": "thead > tr > .fst",
            "value": "tbody > tr > .fst",
            "literalProperty": false,
            "valueAttribute": null
        },
        {
            "property": "thead > tr > :nth-child(2)",
            "value": "tbody > tr > :nth-child(2)",
            "literalProperty": false,
            "valueAttribute": null
        },
        {
            "property": "thead > tr > :nth-child(3)",
            "value": "tbody > tr > :nth-child(3)",
            "literalProperty": false,
            "valueAttribute": null
        },
        {
            "property": "thead > tr > :nth-child(4)",
            "value": "tbody > tr > :nth-child(4) > span > p",
            "literalProperty": false,
            "valueAttribute": null
        },
        {
            "property": "thead > tr > :nth-child(5)",
            "value": "tbody > tr > :nth-child(5) > span > p",
            "literalProperty": false,
            "valueAttribute": null
        },
        {
            "property": "thead > tr > :nth-child(6)",
            "value": "tbody > tr > :nth-child(6) > span > p",
            "literalProperty": false,
            "valueAttribute": null
        },
        {
            "property": "thead > tr > :nth-child(7)",
            "value": "tbody > tr > :nth-child(7) > span > p",
            "literalProperty": false,
            "valueAttribute": null
        },
        {
            "property": "thead > tr > :nth-child(8)",
            "value": "tbody > tr > :nth-child(8) > span > p",
            "literalProperty": false,
            "valueAttribute": null
        },
        {
            "property": "thead > tr > :nth-child(9)",
            "value": "tbody > tr > :nth-child(9) > span > p",
            "literalProperty": false,
            "valueAttribute": null
        },
        {
            "property": "thead > tr > .end",
            "value": "tbody > tr > .end > span > p",
            "literalProperty": false,
            "valueAttribute": null
        }
    ],
    "multiEntity": true,
    "type": "extraction",
    "recursiveCrawling": {
        "selector": "#new_asf > ul > li.pageSkip > ul > li.nxt > a",
        "attribute": "href"
    },
    "inputFrom": {
        "rule": "MADB-AnimeSeries",
        "dataset": "233a06ec-8ae2-4ab0-ba2d-e4a85d1b8988",
        "property": "episodesUrl"
    },
}
```

付録 4　実験で用いた変換規則

```
@prefix rr: <http://www.w3.org/ns/r2rml#>.
@prefix ql: <http://semweb.mmlab.be/ns/ql#>.
@prefix rml: <http://semweb.mmlab.be/ns/rml#>.
@prefix rmle: <https://mdlab.slis.tsukuba.ac.jp/ns/rmle/#>.
@prefix pcmm: <https://mdlab.slis.tsukuba.ac.jp/ns/pcmm/#>.
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>.

#------------------------------------------------
# データソース定義
#------------------------------------------------
<#MADB-Syobo_MADB>
    a rmle:IdentifiedSource ;
    rdfs:label "メディア芸術データベース AnimeEdition" ;
    rml:source <file://data/madb-syobo/MADB-AnimeSeries.json> ;
    rml:referenceFormulation ql:JSONPath;
    rmle:priorityValue 0
    .

<#MADB-Syobo_Syobo>
    a rmle:IdentifiedSource ;
    rdfs:label "しょぼかる AnimeEdition" ;
    rml:source <file://data/madb-syobo/Syobo-Titles.json> ;
    rml:referenceFormulation ql:JSONPath;
    rmle:priorityValue 1
    .

<#MADB-Episodes>
    a rmle:IdentifiedSource ;
    rdfs:label "メディア芸術データベース AnimeEpisode" ;
    rml:source <file://data/madb-syobo/MADB-AnimeEpisodes.json> ;
    rml:referenceFormulation ql:JSONPath;
    rmle:priorityValue 0
    .

<#Syobo-Episodes>
    a rmle:IdentifiedSource ;
    rdfs:label "しょぼかる AnimeEpisode" ;
    rml:source <file://data/madb-syobo/Syobo-Episodes.json> ;
    rml:referenceFormulation ql:JSONPath;
    rmle:priorityValue 1
    .

<#MADB-AnimeHack_MADB>
    a rmle:IdentifiedSource ;
    rdfs:label "メディア芸術データベース AnimeEdition" ;
    rml:source <file://data/madb-animehack/MADB-AnimeSeries.json> ;
    rml:referenceFormulation ql:JSONPath;
    rmle:priorityValue 0
    .

<#MADB-AnimeHack_AnimeHack>
    a rmle:IdentifiedSource ;
    rdfs:label "アニメハック AnimeEdition" ;
    rml:source <file://data/madb-animehack/AnimeHack-AnimeSeries.json> ;
    rml:referenceFormulation ql:JSONPath;
    rmle:priorityValue 1
    .

#------------------------------------------------
# マッピング定義
#------------------------------------------------
<#AnimeEditionMapping>
    a rr:TriplesMap;

    rmle:identifiedSource <#MADB-Syobo_MADB>;
    rmle:identifiedSource <#MADB-Syobo_Syobo>;

    rr:subjectMap [
        rr:template "http://example.com/anime-edition/{アニメシリーズID}";
        rr:class pcmm:AnimeEdition;
        rmle:mappedSource <#MADB-Syobo_MADB>;
    ];

    rr:predicateObjectMap [
        rr:predicate pcmm:title;
        rr:objectMap [
            rmle:mappedSource <#MADB-Syobo_MADB>;
            rml:reference "$.タイトル"
        ];
        rr:objectMap [
            rmle:mappedSource <#MADB-Syobo_Syobo>;
            rml:reference "$.Title"
```

```
        ]
    ];

    rr:predicateObjectMap [
        rr:predicate pcmm:startDate;
        rr:objectMap [
            rmle:mappedSource <#MADB-Syobo_MADB>;
            rml:reference "$.開始年月日"
        ]
    ];

    rr:predicateObjectMap [
        rr:predicate pcmm:endDate;
        rr:objectMap [
            rmle:mappedSource <#MADB-Syobo_MADB>;
            rml:reference "$.終了年月日"
        ]
    ];

    rr:predicateObjectMap [
        rr:predicate pcmm:numberOfEpisodes;
        rr:objectMap [
            rmle:mappedSource <#MADB-Syobo_MADB>;
            rml:reference "$.話数"
        ]
    ];

    rr:predicateObjectMap [
        rr:predicate pcmm:numberOfBroadcasts;
        rr:objectMap [
            rmle:mappedSource <#MADB-Syobo_MADB>;
            rml:reference "$.放送回数"
        ]
    ];

    rr:predicateObjectMap [
        rr:predicate pcmm:firstChannel;
        rr:objectMap [
            rmle:mappedSource <#MADB-Syobo_Syobo>;
            rml:reference "$.FirstCh"
        ]
    ];

    rr:predicateObjectMap [
        rr:predicate pcmm:wikipedia;
        rr:objectMap [
            rmle:mappedSource <#MADB-Syobo_Syobo>;
            rml:reference "$.Wikipedia"
        ]
    ];

    rr:predicateObjectMap [
        rr:predicate pcmm:titleYomi;
        rr:objectMap [
            rmle:mappedSource <#MADB-Syobo_MADB>;
            rml:reference "$.よみがな"
        ];
        rr:objectMap [
            rmle:mappedSource <#MADB-Syobo_Syobo>;
            rml:reference "$.TitleYomi"
        ];
        rmle:condition rmle:all
    ];

    rr:predicateObjectMap [
        rr:predicate pcmm:episode;
        rr:objectMap [
            rr:parentTriplesMap <#EpisodeMapping>;
            rr:joinCondition [
                rr:child "$.episodesUrl";
                rr:parent "$._meta.url";
                rmle:childSource <#MADB-Syobo_MADB>;
                rmle:parentSource <#MADB-Episodes>;
            ]
        ]
    ];

    rr:predicateObjectMap [
        rr:predicate pcmm:broadcastedBy;
        rr:objectMap [
            rr:parentTriplesMap <#BroadcastMapping>;
            rr:joinCondition [
                rr:child "$.TID";
                rr:parent "$.TID";
                rmle:childSource <#MADB-Syobo_Syobo>;
```

```
            ]
        ]
    ]
    .

<#EpisodeMapping>
    a rr:TriplesMap;

    rmle:identifiedSource <#MADB-Episodes>;
    rmle:identifiedSource <#Syobo-Episodes>;

    rr:subjectMap [
        rr:template "http://example.com/anime-episode/{$._id.$oid}";
        rr:class pcmm:AnimeEpisode;
        rmle:mappedSource <#MADB-Episodes>;
    ];

    rr:predicateObjectMap [
        rr:predicate pcmm:title;
        rr:objectMap [
            rmle:mappedSource <#MADB-Episodes>;
            rml:reference "$.各話タイトル";
            rr:language "ja"
        ];
    ];

    rr:predicateObjectMap [
        rr:predicate pcmm:date;
        rr:objectMap [
            rmle:mappedSource <#MADB-Episodes>;
            rml:reference "$.公開日"
        ];
    ];

    rr:predicateObjectMap [
        rr:predicate pcmm:number;
        rr:objectMap [
            rmle:mappedSource <#Syobo-Episodes>;
            rml:reference "$.Number"
        ];
    ];
    .

<#BroadcastMapping>
    a rr:TriplesMap;

    rml:logicalSource [
       rml:source <file://data/Syobo-Broadcast.json> ;
       rml:referenceFormulation ql:JSONPath ;
       rml:iterator "$.[*]"
   ];

    rr:subjectMap [
        rr:template "http://example.com/anime-broadcast/{$.PID}";
        rr:class pcmm:Broadcast;
    ];

    rr:predicateObjectMap [
        rr:predicate pcmm:startTime;
        rr:objectMap [
            rml:reference "$.StTime";
        ];
    ];

    rr:predicateObjectMap [
        rr:predicate pcmm:startTimeOffset;
        rr:objectMap [
            rml:reference "$.StOffset";
        ];
    ];

    rr:predicateObjectMap [
        rr:predicate pcmm:endTime;
        rr:objectMap [
            rml:reference "$.EdTime"
        ];
    ];

    rr:predicateObjectMap [
        rr:predicate pcmm:count;
        rr:objectMap [
            rml:reference "$.Count"
        ];
    ];
```

```
    rr:predicateObjectMap [
        rr:predicate pcmm:chID;
        rr:objectMap [
            rr:template "http://example.com/channel/{$.ChID}";
        ];
    ];
    .

<#MADB-AnimeHackMapping>
    a rr:TriplesMap;

    rmle:identifiedSource <#MADB-AnimeHack_MADB>;
    rmle:identifiedSource <#MADB-AnimeHack_AnimeHack>;

    rr:subjectMap [
        rr:template "http://example.com/anime-edition/{アニメシリーズID}";
        rr:class pcmm:AnimeEdition;
        rmle:mappedSource <#MADB-AnimeHack_MADB>;
    ];
    .

<#EventMapping>
    a rr:TriplesMap;

    rml:logicalSource [
       rml:source <file://data/AnimeHack-Event.json> ;
       rml:referenceFormulation ql:JSONPath ;
       rml:iterator "$.[*]"
    ];

    rr:subjectMap [
       rr:template "http://example.com/anime-event/{$._id.$oid}";
       rr:class pcmm:Event;
       rmle:mappedSource <#MADB>;
    ];

    rr:predicateObjectMap [
       rr:predicate pcmm:date;
       rr:objectMap [
           rml:reference "$.開催日"
       ];
    ];

    rr:predicateObjectMap [
      rr:predicate pcmm:time;
      rr:objectMap [
          rml:reference "$.時間"
      ];
    ];

    rr:predicateObjectMap [
      rr:predicate pcmm:location;
      rr:objectMap [
          rml:reference "$.場所"
      ];
    ];

    rr:predicateObjectMap [
      rr:predicate pcmm:price;
      rr:objectMap [
          rml:reference "$.価格"
      ];
    ];

    rr:predicateObjectMap [
        rr:predicate pcmm:cast;
        rr:objectMap [
            rml:reference "$.出演者"
        ];
    ];

    rr:predicateObjectMap [
        rr:predicate pcmm:website;
        rr:objectMap [
            rml:reference "$.公式サイト"
        ];
    ];

   rr:predicateObjectMap [
       rr:predicate pcmm:relatesTo;
       rr:objectMap [
           rr:parentTriplesMap <#MADB-AnimeHackMapping>;
           rr:joinCondition [
               rr:child "$.animeUrl";
               rr:parent "$._meta.url";
```

```
            rmle:parentSource <#MADB-AnimeHack_AnimeHack>;
        ]
    ]
];
.
```

データソース定義で参照しているファイルについて以下に補足しておく.

- data/madb-syobo/MADB-AnimeSeries.json
- data/madb-syobo/Syobo-Titles.json
  メディア芸術データベースとしょぼいカレンダー間の AnimeEdition に対するインスタンスマッチング結果ファイル
- data/madb-syobo/MADB-AnimeEpisodes.json
- data/madb-syobo/Syobo-Episodes.json
  メディア芸術データベースとしょぼいカレンダー間の AnimeEpisode に対するインスタンスマッチング結果ファイル
- data/madb-animehack/MADB-AnimeSeries.json
- data/madb-animehack/AnimeHack-AnimeSeries.json
  メディア芸術データベースとアニメハック間の AnimeEdition に対するインスタンスマッチング結果ファイル
- data/Syobo-Broadcast.json
  しょぼいカレンダーから抽出した Broadcast の抽出データファイル
- data/AnimeHack-Event.json
  アニメハックから抽出した Event の抽出データファイル