UNIVERSIDADE DE SÃO PAULO

FACULDADE DE FILOSOFIA, CIÊNCIAS E LETRAS DE RIBEIRÃO PRETO

PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO APLICADA

Suporte à geração de dados abertos ligados em bioinformática

Gabriel do Couto Seabra Gusmão de Paula

Dissertação apresentada à Faculdade de Filosofia, Ciências e Letras de Ribeirão Preto da USP, como parte das exigências para a obtenção do título de Mestre em Ciências, Área: Computação Aplicada.

Ribeirão Preto–SP

2019

Gabriel do Couto Seabra Gusmão de Paula

Suporte à geração de dados abertos ligados em bioinformática

Versão **Revisada**

(Versão original encontra-se na unidade que aloja o Programa de Pós-graduação)

Dissertação apresentada à Faculdade de Filosofia, Ciências e Letras de Ribeirão Preto da USP, como parte das exigências para a obtenção do título de Mestre em Ciências, Área: Computação Aplicada.

Orientador: Prof. Dr. Cléver Ricardo Guareis de Farias

Ribeirão Preto–SP

2019

Gabriel do Couto Seabra Gusmão de Paula
Suporte à geração de dados abertos ligados em bioinformática. Ribeirão Preto–SP, 2019.
138p. : il.; 30 cm.

Dissertação apresentada à Faculdade de Filosofia, Ciências e Letras
de Ribeirão Preto da USP, como parte das exigências para
a obtenção do título de Mestre em Ciências,
Área: Computação Aplicada.

Orientador: Prof. Dr. Cléver Ricardo Guareis de Farias

1. Dados Abertos Ligados. 2. Bioinformática. 3. Dados Semiestruturados.
4. Regras de Transformação.

Gabriel do Couto Seabra Gusmão de Paula

Support for the generation of linked open data in bioinformatics

Revised Version

(Original version is in the unit that hosts
the Graduate Program)

Supervisor: Prof. Dr. Cléver Ricardo Guareis de Farias

Ribeirão Preto–SP

2019

Gabriel do Couto Seabra Gusmão de Paula

Suporte à geração de dados abertos ligados em bioinformática

Dissertação apresentada à Faculdade de Filosofia, Ciências e Letras de Ribeirão Preto da USP, como parte das exigências para a obtenção do título de Mestre em Ciências, Área: Computação Aplicada.

Trabalho aprovado. Ribeirão Preto–SP, 20 de Novembro de 2019:

Prof. Dr. Cléver Ricardo Guareis de Farias
Orientador

Prof. Dr. Ricardo Zorzetto Nicoliello Vêncio

Profa. Dra. Marilde Terezinha Prado Santos

Prof. Dr. José Eduardo Santarém Segundo

Ribeirão Preto–SP
2019

Agradecimentos

Agradeço primeiramente aos meus pais por todo o carinho, apoio psicológico e financeiro durante essa jornada. Sem vocês essa conquista não teria sido possível, muito obrigado e eu amo vocês!

Agradeço também à minha namorada, Verena, por estar sempre comigo, compartilhando os momentos de alegria e de desânimo, me dando apoio e tendo esperança junto comigo. Te amo, linda.

Eu gostaria de agradecer ao meu orientador, Cléver, o qual me ensinou muitas coisas, tais como: pensamento crítico, clareza no texto, ética na pesquisa e dedicação. Obrigado, Cléver!

Não poderia deixar de agradecer também a todas as novas amizades que fiz em Ribeirão Preto e também a todas as amizades antigas as quais foram muito importantes durante a construção desse trabalho. Obrigado Espin, Lúcia, Diógenes, Patrícia, George, Angélica e outras pessoas nas quais tenho carinho. Em especial, eu gostaria de agadecer os amigos de laboratório: Ricardo (Cawal), Amr, Danillo (Torresmo), Yagoub e Guilherme. Muito obrigado, galera!

Resumo

DE PAULA, Gabriel Couto Seabra Gusmão. **Suporte à geração de dados abertos ligados em bioinformática**. 2019. 138p. Dissertação (Mestrado em Ciências) - Faculdade de Filosofia, Ciências e Letras de Ribeirão Preto, Universidade de São Paulo, 2019.

Diferentes conjuntos de dados na web encontram-se em formatos que dificultam o processamento e a extração automática de informação. A transformação desses dados em Dados Abertos Ligados (DAL) pode facilitar o processamento e a obtenção de novos conhecimentos. DAL define um conjunto de dados compreendido/interpretável por computadores, interconectado e semanticamente anotado. Diversos domínios de conhecimento podem ser beneficiados com o uso de DAL, dentre os quais destaca-se a bioinformática. A bioinformática é caracterizada pelo uso e a disponibilização de grandes quantidades de dados na web, normalmente armazenados em arquivos texto semiestruturados. Existem diferentes abordagens de transformação de dados estruturados e semiestruturados para DAL no domínio biomédico. Porém, essas abordagens não podem ser facilmente estendidas para a bioinformática. Este trabalho teve por objetivo propor uma abordagem de transformação de dados semiestruturados de bioinformática para DAL, chamada de SSD2LOD Transformation Approach. Esta abordagem é composta por quatro atividades, as quais orientam a definição de questões de competência (perguntas de interesse), a especificação de regras de transformação, a transformação dos dados, e, finalmente, a exploração do conjunto DAL resultante. Adicionamente, desenvolvemos um conjunto de ferramentas de suporte para executar o processo de transformação e exploração dos dados, de modo a facilitar a aplicação da abordagem. Na sequência, aplicamos a abordagem proposta em uma prova de conceito utilizando dados de um experimento de genômica funcional disponibilizado na plataforma ArrayExpress. A abordagem SSD2LOD Transformation Approach representa uma solução adequada para a transformação de dados de bioinformática em DAL, permitindo alcançar os benefícios da web semântica neste domínio.

Palavras-chave: dados abertos ligados; bioinformática; dados semiestruturados; regras de transformação.

Abstract

DE PAULA, G. C. S. G. **Support for the generation of linked open data in bioinformatics**. 2019. 138p. Dissertação (Mestrado) - Faculdade de Filosofia, Ciências e Letras de Ribeirão Preto, Universidade de São Paulo, São Paulo, 2019.

Data on the web are frequently stored in formats that hinder the automatic processing and extraction of knowledge. However, the transformation of those data into Linked Open Data (LOD) may facilitate the discovery of new knowledge. LOD defines a machine-readable, interconnected and semantically annotated dataset. The bioinformatics domain may benefit from the use of LOD, since it is characterized by the use and availability of large amounts of data on the web, usually stored in semi-structured text files. There are different approaches to support the transformation of structured and semi-structured data into LOD in the biomedical domain. However, these approaches are not easily applied to the bioinformatics domain. This work aimed at developing a transformation approach from semi-structured bioinformatics data into LOD, called SSD2LOD Transformation Approach. The proposed approach consists of four activities, which guide the definition of competency questions (questions of interest), specification of transformation rules, data transformation, and, finally, exploration of the produced LOD set. Additionally, we have developed a toolset to support the process of data transformation and exploration of our LOD transformation approach. Next, we have applied the proposed approach in a proof of concept using source data from a functional genomics experiment available at ArrayExpress. Our approach supports the transformation of bioinformatics data into LOD, thus enabling the benefits of the semantic web in this domain.

Keywords: linked open data; bioinformatics; semi-structured data; transformation rules.

Lista de figuras

Lista de tabelas

Lista de Listagens

Sumário

1

Introdução

A web semântica é uma das áreas da computação que tem tido um grande crescimento nos últimos anos. O desafio principal da web semântica consiste em tornar a semântica dos dados identificável e compreendida por computadores e não somente por seres humanos, permitindo que máquinas possam interpretar e analisar tais dados de forma autônoma. Todas as áreas de conhecimento têm o potencial para beneficiar-se dessa capacidade computacional. Dentro das ciências biomédicas, a bioinformática apresenta-se como uma área promissora para a web semântica em geral e Dados Abertos Ligados (DAL) em particular, dada a disponibilidade crescente de informações (semânticas) na área disponíveis publicamente na web. Desta forma, este capítulo apresenta a motivação, o objetivo e a metodologia utilizada no desenvolvimento deste trabalho.

O restante deste capítulo está estruturado da seguinte forma: a seção 1.1 apresenta a motivação para o desenvolvimento de uma nova abordagem de transformação de dados de bioinformática para o formato DAL; a seção 1.2 apresenta o objetivo do trabalho; a seção 1.3 apresenta a metodologia de desenvolvimento utilizada; e por fim, a seção 1.4 apresenta a estrutura dos demais capítulos da dissertação.

1.1 Motivação

O volume de dados disponibilizados na web vem crescendo desde sua criação. Um dos motivos desse crescimento está relacionado à democratização mundial do acesso à internet, permitindo que pessoas de todas as classes sociais possam publicar conteúdo (multimídia e textual) em redes sociais, trocar emails profissionais e pessoais, etc. Porém, tais dados

são, em sua grande maioria, disponibilizados em um formato que apenas seres humanos são capazes de interpretá-los.

A fim de solucionar tal problema, esforços para a disseminação da web semântica têm sido desenvolvidos. A web semântica preconiza o uso de formatos para a publicação de dados que facilitem o processamento e a compreensão automática da semântica dos mesmos. A capacidade de interpretar dados disponíveis na web permite que serviços computacionais (aplicações), existentes ou futuros, possam executar automaticamente tarefas a partir da descoberta (inferência) de conhecimento sob demanda.

A web semântica pressupõe que os dados disponibilizados na web estejam em forma de Dados Abertos Ligados (DAL) (BERNERS-LEE, 2006), também chamados de Dados Abertos Conectados (ISOTANI; BITTENCOURT, 2015). Dados abertos ligados não devem possuir restrições de uso, devem possuir uma representação padronizada pela W3C (*World Wide Web Consortium*) e devem referenciar outros dados já disponibilizados na web. Dados abertos ligados são semanticamente anotados de acordo com os termos de uma ontologia. Uma ontologia define formalmente um domínio de conhecimento (GRUBER, 1995). Desta forma, softwares automatizados podem entender os dados publicados e permitir que informações semânticas associadas a um outro conjunto de dados possam ser utilizadas para a descoberta de novos conhecimentos.

Dentre as áreas de conhecimento que podem ser beneficiadas com a utilização da web semântica, destaca-se a bioinformática. Bioinformática pode ser definida como a aplicação de técnicas estatísticas e computacionais para entender e organizar dados associados a moléculas (PEVSNER, 2015; CHRISTENSEN, 2018). Mais especificamente, a genômica funcional, um dos campos da bioinformática, destaca-se em razão do uso e disponibilização de uma quantidade crescente de dados de maneira pública. Genômica funcional tem como objetivo atribuir função às informações genéticas armazenadas no genoma tendo como base o nível de expressão dos genes em diferentes condições experimentais (GUARDIA, 2016). A partir do conhecimento das funções gênicas, novas descobertas sobre o funcionamento dos sistemas biológicos podem ser feitas, permitindo responder a uma questão biológica de interesse.

Dados públicos de expressão gênica são tipicamente armazenados em arquivos semiestruturados e disponibilizados em repositórios, tais como *Gene Expression Omnibus*

(GEO) (National Center for Biotechnology Information, 2017) e ArrayExpress (BRAZMA et al., 2003). Os dados disponíveis nesses repositórios públicos podem ser brutos e/ou processados. Dados brutos representam dados obtidos a partir de diferentes técnicas de obtenção de dados de expressão gênica, tais como microarray de DNA e sequenciamento de RNA (RNA-seq). Dados processados representam dados de expressão gênica obtidos a partir da realização de alguma atividade de análise posterior a sua extração, como, por exemplo, normalização ou análise diferencial.

Em geral, as ferramentas e/ou plataformas integradas de suporte à análise de dados fornecem os resultados destas análises em formatos puramente sintáticos, isto é, sem qualquer vínculo explícito com ontologias relacionadas. Dessa forma, dados obtidos a partir das análises realizadas normalmente não possuem valor semântico explícito. A ausência explícita de representação semântica dificulta a execução de novas análises que utilizem o relacionamento entre os dados como premissa, assim como dificulta a exploração semântica destes dados. Desta forma, faz-se necessária a transformação dos dados processados em DAL a fim de usufruir dos benefícios da web semântica na bioinformática. No entanto, a transformação de dados processados em DAL requer um conhecimento técnico da web semântica dificilmente dominado por bioinformatas e biologistas.

Acreditamos que, da mesma forma que uma metodologia de projeto, uma abordagem para a transformação de dados de bioinformática semiestruturados em DAL deva aderir a um número de propriedades gerais de qualidade (FARIAS, 2002). Particularmente, acreditamos que essa abordagem deva ser simples, sistemática e flexível. Uma abordagem simples depende de um conjunto mínimo de conceitos para representar as regras de transformação de DAL, o que facilita seu uso como um todo. Uma abordagem sistemática fornece um processo passo a passo para orientar a transformação de dados de origem em DAL. Finalmente, uma abordagem flexível pode ser usada em uma variedade de situações, sem (grandes) alterações ou adaptações.

Diferentes abordagens foram propostas para transformar dados biomédicos em DAL, como, por exemplo, Belleau et al. (2008), Auer et al. (2009), Jupp et al. (2014), Merrill et al. (2014), Legaz-García et al. (2016), Kaalia e Ghosh (2016), Jovanovik e Trajanov (2017) e Sernadela, González-Castro e Oliveira (2017). Apesar de seu foco no domínio biomédico, essas abordagens são geralmente inadequadas para a transformação de dados semiestruturados de bioinformática em DAL. Algumas abordagens, por exemplo,

não permitem a customização do processo de transformação por meio do uso de diferentes ontologias (OWL) (BELLEAU et al., 2008; MERRILL et al., 2014), enquanto que outras abordagens não utilizam explicitamente uma linguagem para a especificação de mapeamento/regras de transformação de dados de entrada para DAL, limitando assim sua aplicação.

A abordagem proposta por Legaz-García et al. (2016) pode ser considerada uma exceção nesse sentido. Esta abordagem permite, explicitamente, o uso de diferentes ontologias no processo de transformação, bem como define uma linguagem para a especificação de regras de mapeamento. No entanto, uma vez que esta abordagem foi concebida para a transformação de dados estruturados, esta abordagem e ferramenta de suporte associada não podem ser facilmente adaptadas ao domínio da bioinformática.

1.2 Objetivo

Dados processados de bioinformática de maneira geral e de genômica funcional em particular normalmente não possuem estrutura formal definida, mas uma estrutura flexível e implicitamente definida por meio de rótulos. Esses dados são tipicamente armazenados em arquivos textos ASCII nos formatos *tab-separated values* (TSV) e *comma-separated values* (CSV). Adicionalmente, dados processados não possuem qualquer tipo de informação semântica associada, o que dificulta a exploração semântica dos resultados de uma análise. Neste sentido, este trabalho teve por objetivo propor uma abordagem sistemática para a transformação de dados semiestruturados de bioinformática em DAL. Os objetivos específicos deste projeto incluíram:

1. A definição de uma abordagem de geração de DAL a partir de dados em formato texto semiestruturado;

2. O desenvolvimento de ferramentas de suporte ao processo de transformação de dados semiestruturados em DAL.

1.3 Metodologia

Inicialmente fizemos estudos bibliográficos sobre web semântica, dados abertos ligados e tecnologias associadas. Dentre as tecnologias estudadas, destacam-se: Extensible Markup Language (XML), Resource Description Framework (RDF), Resouce Description Framework Schema (RDFS), Web Ontology Language (OWL) e SPARQL Protocol and RDF Query Language (SPARQL).

Posteriormente iniciamos estudos bibliográficos sobre o estado da arte de abordagens de transformação de dados estruturados e/ou semiestruturados para DAL, aplicados a bioinformática ou a áreas de conhecimento relacionadas. Estes estudos foram utilizados na identificação dos métodos, ferramentas e mecanismos propostos para a transformação de dados (estruturados ou semiestruturados) em DAL.

A seguir, propusemos uma abordagem sistemática de transformação de dados semiestruturados em DAL, chamada SSD2LOD Transformation Approach. Tal abordagem auxilia o bioinformata/biologista a identificar questões de interesse para a transformação dos dados, auxilia na especificação da equivalência semântica dos dados com termos de uma ontologia por meio de uma linguagem de especificação de regras de transformação, chamada SSD2LOD Transformation Language, auxilia na transformação de dados semiestruturados para DAL, e finalmente, auxilia na exploração dos DAL transformados permitindo alcançar as respostas para as questões identificadas.

Em seguida, desenvolvemos um conjunto de ferramentas de suporte à abordagem proposta. A primeira ferramenta consiste de um sistema com interface de linha de comando, chamado SSD2LOD Transformation Tool, que interpreta as regras de transformação de modo a produzir conjuntos DAL. A segunda ferramenta de suporte consiste de um serviço web RESTful, chamada de SSD2LOD Web Service, que encapsula a primeira ferramenta de modo a permitir sua utilização de forma remota pela web. Por fim, desenvolvemos um sistema web, chamado SSD2LOD Web, que utiliza as operações expostas pelo serviço web para facilitar a execução de cada etapa da abordagem.

Finalmente, desenvolvemos uma prova de conceito a fim de explorar o potencial da nossa abordagem de transformação e do conjunto de ferramentas de suporte. Nessa prova de conceito foi possível explorar a linguagem de transformação para especificar

corretamente as equivalências semânticas existentes entre os itens de dados SSD e os termos de uma ontologia desenvolvida para suporte ao processo de transformação. Ao fim deste processo, demonstramos o potencial de nossa abordagem para a descoberta de informações a partir da transformação de dados semiestruturados de genômica funcional em DAL.

1.4 Estrutura do documento

Os demais capítulos deste documento estão estruturados da seguinte forma: o capítulo 2 apresenta a fundamentação teórica da web semântica e suas principais tecnologias, bem como apresenta uma visão geral dos principais formatos de representação de dados brutos de bioinformática; o capítulo 3 apresenta um conjunto de trabalhos relacionados à transformação de dados (semi)estruturados em DAL, principalmente no domínio biomédico; o capítulo 4 descreve cada etapa da proposta de abordagem de transformação de dados semiestruturados em DAL; o capítulo 5 apresenta uma prova de conceito ilustrando a aplicação de nossa abordagem de transformação usando um conjunto de dados de genômica funcional; e por fim, o capítulo 6 apresenta a conclusão do trabalho e elenca futuras direções para esta pesquisa.

2

Fundamentação Teórica

Dados de maneira geral são disponibilizados na web a fim de prover informações aos seus diferentes usuários. Dados de bioinformática em particular são compartilhados na web a fim de colaborar com a descoberta de novos conhecimentos e também permitir a reprodução de experimentos científicos. O conjunto de tecnologias atualmente utilizado na web permite essencialmente que apenas pessoas compreendam dados de uma página web exibida por um navegador. Tim Berners-Lee propôs uma extensão da web chamada de web semântica a fim de permitir que sistemas computacionais também possam compreender este conjunto de dados. Essa extensão não altera a arquitetura de comunicação da web, entretanto apresenta algumas alterações quanto às tecnologias utilizadas para a representação de conteúdo. Este capítulo apresenta uma classificação geral de dados na web e uma visão geral de dados na bioinformática. Adicionalmente, este capítulo apresenta uma visão geral dos principais conceitos e tecnologias relacionados à web semântica e a dados abertos ligados.

O restante do capítulo está estruturado da seguinte forma: a seção 2.1 apresenta uma classificação geral de dados; a seção 2.2 apresenta uma visão geral de dados na bioinformática; a seção 2.3 apresenta uma visão geral da web semântica e suas principais tecnologias; e finalmente, a seção 2.4 apresenta o conceito de dados abertos ligados.

2.1 Categorias de dados

Dados podem ser classificados em três grandes categorias com base na sua estrutura (ABITEBOUL; BUNEMAN; SUCIU, 2000; SINT et al., 2009; RUSU et al., 2013; SAM-

BREKAR; RAJPUROHIT; JOSHI, 2018):

- *Dados estruturados*, os quais consistem de um conjunto de dados definidos a partir de uma estrutura (*schema*) formalmente definida. A existência de uma descrição formal do conteúdo de um conjunto de dados facilita a compreensão e a extração de informações. Exemplos de dados estruturados incluem dados armazenados em um banco de dados relacional, cuja estrutura é formalmente definida por um diagrama entidade-relacionamento (DER), e um documento XML, cuja estrutura está descrita por um documento XSD;
- *Dados não estruturados*, os quais consistem de um conjunto de dados que não possui uma estrutura formalmente definida. Em razão da ausência de uma estrutura formal, a extração de informações de conjuntos de dados não estruturados é muito mais custosa, sob o ponto de vista computacional (SINT et al., 2009; RUSU et al., 2013; SAMBREKAR; RAJPUROHIT; JOSHI, 2018). Exemplos de dados não estruturados incluem imagens, vídeos, áudios e documentos de texto livre;
- *Dados semiestruturados*, os quais possuem uma estrutura implícita e flexível. Ainda que a estrutura de um conjunto de dados semiestruturados não seja rígida, a existência de uma estrutura implícita facilita a extração de informações, se comparado com conjuntos de dados não estruturados. Exemplos de dados semiestruturados incluem dados tabulares armazenados em uma planilha (formato TSV ou CSV).

Abiteboul (1997) apresenta diferentes características de um conjunto de dados semiestruturados. No contexto deste trabalho, destacam-se:

- *Estrutura irregular*, o que indica que uma mesma informação pode ser representada de diferentes formas. Por exemplo, um endereço pode ser representado em partes de um conjunto de dados em formato de *string* e em outras partes do conjunto de dados em uma estrutura diferente, como rua, número, complemento, etc;
- *Estrutura implícita*, o que indica que os (tipos de) dados podem seguir um padrão, ou seja, podem ser homogêneos, porém sem que haja uma gramática formal (*schema*) usada como referência para validar tal padrão/estrutura. Tal estrutura pode ser identificada por meio de softwares de análise. Por exemplo, uma tabela em que cada coluna seja separada por ponto-e-vírgula.

- *Estrutura indicativa*, o que aponta que a estrutura dos dados é autodescrita, ou seja, os dados do próprio conjunto indicam sua estrutura. Contudo, tal indicação não é impositiva, permitindo que outros tipos de dados sejam adicionados, tornando a estrutura heterogênea. Por exemplo, lista de preços expressos essencialmente em reais, acrescida de valores em dólares ao final.

2.2 Dados em bioinformática

Dados de bioinformática obtidos diretamente a partir de um processo de extração das informações biológicas das amostras utilizadas na investigação são chamados de dados brutos. Por exemplo, em investigações sobre análise funcional de genes, os níveis de expressão de cada gene são extraídos por métodos como Microarray de DNA (TREVINO; FALCIANI; BARRERA-SALDAÑA, 2007) e RNA-Seq (WANG; GERSTEIN; SNYDER, 2009). Ambos métodos geram um conjunto de dados os quais devem ser pré-processados (normalizados) a fim de tornar esses dados comparáveis. Posteriormente, dados normalizados devem ser processados por meio de diferentes atividades de análise, tais como, análise diferencial, clusterização e análise funcional. Dados obtidos por meio de uma atividade de análise são chamados de dados processados.

Dados brutos embutem informação sobre o estado de um organismo em um dado contexto (condição experimental). Porém esta informação pode não ser (facilmente) compreendida. O processamento dos dados tem então por objetivo extrair/revelar informações capturadas pelos dados brutos de tal forma que um fenômeno biológico possa ser melhor compreendido. Esta compreensão irá, por sua vez, permitir o avanço do conhecimento.

2.2.1 Organização dos dados

Dados (pré-)processados de bioinformática são semiestruturados por natureza. Tais conjuntos de dados são essencialmente armazenados em arquivos ASCII semiestruturados organizados de acordo com diferentes estilos: i) lista de valores; ii) matriz de valores; e iii) combinação de lista e matriz de valores.

De acordo com o primeiro estilo, lista de valores, os dados são organizados sequencialmente em uma lista, onde cada linha tipicamente corresponde exclusivamente a um

tipo de informação (podendo ser diferente ou não da linha anterior). Um rótulo (tag) pode ser usado para descrever o tipo de conteúdo da linha, seguido pelos dados em si (zero ou mais). Estes ainda podem ser delimitados posteriormente, usando, por exemplo, tabulação ou qualquer outro formato de separação personalizado.

A Figura 1 apresenta um exemplo do estilo de organização de dados em lista. Neste exemplo, rótulos, tais como `MAGE-TAB Version`, `Experimental Factor Name` e `Person Email`, indicam o tipo de dado representado e na sequência são apresentados, em um formato separado por tabulações, os respectivos valores destes dados.

Figura 1 – Dados organizados em lista.

```
Comment[ArrayExpressAccession]  E-MTAB-7834
MAGE-TAB Version    1.1
Comment[Submitted Name] Custom home-build DNA-microarray to compare transcriptomes of wild-
Investigation Title Custom home-build DNA-microarray to compare transcriptomes of wild-type
Experiment Description  The operon for a phosphate-specific ABC transporter (HVO_A0477-80)
Experimental Design genetic modification design
Experimental Design Term Source REF EFO
Experimental Design Term Accession Number   EFO:0001758
Experimental Factor Name    genotype
Experimental Factor Type    genotype
Experimental Factor Term Source REF EFO
Experimental Factor Term Accession Number   EFO_0000513
Person Last Name    Borst
Person First Name   Andreas
Person Mid Initials J
Person Email    Borst@bio.uni-frankfurt.de
Person Phone
Person Fax
Person Address  Max-von-Laue-Str. 9 60438 Frankfurt am Main
Person Affiliation  Goethe-Universität Frankfurt, Campus Riedberg Biozentrum, N240
Person Roles    submitter
Public Release Date 2019-03-26
Protocol Name   P-MTAB-84633    P-MTAB-84634    P-MTAB-84635    P-MTAB-84636    P-MTAB-8463
Protocol Type   sample collection protocol  nucleic acid extraction protocol    nucleic aci
Protocol Term Source REF    EFO EFO EFO EFO EFO EFO
Protocol Term Accession Number  EFO_0005518 EFO_0002944 EFO_0003813 EFO_0003815 EFO_0003814
Protocol Description    To characterize the role of sRNA132 in phosphate-dependent regulati
```

Fonte: Autoria própria.

De acordo com o segundo estilo, matriz de valores, os dados são organizados em um formato de tabela, normalmente delimitada por tabulações (TSV) ou vírgulas (CSV). Cada coluna geralmente corresponde a um tipo diferente de informação, enquanto cada linha armazena valores diferentes para esse tipo de informação. A primeira linha do arquivo pode conter rótulos descrevendo o tipo de informação armazenada em cada coluna.

A Figura 2 apresenta um exemplo do estilo de organização de dados como uma matriz de valores. Nesta figura, rótulos, tais como `Source Name`, `Characteristics [genotype]` e `Material Type`, indicam o tipo de dado da coluna. Cada coluna é separada por uma tabulação.

Figura 2 – Dados organizados em uma matriz.

Source Name	Characteristics[organism]	Characteristics[genotype]	Material Type
Sample 1	Haloferax volcanii	mutant (del srNA132)	whole organism
Sample 2	Haloferax volcanii	wild type	whole organism
Sample 3	Haloferax volcanii	mutant (del srNA132)	whole organism
Sample 4	Haloferax volcanii	wild type	whole organism
Sample 5	Haloferax volcanii	wild type	whole organism
Sample 6	Haloferax volcanii	mutant (del srNA132)	whole organism
Sample 7	Haloferax volcanii	wild type	whole organism
Sample 8	Haloferax volcanii	mutant (del srNA132)	whole organism

Fonte: Autoria própria.

Finalmente, os dois tipos de organização de dados podem ser combinados em um único conjunto de dados, formando um estilo de organização híbrido. As primeiras linhas do conjunto de dados geralmente contêm dados organizados em um formato de lista, seguidos por dados organizados em um formato de matriz.

A Figura 3 apresenta um exemplo desse estilo de organização de dados. As linhas 1 a 15 apresentam o padrão de organização em lista e em seguida, as linhas 17 a 27 apresentam o padrão de organização em matriz.

Figura 3 – Dados organizados de forma híbrida.

```
1  Comment[ArrayExpressAccession]  A-MTAB-658
2  Array Design Name   Custom Haloferax volcanii home-build DNA-microarray Druck_21 (AG Soppa, Goethe-Univer
3  Version Druck_21
4  Provider    Andreas Borst (Borst@bio.uni-frankfurt.de)
5  Comment[Organism]   Haloferax volcanii
6  Comment[Description]    This custom microarray consists of 3678 (3872 total including 194 control probes)
7  Comment[ArrayExpressReleaseDate]    2019-06-01
8  Printing Protocol   Four arrays containing custom ds-DNA reporters(~1000-1500nt) and 60-mer ss-Oligonucle
9  Technology Type spotted ss oligo feature  & spotted ds DNA features
10 Surface Type    polylysine
11 Substrate Type  glass
12 Sequence Polymer Type   DNA
13 Term Source Name
14 Term Source File
15 Comment[AdditionalFile:txt] A-MTAB-658_comments.txt
16
17 [main]
18 Block Column    Block Row   Column  Row Reporter Name   Reporter Group[role]    Control Type
19 1   1   1   1   Cy5/3   Control array control label
20 1   1   2   1   Cy5/3   Control array control label
21 1   1   3   1   Hs-5SrDNA   Control array control biosequence
22 1   1   4   1   Hs-16SrDNA  Control array control biosequence
23 1   1   5   1   Hs-23SrDNA  Control array control biosequence
24 1   1   6   1   rpoS    Control array control biosequence
25 1   1   7   1   sRNA288 Experimental
26 1   1   8   1   HVO_A0637   Experimental
27 1   1   9   1   438-A04 Experimental
```

Fonte: Autoria própria.

2.2.2 Representação de dados processados

Dados processados em bioinformática podem ser usados como entrada em outras atividades de análise ou diretamente interpretados por um biologista. Estes dados podem ser descritos

por um conjunto de rótulos definidos explicitamente e dependentes da ferrramenta de análise usada para produzir os mesmos. Contudo, tal descrição pode frequentemente ser omitida (implícita).

A Figura 4 apresenta um exemplo de dados de expressão gênica obtidos após uma atividade de análise diferencial. Nesta figura os dados estão definidos por meio de um conjunto de rótulos separados por tabulação, da seguinte forma: i) identificador do genes; ii) média dos valores da contagem normalizada; iii) medida de alteração do nível expressão do gene medido pelo método Log2Fold; iv) desvio padrão da medida de expressão; e, finalmente, v) resultado estatístico do teste de hipótese.

Figura 4 – Exemplo de resultado de atividade de análise diferencial.

	gene.id	baseMean	log2FoldChange	lfcSE	stat
2	ENSG00000135677	23177.6085019946	-1.36168976591573	0.0365926150069627	-37.2121469224495
3	ENSG00000119917	10509.4437320546	1.68752181977809	0.0455193136223879	37.0726552200935
4	ENSG00000090520	8463.84847778748	-1.62174802312958	0.0441417227699095	-36.7395724807344
5	ENSG00000141458	14022.6465837973	1.33406779365573	0.0366841127852658	36.3663638661464
6	ENSG00000003249	4702.01942640927	1.82167126623239	0.0501956655676366	36.2914057545021
7	ENSG00000197106	8672.11856119375	1.69026924731219	0.04687535873278	36.0588013192138
8	ENSG00000049245	7209.71776915905	-1.65978762792526	0.0461624382417009	-35.9553717512669
9	ENSG00000117335	5258.3605519377	-1.7872235520884	0.0499609826239953	-35.7723859344197
10	ENSG00000156535	21581.8856308388	1.49508598666535	0.0418779791537096	35.7010060389438
11	ENSG00000107165	918585.30858274	-1.23088017328153	0.0347169552719652	-35.4547270530805
12	ENSG00000132386	10663.7184949654	-1.35932972105123	0.0387855629128718	-35.0473119110026

Fonte: Autoria própria.

Dados processados podem ser disponibilizados em repositórios públicos, tais como ArrayExpress e NCBI Gene Expression Omnibus (GEO), de modo a facilitar a reprodução de experimentos. Nesses repositórios os dados processados seguem formatos de representação padrão. Exemplos de tais formatos incluem MicroArray Gene Expression Tabular (MAGE-TAB) (RAYNER et al., 2006) e Simple Omnibus Format in Text (SOFT) (BARRETT et al., 2007).

MAGE-TAB é um formato projetado para armazenar dados de microarray e de RNA-Seq. O formato MAGE-TAB consiste em quatro tipos diferentes de arquivos: i) arquivo *Array Design Format* (ADF), descrevendo o design do array, isto é, os locais das sequências; ii) arquivo *Investigation Description Format* (IDF), contendo informações gerais sobre a própria investigação; iii) arquivo *Sample and Data Relationship Format* (SDRF), descrevendo as relações entre amostras, matrizes, dados e outros objetos usados ou produzidos na investigação; e, finalmente, iv) arquivos de dados brutos e processados. Os arquivos IDF, ADF e SDRF são arquivos simples, delimitados por tabulações e contendo campos obrigatórios e opcionais. As Figuras 1, 2 e 3, apresentam partes de um arquivo

IDF, SDRF e ADF, respectivamente.

SOFT é um formato de texto simples, baseado em linhas, usado para armazenar e organizar dados genômicos em um único documento. O formato SOFT consiste em três seções: i) uma seção de plataforma, contendo uma descrição resumida do array ou sequenciador usado em um determinado experimento; ii) uma seção de amostra, contendo uma descrição resumida de cada amostra usada no experimento; e iii) uma seção de séries, contendo um conjunto de amostras relacionadas ao estudo. Cada seção contém campos obrigatórios e opcionais.

A Figura 5 apresenta parte de um arquivo SOFT. As linhas 1 a 7 representam a seção de séries. As linhas 9 a 14 representam a seção de plataforma. Finalmente, as linhas 16 a 22 representam a seção de amostra. Os três pontos ("...") indicam a supressão de uma ou mais linhas para concisão.

Figura 5 – Exemplo de parte de um arquivo com formato SOFT.

```
1  ^SERIES = GSE129884
2  !Series_title = ChiCMaxima: a robust and simple pipeline for detection and visualization of
3  !Series_geo_accession = GSE129884
4  !Series_submission_date = Apr 16 2019
5  !Series_last_update_date = Apr 20 2019
6  !Series_summary = Capture Hi-C (CHi-C) is a new technique for assessing genome organization
7  !Series_contact_institute = IGBMC (Institute of Genetics and Molecular and Cellular Biology
8  ...
9  ^PLATFORM = GPL21103
10 !Platform_title = Illumina HiSeq 4000 (Mus musculus)
11 !Platform_geo_accession = GPL21103
12 !Platform_submission_date = Nov 04 2015
13 !Platform_technology = high-throughput sequencing
14 !Platform_organism = Mus musculus
15 ...
16 ^SAMPLE = GSM3724176
17 !Sample_title = mES 4C-seq (Dek)
18 !Sample_geo_accession = GSM3724176
19 !Sample_submission_date = Apr 16 2019
20 !Sample_type = SRA
21 !Sample_growth_protocol_ch1 = J1 mouse ES cells were grown on gamma-irradiated mouse embryo
22 !Sample_molecule_ch1 = genomic DNA
```

Fonte: Autoria própria.

2.3 Web semântica

2.3.1 Visão geral

No início dos anos 90, Tim Berners-Lee propôs um conjunto de tecnologias e protocolos de comunicação que criaram as bases para o surgimento da web. Desde então, a web tem

sido predominantemente utilizada para a apresentação e recuperação de conteúdos por meio de páginas web (documentos HTML) e ligações entre estas páginas (*hyperlinks*).

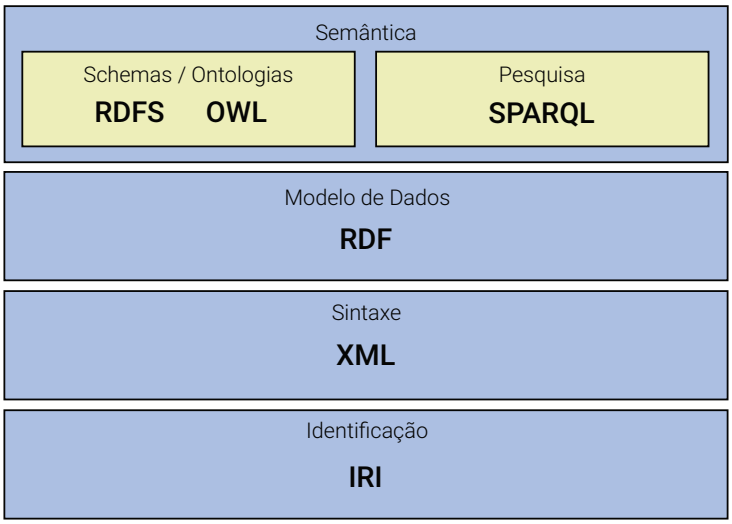
HTML consiste em uma linguagem de marcação que define a forma de apresentação visual de dados de uma página web por meio de um conjunto de tags. Tags são marcações textuais que delimitam o início e o fim de um conteúdo (e.g., <title> para o início e </title> para o fim). Entretanto, tags não agregam informações semânticas aos dados para que um sistema computacional consiga compreender seu conteúdo de forma automática, tornando uma página web fundamentalmente direcionada a pessoas.

Com o objetivo de sanar tal limitação, Tim Berners-Lee propôs uma extensão da web chamada de web semântica, que preserva sua estrutura atual ao mesmo tempo que, por meio de anotações semânticas de conteúdo, permite sua compreensão por sistemas computacionais (BERNERS-LEE, 1994; BERNERS-LEE; HENDLER; LASSILA, 2001). Páginas web devem possuir estrutura e semântica representadas em um formato que possa ser (automaticamente) compreendido, ou seja, as páginas devem possuir estrutura e semântica formalmente definidas, sem a presença de ambiguidades sintáticas e de vocabulário (ANDREAS; KATJA; RALF, 2014).

Softwares automatizados (agentes) podem executar diferentes tarefas, tais como inferências de informação e integração de dados, considerando-se o significado do conteúdo de uma página web. Por exemplo, se uma clínica médica disponibiliza informações semanticamente anotadas sobre os médicos desta clínica e suas agendas de trabalho, um agente automatizado pode explorar (semanticamente) estas informações e identificar quando um determinado médico possui horários disponíveis compatíveis com a agenda do paciente, efetuando assim a marcação automática de uma consulta (BERNERS-LEE; HENDLER; LASSILA, 2001).

A W3C propôs uma arquitetura tecnológica em camadas para a web semântica. Ao longo dos anos tal arquitetura foi alterada à medida em que diferentes tecnologias de suporte foram desenvolvidas. Andreas, Katja e Ralf (2014) propõem uma pilha tecnológica derivada da arquitetura proposta pela W3C, onde, para cada camada desta pilha, uma ou mais tecnologias existentes podem ser utilizadas. A Figura 6 apresenta uma adaptação da pilha tecnológica proposta por Andreas, Katja e Ralf, com foco no conjunto de tecnologias relevantes para este projeto.

Figura 6 – Pilha tecnológica da web semântica.



Fonte: Adaptado de Andreas, Katja e Ralf (2014)

A camada de `Identificação` utiliza o padrão *Internationalized Resource Identifier* (IRI) (The Internet Society, 2005a) na identificação de cada elemento passível de referenciação. O IRI consiste em uma extensão do padrão *Uniform Resource Identifier* (URI) (The Internet Society, 2005b) para a identificação única de recursos na web. Enquanto o URI permite apenas a utilização de um determinado subconjunto de caracteres ASCII, o IRI internacionaliza a tecnologia de identificação de recursos da web, permitindo a utilização de outros caracteres existentes, como, por exemplo, caracteres árabes e japoneses.

A camada de `Sintaxe` provê suporte à definição de dados estruturados, facilitando desta forma a manipulação dos mesmos por meio de sistemas computacionais. A linguagem XML é normalmente utilizada na camada de sintaxe para a representação de dados (W3C, 2008).

A camada de `Modelo de Dados` provê suporte à definição de uma estrutura de dados comum entre diferentes sistemas computacionais de forma a uniformizar o entendimento de uma informação. Entretanto, tal estrutura agrega pouco significado aos dados, sendo portanto insuficiente para a web semântica. O RDF é frequentemente utilizado como tecnologia padrão para a definição de modelos de dados.

A camada de `Semântica` provê suporte à representação formal do conhecimento de modo a permitir que sistemas computacionais consigam compreendê-lo e sejam capazes de inferir novas informações. Esta camada está estruturada em duas subcamadas:

`schemas/ontologias` e `pesquisa`.

A subcamada `Schemas/Ontologias` utiliza ontologias para a representação do conhecimento de um domínio. Uma ontologia representa conhecimento por meio de uma especificação explícita de uma conceitualização (GRUBER, 1995). Neste contexto, conceitualização consiste de uma visão abstrata e simplificada de um domínio, formado por conceitos e relacionamentos que possam existir nesse domínio (GENESERETH; NILSSON, 1987). Diferentes tecnologias têm sido propostas para a estruturação de dados e a representação das ontologias de forma que possam ser interpretadas computacionalmente, tais como RDFS (W3C, 2014b) e OWL (W3C, 2012e).

A subcamada de `Pesquisa` provê suporte à recuperação de informações a partir de um ou mais conjuntos de dados semanticamente anotados. Neste contexto, a principal tecnologia utilizada é o SPARQL, o qual permite a exploração das ligações entre os dados, a aplicação de diversos tipos de filtros de conteúdo, a transformação de conteúdos, dentre outras funções.

2.3.2 Tecnologias

2.3.2.1 Extensible Markup Language

Extensible Markup Language (XML) (W3C, 2008) consiste de um conjunto de regras padronizadas pela *World Wide Web Consortium* (W3C) para o desenvolvimento de dados estruturados (BOSAK; BRAY, 1999). Os objetivos que nortearam a concepção do XML incluem o suporte à criação de documentos legíveis por pessoas, o suporte a uma ampla variedade de softwares e o suporte à definição de uma estrutura formal e concisa.

Documentos XML consistem de texto puro, tornando-os legíveis a uma pessoa usando um simples editor de texto. Assim, documentos XML não necessitam necessariamente de softwares intermediários para sua manipulação. Além disso, a sintaxe XML baseia-se na utilização de tags para a estruturação de seu conteúdo e para a associação informal de uma pseudo estrutura semântica aos dados do documento. A semântica agregada aos dados por meio das tags facilita a compreensão dos mesmos por pessoas, mas não o suficiente para permitir sua compreensão por sistemas computacionais.

O XML também não está limitado a um domínio de conhecimento específico,

podendo ser utilizado em qualquer área do conhecimento que necessite representar informações estruturadas e interoperáveis. Diversas linguagens foram criadas utilizando XML para apresentar dados estruturados em diferentes domínios (GEROIMENK; CHEN, 2003), tais como: *eXtensible Hypertext Markup Language* (XHTML) para apresentação de conteúdo web e *Web Services Description Language* (WSDL) para descrição de serviços web.

Documentos XML possuem estrutura formal e concisa pois utilizam uma estrutura hierárquica de *tags*, indicando o início (e.g., <pesquisa>) e o fim (e.g., </pesquisa>) de uma determinada *tag*. Inicialmente, propôs-se o uso da linguagem *Document Type Definition* (DTD) para garantir tal estrutura. DTD especifica quais elementos e suas propriedades são permitidos em cada documento XML. Entretanto, a linguagem DTD apresentava algumas limitações (LEE; CHU, 2000; DACONTA; OBRST; SMITH, 2003), tais como: a necessidade de aprender uma nova sintaxe para a especificação de um documento DTD, a não utilização de *namespaces*, a incapacidade de importação de documentos externos e o suporte a uma pequena quantidade de tipos de dados nativos em comparação a outras abordagens. Tais limitações contribuíram para que a DTD fosse substituída pela linguagem *XML Schema Definition* (XSD) (W3C, 2012f).

XSD consiste de uma linguagem para a definição da estrutura de um documento XML. Documentos XSD definem um conjunto de critérios que um documento XML deve seguir para que possa ser considerado válido com base em um documento XSD de referência. Esses critérios incluem *tags* e atributos permitidos e como eles devem ser estruturados e utilizados no documento XML. Além disso, são definidos também valores ou tipos de valores permitidos em cada *tag*/atributo. A 7 ilustra a especificação XSD de uma *tag* chamada *pesquisa*.

A *tag pesquisa* representa um tipo complexo formado por outras quatro *tags*: *titulo*, *data-submissao*, *data-publicacao* e *paginas*. Dentre essas *tags*, duas foram definidas como sendo do tipo *date*, uma foi definida como do tipo *integer* e uma foi definida como sendo do tipo *string*. Além disso, a *tag pesquisa* ainda possui um atributo denominado *id* do tipo *string*. A 8 ilustra um trecho de um documento XML criado em conformidade com a especificação XSD da *tag pesquisa*.

A conformidade de um documento XML pode ser avaliada de acordo tanto com os

Figura 7 – Trecho de um documento XSD.

```
<xs:element name="pesquisa">
<xs:complexType>
  <xs:sequence>
    <xs:element name="titulo" type="xs:string" />
    <xs:element name="data-submissao" type="xs:date" />
    <xs:element name="data-publicacao" type="xs:date" />
    <xs:element name="paginas" type="xs:integer" />
  </xs:sequence>
  <xs:attribute name="id" type="xs:string" />
</xs:complexType>
</xs:element>
```

Fonte: Autoria própria.

Figura 8 – Trecho de um documento XML.

```
<pesquisa id="12345">
  <titulo>Pesquisa 12345</titulo>
  <data-submissao>2017-01-01</data-submissao>
  <data-publicacao>2017-02-01</data-publicacao>
  <paginas>30</paginas>
</pesquisa>
```

Fonte: Autoria própria.

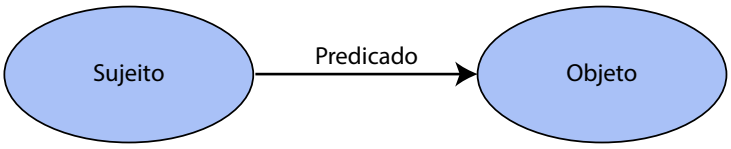
princípios do XML quanto com as especificações de um documento XSD. Desta forma, documentos XML que estão de acordo com os princípios da sintaxe XML (e.g., abertura e fechamento de *tags*) são classificados como bem formados. Documentos XML que estão de acordo com um XSD de referência são classificados como válidos. Todo documento XML deve ser obrigatoriamente bem formado, mas não necessariamente válido (DACONTA; OBRST; SMITH, 2003).

Um documento XSD pode então ser utilizado para duas finalidades (DACONTA; OBRST; SMITH, 2003): i) como modelo para um gerador de documentos XML ou ii) para validar documentos XML. No primeiro caso, utiliza-se um documento XSD como base para a geração automática de documentos XML válidos. No segundo caso, utiliza-se um documento XSD como recurso para verificar se um determinado documento XML está de acordo com sua especificação. Essa validação considera características, tais como: a ordem hierárquica de *tags*, os tipos de dados e os atributos obrigatórios. Essa validação pode ser efetuada quando um documento XML indica o documento XSD utilizado como referência.

2.3.2.2 Resource Description Framework

Resource Description Framework (RDF) (W3C, 2014a) consiste de uma estrutura para representar conhecimento na web utilizando *triplas*. Uma tripla representa dois recursos (sujeito e objeto) e um relacionamento (predicado) definido entre estes recursos. Um recurso representa um conceito concreto ou abstrato de um dado domínio, e.g., DNA, medula ou coração, enquanto um predicado representa um relacionamento existente entre o recurso sujeito e o recurso objeto, e.g., é-um, pertence, processa (ANTONIOU et al., 2012). A Figura 9 representa a estrutura genérica de uma tripla.

Figura 9 – Representação gráfica de uma tripla.



Fonte: Adaptado de W3C (2014a).

Um recurso definido em uma tripla pode ser utilizado como recurso em diversas outras triplas, permitindo assim diversos tipos de relacionamentos ligando um mesmo recurso a outros recursos. IRIs são utilizados para identificar unicamente recursos e relacionamentos. A utilização de identificação única desses elementos permite que não somente eles possam ser referenciados/reutilizados como também evita a duplicação da especificação de recursos e relacionamentos já existentes. Desta forma, o sujeito e o predicado de uma tripla devem indicar o IRI do elemento desejado. Por sua vez, o objeto de uma tripla pode ter como valor tanto um IRI quanto um literal (letras e/ou números) (BERNERS-LEE; HENDLER; LASSILA, 2001; ANDREAS; KATJA; RALF, 2014; W3C, 2014a). Assim, um determinado sujeito pode estar vinculado tanto a um valor primitivo (letras e/ou números) quanto a um outro recurso. Um conjunto de triplas (grafos) é chamado de *dataset*. Um *dataset* deve ser disponibilizado na web para que agentes automatizados possam localizá-lo e utilizá-lo.

A reutilização de IRIs possui a desvantagem de criar redundância dos endereços nos documentos RDF, pois apenas parte dos endereços IRIs são alterados entre os elementos. Desta forma, utiliza-se *namespaces* para a redução da redundância das partes inalteradas dos endereços IRIs. Um *namespace* não possui definição formal no contexto RDF. Porém, informalmente, *namespace* é tratado como um vocabulário de elementos utilizado por

outros documentos (W3C, 2014a). Adicionalmente, o endereço de identificação de um *namespace* RDF pode ser abreviado utilizando um prefixo de modo a simplificar a escrita dos caminhos dos documentos. Tais prefixos abreviam apenas a parte constante dos IRIs, reduzindo desta forma a quantidade de texto em um documento RDF e aumentando a legibilidade do mesmo.

A Tabela 1 apresenta alguns exemplos de *namespaces* frequentemente utilizados em documentos RDF (W3C, 2014a). Esses vocabulários são distribuídos e definidos pela W3C fazendo com que todos os documentos RDF utilizem/compreendam as mesmas palavras-chave. O significado desses vocabulários independe da sintaxe utilizada para a especificação do documento RDF.

Tabela 1 – Exemplos de *namespaces* (adaptado de W3C (2014a)).

Prefixo *Namespace*	*Namespace* IRI
rdf	http://www.w3.org/1999/02/22-rdf-syntax-ns#
xsd	http://www.w3.org/2001/XMLSchema#

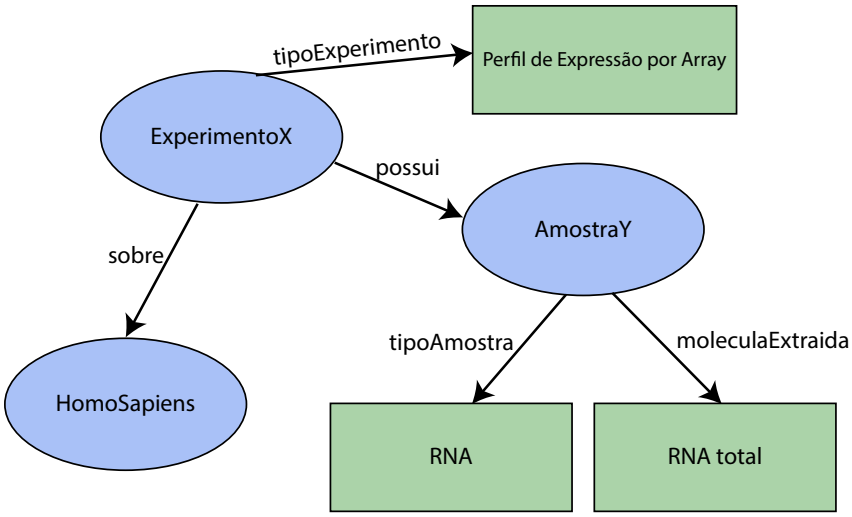
Um grafo RDF consiste de um conjunto de triplas visualmente representado por meio de uma rede de nós conectados por arestas. Em um grafo RDF, os nós são os recursos e as arestas são os predicados. Nós normalmente são representados por elipses, enquanto os predicados por setas. Caso o objeto de uma tripla seja um literal, este é representado por um retângulo.

A Figura 10 ilustra um grafo RDF contendo cinco triplas. As triplas `ExperimentoX-sobre-HomoSapiens` e `ExperimentoX-possui-AmostraY` relacionam dois recursos, enquanto as triplas `ExperimentoX-tipoExperimento-Perfil de Expressão por Array`, `AmostraY-tipoAmostra-RNA` e `AmostraY-moleculaExtraida-RNA total` relacionam um recurso a um valor literal.

2.3.2.3 Resource Description Framework Schema

Resource Description Framework Schema (RDFS) (W3C, 2014b) consiste em uma extensão do RDF para a representação de informações semânticas por meio da especificação de critérios a serem respeitados por documentos RDF. Essa especificação resulta em um conjunto de elementos passíveis de utilização por documentos RDF (ANTONIOU et al., 2012). Desta forma, documentos RDF passam a possuir uma estrutura formal, garantindo

Figura 10 – Grafo RDF.



Fonte: Autoria própria.

a correta representação do conhecimento desejado. A utilização de um documento RDFS como referência para um documento RDF garante que as triplas RDF sejam limitadas à especificação RDFS, promovendo assim uma maior conformidade com o conhecimento existente em um dado domínio.

Um documento RDFS utiliza o conceito de triplas RDF para a especificação de conhecimento. Entretanto, os recursos e predicados possuem nomenclaturas diferentes. Um recurso é chamado de classe, enquanto que um predicado é chamado de propriedade. Um documento RDFS define as características de cada classe e de cada propriedade. Desta forma, o RDFS provê suporte à especificação de um vocabulário de classes e suas hierarquias, assim como a especificação de propriedades a serem aplicadas a estas classes (DECKER et al., 2000).

As especificações do documento RDFS são definidas utilizando um vocabulário de termos definidos pela W3C, chamado de vocabulário RDFS. Os principais termos do vocabulário RDFS são: i) *Class*, para a definição de uma nova classe; ii) *subClassOf*, para a definição de uma subclasse (hierarquia); e iii) *subPropertyOf*, para a definição de uma subpropriedade (hierarquia). Além disso, os termos *domain* e *range* limitam a forma com que uma propriedade pode ser definida em uma tripla RDF. *Domain* define o tipo de sujeito permitido em uma propriedade, enquanto *range* define o tipo de objeto permitido em uma propriedade (DECKER et al., 2000).

A nomenclatura utilizada no RDFS possui semelhança com a nomenclatura utili-

zada no desenvolvimento de software orientado a objetos. Características como classes, hierarquia de classes e propriedades fazem parte dos conhecimentos básicos para o desenvolvimento de software orientado a objetos. Entretanto, tais características possuem diferenças quando utilizadas no RDFS. No desenvolvimento orientado a objetos as propriedades são definidas dentro das classes e qualquer alteração nestas propriedades resultam na alteração da classe. No RDFS novas propriedades podem ser adicionadas às classes sem que haja a necessidade de alterar a especificação original da classe, permitindo desta forma que outros documentos RDFS possam estender a semântica da classe externamente (ANTONIOU et al., 2012). Desta forma, novas características e relacionamentos não influenciam instâncias (dados em documentos RDF) já existentes das classes, pois o documento RDFS de referência não foi alterado.

A 11 apresenta um exemplo de uma especificação RDFS. Esta especificação foi utilizada como base para a criação do grafo RDF apresentado na Figura 10. De acordo com esta especificação um `experimento` possui `amostra`, está associado a um `organismo` e possui uma propriedade `tipoExperimento`. Os elementos `experimento`, `amostra` e `organismo` são representados como classes.

A propriedade `sobre` representa um relacionamento definido entre as classes `experimento` e `organismo`. Um `experimento` contém uma propriedade `tipoExperimento` do tipo `string`. A propriedade `possui` representa um relacionamento definido entre as classes `experimento` e `amostra`. Uma `amostra` contém duas propriedades, ambas do tipo `string`: `tipoAmostra` e `moleculaExtraida`.

2.3.2.4 Web Ontology Language

Embora o RDFS permita a representação de informações semânticas em um domínio do conhecimento, existem características que não podem ser expressas usando esta linguagem (ANTONIOU et al., 2012), tais como disjunção de valores e propriedades transitivas. Estas e outras limitações motivaram o desenvolvimento de uma linguagem ontológica com maior expressividade semântica (ANTONIOU; HARMELEN, 2008), a linguagem *Web Ontology Language* (OWL) (W3C, 2012c).

OWL consiste de uma linguagem criada para a especificação de ontologias na web semântica. Uma ontologia OWL é composta por entidades, expressões e axiomas. Entidades

Figura 11 – Especificação RDFS.

```
1   <rdfs:Class rdf:ID="Organismo">
2     <rdfs:subClassOf rdf:resource="http://www.w3.org/2000/01/rdf-schema#
          Class"/>
3   </rdfs:Class>
4   <rdfs:Class rdf:ID="Experimento">
5     <rdfs:subClassOf rdf:resource="http://www.w3.org/2000/01/rdf-schema#
          Class"/>
6   </rdfs:Class>
7   <rdfs:Class rdf:ID="Amostra">
8     <rdfs:subClassOf rdf:resource="http://www.w3.org/2000/01/rdf-schema#
          Class"/>
9   </rdfs:Class>
10  <rdf:Property rdf:ID="sobre">
11    <rdfs:range rdf:resource="#Organismo"/>
12    <rdfs:domain rdf:resource="#Experimento"/>
13  </rdf:Property>
14  <rdf:Property rdf:ID="tipoExperimento">
15    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
16    <rdfs:domain rdf:resource="#Experimento"/>
17  </rdf:Property>
18  <rdf:Property rdf:ID="moleculaExtraida">
19    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
20    <rdfs:domain rdf:resource="#Amostra"/>
21  </rdf:Property>
22  <rdf:Property rdf:ID="possui">
23    <rdfs:range rdf:resource="#Amostra"/>
24    <rdfs:domain rdf:resource="#Experimento"/>
25  </rdf:Property>
26  <rdf:Property rdf:ID="tipoAmostra">
27    <rdfs:domain rdf:resource="#Amostra"/>
28    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
29  </rdf:Property>
```

Fonte: Autoria própria.

representam o mundo real por meio de classes, propriedades e indivíduos. Indivíduos representam elementos concretos ou abstratos de um domínio. Classes representam categorias de indivíduos. Propriedades representam relacionamentos entre dois indivíduos, também chamados de objetos. Propriedades são segmentadas em propriedades de objetos e propriedades de tipo de dados. A primeira relaciona um objeto a um outro objeto, enquanto que a segunda relaciona um objeto a um tipo de dados (letras e/ou números) (W3C, 2012c).

Expressões combinam entidades (elementos atômicos) com a finalidade de criar descrições complexas do conhecimento. As entidades utilizadas em uma expressão são avaliadas de acordo com os critérios estabelecidos pela expressão. Uma expressão torna-se verdadeira quando as entidades e seus respectivos valores atendem aos critérios estabelecidos pela expressão. Expressões podem ser utilizadas tanto para identificar instâncias dessas expressões quanto para compor novas expressões (W3C, 2012a; W3C, 2012c).

Finalmente, axiomas representam uma verdade definida (afirmação) sobre um domínio. A definição de um axioma pode levar em consideração definições de classes (e.g., classe `Mulher` é subclasse de `Pessoa`, logo, toda mulher é uma pessoa), propriedades (e.g., propriedade `Contem` com *domain* `Organismo` e *range* `Proteina`, logo, organismos podem conter proteína), entre outros (W3C, 2012a; W3C, 2012c).

Novos conhecimentos podem ser inferidos por meio de motores (mecanismos) de inferência. Mecanismos de inferências utilizam axiomas, expressões e entidades para explorar o conhecimento representado por meio de deduções lógicas. Desta forma, estes mecanismos permitem a realização (automática) de inferências com base no conhecimento previamente representado por uma ontologia (W3C, 2012e).

Existem duas abordagens distintas para a especificação de semântica em OWL (W3C, 2012c): *Direct Semantics* e *RDF-Based Semantics*. Direct Semantics utiliza lógica descritiva (subconjunto da lógica de predicados) para a definição da semântica dos dados. RDF-Based Semantics utiliza os mecanismos de representação semântica definidos em documentos RDFS e adiciona novos mecanismos, tais como conectores booleanos e cadeias de sub-propriedades (W3C, 2012e; W3C, 2012b).

Documentos que utilizam RDF-Based Semantics são conhecidos por documentos do tipo OWL *Full*. Estes documentos permitem o uso de todos os recursos da linguagem OWL, o que garante total compatibilidade com RDF (sintaticamente e semanticamente). Como desvantagem desta abordagem, destaca-se a indecidibilidade nas inferências em razão da complexidade envolvida nos recursos da linguagem. Documentos que utilizam Direct Semantics são conhecidos por documentos OWL *Description Logic* (OWL DL). OWL DL consiste em um conjunto limitado do OWL *Full*. Dado que OWL DL não possui todos os recursos da linguagem OWL, especificações OWL DL são decidíveis. Por outro lado, a ausência de alguns recursos do OWL *Full* faz com que OWL DL perca a compatibilidade integral com documentos RDF (ANTONIOU et al., 2012; W3C, 2012e).

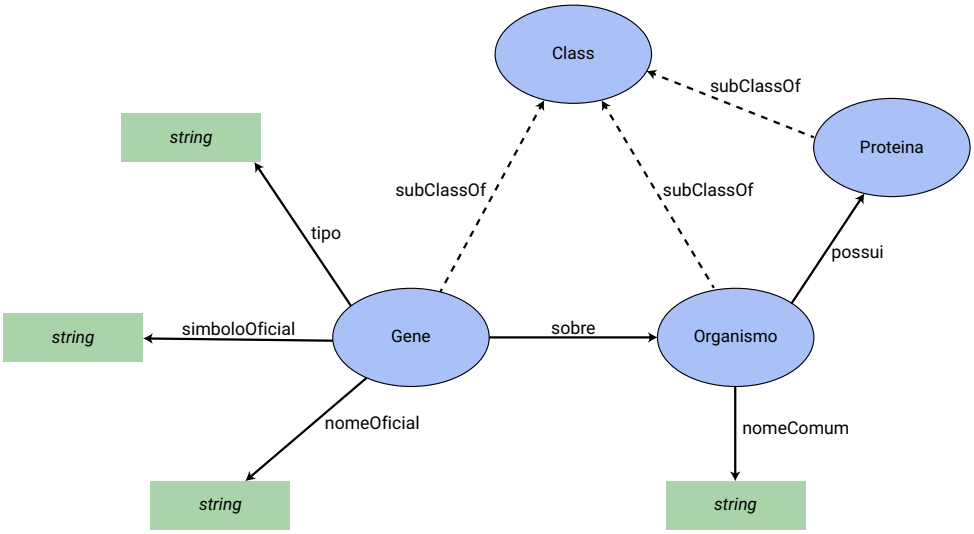
Um perfil de linguagem representa uma sublinguagem definida com um propósito específico. Neste sentido, OWL possui três perfis de linguagem distintos e independentes uns dos outros (W3C, 2012c): OWL *Existential Language* (OWL EL), OWL *Query Language* (OWL QL) e OWL *Rule Language* (OWL RL).

O perfil OWL EL foi definido para a especificação de ontologias que contêm uma

grande quantidade de propriedades e/ou classes, de modo a permitir a realização de inferência em tempo polinomial. O perfil OWL QL foi definido para a especificação de ontologias que contêm uma grande quantidade de dados, sendo a tarefa de consulta (veja subseção 2.3.2.5) desses dados mais importante do que a realização de inferências. Finalmente, o perfil OWL RL foi definido para a especificação de ontologias que requerem que o processamento de inferências seja capaz de escalar sem sacrificar muito da sua capacidade de expressão semântica. Além disso, OWL RL pode ser implementado utilizando máquina de inferência baseada em regras (W3C, 2012d).

A Figura 12 ilustra graficamente uma especificação OWL. Círculos representam classes, enquanto linhas representam propriedades. A propriedade *subClassOf* especialmente é graficamente representada por uma seta pontilhada. Os nomes das propriedades e os tipos de dados são ilustrados por retângulos. De acordo com esta especificação, uma classe *Gene* possui três propriedades do tipo *string* (tipo, simboloOficial e nomeOficial) e uma propriedade *sobre* ligando a uma classe *Organismo*. A classe *Organismo* possui uma propriedade do tipo *string* (nomeComum) e uma propriedade *possui* ligando a uma classe *Proteina*.

Figura 12 – Representação gráfica OWL.



Fonte: Autoria própria.

A 13 um trecho da especificação OWL usada como base para a Figura 12. De acordo com esta especificação as classes foram definidas como disjuntas umas das outras, ou seja, um mesmo indivíduo não pode ter como tipo duas dessas classes simultaneamente (veja por exemplo as linhas 27, 28 e 32). A representação de tal característica não era

possível usando o RDFS.

Figura 13 – Exemplo de uma especificação OWL.

```
1   <owl:ObjectProperty rdf:about="http://example.org/ontologiaOWL#possui">
2       <rdfs:domain rdf:resource="http://example.org/ontologiaOWL#Organismo"
            />
3       <rdfs:range rdf:resource="http://example.org/ontologiaOWL#Proteina"/>
4   </owl:ObjectProperty>
5   <owl:ObjectProperty rdf:about="http://example.org/ontologiaOWL#sobre">
6       <rdfs:domain rdf:resource="http://example.org/ontologiaOWL#Gene"/>
7       <rdfs:range rdf:resource="http://example.org/ontologiaOWL#Organismo"/>
8   </owl:ObjectProperty>
9   <owl:DatatypeProperty rdf:about="http://example.org/ontologiaOWL#nomeComum
        ">
10      <rdfs:domain rdf:resource="http://example.org/ontologiaOWL#Organismo"
            />
11      <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
12  </owl:DatatypeProperty>
13  <owl:DatatypeProperty rdf:about="http://example.org/ontologiaOWL#
        nomeOficial">
14      <rdfs:domain rdf:resource="http://example.org/ontologiaOWL#Gene"/>
15      <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
16  </owl:DatatypeProperty>
17  <owl:DatatypeProperty rdf:about="http://example.org/ontologiaOWL#
        simboloOficial">
18      <rdfs:domain rdf:resource="http://example.org/ontologiaOWL#Gene"/>
19      <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
20  </owl:DatatypeProperty>
21  <owl:DatatypeProperty rdf:about="http://example.org/ontologiaOWL#tipo">
22      <rdfs:domain rdf:resource="http://example.org/ontologiaOWL#Gene"/>
23      <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
24  </owl:DatatypeProperty>
25  <owl:Class rdf:about="http://example.org/ontologiaOWL#Gene">
26      <rdfs:subClassOf rdf:resource="http://www.w3.org/2000/01/rdf-schema#
            Class"/>
27      <owl:disjointWith rdf:resource="http://example.org/ontologiaOWL#
            Organismo"/>
28      <owl:disjointWith rdf:resource="http://example.org/ontologiaOWL#
            Proteina"/>
29  </owl:Class>
30  <owl:Class rdf:about="http://example.org/ontologiaOWL#Organismo">
31      <rdfs:subClassOf rdf:resource="http://www.w3.org/2000/01/rdf-schema#
            Class"/>
32      <owl:disjointWith rdf:resource="http://example.org/ontologiaOWL#
            Proteina"/>
33  </owl:Class>
34  <owl:Class rdf:about="http://example.org/ontologiaOWL#Proteina">
35      <rdfs:subClassOf rdf:resource="http://www.w3.org/2000/01/rdf-schema#
            Class"/>
36  </owl:Class>
```

Fonte: Autoria própria.

2.3.2.5 SPARQL Protocol And RDF Query Language

SPARQL Protocol And RDF Query Language (SPARQL) (W3C, 2013) consiste de uma linguagem e de um protocolo que provê suporte à pesquisa e manipulação de triplas RDF (CURÉ; BLIN, 2014). A linguagem SPARQL foi concebida para permitir que triplas RDF possam ser recuperadas, filtradas e manipuladas. Por sua vez o protocolo SPARQL é

utilizado para o desenvolvimento de um motor de pesquisa SPARQL. Um motor SPARQL permite o intercâmbio e o processamento de instruções SPARQL (DUCHARME, 2013).

A linguagem SPARQL provê suporte à realização de diversas operações, tais como seleção de dados, subconsultas e atribuição de valores. Uma das características do SPARQL consiste em considerar a semântica dos dados para a execução das operações, permitindo, por exemplo, efetuar uma busca por uma propriedade e recuperar automaticamente os dados das propriedades filhas (W3C, 2013).

A sintaxe do SPARQL assemelha-se sintaticamente ao SQL, e sendo que esta linguagem está para o RDF assim como a linguagem SQL está para bancos de dados relacionais (CURÉ; BLIN, 2014). Entretanto, o funcionamento interno dessas linguagens difere. Enquanto o SQL trabalha com a junção de tabelas, o SPARQL utiliza a exploração de nós RDF (navegando de nó em nó, acessando os IRIs) para a recuperação de informações (CURÉ; BLIN, 2014). A sintaxe do SPARQL utiliza alguns elementos, tais como cláusulas, *namespaces* e variáveis.

A Listagem 1 ilustra uma consulta simples para a recuperação do nome comum de um *Organismo*. As cláusulas presentes na figura são: *PREFIX*, *SELECT* e *WHERE*. A cláusula PREFIX define um *namespace*, a cláusula WHERE define os filtros que devem ser aplicados nos dados e a cláusula SELECT determina quais campos devem ser apresentados no resultado (DUCHARME, 2013). *Namespace* são definidos por meio da associação ‘rdf:’, ‘owl:’, ‘rdfs:’, ‘xsd:’ e ‘onto:’ com seus respectivos IRIs. O *namespace* ‘onto:’ é uma abreviação para o IRI da ontologia OWL, onde estão especificados as classes e propriedades enquanto que os outros *namespaces* são vocabulários padrões. Esta consulta tem por objetivo retornar os sujeitos e objetos das triplas que contenham como predicado ‘onto:nomeComum’.

As consultas SPARQL devem ser executadas em um software de processamento de instruções SPARQL (motor). Existem motores tanto para a execução de consultas locais (e.g., ARQ (Apache Software Foundation, 2017a)) quanto para a execução de consultas na web (e.g., SNORQL[1]). Motores locais permitem que sejam carregados conjuntos de dados presentes na máquina local, possibilitando assim que sejam feitos testes e exploração de um conjunto de dados ainda não disponível na web. Por sua vez, motores web não requerem a instalação local de softwares no computador para que seja feita a exploração

[1] http://dbpedia.org/snorql/

```
1  PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
2  PREFIX owl: <http://www.w3.org/2002/07/owl#>
3  PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
4  PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
5  PREFIX onto: <http://www.example.org/ontologiaOWL#>
6  SELECT ?subject ?object WHERE {
7
8    ?subject onto:nomeComum ?object
9
10 }
```

Listagem 1 – Exemplo de consulta SPARQL.

dos dados (DUCHARME, 2013).

Após a realização das consultas, os resultados das operações efetuadas podem ser exportados. Os dados exportados podem ser formatados seguindo um dos tipos padronizados: XML, JSON, *Comma Separated Values* (CSV) ou *Tab Separated Values* (TSV) (W3C, 2013; DUCHARME, 2013). Desta forma, os dados recuperados pela consulta SPARQL podem ser manipulados por outros softwares para análises subsequentes.

Triplas RDF podem ser armazenadas em bancos de dados especializados chamados *triplestores*. Ao contrário de bancos de dados relacionais, os quais em geral apresentam baixo desempenho no armazenamento e recuperação de triplas RDF, *triplestores* representam uma solução otimizada para este propósito (DUCHARME, 2013). Exemplos de *triplestores* incluem Virtuoso[2] e TDB (Apache Software Foundation, 2017b)

2.4 Dados abertos ligados

2.4.1 Visão geral

A web semântica tem como objetivo permitir a associação entre dados e informações relacionadas, de forma que computadores consigam explorar o conhecimento associado aos dados presentes em documentos HTML. A associação de dados semanticamente anotados forma a chamada web de dados. Na web dos dados conceitos associados a dados presentes em diferentes tipos de documentos são explicitamente ligados a outros conceitos

[2] https://virtuoso.openlinksw.com/

(BERNERS-LEE, 2006). Essas associações entre conceitos podem ser chamadas de dados ligados. Dados disponibilizados na web são considerados dados ligados quando seguem quatro princípios (BERNERS-LEE, 2006): i) utilizam IRIs (anteriormente chamado de URI) para a identificação de conceitos; ii) utilizam IRI resolvível por protocolo HTTP; iii) estão associados a informações úteis; e iv) incluem outras IRIs para que sejam descobertos novos conceitos.

A utilização de um IRI para identificar univocamente um conceito facilita a referenciação deste elemento. Assim como uma URL identifica univocamente uma página HTML, um IRI deve identificar univocamente um recurso na web de dados. Ao utilizar um IRI resolvível por protocolo HTTP, procura-se evitar o uso de outros protocolos/tecnologias (e.g. FTP, *urn:schemas*). Adicionalmente, ao acessar este IRI pode-se recuperar um conjunto relevante de informações relacionadas. Finalmente, ao acessar um recurso pode-se identificar e explorar todos os outros recursos relacionados ao mesmo.

O termo dados abertos é utilizado para indicar um conjunto de dados disponibilizado na web com licença livre para acesso de terceiros. Assim, Dados Abertos Ligados (DAL) ou Dados Abertos Conectados (do inglês *Linked Open Data*) remete à junção das características de dados abertos e de dados ligados. DAL permite o reuso de conceitos de uma ontologia, facilita o acesso a informações semânticas na web e possibilita a descoberta e a inferência de novas informações (BIZER, 2009; BERNERS-LEE, 2006).

O reuso de conceitos é obtido mediante a utilização de elementos existentes em diferentes ontologias, evitando a duplicação dos mesmos. O reuso de conceitos proporciona a diminuição do esforço e da complexidade na elaboração de novas ontologias. A criação de referências entre conjuntos de dados facilita o acesso a novas informações semânticas, enriquecendo as informações já existentes. Tais referências representam dados relacionados ou informações complementares em relação a um determinado conceito. A descoberta de informações consiste na exploração de referências entre os conjuntos de dados em busca explícita por novas informações. A inferência de informações utiliza diferentes tipos de relacionamentos definidos entre os conceitos, permitindo que uma máquina de inferência consiga percorrer o conjunto de dados e descobrir (inferir) novos conhecimentos (BIZER, 2009).

Uma das limitações da web consiste em que a maioria das informações disponíveis

estão armazenadas em diferentes bases de dados. Essa distribuição dificulta a utilização desses dados, havendo assim frequentemente a necessidade do desenvolvimento de softwares capazes de localizar, acessar, converter e combinar diferentes fontes de dados. A utilização de DAL resolve potencialmente esse problema de distribuição de dados, pois dados de diversas fontes podem ser utilizados sem a execução de processos específicos para a recuperação da informação (WOOD et al., 2013). Entretanto, deve haver um esforço para a disponibilização dos dados em formato DAL. Em casos de bases de dados existentes, há a necessidade de transformação desses dados.

2.4.2 Classificação de dados na web

Com o objetivo de classificar de forma geral os dados disponíveis na web, Tim Berners-Lee propôs uma categorização baseada em cinco estrelas (BERNERS-LEE, 2006) (veja Figura 14). Cada categoria adiciona novos critérios a partir da categoria imediatamente anterior.

Figura 14 – Representação gráfica da classificação em estrelas.

Fonte: Autoria própria.

Dados classificados com uma estrela encontram-se disponíveis livremente na web em qualquer formato, e.g., jpg, pdf, doc. Nesta categoria, o principal fator consiste na disponibilização dos dados na web e na ausência de restrições quanto ao uso dos mesmos, permitindo que qualquer pessoa possa compartilhá-los, distribuí-los ou adaptá-los. A utilização de licença livre não indica a renúncia de autoria, ou seja, mantém-se a obrigação de citar a autoria dos dados.

Dados classificados com duas estrelas encontram-se disponíveis em arquivos (semi) estruturados processáveis computacionalmente, e.g., arquivo XSL (Excel). Desta forma, dados de uma tabela armazenada em um arquivo XSL podem ser processados computacio-

nalmente, ao contrário de uma imagem/fotografia de uma tabela na qual não armazena os dados propriamente ditos.

Arquivos que utilizam formatos proprietários podem limitar o acesso aos dados, pois os mesmos estão contingenciados sob uma licença restrita. Dados classificados com três estrelas encontram-se disponíveis em arquivos (semi)estruturados, usando-se formatos não proprietários. Com a utilização de formatos não proprietários, o acesso atual ou futuro aos dados está garantido. Desta forma, dados originalmente armazenados em arquivos XSL (formato proprietário) podem, por exemplo, ser armazenados em arquivos CSV ou TSV (formatos de licença livre).

A W3C padroniza tipos de arquivos e suas formas de utilização a fim de homogeneizar as tecnologias utilizadas pela comunidade. Dados classificados com quatro estrelas encontram-se disponíveis em um formato estabelecido pela W3C, e.g., RDF. Tim Berners-Lee recomenda a utilização do RDF para a representação dos dados e a utilização do SPARQL para explorar os conjuntos de dados já existentes.

Dados classificados com cinco estrelas encontram-se disponíveis de forma ligada (BERNERS-LEE, 2006; ISOTANI; BITTENCOURT, 2015). Dados ligados podem não apenas referenciar outros conjuntos de dados mas também ser referenciados pelos mesmos. Desta forma, para que os dados sejam caracterizados como DAL, eles devem ser classificados com cinco estrelas. Tim Berners-Lee ainda afirma que esse sistema de classificação de estrelas é específico para dados abertos, pois mesmo que seja benéfico o uso de dados ligados em um ambiente interno, a primeira estrela representa justamente a disponibilidade irrestrita dos dados na web (BERNERS-LEE, 2006).

2.4.3 Repositórios de dados abertos ligados

DAL podem ser utilizados para diversas finalidades, como, por exemplo, na complementação de informações disponibilizadas em *websites*. Uma equipe de desenvolvimento da BBC (portal de notícias) resolveu utilizar DAL no desenvolvimento de três portais: *BBC Programmes*[3], *BBC Music*[4] e *BBC Nature Wildlife*[5]. Somente o BBC Programmes cria páginas web para mais de 1.500 programas de TV e rádio diariamente. No desenvolvimento

[3] http://www.bbc.co.uk/programmes
[4] http://www.bbc.co.uk/music
[5] http://www.bbc.co.uk/nature/wildlife

de tais portais foram utilizados DAL para consumir dados disponíveis na web e disponibilizá-los nos portais desenvolvidos. Ao utilizar DAL, o gerenciamento deste conteúdo teve que ser alterado, pois os editores devem alterar os dados diretamente na fonte (possivelmente em websites terceiros) (WOOD et al., 2013).

DBpedia[6] consiste em um dos maiores conjuntos DAL atualmente disponíveis, sendo amplamente utilizado como fonte de pesquisas (SIMPERL; BU; LI, 2015). O DBpedia foi criado com o objetivo de transformar dados da Wikipedia em dados da web semântica, extraindo informações de mais de 111 edições do Wikipedia disponíveis em diferentes idiomas. Somente da edição americana, o conjunto DAL DBpedia converteu mais de 5.5 milhões de recursos: 1.5 milhão de pessoas, 840 mil lugares, 496 mil obras (álbuns de música, filmes e video games), 286 mil organizações, 306 mil espécies, 58 mil plantas e 6 mil doenças. O DBpedia tornou-se referência da web semântica e um dos principais conjuntos de dados do Projeto *Linking Open Data* (Projeto LOD) (Linking Open Data Project, 2017).

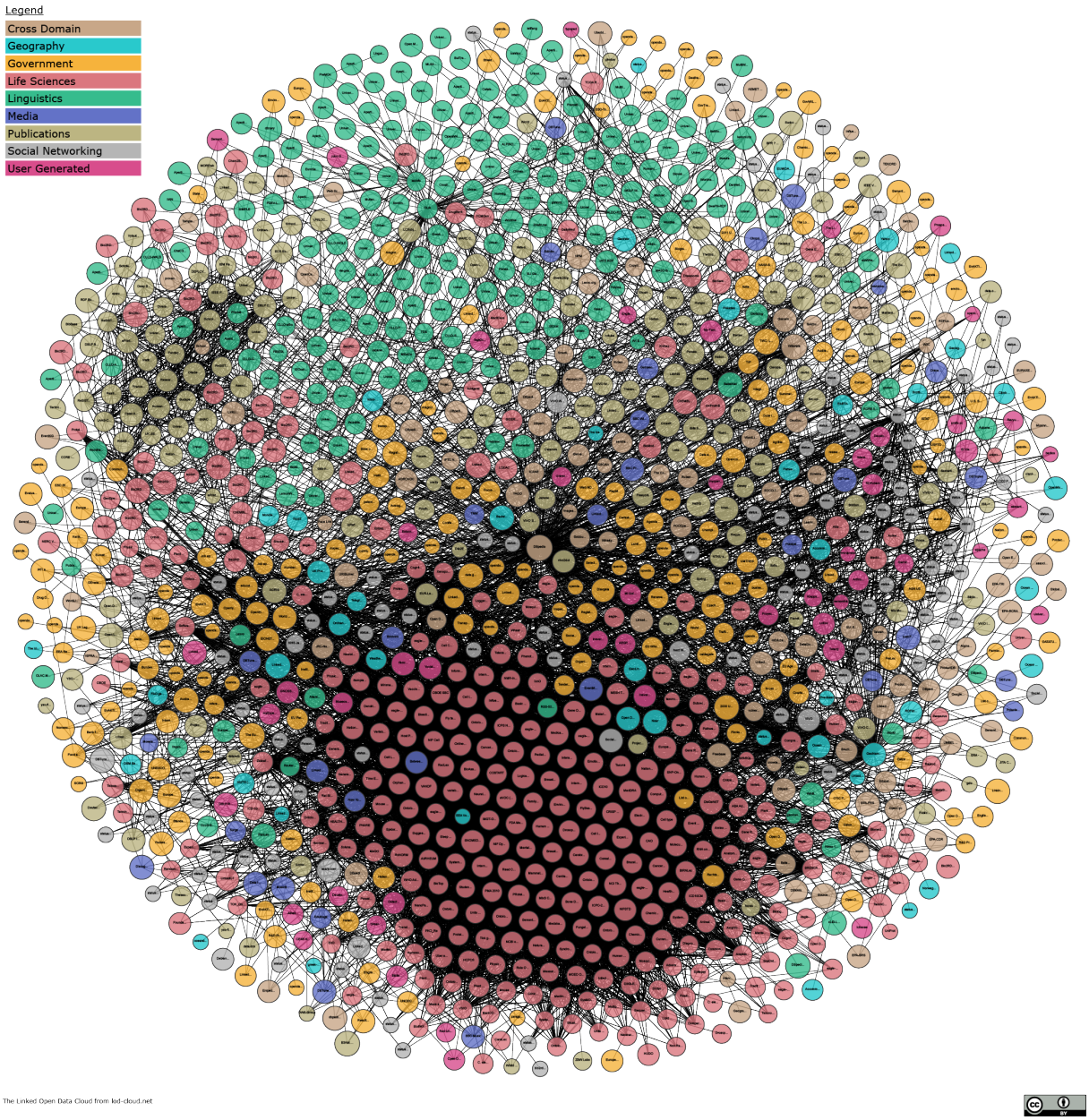
O projeto LOD consiste em uma atividade comunitária iniciada pelo grupo da W3C *Semantic Web Education and Outreach* (SWEO) (WOOD et al., 2013; Semantic Web Education and Outreach Interest Group, 2017). Este projeto foi criado com o objetivo de incentivar a transformação de dados da web tradicional em dados da web semântica. Dados da web tradicional são transformados em dados da web semântica em três passos (BIZER, 2009; BIZER; HEATH; BERNERS-LEE, 2009): i) identificação dos conjuntos de dados com licença livre; ii) transformação desses conjuntos de dados em documentos RDF aplicando os princípios DAL; e iii) publicação desses documentos RDFs na web. O projeto conta com conjuntos de dados de diversas temáticas, como, por exemplo, localizações geográficas, programas de rádio, publicações científicas, companhias, livros, pessoas, dados governamentais e ciências da vida.

A Figura 15 ilustra um diagrama chamado de *nuvem LOD* (MCCRAE et al., 2019) que representa as ligações entre os conjuntos de dados DAL identificados pelo projeto LOD. A nuvem LOD representa alguns dos mais de 1.200 conjuntos de dados identificados pelo Projeto LOD. As linhas entre os círculos indicam que há ao menos 50 ligações entre os conjuntos de dados.

A área biomédica já conta com uma grande quantidade de repositórios de dados

[6] http://wiki.dbpedia.org/

Figura 15 – Nuvem LOD.



Fonte: McCrae et al. (2019)

abertos ligados. O projeto Bio2RDF[7] consiste em uma iniciativa para a expansão destes dados. Bio2RDF tem como objetivo a transformação de dados de portais públicos da área de ciências da vida para DAL. Até seu último relatório (DUMONTIER et al., 2014), o projeto Bio2RDF havia transformado 30 bases de dados em DAL, resultando em aproximadamente 11 bilhões de triplas. Dentre essas bases transformadas encontram-se os dados dos portais PudMed, iProsClass, NCBI Gene e do GO Triplestore.

O conjunto de dados do portal PubMed[8] consiste na maior base de dados transformada pelo projeto Bio2RDF, contendo mais de 5 bilhões de triplas. O portal PubMed disponibiliza um conjunto de mais de 26 milhões documentos específicos da área biomédica, contendo, entre outros, citações da literatura, jornais (periódicos) e livros (National Center for Biotechnology Information, 2016).

O conjunto de dados do portal iProClass[9] contém mais de 3 bilhões de triplas, sendo o segundo maior do Bio2RDF. A base de dados iProClass provê descrições de proteínas, classificação funcional de proteínas e classificação estrutural de proteínas. Essas informações são vinculadas a mais de 50 bancos de dados de diversos domínios: sequências protéicas, interações proteína-proteína, genes, ontologias, taxonomias, entre outros (HUANG et al., 2003).

O conjunto de dados do portal *National Center for Biotechnology Information's Gene* (NCBI Gene)[10] contém mais de 1 bilhão de triplas, sendo o terceiro maior do Bio2RDF. O Portal NCBI Gene centraliza informações biomédicas sobre organismos, originárias de diversas fontes de dados. Dentre as informações integradas, há, por exemplo, nomenclaturas, sequências, fenótipos e interações (BROWN et al., 2015).

Além dos conjuntos de dados transformados pelo projeto Bio2RDF, existem conjuntos de dados disponibilizados de forma autônoma pelos próprios mantenedores dos dados, como, por exemplo, o GO Triplestore e o Wikipathways.

O conjunto de dados do portal *Gene Ontology Public Triple Store* (GO Triplestore)[11] armazena as informações do *Gene Ontology Causal Activity Model* (GO-CAM). GO-CAM consiste na interligação de múltiplas anotações do Gene Ontology (função molecular,

[7] http://bio2rdf.org/
[8] https://www.ncbi.nlm.nih.gov/geo/
[9] http://pir.georgetown.edu/pirwww/dbinfo/iproclass.shtml
[10] https://www.ncbi.nlm.nih.gov/gene
[11] http://rdf.geneontology.org

processo biológico ou componente celular) a fim de fornecer uma informação mais completa acerca das funções moleculares (Gene Ontology, 2019).

O conjunto de dados do portal *Wikipathways*[12] armazena informações de vias metabólicas com curadoria colaborativa da comunidade científica. Neste portal são mantidas informações de mais de 2.600 vias metabólicas de 25 espécies diferentes, utilizadas por diferentes comunidades de pesquisa em ciências da vida. Tais informações também são anotadas utilizando identificadores de bancos de dados de referência, como, por exemplo, Entrez Gene, Ensembl, UniProt, entre outros (WAAGMEESTER et al., 2016; SLENTER et al., 2017). Essas referências são utilizadas para vincular os dados do Wikipathways com os dados já existentes nos bancos de dados de referência.

[12] https://www.wikipathways.org/

3

Abordagens de Transformação

Este capítulo apresenta diversas abordagens de transformação de dados em DAL. Essas abordagens estão classificadas de acordo com o tipo de dado de entrada usado no processo de transformação. Para cada abordagem apresentada destacamos suas principais características.

Este capítulo está estruturado da seguinte forma: a seção 3.1 apresenta uma visão geral sobre abordagens de transformação de dados, principalmente aplicada na área biomédica; a seção 3.2 apresenta as abordagens que utilizam dados estruturados como entrada; a seção 3.3 apresenta as abordagens que utilizam dados semiestruturados como entrada; a seção 3.4 apresenta as abordagens que utilizam tanto dados estruturados quanto dados semiestruturados como entrada; finalmente, a seção 3.5 apresenta uma avaliação acerca das abordagens estudadas.

3.1 Visão geral

O conhecimento da área biomédica encontra-se segmentado, entre outros, em mais de 1600 bancos de dados temáticos (RIGDEN; FERNANDEZ-SUAREZ; GALPERIN, 2016). Em razão dessa fragmentação, cientistas da área enfrentam diferentes problemas, tais como: i) quais bancos de dados estão disponíveis e quais contêm as informações desejadas; ii) quais campos são utilizados em cada base de dados e quais são os significados desses campos; e iii) como os bancos de dados podem ser acessados e consultados. Estes problemas podem, em grande parte, ser solucionados por meio da integração (semântica) dos dados (LEGAZ-GARCÍA et al., 2016).

Uma das principais abordagens utilizadas na integração semântica de dados consiste no uso de uma ontologia para representar o conhecimento existente em bancos de dados heterogêneos. A representação do conhecimento de mais de um banco de dados utilizando uma mesma ontologia viabiliza a integração dos dados de forma a manter a equivalência semântica de cada item de dado extraído dos diferentes bancos de dados. Essa equivalência semântica pode ser utilizada para a persistência dos dados em um banco de dados unificado ou utilizada como ferramenta para uma plataforma de conversão de consultas em tempo de execução.

Mesmo que uma mesma organização integre semanticamente os bancos de dados de um mesmo domínio, a integração de dados de diferentes organizações ou pertencentes a diferentes domínios do conhecimento continua a ser um desafio. Este problema pode ser minimizado por meio de um processo de integração que utilize dados abertos ligados.

Diferentes abordagens para a geração de dados abertos ligados têm sido propostas na literatura (BELLEAU et al., 2008; AUER et al., 2009; JUPP et al., 2014; MERRILL et al., 2014; KAALIA; GHOSH, 2016; LEGAZ-GARCÍA et al., 2016; JOVANOVIK; TRAJANOV, 2017; SERNADELA; GONZÁLEZ-CASTRO; OLIVEIRA, 2017). Estas abordagens têm sido aplicadas principalmente no domínio biomédico e podem ser classificadas de acordo com o tipo de dado de entrada em: i) abordagens baseadas em dados estruturados; ii) abordagens baseadas em dados semiestruturados; e iii) abordagens híbridas, isto é, que aceitam tanto dados estruturados quanto semiestruturados como entrada para o processo de transformação.

3.2 Abordagens baseadas em dados estruturados

3.2.1 Abordagem de Auer et al.

Auer et al. (2009) afirmam que a maioria das aplicações web utiliza uma quantidade relativamente pequena de consultas SQL (entre 5 e 20) para a recuperação dos dados mais importantes. Desta forma, os autores propõem uma abordagem de geração de DAL a partir de bancos de dados relacionais transformando os resultados das pesquisas SQL em triplas RDF.

As consultas SQL são utilizadas para extrair os dados do banco de dados e também como base para o mapeamento semântico entre os dados e os termos das ontologias. Os nomes das colunas são utilizados como propriedades das triplas. As células da tabela resultante da pesquisa SQL consistem nos objetos das triplas RDF, podendo ser representados por tipos de dados (e.g., string, integer) ou referências para outros indivíduos.

Ao receber uma requisição HTTP solicitando informações sobre um recurso, instruções SQL são executadas e retornam o resultado em forma de dados ligados. Essas requisições podem conter diferentes tipos de informação, como, por exemplo, todas as categorias disponíveis, todos os indivíduos de uma determinada categoria ou todos os dados de um determinado indivíduo.

Segundo Auer et al., um diferencial da abordagem proposta por eles consiste da utilização da linguagem SQL como mecanismo de mapeamento dos dados entre as bases de dados e as ontologias, reduzindo a curva de aprendizado para o início da aplicação da abordagem. Adicionalmente, esta abordagem provê suporte à geração de atualizações de dados, tal que dados alterados após a geração de um conjunto DAL são disponibilizados na forma de um novo conjunto, permitindo que softwares clientes tenham acesso às modificações separadamente.

Auer et al. desenvolveram em PHP uma ferramenta intitulada Triplify para suporte ao processo de transformação. Esta ferramenta foi utilizada na iniciativa *OpenStreetMap Geo* para a geração de dados geográficos abertos e ligados. Como resultado desta aplicação, o Triplify produziu um conjunto DAL contendo mais de 1 bilhão de triplas. Embora a abordagem de Auer et al. não tenha sido utilizada no domínio biomédico, esta pode ser considerada indepedente de domínio, permitindo sua utilização para a transformação de dados biomédicos armazenados em bancos de dados relacionais.

3.2.2 Abordagem de Jupp et al.

Jupp et al. (2014) propõem uma abordagem de geração de DAL a partir de dados de bioinformática mantidos pelo Instituto de Bioinformática Europeu (*European Bioinformatics Institute* - EBI) (European Bioinformatics Institute, 2017). O EBI consiste no maior provedor de recursos de bioinformática da Europa, provendo acesso a diversos bancos de dados públicos e de interesse para a área, tais como, Gene Expression Atlas, ChEMBL e

BioModels.

A abordagem de geração de DAL foi proposta para responder perguntas científicas cada vez mais complexas e suprir a demanda por disponibilização de dados em formato RDF. Requisitos técnicos e científicos foram levantados a partir de casos de uso e perguntas técnicas feitas a cientistas e usuários do EBI. As perguntas foram formuladas de forma que estas devessem envolver múltiplos bancos de dados e não fossem triviais de responder considerando-se a estrutura atual do EBI. As perguntas foram então utilizadas para a identificação dos pontos de integração entre os bancos de dados, enquanto que os casos de uso foram utilizados para levantar os requisitos de infraestrutura (software e hardware) necessários para proporcionar a solução desejada.

Em razão do uso de IRIs na identificação dos elementos de um documento RDF, esta abordagem então definiu um padrão a ser seguido pelos bancos de dados do EBI. Bancos de dados que tivessem IRIs estáveis, i.e., que não poderiam ser alterados em algum momento, usariam seus próprios IRIs. Os bancos de dados que não tinham essa garantia, e.g., bancos de dados de parceiros do EBI, utilizaram o serviço *Identifier.org*, o qual fornece um IRI único. Foi feito também um esforço para que todos os dados disponibilizados por meio de qualquer um desses IRIs direcionassem tanto para uma página HTML quanto para um documento RDF. A página HTML permite a visualização por pessoas, enquanto que o documento RDF permite que computadores tenham acesso às triplas RDF e possam utilizá-las para fazer inferências.

Ontologias foram utilizadas para anotar os bancos de dados e com isso permitir as ligações entre as triplas RDF geradas. As ontologias Gene Ontology, Chemical Entities of Biological Interest, Cell Type Ontology e Pathway Exchange, entre outras, foram utilizadas para essa anotação. Além disso, foram utilizados vocabulários de termos para anotação de metadados como, por exemplo, o vocabulário Dublin Core. As triplas RDF foram armazenadas em um *triplestore* fornecido pelo OpenLink[1].

Um mecanismo de execução de instruções SPARQL e um *browser* foram disponibilizados ao final da abordagem. Tal mecanismo, chamado LODEStar, provê uma API de acesso, bem como facilidades para a exploração visual dos dados.

[1] https://virtuoso.openlinksw.com/

3.3 Abordagens baseadas em dados semiestruturados

3.3.1 Abordagem de Merrill et al.

Merrill et al. (2014) propõem uma abordagem de geração de DAL por meio do framework eXframe. Este framework permite a criação de repositórios web de experimentos genômicos, disponibiliza os dados de forma ligada e provê um mecanismo de busca utilizando SPARQL. O framework eXframe utiliza um sistema *open source* de gerenciamento de conteúdo, chamado Drupal[2], com modificações para suportar dados de experimentos genômicos. Além disso, o eXframe utiliza uma biblioteca PHP baseada em MySQL para o armazenamento das triplas RDF, chamada ARC2[3].

O framework eXframe suporta o armazenamento de experimentos genômicos com base na utilização de tipos de conteúdo (e.g., experimentos, ensaios, biomateriais entre outros) e relacionamentos entre os mesmos. Os diferentes tipos de conteúdo são definidos e disponibilizados por padrão junto ao framework. No entanto, esses tipos de conteúdo podem ser modificados/configurados pelo usuário no momento da instalação para que possam se adequar melhor ao conhecimento a ser armazenado. Tanto os tipos de conteúdo quanto seus relacionamentos são vinculados a uma ontologia fixa na instalação.

A alimentação do repositório é feita por meio do preenchimento de um formulário de cadastro, onde cada campo do formulário está associado internamente a um tipo de conteúdo. Desta forma, ao preencher o formulário os dados são armazenados de forma ligada utilizando os relacionamentos previamente definidos. As triplas RDF ligadas são produzidas pelo módulo Drupal RDF (CORLOSQUET et al., 2009), enquanto que a biblioteca ARC2 fica responsável pelo armazenamento e a disponibilização do mecanismo de execução de instruções SPARQL.

O framework eXframe trabalha com dois armazenamentos distintos: público e privado. O primeiro disponibiliza de forma livre os dados sinalizados como finalizados e publicados. O segundo disponibiliza os dados de pesquisas em andamento apenas para as pessoas envolvidas nestas pesquisas. Assim, as consultas SPARQL não estão disponíveis para todos os dados.

[2] https://www.drupal.org/
[3] https://github.com/semsol/arc2

O eXframe foi utilizado para a integração dos bancos de dados Blood Genomics e o Stem Cell Discovery Engine (SCDE) (CORLOSQUET et al., 2009), que juntos possuíam dados de cerca de 20 laboratórios. Essa integração resultou em 200 conjuntos de dados abertos ligados, os quais representam informações sobre 4 organismos, 119 tipos diferentes de células e 39 tipos de tecidos.

3.3.2 Abordagem de Kaalia e Ghosh

Kaalia e Ghosh (2016) propõem uma abordagem para a geração de DAL usada pela ferramenta *InstanceLoaderDB* do *framework* DAO-db. O framework DAO-db foi desenvolvido para integrar informações heterogêneas associadas com a *diabetes mellitus*. DAO-db consiste em uma plataforma orientada a ontologia para a integração (semântica) de associações funcionais (e.g., gene-doença, gene-componente celular, proteína-proteína) e para a descrição de genes, proteínas e vias metabólicas envolvidas com esta doença.

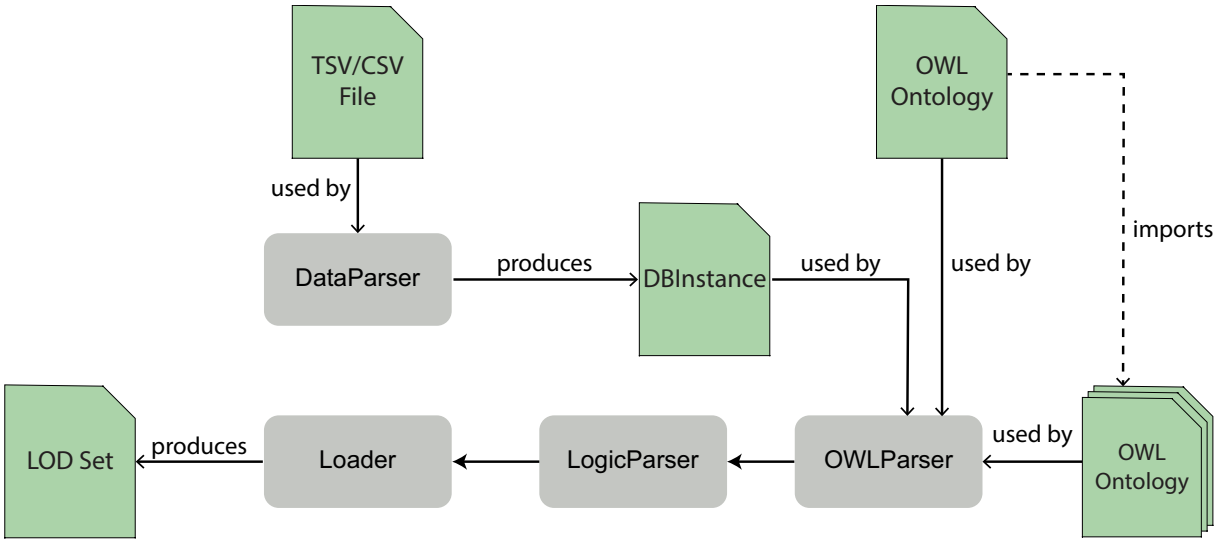
InstanceLoaderDB consiste em uma ferramenta de inserção de indivíduos da ontologia em larga escala. Esta ferramenta recebe como entrada uma ontologia OWL e diferentes conjuntos de dados, os quais são convertidos em indivíduos (instâncias) das classes definidas na ontologia por meio de quatro módulos: `DataParser`, `OWLParser`, `LogicParser` e `Loader`.

O módulo de `DataParser` recebe conjuntos de dados como entrada e os converte em um tipo padrão do Framework (objeto chamado *DBInstance*). Esses dados de entrada são arquivos de texto, podendo ter uma estrutura de separação tabular (arquivos TSV) ou separação por vírgulas (arquivos CSV). Com a conversão dos dados para um tipo padrão, elimina-se a heterogeneidade dos tipos de arquivos, facilitando assim o processamento subsequente.

O módulo `OWLParser` lê uma ontologia e importa quaisquer ontologias dependentes referenciadas. Com isso, o InstanceLoaderDB identifica as classes e propriedades assim como suas restrições, i.e., as características que os indivíduos da ontologia devem respeitar. O módulo `LogicParser` cruza os objetos (dados) com as restrições definidas pela ontologia. Este módulo executa inferências a respeito do dado por meio das restrições carregadas no módulo anterior. Desta forma são identificados, por exemplo, a qual classe um dado pertence, assim como se já existe um indivíduo com o mesmo nome na ontologia.

Finalmente, o módulo `Loader` cria os indivíduos na ontologia considerando as inferências do módulo lógico. Desta forma, os dados, antes presentes em diferentes arquivos, são agora instanciados em uma ontologia contendo características (e.g., relacionamentos) passíveis de inferência automática. A Figura 16 ilustra a arquitetura da ferramenta InstanceLoaderDB.

Figura 16 – Arquitetura da ferramenta InstanceLoaderDB.



Fonte: Autoria própria. Um retângulo com borda arredondada representa um módulo da ferramenta, enquanto uma linha com uma ponta de seta sólida conectando dois módulos define a ordem na qual esses módulos são executados. Um retângulo verde dobrado representa um artefato produzido ou consumido por um módulo, enquanto uma linha sólida com uma ponta de seta conectando um módulo a um artefato indica a produção ou o consumo do artefato. Finalmente, uma linha tracejada com uma ponta de seta sólida define a importação de ontologias referenciadas por outra ontologia.

3.3.3 Abordagem de Jovanovik e Trajanov

Informações globais sobre medicamentos são normalmente disponibilizadas de forma não aberta. Entretanto, quando disponíveis, estas informações são disponibilizadas de forma descentralizada, em portais especializados. Estes portais fornecem informações sobre: i) público-alvo, dosagens e avisos; ii) princípios ativos, fórmulas químicas, toxidade e interações com comidas e com outros medicamentos; e iii) medicamentos comercializados em determinados países. Assim, Jovanovik e Trajanov (2017) propuseram uma metodologia de transformação de dados no formato CSV em DAL para a construção de uma base global contendo informações oficiais de medicamentos utilizados/comercializados em diferentes

países.

A metodologia proposta por Jovanovik e Trajanov engloba cinco etapas, as quais são normalmente executadas em sequência: i) conhecimento do domínio e dos dados; ii) modelagem e alinhamento dos dados; iii) transformação dos dados em DAL; iv) publicação do conjunto de dados na web; e v) casos de uso, aplicações e serviços. A Figura 17 ilustra uma visão geral destas etapas.

Figura 17 – Etapas propostas por Jovanovik e Trajanov.

Fonte: Adaptado de Jovanovik e Trajanov (2017)

A primeira etapa da metodologia aborda o conhecimento dos dados a serem trabalhados. De forma geral, ao trabalhar em um domínio de conhecimento deve-se conhecer os tipos de dados existentes, o significado de cada um deles e a importância/utilidade deles para o contexto. Este conhecimento pode ser adquirido por meio de um especialista de domínio ou pela exploração de conjuntos DAL semelhantes ou de um mesmo domínio.

A segunda etapa da metodologia aborda a modelagem dos dados a serem transformados e o alinhamento destes com os dados presentes em outros conjuntos de dados já existentes. Nessa etapa a semântica dos dados deve ser corretamente especificada, considerando-se a integração dos dados transformados com conjuntos de dados já existentes e a reutilização de ontologias a fim de aproveitar o benefício dos dados ligados e da vinculação de dados semelhantes. No estudo de caso realizado foi desenvolvido um

arquivo modelo em formato CSV e uma ontologia RDFS. O arquivo CSV foi utilizado para padronizar os dados providos pelos portais. Cada portal poderia armazenar seus dados em arquivos CSV diferentes ou em um único arquivo CSV, seguindo o modelo criado. Já a ontologia RDFS foi utilizada para anotar semanticamente as colunas dos arquivos CSV, utilizando classes e propriedades do vocabulário Schema.org[4].

A terceira etapa aborda a transformação dos dados originais em DAL. Essa transformação pode ser realizada utilizado-se, em princípio, qualquer ferramenta de suporte. No estudo de caso realizado, a transformação foi executada utilizando uma ferramenta *open-source* para tratamento de dados chamada OpenRefine[5]. Durante o processo de transformação, também foi efetuada a ligação dos dados do arquivo CSV com medicamentos relacionados de outras fontes de dados, tais como DrugBank e DBpedia. O código *Anatomical Therapeutic Chemical* (ATC) (World Health Organization, 2017), que indica as propriedades terapêuticas, farmacológicas e químicas do medicamento, foi utilizado para a identificação de relações entre os medicamentos. Além disso, os medicamentos do próprio conjunto de dados RDF gerado também foram ligados com base em seus códigos ATCs.

A quarta etapa aborda a publicação do conjunto de dados na web. Nesta etapa, Jovanovik e Trajanov (2017) sugerem disponibilizar uma API RESTful, uma interface de pesquisa SPARQL e/ou o conjunto de dados em um arquivo para *download*. Adicionalmente, os autores recomendam a publicação do conjunto de dados em uma plataforma de gerenciamento de dados abertos, chamada Datahub.io[6], aumentando assim a visibilidade do conjunto de dados transformado.

A quinta etapa aborda o desenvolvimento de casos de uso, aplicações e serviços utilizando o conjunto de dados transformado. Esta etapa é importante para ressaltar a usabilidade do conjunto de dados e suas ligações, descrevendo a forma segundo a qual o conjunto de dados pode ser explorado, recuperado e utilizado. Esses casos de uso podem ser cenários em formato de texto, instruções específicas SPARQL ou protótipos de aplicações. Além disso, os autores recomendam que os casos de uso exemplifiquem a utilização do conjunto de dados convertido em DAL, ressaltando os resultados possíveis de serem alcançados e que não poderiam ser facilmente obtidos utilizando as bases de dados

[4] http://schema.org/
[5] http://openrefine.org/
[6] https://datahub.io/

distribuídas.

3.4 Abordagens híbridas

3.4.1 Abordagem de Belleau et al.

Belleau et al. (2008) apresentam uma abordagem para a transformação de dados de bancos de dados públicos da área de bioinformática em triplas RDF. O processo de transformação engloba tanto a conversão para documentos RDF quanto a integração semântica desses dados e a capacidade de pesquisa e de visualização dos dados transformados. A metodologia de transformação possui duas etapas: i) listagem de bancos de dados e extração do conhecimento; e ii) conversão de dados e normalização de IRIs.

A atividade de listagem de bancos de dados consiste na identificação das fontes de dados que serão utilizadas para a extração dos dados a serem transformados. A atividade de modelagem de conhecimento consiste na extração do conhecimento dos portais das fontes de dados para a modelagem de uma ontologia OWL. Nesta atividade são exploradas as páginas HTML existentes em cada portal selecionado a fim de gerar uma ontologia a partir das informações exibidas nestas páginas. Nessa exploração, a própria página acessada representa a classe (sujeito), os rótulos presentes na página representam as propriedades e os *hyperlinks* indicados pelos rótulos representam os indivíduos da ontologia. Uma ontologia OWL deve ser gerada para cada portal analisado. Ao final, todas as ontologias OWL devem ser combinadas em uma única ontologia.

A atividade de conversão de dados consiste no desenvolvimento de softwares específicos para a criação de documentos RDF a partir da forma de acesso aos dados dos portais. Os dados podem estar armazenados em arquivos tais como, documentos XML, arquivos texto e documentos RDF, ou podem ser acessados diretamente em um banco de dados relacional. Diversas técnicas são utilizadas para o processo de conversão com base na forma de acesso disponibilizado: *XML Path Language* (XPath), para converter dados de documentos XML; expressões regulares, para converter dados em texto livre; e SQL, para converter dados de bancos de dados relacionais.

A atividade de normalização de IRIs tem como objetivo a diminuição/eliminação de

possíveis redundâncias dos IRIs encontrados. Por exemplo, um documento do PubMed com identificador `12728276` pode ser referenciado por `PMID:12728276` ou `pubmed:12728276`. Desta forma, ao integrar diversos bancos de dados torna-se necessária a normalização dos IRIs para garantir a correta estrutura de dados ligados, ou seja, múltiplos identificadores que tratem sobre a uma mesma informação devem ser unificados em um único identificador. Ao explorar os dados dos bancos de dados, os IRIs são criados e, ao encontrar um outro IRI identificando o mesmo conceito, cria-se uma propriedade *owl:sameAs*, para que sejam armazenados os IRIs sinônimos.

O processo de transformação dos dados utiliza alguns softwares abertos a fim de viabilizar a reprodutibilidade do projeto. Os softwares utilizados incluem: triplestore Sesame[7], para o armazenamento de triplas; RDF Elmo[7], para explorar documentos RDFs (criados pelos softwares específicos) e instanciar novas triplas; *JavaServer Pages Technology* (JSP) (ORACLE, 2017b) e *JavaServer Pages Standard Tag Library* (JSTL) (ORACLE, 2017a), para a geração de páginas web; e a biblioteca *URLrewrite* (TUCKEY, 2017), para a intermediação de requisições HTTP.

Após a transformação e o armazenamento dos dados transformados no *triplestore* Sesame, os dados podem ser pesquisados/recuperados por meio da linguagem *Sesame RDF Query Language* (SeRQL) (Aduna Software, 2017), uma linguagem sintaticamente semelhante ao SPARQL. Uma vez recuperados da *triplestore*, os dados são convertidos em RDF e disponibilizados para aplicações clientes.

A abordagem de Belleau et al. foi utilizada para o desenvolvimento do projeto Bio2RDF, o qual havia transformado, até julho de 2014, dados de mais de 30 fontes de dados totalizando mais de 11 bilhões de triplas (DUMONTIER et al., 2014).

3.4.2 Abordagem de Legaz-García et al.

Legaz-García et al. (2016) propõem uma abordagem de transformação que utiliza regras de equivalência entre elementos que descrevem a estrutura (*schema*) dos dados e termos de uma ontologia para a integração de diferentes bancos de dados e documentos XML. A geração dos dados abertos ligados é realizada ao longo de três etapas: i) definição de regras de mapeamento; ii) definição de padrões de transformação; e iii) definição de regras

[7] http://rdf4j.org/

de identidade.

A etapa de definição de regras de mapeamento consiste na elaboração de um conjunto de regras básicas de mapeamento. Essas regras básicas são divididas em três categorias: regras de entidades, regras de atributos e regras de relacionamento. Regras de entidades garantem a equivalência semântica das entidades dos dados de entrada na geração de indivíduos das classes da ontologia. Regras de atributos garantem a equivalência semântica dos atributos das entidades dos dados de entrada nos indivíduos da ontologia. Finalmente, regras de relacionamento garantem a equivalência semântica dos relacionamentos entre as entidades nos indivíduos da ontologia.

A etapa de definição de padrões de transformação consiste na definição de regras mais complexas de mapeamento, também chamadas de padrões de mapeamento. Esses padrões permitem a utilização de múltiplos elementos do *schema* dos dados de entrada e múltiplos conceitos definidos em ontologias, assim como a inserção de valores fixos para informações de entrada omissas ou complementares. A definição de um padrão de mapeamento pode envolver o uso de um conjunto de conceitos definidos em uma ontologia, um conjunto de propriedades de tipo de dados (atributos de uma classe), um conjunto de propriedades de objetos (relacionamentos) e um conjunto de restrições de aplicabilidade.

Finalmente, a etapa de definição de regras de identidade utiliza as propriedades de tipo de dados e as propriedades de objetos como critérios para a identificação de um indivíduo de uma classe. Essa identificação impede transformações que possuam os mesmos valores para as mesmas propriedades, prevenindo assim a criação de indivíduos duplicados, mesmo que estes indivíduos possuam IRIs distintos.

Legaz-García et al. implementaram uma ferramenta intitulada *Semantic Web Integration Tool* (SWIT) que provê suporte a essa abordagem de geração de DAL. SWIT fornece uma interface web que guia o usuário em todos os passos do processo, permitindo, ao final, gerar um conjunto de dados no formato RDF ou OWL. Desta forma, a ferramenta SWIT foi utilizada para a integração de diversos bancos de dados sobre genes ortólogos utilizando a ontologia OrthoXML. A utilização nesse estudo de caso teve como resultado a integração e disponibilização de um conjunto DAL com mais de 2 bilhões de triplas.

Recentemente, Bernabé-Díaz et al. publicaram uma atualização da ferramenta SWIT (BERNABÉ-DÍAZ et al., 2019), incluindo suporte à transformação de dados

semiestruturados em formato TSV/CSV, verificação da consistência lógica das triplas com base na ontologia de entrada, uso de IRIs específicos para cada regra de transformação e, finalmente, busca de IRIs em *endpoint* SPARQL externos.

3.4.3 Abordagem de Sernadela, González-Castro e Oliveira

Sernadela, González-Castro e Oliveira (2017) propõem uma abordagem de transformação de dados que permite que usuários tenham acesso a um conjunto de serviços semânticos para a integração, o gerenciamento e a exportação dos dados abertos ligados. Essa abordagem foi utilizada para o desenvolvimento de uma ferramenta chamada Scaleus.

A arquitetura da ferramenta é dividida em três camadas: base de conhecimento, abstração e serviços. A camada de base de conhecimento utiliza um banco de dados transacional (TDB) para a persistência das triplas RDF e a execução de operações de pesquisa de forma eficiente. Ao utilizar transações, os dados são protegidos de fatores, tais como corrupção, terminação inesperada de processos e falhas de sistema. Além disso, o TDB permite o gerenciamento de diversos conjuntos de dados de forma independente uns dos outros.

A camada de abstração tem como função a administração de informações semânticas (e.g., adicionar e remover triplas e *datasets*) e o suporte à inferência. Essas funções foram desenvolvidas utilizando o conjunto ferramental Jena[8]. Desta forma, Scaleus fornece um conjunto de conectores e interfaces que auxiliam no processo de transformação dos dados, de modo a integrar dados de diferentes formatos de arquivos em triplas. Esse processo de transformação permite que sejam importados tanto documentos RDF, contendo triplas prontas, quanto dados não transformados (e.g., planilhas XSL).

Ao importar uma planilha, a ferramenta permite que o usuário crie IRIs para uma coluna (um IRI para cada elemento da coluna) e especifique mapeamentos de relacionamentos entre as colunas. Assim, o mapeamento é criado por meio da definição de uma coluna como sujeito, uma propriedade disponível na web como predicado e uma outra coluna como objeto da tripla. Esse mapeamento permite que a ferramenta Scaleus transforme os dados de uma planilha em triplas RDF.

Na camada de abstração também são trabalhados os métodos de inferência de

[8] https://jena.apache.org/

dados, os quais são divididos em regras nativas e regras definidas pelo usuário. As regras nativas são baseadas na utilização dos axiomas (classes e propriedades) RDFS, permitindo a realização de inferências ao analisar essas características. Regras definidas pelo usuário permitem que novos relacionamentos possam ser criados a partir de triplas já existentes, seguindo critérios definidos pelo próprio usuário.

Finalmente, a camada de serviços provê suporte à camada de abstração por meio de uma API *REpresentional State Transfer* (REST). Esta API fornece diversas operações para o gerenciamento de triplas (implementadas pela camada de abstração), tais como a listagem, a adição e a remoção de triplas, além de prover acesso a um motor SPARQL para suporte à execução de pesquisas.

A abordagem de Sernadela, González-Castro e Oliveira foi utilizada no projeto RD-Connect para importar e explorar semanticamente as informações de um conjunto de bancos de dados de doenças raras (THOMPSON et al., 2014).

3.5 Avaliação

Uma abordagem para transformação de dados de bioinformática para dados abertos ligados deve ser simples, sistemática e flexível. Uma abordagem simples envolve a utilização de um pequeno conjunto de conceitos, o que facilita seu aprendizado e aplicação. Uma abordagem sistemática possui um conjunto de etapas pré-definidas para guiar o processo de transformação. Uma abordagem flexível pode ser utilizada em diferentes contextos sem grande modificações. Uma abordagem flexível também deve suportar a utilização de múltiplos arquivos de entrada e o uso de regras para a definição de relações de equivalência semântica entre itens de dados e conceitos de uma ontologia. Adicionalmente, uma abordagem para a transformação de dados de bioinformática em DAL deve suportar a transformação de dados semiestruturados, em razão da ampla utilização destes tipos de dados no domínio.

A Tabela 2 apresenta uma análise de adequação dos requisitos de simplicidade, sistematicidade, flexibilidade e suporte a dados semiestruturados em relação às abordagens apresentadas neste capítulo. O símbolo '+' indica que o critério foi integralmente atendido, o símbolo '○' indica que o critério foi parcialmente atendido e, finalmente, o símbolo '–'

Tabela 2 – Comparação das abordagens de transformação.

Requisitos	Auer et al.	Jupp et al.	Merrill et al.	Kaalia e Ghosh	Jovanovik e Trajanov	Belleau et al.	Legaz-García et al.	Sernadela, González-Castro e Oliveira
Simplicidade	–	+	–	–	○	–	+	+
Sistematicidade	–	–	–	+	+	–	+	–
Flexibilidade	–	–	–	–	+	○	○	+
Suporte a dados semiestruturados	–	–	+	+	+	+	+	+

indica que o critério não foi atendido. Por meio desta tabela podemos identificar que nenhuma das abordagens suporta integralmente estes requisitos.

As abordagens Auer et al., Merrill et al. e Belleau et al. não podem ser consideradas simples pois requerem a utilização de conceitos alheios à web semântica. O primeiro requer a utilização de SQL, enquanto o segundo requer a utilização de PHP. A abordagem Belleau et al. requer o desenvolvimento de softwares específicos para a transformação de cada tipo de dados de entrada. A abordagem Kaalia e Ghosh não está suficientemente detalhada para uma avaliação adequada quanto a sua simplicidade. Finalmente, consideramos que a abordagem de Jovanovik e Trajanov satisfaça parcialmente o requisito de simplicidade, pois embora os conceitos envolvidos nesta abordagem sejam aparentemente simples, as diretrizes apresentadas para o uso dos mesmos são, em sua maioria, bastante genéricas, o que dificulta a utilização desta abordagem em cenários concretos.

As abordagens Auer et al., Jupp et al., Merrill et al., Belleau et al. e Sernadela, González-Castro e Oliveira não apresentam metodologias de transformação sistemática, mas apenas um conjunto de ferramentas para suporte à transformação dos dados.

As abordagens de Auer et al., Jupp et al. e Merrill et al. não suprem nenhum dos critérios do requisito de flexibilidade, pois essas abordagens não aceitam múltiplos arquivos de dados e múltiplas ontologias como entrada e não suportam a especificação de regras de transformação. Adicionalmente, a abordagem de Kaalia e Ghosh, apesar de suportar múltiplos arquivos de entrada, não pode nem ao menos ser considerada parcialmente flexível, uma vez que esta abordagem requer a modificação do software de aplicação para se adequar a outros domínios, além de não permitir a utilização de regras de transformação.

As abordagens de Legaz-García et al. e Belleau et al. podem ser consideradas parcialmente flexíveis. A abordagem de Legaz-García et al. utiliza regras de transformação, entretanto não aceita múltiplos arquivos de dados e múltiplas ontologias como entrada. A

abordagem de Belleau et al. aceita múltiplos arquivos de dados de entrada, no entanto não aceita a entrada de múltiplas ontologias. Adicionalmente, esta abordagem possui aplicabilidade restrita em razão do desenvolvimento de softwares específicos para cada tipo de fonte de dados.

Finalmente, exceto pelas abordagens de Auer et al. e Jupp et al., todas as demais abordagens suportam a transformação de dados semiestruturados.

4

SSD2LOD Transformation Approach

Este capítulo apresenta uma abordagem para a transformação de dados semiestruturados em DAL chamada SSD2LOD Transformation Approach. Essa abordagem permite a especificação da equivalência semântica de dados semiestruturados com termos de uma ou mais ontologias a fim de agregar valor semântico explícito aos dados após a transformação. Para isso, foi desenvolvida a linguagem de especificação de regras de transformação chamada SSD2LOD Transformation Language e um conjunto de ferramentas de suporte para executar a transformação propriamente dita.

Este capítulo está estruturado da seguinte forma: a seção 4.1 apresenta uma visão geral sobre a abordagem SSD2LOD Transformation Approach; a seção 4.2 apresenta a linguagem de especificação de regras de transformação chamada SSD2LOD Transformation Language; finalmente, a seção 4.3 apresenta as ferramentas de suporte à transformação desenvolvidas.

4.1 Visão geral

Diversas abordagens de transformação de dados para DAL foram propostas na literatura. No entanto, apenas algumas delas permitem o uso de dados semiestruturados como fonte de dados. Desta forma, propomos a abordagem SSD2LOD Transformation Approach para a transformação de dados semiestruturados (TSV ou CSV) em DAL introduzindo semântica aos mesmos por meio da utilização de uma ou mais ontologias. De modo a transformar SSD em DAL, faz-se necessário especificar como os dados semiestruturados se relacionam com um conjunto ontológico de conceitos. Tal relacionamento pode ser definido

por meio de um conjunto de regras de transformação. Assim, uma regra de transformação especifica a equivalência semântica entre um ou mais itens de dados semiestruturados e um conceito definido em uma ontologia OWL. Uma regra de transformação não apenas vincula o significado aos dados, mas também especifica como os dados relacionam-se uns com os outros no conjunto DAL resultante.

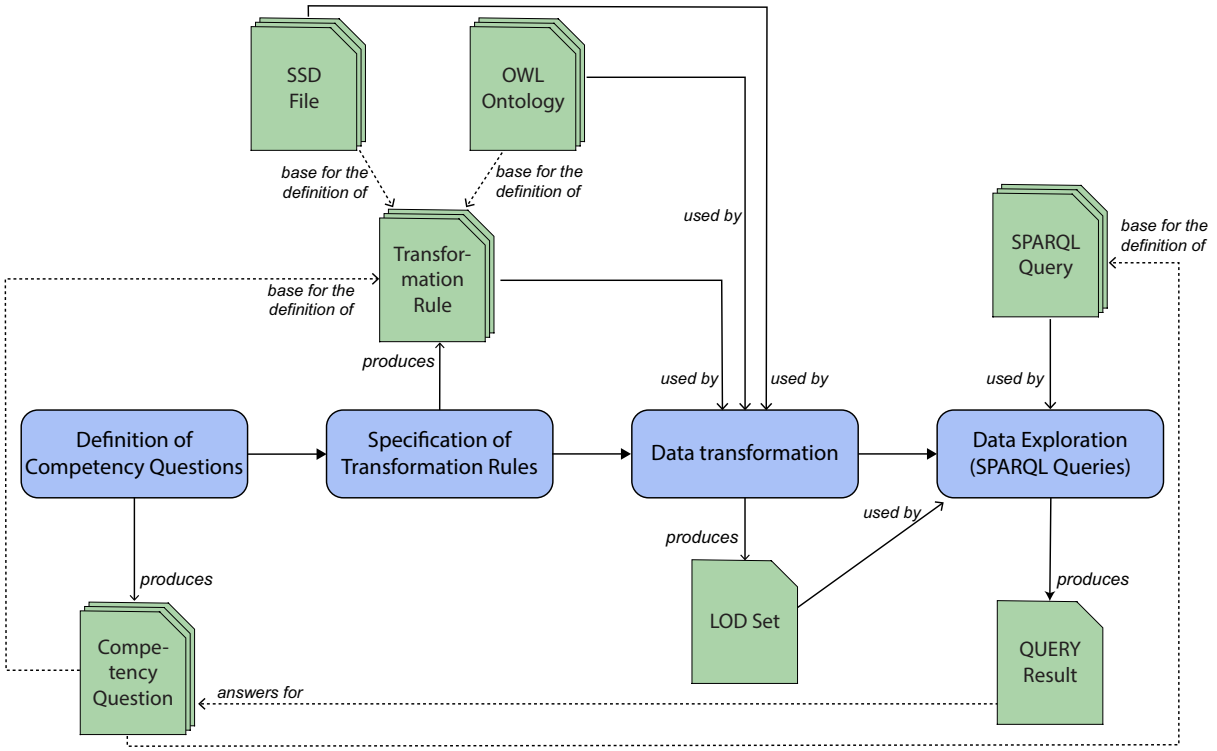
A abordagem SSD2LOD Transformation Approach proposta neste trabalho é sistemática, isto é, segue um conjunto ordenado de atividades a fim de transformar um conjunto de dados semiestruturados presentes em um ou mais arquivos em um conjunto DAL. A abordagem é orientada a questões de competência e utiliza regras de transformação definidas pelo usuário como mecanismo de transformação. A abordagem de transformação consiste das seguintes atividades: i) definição de Questões de Competências (QC); ii) especificação de regras de transformação; iii) transformação de dados; e iv) exploração de dados. A Figura 18 fornece uma visão geral do processo de transformação de SSD em DAL.

4.1.1 Processo de transformação

A primeira atividade visa definir um conjunto de questões de competências que serão usadas para orientar a criação das regras de transformação. Questão de competência é uma técnica frequentemente utilizada durante o desenvolvimento de ontologias para a identificação de perguntas que devem ser respondidas por uma ontologia após sua construção (GRÜNINGER; FOX, 1995; FERNÁNDEZ-LÓPEZ; GÓMEZ-PÉREZ; JURISTO, 1997; NOY; MCGUINNESS, 2001; NICOLA; MISSIKOFF; NAVIGLI, 2005; FALBO, 2014). No contexto deste trabalho, as questões de competência definem o escopo do processo de transformação, guiando a identificação dos dados que serão posteriormente transformados. Assim, a definição de questões de competência direciona a criação de regras de transformação para tornar os dados semanticamente ligados e, eventualmente, permitir a descoberta do conhecimento por meio da exploração das relações semânticas criadas. Questões de competência devem ser elaboradas na forma de perguntas específicas, cujas respostas devem ser obtidas ao final do processo de transformação. Essas questões serão utilizadas como base para a próxima atividade da abordagem de transformação.

A segunda atividade visa especificar um conjunto de regras de transformação para

Figura 18 – Visão geral do processo de transformação.



Fonte: Autoria própria. Um retângulo azul com cantos arredondados representa uma atividade do processo de transformação. Uma linha com uma ponta de seta sólida conectando duas atividades define a ordem na qual essas atividades são executadas. Um retângulo verde com o canto superior direito cortado representa um artefato produzido ou consumido por uma atividade. Uma linha sólida com uma ponta de seta conectando uma atividade a um artefato indica que o artefato é produzido ou consumido pela atividade. Por fim, uma linha tracejada com uma ponta de seta conectando dois artefatos indica a existência de uma relação entre os artefatos.

cada questão de competência definida na atividade anterior. Uma regra de transformação define relações semânticas entre itens de dados de um documento SSD e conceitos de uma ou mais ontologias OWL. Uma regra de transformação normalmente especifica dois tipos de relações semânticas simultaneamente. O primeiro tipo de relação semântica define um relacionamento do tipo *is_a* (representado no RDF por `rdf:type`) entre um conjunto de itens de dados e um conceito de uma ontologia. Esse tipo de relação produz os diferentes nós RDF que serão utilizados como sujeitos das triplas a serem criadas pelo segundo tipo de relação. Assim, o segundo tipo de relação semântica utiliza os sujeitos criados pelo primeiro tipo de relação semântica e os predicados e objetos definidos pelo usuário para criar novos conjuntos de triplas. Os predicados definidos pelo usuário referem-se às propriedades (de tipo de dados ou de objeto) especificadas nas ontologias OWL. Os objetos definidos pelo usuário podem ser tanto um outro nó RDF quanto um valor literal.

A terceira atividade consiste em executar a transformação de SSD para DAL. Esta atividade executa as regras de transformação especificadas para cada questão de competência, gerando os chamados *slices* de dados. Um *slice* consiste em um conjunto DAL obtido como resultado da execução de pelo menos uma regra de transformação. Ao final do processo de transformação, a união lógica dos diferentes *slices* de dados representa o conjunto DAL resultante da transformação. O escopo de um *slice* é restrito pelo conjunto de questões de competência especificadas durante a transformação. Os *slices* podem ser tanto armazenados em uma *triplestore* quanto disponibilizados para *download*, como também preconizado por Jovanovik e Trajanov (2017).

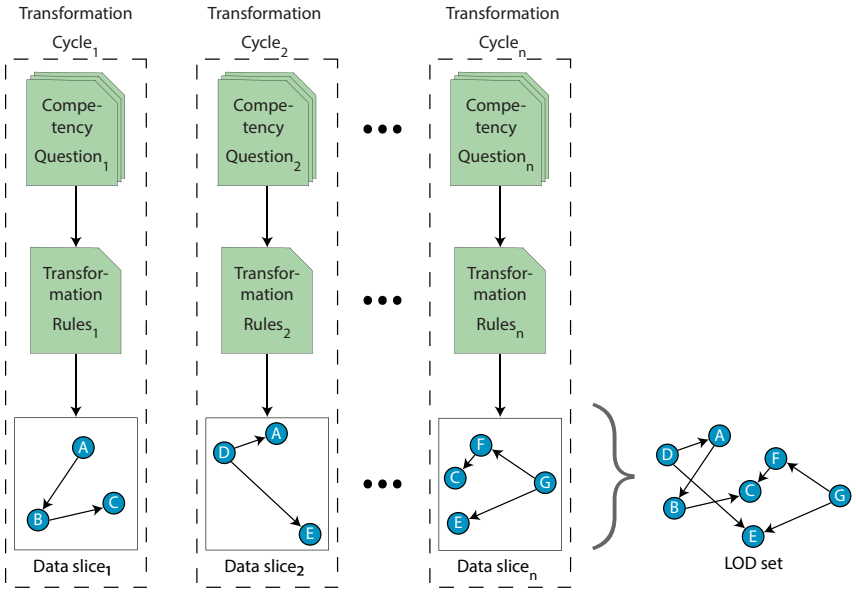
Finalmente, a quarta atividade visa explorar semanticamente os dados transformados a fim de obter as respostas às questões de competência. A W3C recomenda utilizar SPARQL para explorar e manipular dados DAL. Desta forma, para cada questão de competência definida, uma ou mais consultas SPARQL devem ser escritas para explorar, recuperar e manipular o conjunto DAL produzido na atividade anterior. As respostas obtidas em cada consulta (ou em um conjunto) representam as respostas às próprias questões de competência.

Em geral, as diferentes atividades da abordagem SSD2LOD Transformation Approach são executadas sequencialmente. No entanto, após a definição das questões de competência, um conjunto de questões (relacionadas) pode ser agrupado para facilitar o desenvolvimento das atividades de transformação remanescentes. Tal agrupamento representa uma compartimentalização de um processo de transformação, produzindo assim um ciclo de transformação independente a partir do conjunto agrupado de questões de competência.

4.1.2 Ciclos de transformação

Um ciclo de transformação possui pelo menos uma questão de competência, o que permite que, em casos de transformações complexas, sejam criados ciclos de transformação com um conjunto reduzido de questões de competência a fim de dividir este problema complexo em partes menores e mais facilmente tratáveis. A Figura 19 ilustra como slices produzidos em diferentes ciclos de transformação de dados são logicamente combinados para formar um conjunto DAL.

Figura 19 – Slices de dados e conjunto DAL resultante.



Fonte: Autoria própria.

Diferentes critérios podem ser usados para agrupar um conjunto de questões de competência em um ciclo de transformação, tais como: i) o uso do(s) mesmo(s) arquivo(s) SSD de origem, ii) a complexidade da especificação das regras de transformação, ou iii) questões de desempenho relacionados ao processo de transformação.

Uma questão de competência pode envolver um ou mais arquivos SSD de entrada. Desta forma, podemos agrupar as questões de competência em ciclos de transformação com base em um mesmo conjunto de arquivos de entrada. Em outros casos, a presença de múltiplas questões de competência pode contribuir para aumentar a complexidade das regras de transformação. Nestes casos, pode ser interessante separar as questões de competência em diferentes ciclos de transformação. Dado que somente um subconjunto das questões de competência identificadas e os dados relacionados às mesmas são considerados, a especificação das regras de transformação torna-se mais simples.

O desempenho do processo de transformação também pode influenciar o agrupamento das questões de competência em diferentes ciclos de transformação. Por exemplo, um problema de desempenho pode ser ocasionado pela utilização da busca externa de nós RDF em conjuntos DAL externos. Essa busca externa envolve fatores imponderáveis, tais como transmissão de dados pela rede, resposta do servidor remoto, tempo de execução de consultas no servidor remoto, entre outros, os quais podem resultar em um erro durante

a transformação. A compartimentalização das questões de competência em ciclos de transformação pode facilitar o processo de transformação como um todo, pois a presença de um erro durante uma determinada transformação não irá afetar as transformações executadas nos demais ciclos.

4.2 Especificação de regras de transformação

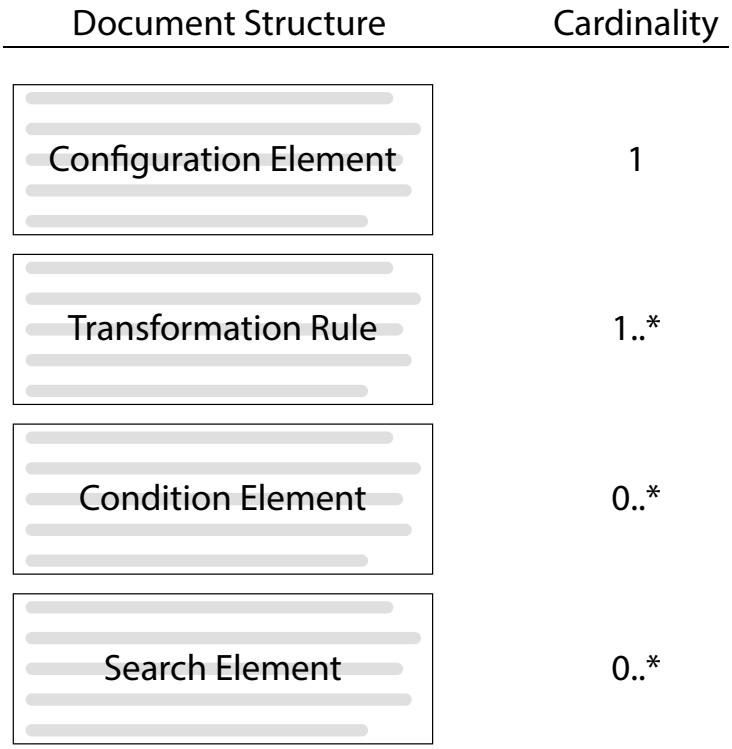
Uma regra de transformação especifica como um conjunto de dados semiestruturados deve ser transformado em DAL. Por meio de regras de transformação pode-se definir diversas declarações de transformação, tais como: quais dados devem ser transformados, quais critérios devem ser utilizados para habilitar uma transformação, quais as relações semânticas entre estes dados e conceitos de uma ontologia, entre outros.

Ao especificar uma regra de transformação, assumimos que os dados estão representados de acordo com o estilo de matriz de valores, uma vez que este estilo é o mais comum no domínio de bioinformática. Caso os dados estejam representados de acordo com o estilo em lista ou o estilo híbrido, esses dados devem ser primeiro transformados para o estilo de matriz de valores antes da transformação para DAL.

A especificação de uma transformação contém quatro diferentes tipos de elementos: i) Elemento de Configuração; ii) Regra de Transformação; iii) Elemento de Condição; e iv) Elemento de Pesquisa. Elemento de Configuração deve aparecer apenas uma única vez na especificação. Regra de Transformação deve aparecer no mínimo uma vez. Elemento de Condição e Elemento de Pesquisa são opcionais, podendo aparecer múltiplas vezes na especificação. A Figura 20 ilustra uma visão geral da estrutura de uma especificação de transformação e a respectiva cardinalidade de cada elemento.

Os elementos Regra de Transformação, Elemento de Condição e Elemento de Pesquisa são identificados univocamente na especificação. Tais identificadores são utilizados para referenciar esses elementos sempre que necessário na transformação. No caso de uma regra de transformação, o identificador é utilizado para que a regra possa ser referenciada por outra regra (encadeamento de regras). No caso do Elemento de Condição e do Elemento de Pesquisa, os identificadores são utilizados por uma regra de transformação para indicar respectivamente qual elemento de condição ou elemento de pesquisa deve ser processado

Figura 20 – Visão geral da estrutura de uma especificação de regras de transformação.



Fonte: Autoria própria.

naquele ponto específico da transformação.

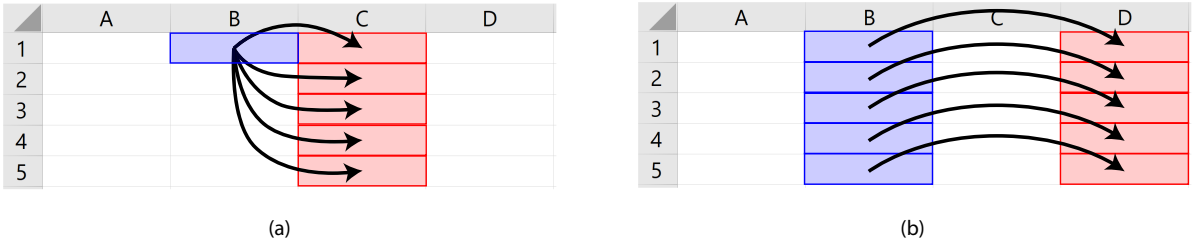
Ao contrário dos demais elementos, o Elemento de Configuração não requer a definição de um identificador único, pois um documento de especificação de regras de transformação contém apenas uma ocorrência deste elemento.

4.2.1 Elemento de configuração

Elemento de Configuração define um conjunto de parâmetros que devem ser usados em um processo de transformação. Os seguintes parâmetros podem ser especificados:

- **Default Base IRI**, o qual consiste de um parâmetro obrigatório que define um identificador padrão a ser usado como base de um IRI em um nó de uma tripla;
- **Has Non Processable Header**, o qual consiste de um parâmetro opcional que especifica se o conjunto SSD contém, ou não, um cabeçalho não processável, ou seja, se a primeira linha (cabeçalho) do conjunto SSD não deve ser processada durante a transformação por se tratar apenas da informação de qual tipo de dados a

Figura 21 – Tipos de regras de transformação.



Fonte: Autoria própria. a) regra baseada em coluna; b) regra baseada em linha.

coluna possui. Caso este parâmetro seja omitido, assume-se o valor padrão `"true"`, indicando a presença de um cabeçalho não processável;

- **Export Syntax**, o qual consiste de um parâmetro opcional que define a sintaxe de representação do conjunto DAL. Exemplos de sintaxe suportada incluem RDF/XML, N-Triples e Turtle. Caso este parâmetro seja omitido, assume-se a sintaxe padrão RDF/XML;

- **Namespace**, o qual consiste de um parâmetro opcional que especifica um nome abreviado (*namespace*) a ser usado como referência para uma dada ontologia. O parâmetro *namespace* é necessário apenas quando várias ontologias são usadas em uma mesma especificação de um conjunto de regras de transformação.

4.2.2 Regra de transformação

O elemento Regra de Transformação define as equivalências semânticas entre itens de dados e conceitos de um ontologia. Existem dois tipos de regra de transformação, viz., regras baseadas em colunas e regras baseadas em linhas. Uma regra baseada em coluna cria um conjunto de triplas associando um único item de dados (o primeiro elemento de uma coluna de origem) a um conjunto de itens de dados fornecidos por uma coluna de destino, enquanto uma regra baseada em linha cria um conjunto de triplas associando itens de dados fornecidos por uma coluna de origem aos itens de dados correspondentes fornecidos por uma coluna de destino (uma associação para cada linha existente da coluna de origem). A Figura 21 ilustra a execução dos diferentes tipos de regras de transformação.

Uma regra de transformação de qualquer tipo pode referenciar qualquer outra regra de transformação, criando, portanto, um encadeamento de regras de transformação.

O encadeamento de regras permite a especificação de transformações complexas que combinem os dois tipos de regras. Desta forma, cada regra de transformação presente na especificação pode ser classificada como regra referenciada ou regra não referenciada. Como o próprio nome indica, uma regra é dita referenciada se esta for referenciada por outra regra (referenciada ou não). Ao referenciar uma regra de transformação, os sujeitos das triplas criadas pela regra referenciada são utilizados como objetos das triplas criadas pela regra que utiliza tal referência.

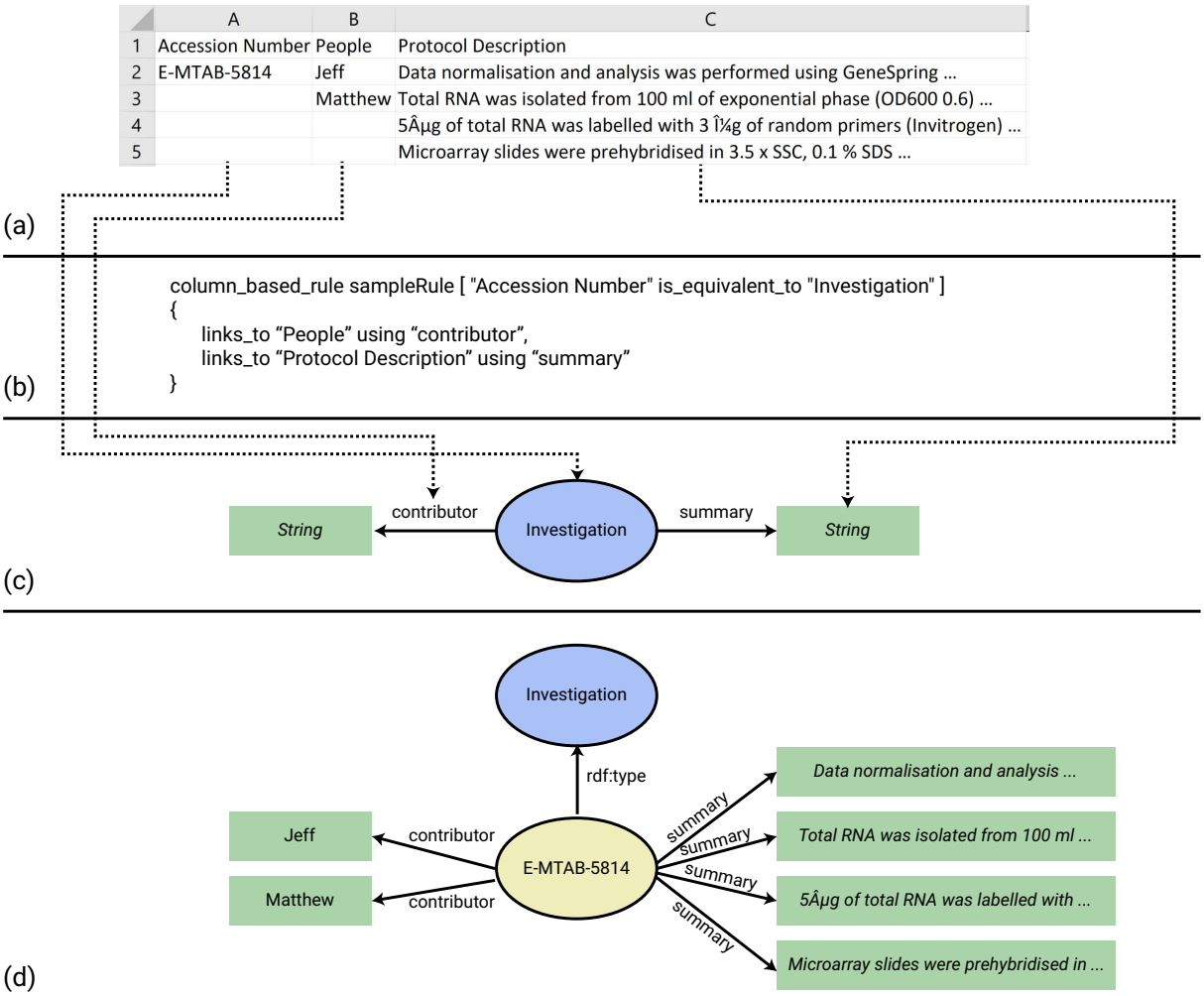
Uma regra de transformação contém um cabeçalho e um corpo. O cabeçalho especifica o tipo da regra (baseado em linha ou baseado em coluna), um identificador de regra exclusivo e o tipo do sujeito.

O tipo do sujeito especifica a equivalência entre uma coluna SSD e uma classe de ontologia. O processo de transformação cria uma ou mais triplas associando o(s) item(ns) de dados da coluna SSD (sujeito da tripla) a uma classe de ontologia (objeto da tripla) usando o predicado `rdf:type`. Dependendo do tipo da regra de transformação, o processo de transformação produz uma única tripla ou várias triplas. No caso de uma regra baseada em coluna, uma única tripla é criada para representar esta associação, enquanto que no caso de uma regra baseada em linha, várias triplas são criadas (uma para cada item de dados presente na coluna de origem). O(s) sujeito(s) da(s) tripla(s) produzida(s) é(são) então usado(s) como ponto de partida para a criação de triplas adicionais, conforme especificado pelo corpo da regra.

O corpo de uma regra especifica um conjunto de predicados e objetos (de triplas) a serem ligados aos sujeitos das triplas criadas pelo cabeçalho de regra. Objetos (de triplas) podem ser fornecidos diretamente pelos dados de uma coluna SSD de destino ou indiretamente obtidos por referência a outra regra de transformação (regra encadeada). Finalmente, o predicado representa uma propriedade de objeto ou uma propriedade de dados definida em uma ontologia. A Figura 22 ilustra a aplicação de uma regra de transformação baseada em colunas para transformar um SSD de origem contendo dados relacionados a um experimento biomédico.

O cabeçalho da regra produz uma tripla inicial associando o `Accession Number` do experimento (sujeito da tripla) ao conceito `Investigation` (objeto da tripla) usando o predicado `rdf:type` (“`E-MTAB-5814 rdf:type Investigation`”). Em seguida, para

Figura 22 – Exemplo de aplicação de regra de transformação.



Fonte: Autoria própria. Uma elipse azul representa uma classe de uma ontologia. Uma elipse amarela representa um nó RDF. Um retângulo verde representa um dado primitivo. Uma linha sólida com ponta de seta representa uma propriedade de tipo de dados ou propriedade de tipo de objeto. Uma linha pontilhada com ponta de seta indica uma referência a um conceito ou uma propriedade de uma ontologia. a) arquivo SSD de entrada; b) especificação de regra de transformação; c) ontologia de entrada; d) conjunto DAL resultante.

cada declaração do corpo de regra, o nó RDF `E-MTAB-5814` é usado como sujeito para a criação de um conjunto de triplas ligando esse sujeito a cada elemento contido na coluna de destino utilizando como ligação o predicado especificado. Assim, uma tripla separada `E-MTAB-5814` (sujeito) `contributor` (predicado) *"Coluna_Alvo"* (objeto) é criada para cada elemento contido na coluna `People` do SSD. Analogamente, uma tripla separada `E-MTAB-5814` (sujeito) `summary` (predicado) *"Coluna_Alvo"* (objeto) é criada para cada elemento contido na coluna `Protocol Description` do SSD de destino.

4.2.3 Elementos auxiliares

Elementos auxiliares permitem a definição de operações complementares sobre os itens de dados sendo transformados. Estes elementos são formados pelo Elemento de Condição, pelo Elemento de Pesquisa e por nove diferentes tipos de *flags* auxiliares.

Um elemento de condição especifica um conjunto de condições lógicas que devem ser atendidas para permitir a execução de uma regra de transformação (como um todo) ou a execução de uma declaração específica do corpo de uma regra. Todas as condições lógicas definidas devem ser avaliadas como verdadeiras para satisfazer o elemento de condição. Uma declaração de condição pode comparar valores numéricos (inteiro, float, double) ou valores textuais (string). Valores numéricos podem ser avaliados por meio dos seguintes operadores: `<` (menor que), `<=` (menor ou igual a), `>` (maior que), `>=` (maior ou igual a), `==` (igual a) ou `!=` (diferente de). Já valores textuais (string) podem ser avaliados por meio dos seguintes operadores: `==` (igual a) ou `!=` (diferente de).

Durante uma transformação, frequentemente precisamos vincular um sujeito a um objeto já definido como parte de uma tripla pré-existente em um conjunto DAL remoto. Assim, um elemento de pesquisa especifica um *script* de consulta (*query*) que será executado em um conjunto de dados remoto (*endpoint*) para recuperar o IRI do objeto desejado.

Um elemento de pesquisa contém um cabeçalho e um corpo. O cabeçalho especifica o tipo de elemento (search_element), um identificador único para o elemento de pesquisa e um endpoint SPARQL (URL). O corpo de pesquisa define a consulta SPARQL a ser executada remotamente. O processamento de um elemento de pesquisa usa a consulta SPARQL definida para recuperar remotamente o IRI do objeto desejado. Para permitir o

uso de um item de dado do SSD como parte de uma consulta SPARQL, definimos uma variável por meio da palavra-chave reservada `?tsvData`. Assim, sempre que essa palavra-chave aparecer no corpo de uma pesquisa, o item de dado do SSD sendo transformado irá substituir a variável na consulta.

Uma regra de transformação também pode fazer o uso de *flags*. *Flags* consistem em operações especiais que são executadas sobre os dados semiestruturados, permitindo assim transformações mais complexas. Essas *flags* podem ser aplicadas independentemente a uma definição de sujeito (cabeçalho da regra) ou a uma declaração do corpo de regra. Nove diferentes tipos de *flags* foram definidos:

- **Not Metadata**, a qual indica que o cabeçalho da coluna SSD deve ser usado como um item de dado ao invés de ser considerado como um metadado, isto é, um cabeçalho nomeando a coluna. Essa *flag* substitui a definição geral do elemento de configuração apenas no contexto da declaração em que essa *flag* é especificada;

- **Base IRI**, a qual especifica um IRI personalizado a ser usado como base para a criação de IRIs para os nós RDF, em substituição ao IRI base definido no elemento de configuração. Essa *flag* substitui a definição geral do elemento de configuração apenas no contexto da declaração em que essa *flag* é especificada;

- **Separator**, a qual especifica um critério de separação, como, por exemplo, vírgula (“,”), traço (“-”) ou ponto-e-vírgula (“;”), a ser usado durante o processamento de um item de dado. Durante o processamento, cada elemento separado se torna um item de dado a ser transformado, ou seja, se em um campo possui 10 itens de dados separados por vírgulas esses 10 itens serão transformados em 10 novos objetos de uma tripla;

- **Datatype**, a qual especifica o tipo primitivo de um item de dado, como, por exemplo, string, booleano ou double. Esse tipo é utilizado para especificar a propriedade de tipo de dados criada no conjunto DAL resultante;

- **Default Value**, a qual especifica um valor fixo a ser utilizado na definição do IRI de um nó RDF. Essa *flag* é particularmente útil quando se quer inserir um valor na tripla sendo criada que não faça parte do conjunto SSD de entrada, como, por exemplo, a data da transformação ou outro valor que seja constante;

- **Column**, a qual especifica o número de uma coluna SSD a ser usada como coluna de origem para a transformação. Essa *flag* é necessária sempre que um conjunto SSD não tiver um cabeçalho ou tiver várias colunas com o mesmo título de cabeçalho;

- **Search Element**, a qual especifica o elemento de pesquisa que deve ser usado para recuperar o IRI de um nó a ser usado como um objeto de uma tripla;

- **Condition Element**, a qual especifica o elemento de condição que deve ser satisfeito para permitir a execução da regra associada como um todo ou uma declaração de corpo de regra em específico;

- **Node**, a qual indica que o objeto da tripla deve ser uma instância de uma determinada classe de uma ontologia ao invés de um objeto (de tripla) de um tipo primitivo (e.g., string, inteiro).

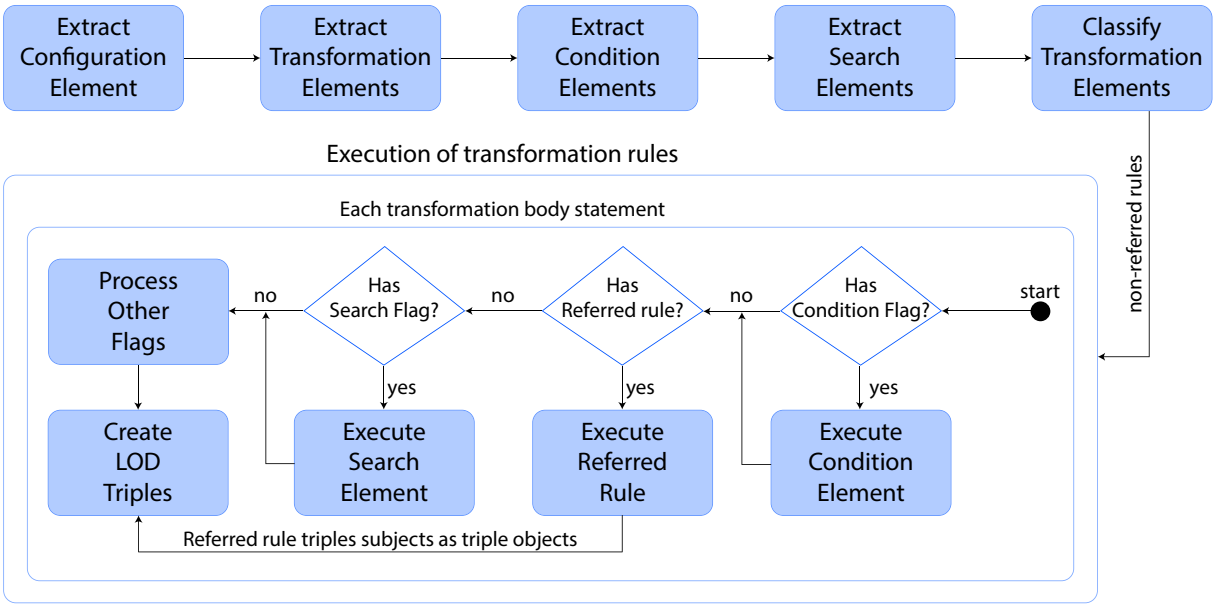
O Apêndice A apresenta a Forma Backus-Naur Estendida (EBNF) que descreve a sintaxe completa de uma especificação de transformação.

4.2.4 Execução das regras de transformação

Uma especificação de transformação de SSD para DAL é processada da seguinte forma (veja Figura 23). Inicialmente, analisamos o elemento de configuração para extrair os parâmetros gerais que devem ser usados durante o processo de transformação. Posteriormente, extraímos as regras de transformação, os elementos de condição e os elementos de pesquisa. Em seguida, classificamos as regras de transformação como referenciadas ou não referenciadas e executamos sequencialmente as regras de transformação não referenciadas.

Durante o processamento de uma regra de transformação, executamos os elementos de condição e elementos de pesquisa assim que identificados. Sempre que uma regra de transformação referenciada for identificada em uma declaração de corpo de regra, interrompemos temporariamente a execução da regra atual até que a regra referenciada associada seja completamente executada, retornando o(s) sujeito(s) da regra referenciada para que seja(m) utilizado(s) como objeto da(s) tripla(s) da regra atual. A Figura 24 ilustra a execução de uma regra baseada em coluna encadeada a uma regra baseada em linha.

Figura 23 – Processamento de uma especificação de regras de transformação.



Fonte: Autoria própria.

O cabeçalho da regra produz uma tripla inicial associando o `Accession Number` do experimento (sujeito da tripla) ao conceito `Investigation` (objeto da tripla) usando o predicado `rdf:type` ("`E-MTAB-5814 rdf:type Investigation`"). Em seguida, o corpo de regra é executado normalmente até quando o processo de transformação atinge a terceira declaração de corpo de regra, onde a regra `rule2` é referenciada. Nesta terceira declaração é ligado o experimento `Accession Number` (sujeito da tripla) a `rule2` usando o predicado `has_specified_input`. Ao executar essa declaração, o processo de transformação da `rule1` é interrompido temporariamente para que a transformação especificada por `rule2` seja executada. Como `rule2` é uma regra baseada em linha, várias triplas são criadas (uma para cada linha), vinculando o experimento `Source Name` (sujeito da tripla) ao respectivo `Genotype` (objeto da tripla) usando o predicado `genotype`. Todas as triplas criadas por `rule2` serão vinculadas ao sujeito da tripla produzido por `rule1` usando o predicado `has_specified_input`.

Figura 24 – Encadeamento de regras de transformação.

(a)

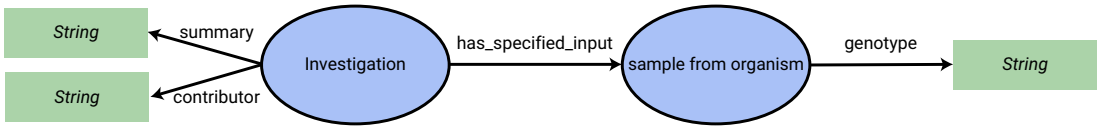
	A	B	C	D	E
1	Accession Number	People	Protocol Description	Source Name	Genotype
2	E-MTAB-5814	Jeff	Data normalisation and analysis was performed using GeneSpring ...	KO_A1-1_culture	whiB1 deletion complemented with wild type whiB1
3		Matthew	Total RNA was isolated from 100 ml of exponential phase (OD600 0.6) ...	WT_A1-1_culture	wild type genotype
4			5ÂÁg of total RNA was labelled with 3 ÌÍg of random primers (Invitrogen) ..	KO_B1-1_culture	whiB1 deletion complemented with wild type whiB1
5			Microarray slides were prehybridised in 3.5 x SSC, 0.1 % SDS ...	WT_B1-1_culture	wild type genotype

(b)

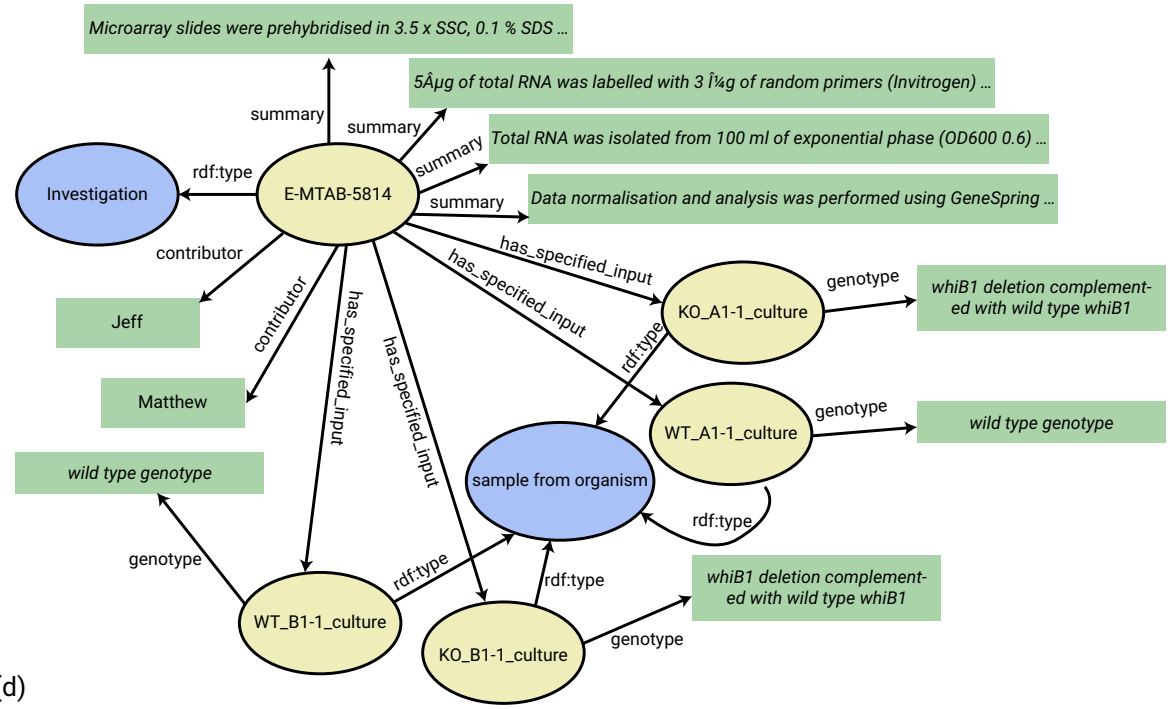
```
column_based_rule rule1 [ "Accession Number" is_equivalent_to "Investigation" ]
{
    links_to “People” using   “contributor”,
    links_to “Protocol Description” using “summary”,
    links_to rule2 using “has_specified_input”
}

row_based_rule rule2 ["Source Name" is_equivalent_to "sample from organism" ]
{
    links_to “Genotype” using “genotype”
}
```

(c)



(d)



Fonte: Autoria própria. a) arquivo SSD de entrada; b) especificação de regras de transformação; c) ontologia de entrada; d) conjunto DAL.

4.3 Ferramentas de Suporte

4.3.1 Visão geral

Três ferramentas foram desenvolvidas para prover suporte ao processo de transformação: i) SSD2LOD Transformation Tool; ii) SSD2LOD Web Service; e iii) SSD2LOD Web. A Figura 25 ilustra a relação entre essas ferramentas.

Figura 25 – Relação entre as ferramentas de suporte desenvolvidas.

Fonte: Autoria própria. Um bloco azul representa uma ferramenta. A linha sólida representa uma comunicação via HTTP. A linha pontilhada representa a invocação via linha de comando de uma ferramenta.

SSD2LOD Transformation Tool consiste em uma ferramenta de linha de comando responsável pela execução da transformação propriamente dita. A ferramenta SSD2LOD Web Service permite acesso ao SSD2LOD Transformation Tool por meio de requisições HTTP. A ferramenta SSD2LOD Web consiste em um website que consome as operações do SSD2LOD Web Service para fornecer uma interface web de acesso à transformação.

De forma geral, todas as ferramentas de suporte foram desenvolvidas utilizando a plataforma Java EE (Enterprise Edition)[1], versão 8, e o gerenciador de projetos Maven[2]. Particularmente, as ferramentas SSD2LOD Transformation Tool e SSD2LOD Web utilizaram recursos adicionais. A ferramenta SSD2LOD Transformation Tool utilizou diversas bibliotecas baseadas em Java, incluindo Jena[3], UniVocity[4], API OWL[5] e Openllet[6]. A

[1] https://www.oracle.com/technetwork/java/javaee/overview/index.html
[2] https://maven.apache.org
[3] https://jena.apache.org/
[4] https://github.com/uniVocity/univocity-parsers
[5] https://github.com/owlcs/owlapi
[6] https://github.com/Galigator/openllet

ferramenta SSD2LOD Web utilizou React[7], HTML5[8], Bootstrap[9], versão 4, e CSS[10], versão 3.

A biblioteca Jena provê um *framework* para a construção de softwares de web semântica e de dados ligados. No contexto do nosso trabalho, esta biblioteca foi usada para criar nós/triplas RDF e armazená-las em uma *triplestore* interna durante o processo de transformação. Adicionalmente, Jena foi usada para persistir em disco (arquivo DAL) as triplas contidas na *triplestore* e também para executar consultas SPARQL utilizadas para responder as questões de competência.

A biblioteca UniVocity consiste em uma coleção de *parsers* que provêem suporte à manipulação de diferentes tipos de arquivos de dados semiestruturados, incluindo arquivos de valores separados por vírgula (CSV), arquivos de valores separados por tabulação (TSV) e arquivos de largura fixa. Assim, o Univocity foi usado para interpretar os arquivos de entrada do SSD a fim de recuperar cada item de dado necessário durante o processo de transformação.

A biblioteca API OWL permite a criação, manipulação e serialização de uma ontologia OWL 2.0. Assim, a API OWL foi usada para analisar uma ontologia de origem para extrair classes e propriedades usadas na criação de uma tripla RDF.

A biblioteca Openllet consiste em uma máquina de inferência OWL2 DL que pode ser utilizada, dentre outras coisas, para verificar a consistência lógica de uma ontologia. Assim, no contexto deste trabalho, o Openllet foi usado para verificar a consistência lógica das triplas RDF criadas em relação as restrições definidas nas ontologias.

React consiste em uma biblioteca Javascript para o desenvolvimento de interfaces de usuários interativas. Bootstrap consiste em um conjunto de ferramentas que facilita o desenvolvimento de websites por meio de recursos pré-definidos de CSS e Javascript.

Após o desenvolvimento da ferramenta de suporte SSD2LOD Transformation Tool, foi utilizada a biblioteca JUnit para a implementação de casos de testes unitários automatizados. As ferramentas disponibilizadas pela biblioteca Jena permitem a implementação de diversos tipos de testes, tais como, verificação de valores retornados, testes parametrizados,

[7] https://reactjs.org/
[8] https://www.w3schools.com/html/html5_intro.asp
[9] https://getbootstrap.com/
[10] https://www.w3schools.com/css/

testes de concorrência e testes de exceções. Ao todo foram implementados 44 casos de testes unitários automatizados. Esses casos de teste variam em abrangência de cobertura de código, ou seja, existem desde testes específicos, para, por exemplo, extrair o valor de uma dada *flag*, até testes mais amplos, para, por exemplo, efetuar uma transformação utilizando múltiplos arquivos de entrada. Além disso, os casos de testes também cobrem situações de falha, garantindo, assim, que a ferramenta está tratando os erros corretamente.

4.3.2 SSD2LOD Transformation Tool

SSD2LOD Transformation Tool consiste em uma ferramenta de linha de comando para a execução de uma transformação de SSD para DAL ou execução de uma consulta SPARQL. Esta ferramenta possui três parâmetros para execução, sendo os dois primeiros obrigatórios e o terceiro opcional. O primeiro parâmetro consiste em uma *string* indicando a execução de uma transformação (valor "`runTransformation`") ou indicando a execução de dada consulta SPARQL (valor "`runQuery`"). O segundo parâmetro consiste no identificador único da transformação para que os arquivos da transformação desejada sejam distinguidos dentre os arquivos das demais transformações. O terceiro parâmetro consiste no identificador único da consulta submetida para que seja distinguida dentre as demais consultas. O terceiro parâmetro torna-se obrigatório caso o primeiro parâmetro indique a execução de uma consulta SPARQL (valor "`runQuery`").

A execução de uma transformação primeiramente carrega o arquivo contendo a especificação da transformação em formato texto e elimina as eventuais quebras de linhas e tabulações que possam existir. Posteriormente, o conteúdo deste arquivo é utilizado para extrair os elementos de configuração, regras (e suas *flags*), condicional e pesquisa. Elementos e *flags* são extraídos por meio de um conjunto de 36 expressões regulares e armazenados em um conjunto de listas, uma para cada tipo de elemento.

Após a extração das regras, são carregadas as ontologias fornecidas e suas ontologias dependentes. Durante esse processo de importação, as classes e propriedades são adicionadas a uma estrutura de dados em forma de chave e valor. Neste caso, é passado como chave o *namespace* da ontologia concatenado com o label do elemento (classe ou propriedade) (e.g., `obi:investigation`) e como valor, o IRI da classe/propriedade. Desta forma, durante o processamento da transformação, a recuperação do IRI de uma

determinada classe ou propriedade é facilitada, bastando informar a chave para obter como resposta o IRI a ser utilizado na criação das triplas.

Em seguida, são separadas as regras referenciadas das não referenciadas e inicia-se o processamento das regras não referenciadas de forma sequencial. O processamento das regras possui dois procedimentos, dependendo do tipo de regra (baseado em linha ou baseado em coluna). Se o tipo de regra for baseada em linha, a ferramenta verifica se a regra já foi processada anteriormente. Em caso positivo, a ferramenta obtém a lista de sujeitos já processados. Caso contrário, a ferramenta obtém o sujeito e verifica se há *flag* de elemento de condição. Caso exista uma *flag* de elemento de condição e esta *flag* seja avaliada como verdadeira (`"true"`), a ferramenta executa o processamento de declarações do corpo da regra para cada linha do arquivo.

Se o tipo de regra for baseada em coluna, a ferramenta verifica se a regra já foi processada anteriormente. Em caso positivo, a ferramenta utiliza a lista de sujeitos já processados. Já em caso negativo, para cada linha, a ferramenta verifica se há *flag* de elemento de condição. Caso exista uma *flag* de elemento de condição e esta *flag* seja avaliada como verdadeira (`"true"`), a ferramenta busca o sujeito da regra e dá início ao processamento das declarações do corpo da regra. Após o processamento das declarações de corpo da regra, os sujeitos da regra são adicionados a uma lista para que sejam recuperados futuramente caso a mesma regra seja novamente referenciada.

Em seguida são processadas as declarações de corpo de regra. O processamento das declarações de corpo de regra divide-se em dois procedimentos, sendo um deles para o processamento de declarações de corpo de regra cujo(s) objeto(s) da(s) tripla(s) originaram de uma coluna SSD e o outro procedimento para o processamento de declarações de corpo de regra cujo(s) objeto(s) da(s) tripla(s) originaram de uma regra referenciada.

Se o objeto for uma coluna de um conjunto SSD, a ferramenta obtém o IRI do predicado, uma lista de valores do SSD (considerando suas *flags*, como, por exemplo, `separator`) e verifica a existência da *flag* `NODE`. Em caso positivo, a ferramenta cria o(s) nó(s) RDF(s) com os valores e adiciona a(s) tripla(s) na *triplestore* do Jena. Caso contrário, a ferramenta considera que o objeto é do tipo primitivo e adiciona a tripla na *triplestore* do Jena.

Se o objeto for uma regra referenciada, a ferramenta verifica se há uma *flag* de

elemento de condição. Caso exista esta *flag* e a mesma seja avaliada como verdadeira (`"true"`), a ferramenta inicia o processamento de regras tal como explicado anteriormente, tornando-se então um processamento recursivo de regras. Após o retorno do processamento das regras (sujeitos da regra referenciada), a ferramenta adiciona na *triplestore* do Jena a tripla associando o sujeito da regra não referenciada com os sujeitos da regra referenciada por meio do predicado definido. Caso não exista uma *flag* de elemento de condição, a ferramenta não executa a regra referenciada e inicia o processamento da próxima declaração de corpo de regra.

Após a transformação de todas as regras não referenciadas, e consequentemente, de todas as regras referenciadas, a ferramenta verifica a consistência lógica da *triplestore* do Jena em relação às ontologias fornecidas. Essa verificação lógica é executada por meio de uma máquina de inferência OWL2 e garante que as restrições definidas na(s) ontologia(s) estejam sendo respeitadas. Após essa verificação, a ferramenta salva as triplas do conjunto DAL no formato especificado pelo elemento de configuração.

4.3.3 SSD2LOD Web Service

O SSD2LOD Web Service consiste em um serviço web RESTful que encapsula a ferramenta SSD2LOD Transformation Tool e disponibiliza um conjunto de operações para suporte ao processo de transformação. Exemplos destas operações incluem operações para a submissão de arquivos de entrada, a execução da transformação, a recuperação de informações de erros e a recuperação do conjunto DAL resultante, bem como operações relacionadas à exploração de dados.

A Tabela 3 apresenta os endpoints das operações disponibilizadas por este serviço. A primeira coluna apresenta cada operação do SSD2LOD Web Service. A segunda coluna apresenta o método HTTP associado a cada operação. Finalmente, a terceira coluna apresenta os endpoints e os parâmetros de URL de cada operação. O parâmetro de URL `transformationId` representa o identificador único de uma dada transformação, o qual é obtido ao criar-se uma nova transformação. O parâmetro de URL `queryId` representa o identificador único de uma dada consulta SPARQL, o qual é obtido ao submeter uma nova consulta para processamento. O parâmetro de URL `datasetName` identifica um arquivo de dados semiestruturados submetido anteriormente em uma dada

transformação. Finalmente, o parâmetro de URL `ontologyName` identifica um arquivo de ontologia submetido anteriormente em uma dada transformação.

Tabela 3 – Endpoints das operações do SSD2LOD Web Service.

URL base: http://kode.ffclrp.usp.br:8081/ssd2lod/api

Operação	Método HTTP	Endpoint
newTransformation	GET	/newTransformation
newTransformation	POST	/newTransformation
setDatasets	POST	/setDatasets/{transformationId}
setOntologies	POST	/setOntologies/{transformationId}
setRules	POST	/setRules/{transformationId}
startTransformation	GET	/startTransformation/{transformationId}
getTransformationStatus	GET	/getTransformationStatus/{transformationId}
getTransformationError	GET	/getTransformationError/{transformationId}
getLODSet	GET	/getLODSet/{transformationId}
executeQuery	POST	/executeQuery/{transformationId}
getQueryStatus	GET	/getQueryStatus/{transformationId}/{queryId}
getQueryError	GET	/getQueryError/{transformationId}/{queryId}
getQueryResult	GET	/getQueryResult/{transformationId}/{queryId}
deleteDataset	DELETE	/deleteDataset/{transformationId}/{datasetName}
deleteOntology	DELETE	/deleteOntology/{transformationId}/{ontologyName}
deleteTransformation	DELETE	/deleteTransformation/{transformationId}

A operação `newTransformation` cria uma estrutura de diretórios e arquivos de suporte para um novo ciclo de transformação a fim de isolar os arquivos relacionados do novo ciclo de transformação de ciclos anteriores. Os diretórios são utilizados no armazenamento dos arquivos de entrada (ontologia(s), especificação de regras de transformação e arquivos semiestruturados de dados) e os arquivos de suporte são utilizados para armazenar informações de *status* e de eventuais erros ocorridos durante o processo de transformação. Os arquivos de entrada podem ser submetidos separadamente por meio de operações específicas. No entanto, esta operação pode ser utilizada também com o método HTTP POST para criar a estrutura de diretórios e submeter todos os arquivos de entrada em uma única requisição. Tal estrutura é identificada univocamente por meio de um identificador alfanumérico. Esse identificador é retornado ao usuário como resposta desta operação e deve ser utilizado nas demais operações que envolvam essa transformação.

A operação `setDatasets` armazena um ou mais arquivos de dados semiestruturados na estrutura de arquivos de uma determinada transformação. Para tanto, essa operação recebe como parâmetro de URL o identificador único da transformação e dois parâmetros de corpo de requisição: `dataset` e `format`. O primeiro parâmetro consiste em um ou mais arquivos de dados semiestruturados no formato “tsv” ou “csv”. O segundo parâmetro

consiste em uma string para informar o formato dos arquivos semiestruturados fornecidos, tendo apenas dois valores aceitáveis: “tsv” ou “csv”. Essa operação retorna os nomes dos arquivos submetidos com sucesso.

A operação `setOntologies` armazena um ou mais arquivos de ontologia na estrutura de arquivos de uma determinada transformação. Para tanto, essa operação recebe como parâmetro de URL o identificador único da transformação e o parâmetro de corpo de requisição `ontology`. Este parâmetro consiste em um ou mais arquivos OWL. Essa operação retorna os nomes das ontologias submetidas com sucesso.

A operação `setRules` armazena um arquivo de especificação de transformação na estrutura de arquivos de uma determinada transformação. Para tanto, essa operação recebe como parâmetro de URL o identificador único da transformação e o parâmetro de corpo de requisição `rules`. Este parâmetro consiste em um arquivo de texto contendo uma especificação de regras de transformação. Essa operação retorna o nome do arquivo enviado, caso este tenha sido armazenado com sucesso.

A operação `getTransformationStatus` recupera uma string indicativa do *status* da transformação. Essa operação pode retornar seis diferentes valores: `"CREATED"`, para indicar que a estrutura de arquivos foi recém criada; `"READY"`, para indicar que os arquivos necessários para iniciar o processo de transformação foram submetidos (arquivos semiestruturados de dados, ontologia e especificação de regras de transformação); `"RUNNING"`, para indicar que o processo de transformação foi iniciado e encontra-se em execução; `"SUCCEEDED"`, para indicar que o processo de transformação encerrou-se com sucesso; `"FAILED"`, para indicar que o processo de transformação finalizou com erro; e finalmente, `"REMOVED"`, para indicar que a estrutura de diretório associada a uma dada transformação, bem como demais informações foram excluídas com sucesso.

A operação `startTransformation` inicia o processo de transformação a ser executado pela ferramenta SSD2LOD Transformation Tool. A operação `getTransformationError` recupera eventuais mensagens de erros que podem ter ocorrido durante a transformação, enquanto que a operação `getLODSet` recupera o conjunto DAL transformado do servidor. A operação `executeQuery` submete uma dada consulta SPARQL e inicia sua execução. Para tanto, essa operação recebe como parâmetro de URL o identificador único da transformação e o parâmetro de corpo de requisição `query` no qual consiste na

consulta SPARQL em texto puro. Ao utilizar essa operação, a ferramenta armazena a consulta fornecida dentro da estrutura de arquivos da transformação indicada, gera um identificador alfanumérico único para a consulta, inicia o processamento da consulta e retorna o identificador único da consulta SPARQL submetida.

A operação `getQueryStatus` recupera uma string indicativa do *status* da consulta SPARQL. Essa operação pode retornar quatro diferentes valores: `"CREATED"`, para indicar que o arquivo contendo a consulta foi recém criado; `"RUNNING"`, para indicar que a consulta SPARQL foi iniciada e encontra-se em execução; `"SUCCEEDED"`, para indicar que a consulta SPARQL finalizou com sucesso; e finalmente, `"FAILED"`, para indicar que a consulta SPARQL finalizou com erro.

A operação `getQueryError` recupera eventuais mensagens de erros que podem ter ocorrido durante a execução da consulta SPARQL. A operação `getQueryResult` recupera o conjunto resultante da consulta SPARQL e o disponibiliza para *download*.

As operações `deleteDataset` e `deleteOntology` removem, respectivamente, um determinado arquivo de dados semiestruturados ou um arquivo de ontologia da estrutura de arquivos de uma determinada transformação. A operação `deleteTransformation` remove toda a estrutura de diretórios e arquivos de uma determinada transformação.

Por meio deste conjunto de operações, a ferramenta provê suporte a diferentes etapas da abordagem SSD2LOD Transformation Approach proposta neste trabalho. Essa ferramenta permite criar uma nova estrutura de arquivos e diretórios para a execução de um ciclo de transformação utilizando a operação `newTransformation`. Adicionalmente, essa ferramenta permite submeter todos os arquivos de entrada necessários utilizando as operações `setDatasets`, `setOntologies`, `setRules` e `startTransformation`. Em seguida, a atividade de transformação em si pode ser iniciada por meio da operação `startTransformation` e o *status* da transformação pode ser monitorado por meio da operação `getTransformationStatus`. Caso a transformação termine com sucesso (situação `"SUCCEEDED"`), o conjunto DAL resultante pode ser recuperado por meio da operação `getLODSet`. Caso contrário, isto é, caso ocorra um erro, a mensagem de erro pode ser recuperada por meio da operação `getTransformationError`.

Após a transformação dos dados, a atividade de exploração pode ser iniciada por meio da submissão de uma consulta SPARQL utilizando a operação `executeQuery`. A

situação da consulta pode ser monitorada por meio da operação `getQueryStatus`. Caso a consulta termine com sucesso (situação `"SUCCEEDED"`), o conjunto resultante da consulta pode ser recuperado por meio da operação `getQueryResult`. Caso contrário, isto é, a consulta termine com um erro, a mensagem de erro pode ser recuperada por meio da operação `getQueryError`.

Finalmente, caso seja necessário remover um arquivo de dados semiestruturados, um arquivo de ontologia ou uma transformação inteira, essas ações podem ser executadas por meio das operações `deleteDataset`, `deleteOntology` e `deleteTransformation`, respectivamente.

4.3.4 SSD2LOD Web

O software SSD2LOD Web[11] consiste em um portal web desenvolvido para facilitar a utilização dos softwares envolvidos na transformação de SSD para DAL. Este sistema provê uma interface web que simplica a execução da abordagem SSD2LOD Transformation Approach sem que seja necessário instalar qualquer software ou haja a preocupação com o ambiente/infraestrutura de execução (máquina) da transformação. O portal permite a submissão dos arquivo(s) SSDs de entrada, arquivo(s) de ontologia(s), arquivo de especificação de regras de transformação, inicialização do processo de transformação e a exploração dos dados por meio de consultas SPARQL.

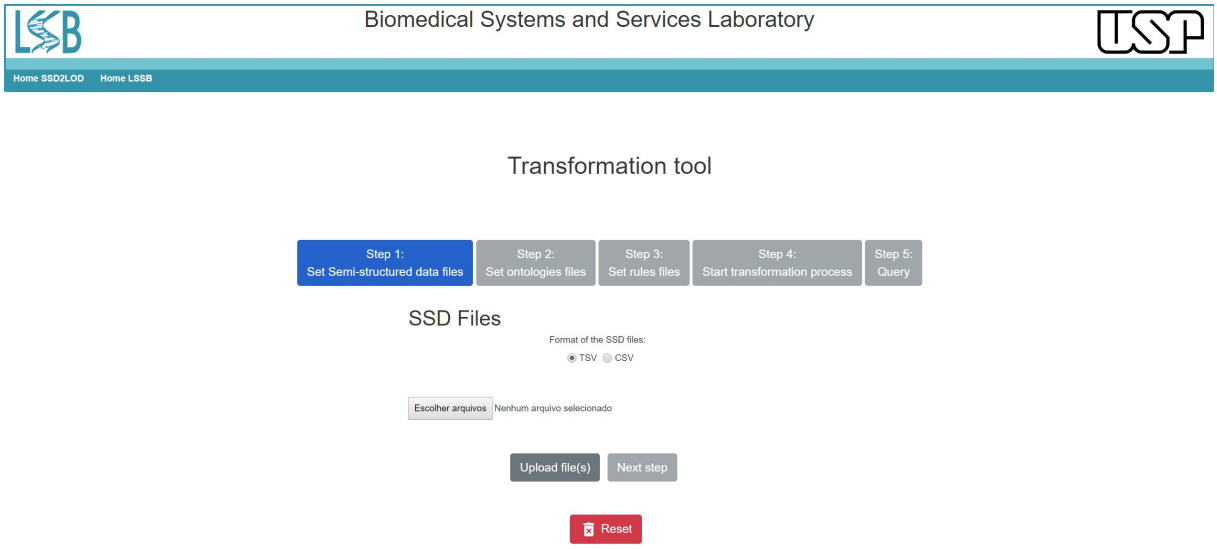
Todas as funcionalidades disponíveis no portal são baseadas no consumo das operações do SSD2LOD Web Service disponibilizadas publicamente na web. Mensagens de sucesso ou de erro são apresentadas a cada operação efetuada no portal. A Figura 26 illustra a interface web da ferramenta.

Após a inicialização do processo de transformação, o próprio portal monitora a situação da transformação periodicamente a fim de notificar o término de uma transformação. Em caso de sucesso na execução, o sistema permite o *download* do conjunto DAL ou permite seguir para o próximo passo (etapa) da abordagem. Caso contrário, em caso de falha na execução, o sistema apresenta a mensagem de erro associada.

Na etapa de exploração, o sistema solicita a inserção da consulta SPARQL para que a mesma seja executada sobre o conjunto DAL gerado no passo anterior. Assim como na

[11] http://purl.org/lssb/ssd2lod

Figura 26 – Ferramenta SSD2LOD Web.



Fonte: Autoria própria.

etapa de transformação, o portal monitora a situação da exploração periodicamente. Ao finalizar a execução da consulta, o sistema permite o *download* do resultado da respectiva consulta ou a realização de uma nova consulta. Neste caso, pode-se modificar a consulta e reexecutar o processo de exploração.

4.3.5 SSD2LOD File Management

SSD2LOD File Management consiste em uma API para o gerenciamento dos arquivos a serem utilizados durante o processo de transformação. Essa API é utilizada pelas ferramentas SSD2LOD Web Service e SSD2LOD Transformation Tool. Essa API foi desenvolvida para padronizar o mecanismo de acesso aos arquivos necessários durante o processo de transformação, assim como para monitorar informações de *status* dos processos de transformação e exploração.

Nessa API foram implementadas 23 funções para gerenciar arquivos semiestruturados, ontologias, consultas SPARQL, arquivos de *status* da transformação, arquivos de erros da transformação, e, finalmente, arquivo DAL. Dentre as principais funções desta API temos:

- **Gerar identificador único**. Cada ciclo de transformação e cada consulta SPARQL

possui um identificador único. Desta forma, foi desenvolvida uma função para a geração de novos identificadores únicos;

- **Controlar estrutura de arquivos**. Funções para a criação e o gerenciamento de uma estrutura de diretórios e um arquivo de *status* para o monitoramento da situação do processo de transformação;

- **Gerenciar arquivos SSD**. Funções para a adição, recuperação e exclusão de arquivos SSD em diretórios separados para cada processo de transformação (localizáveis por meio do identificador único);

- **Gerenciar arquivos de ontologia**. Funções para a adição, recuperação e exclusão de arquivos de ontologia em diretórios separados para cada processo de transformação (localizáveis por meio do identificador único);

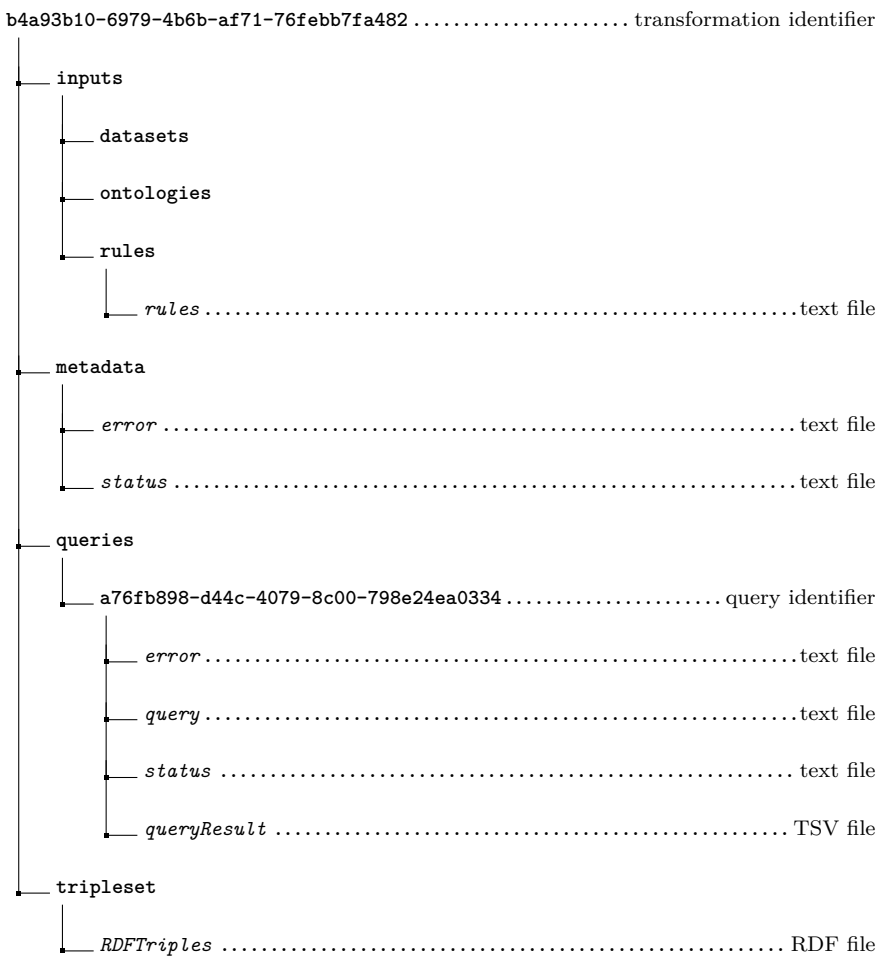
- **Gerenciar consultas SPARQL**. Funções para a geração de arquivo de *status* para monitoramento da exploração do conjunto DAL, e adição e recuperação de arquivos de consulta separados por identificador único;

- **Gerenciar arquivo DAL**. Função de recuperação do arquivo DAL gerado após a transformação.

Ao inicializar um novo ciclo de transformação é gerado um identificador único e criada uma estrutura de diretórios vinculado a este identificador para que os arquivos submetidos sejam armazenados separamente. A Figura 27 ilustra um exemplo da estrutura de diretórios criado pelo SSD2LOD File Management. Os subdiretórios consistem em: i) `inputs`, para armazenar os arquivos de entrada; ii) `metadata`, para armazenar a situação do processo de transformação; iii) `queries`, para armazenar as consultas SPARQL submetidas; e finalmente, iv) `tripleset`, para armazenar o conjunto DAL.

O subdiretório `inputs` contém ainda três subdiretórios: `datasets`, para armazenar os arquivos SSD, `ontologies`, para armazenar os arquivos de ontologia e `rules`, para armazenar o arquivo de regras de transformação. Ao submeter um arquivo de regras, o conteúdo desse arquivo é armazenado dentro do arquivo `rules`, presente dentro do diretório `rules`.

Figura 27 – Estrutura de diretórios gerenciado pelo SSD2LOD File Management.



Fonte: Autoria própria.

O subdiretório `metadata` contém dois arquivos de controle: `error`, para armazenar as mensagens de eventuais erros que possam acontecer durante o processo de transformação, e `status`, para armazenar a situação do processo de transformação.

O subdiretório `queries` contém um subdiretório univocamente identificado para cada consulta submetida. Cada subdiretório de consulta contém quatro arquivos: `error`, para armazenar as mensagens de eventuais erros que possam acontecer durante o processo de exploração; `query` para armazenar a consulta submetida; `status`, para armazenar a situação do processo de exploração; e finalmente, `queryResult`, para armazenar o resultado da execução da consulta.

O subdiretório `tripleset` contém um arquivo com as triplas RDF do conjunto DAL. Esse arquivo é gerado apenas ao final do processo de transformação, seguindo o

formato de dados especificado no elemento de configuração (RDF/XML, N3 ou Turtle).

5

Prova de Conceito

Este capítulo apresenta uma prova de conceito que ilustra a utilização da abordagem SSD2LOD Transformation Approach. Este capítulo objetiva demonstrar a abordagem de transformação e, ao mesmo tempo, o potencial de descoberta de conhecimento a partir de um conjunto de dados transformado.

Este capítulo está estruturado da seguinte forma: a seção 5.1 apresenta uma visão geral da prova de conceito; a seção 5.2 apresenta o desenvolvimento da etapa de definição de questões de competência da abordagem; a seção 5.3 apresenta uma visão geral sobre a ontologia de suporte utilizada na prova de conceito; a seção 5.4 discute a compartimentalização das questões de competência em ciclos de transformação e apresenta a aplicação das atividades de transformação e exploração de cada ciclo de transformação criados; finalmente, a seção 5.5 discute os resultados obtidos por meio desta prova de conceito.

5.1 Visão geral

A fim de ilustrar nosso processo de transformação de SSD para DAL, selecionamos um estudo de genômica funcional, publicado por Spath et al. (2017), contendo um conjunto de dados processados disponíveis no Array Express[1]. Nessa prova de conceito aplicamos a abordagem SSD2LOD Transformation Approach utilizando as ferramentas de suporte para transformar parte dos dados publicados em DAL, a fim de facilitar a descoberta e a exploração de conhecimento biológico relevante por um biologista interessado.

[1] www.ebi.ac.uk/arrayexpress/experimentos/E-MTAB-5412/

O trabalho de Spath et al. (2017) objetivou investigar o papel da citocina Granulocyte-Macrophage Colony-Stimulating Factor (GM-CSF) para a expansão de células mielóides inflamatórias e seu efeito no Sistema Nervoso Central (SNC). Assim, como parte de seu trabalho, os autores realizaram a análise diferencial de dados de RNA-Seq de `mus musculus`, obtidos de células mielóides inflamatórias isoladas de diferentes órgãos individuais, incluindo SNC, rim, fígado, pulmão e baço.

Os arquivos de dados do experimento incluem arquivos contendo informações sobre o próprio experimento (arquivos IDF e SDRF) e arquivos contendo resultados da análise diferencial. Os dados presentes nos arquivos de resultados da análise diferencial e no arquivo SDRF estão organizados de acordo com o estilo em matriz de valores, enquanto que os dados presentes no arquivo IDF estão organizados de acordo com o estilo em lista de valores. Portanto, desenvolvemos uma ferramenta simples para converter o arquivo IDF para o estilo de matriz de valores.

Os resultados da análise diferencial estão disponíveis em arquivos distintos, um para cada tipo de análise diferencial realizada: SNC *versus* rim (*CNSOverKidney.txt*), SNC *versus* fígado (*CNSOverLiver.txt*), SNC *versus* pulmão (*CNSOverLung.txt*) e SNC *versus* baço (*CNSOverSpleen.txt*). Cada um desses arquivos contém diferentes tipos de informações associadas a cada gene, incluindo os resultados da própria análise diferencial (log2 ratio, p-value, fdr, etc) e lista de processos biológicos, funções moleculares e componentes celulares associados à ontologia Gene Ontology. Além disso, há um arquivo adicional de análise diferencial contendo a interseção (CNSOverKidneyLiverLung (Intersection).txt) entre as análises SNC *versus* rim, SNC *versus* fígado e SNC *versus* pulmão.

As seguintes atividades foram realizadas nesta prova de conceito: 1) definição de um conjunto de questões de competência; 2) definição de uma ontologia de suporte; 3) agrupamento de questões de competência em ciclos de transformações e especificação de regras de transformação; e 4) transformação e exploração de dados.

5.2 Definição das questões de competência

A primeira atividade realizada nesta prova de conceito foi a definição de um conjunto de questões de competência (QC). Ao todo foram especificadas treze (13) questões de compe-

tência englobando tanto dados sobre o experimento quanto dados da análise diferencial.

Questões de competência de âmbito geral foram elaboradas a fim de permitir a recuperação das principais características do experimento (QC 01 a 05). Questões de competência envolvendo dados da análise diferencial foram elaboradas a fim de extrair informações biologicamente relevantes (QC 06 a 13). A Tabela 4 mostra as treze questões de competência especificadas.

Tabela 4 – Questões de competência especificadas.

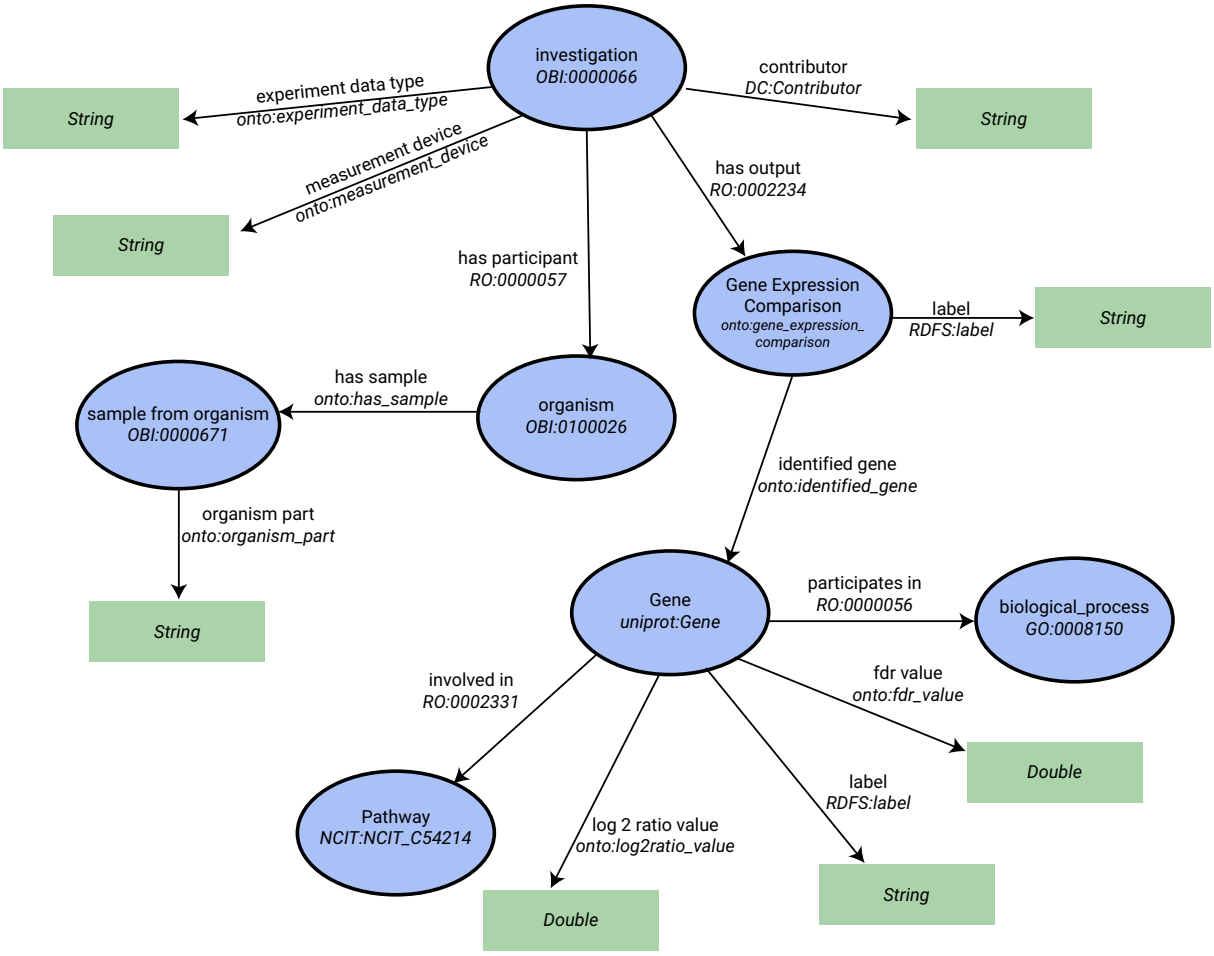
QC 01	Quais são os nomes dos pesquisadores envolvidos no experimento?
QC 02	Qual o tipo de dados de expressão gênica envolvido no experimento?
QC 03	Qual foi a plataforma usada para obter os dados de expressão gênica?
QC 04	Quais organismos estão envolvidas no experimento?
QC 05	Quais tipos de amostras foram usadas no experimento?
QC 06	Quais genes são regulados positivamente na amostra SNC *versus* rim?
QC 07	Quais genes são regulados negativamente na amostra SNC *versus* rim?
QC 08	Quais genes são regulados positivamente na amostra SNC *versus* fígado?
QC 09	Quais genes são regulados negativamente na amostra SNC *versus* fígado?
QC 10	Quais genes são regulados positivamente na amostra SNC *versus* pulmão?
QC 11	Quais genes são regulados negativamente na amostra SNC *versus* pulmão?
QC 12	Quais são as principais vias metabólicas associadas a genes diferencialmente expressos das amostras de SNC *versus* rim, fígado e pulmão?
QC 13	Quais são os principais processos biológicos associados a genes diferencialmente expressos das amostras de SNC *versus* rim, fígado e pulmão?

5.3 Ontologia de suporte

A segunda atividade realizada nesta prova de conceito foi a definição de uma ontologia para suporte ao processo de transformação. Para tanto, definimos uma ontologia personalizada reutilizando um conjunto de conceitos e predicados oriundos de cinco diferentes ontologias: *Ontology for Biomedical Investigations* (OBI)[2] (BANDROWSKI et al., 2016), *Relations Ontology* (RO)[3] (SMITH et al., 2005; ARP; SMITH; SPEAR, 2015), *UniProt RDF Schema* (URDFS)[4], *NCI Thesaurus* (NCIT)[5] (SIOUTOS et al., 2007; ABEYSINGHE et al., 2018) e *Gene Ontology* (GO)[6] (ASHBURNER et al., 2000; DESSIMOZ; ŠKUNCA, 2017).

[2] http://purl.obolibrary.org/obo/obi.owl
[3] http://purl.obolibrary.org/obo/ro.owl
[4] http://www.uniprot.org/core/
[5] http://purl.obolibrary.org/obo/ncit.owl
[6] http://purl.obolibrary.org/obo/go.owl

Figura 28 – Ontologia de suporte.



Fonte: Autoria própria. Uma elipse azul representa uma classe, enquanto um retângulo verde representa um valor literal. Uma flecha conectando duas classes representa uma propriedade de objeto, enquanto uma flecha conectando uma classe a um valor literal representa uma propriedade de tipo de dados.

A Figura 28 ilustra um trecho da ontologia desenvolvida contendo os *labels* das classes e das propriedades juntamente com seus respectivos *namespaces* e identificadores. A classe `investigation` é utilizada para representar o experimento, o qual possui como participante (`has participant`), um organismo (`organism`). Um experimento possui como resultado (`has output`) uma análise diferencial (`Gene Expression Comparison`). Um organismo possui (`has sample`) amostras de organismos (`sample from organism`). Já uma análise diferencial está relacionada (`identified genes`) a diversos genes (`Gene`). Cada gene está envolvido (`involved in`) em uma via metabólica (`Pathway`) e participa (`participates in`) de um processo biológico (`biological_process`).

5.4 Ciclos de transformação

A terceira atividade realizada nesta prova de conceito foi a especificação das regras de transformação. Contudo, antes que as regras fossem especificadas, agrupamos as questões de competência em quatro ciclos de transformação para facilitar tanto a especificação das regras de transformação quanto a execução das transformações propriamente ditas. Cada ciclo de transformação criado envolveu uma ou mais questões de competência. A Tabela 5 apresenta a divisão das questões de competência entre os diferentes ciclos de transformação (CT).

Tabela 5 – Divisão de questões de competência em ciclos de transformação.

CT 01	QC 01	Quais são os nomes dos pesquisadores envolvidos no experimento?
	QC 02	Qual o tipo de dados de expressão gênica envolvido no experimento?
	QC 03	Qual foi a plataforma usada para obter os dados de expressão gênica?
	QC 04	Quais organismos estão envolvidas no experimento?
	QC 05	Quais tipos de amostras foram usadas no experimento?
CT 02	QC 06	Quais genes são regulados positivamente na amostra SNC *versus* rim?
	QC 07	Quais genes são regulados negativamente na amostra SNC *versus* rim?
	QC 08	Quais genes são regulados positivamente na amostra SNC *versus* fígado?
	QC 09	Quais genes são regulados negativamente na amostra SNC *versus* fígado?
	QC 10	Quais genes são regulados positivamente na amostra SNC *versus* pulmão?
	QC 11	Quais genes são regulados negativamente na amostra SNC *versus* pulmão?
CT 03	QC 12	Quais são as principais vias metabólicas associadas a genes diferencialmente expressos das amostras de SNC *versus* rim, fígado e pulmão?
CT 04	QC 13	Quais são os principais processos biológicos associados a genes diferencialmente expressos das amostras de SNC *versus* rim, fígado e pulmão?

O conjunto de arquivos SSD de origem usados no processo de transformação foi o critério básico usado para agrupar as questões de competência 01 a 05 no ciclo de transformação 01 e as questões de competência remanescentes nos demais ciclos. O primeiro ciclo envolveu o uso de um conjunto de arquivos SSD de origem contendo informações sobre o próprio experimento de expressão gênica, enquanto que os ciclos de transformação

02 a 04 envolveram o uso de arquivos SSD de origem contendo os resultados dos dados de análise diferencial.

Além disso, as questões de competência 06 a 11 foram agrupadas no ciclo de transformação 02, pois essas questões visavam extrair conhecimento biológico diretamente do SSD de entrada, enquanto as questões de competência 12 e 13 envolviam não apenas a extração do conhecimento biológico do SSD de entrada, mas também sua conexão com um conjunto de dados remoto. Finalmente, as questões de competência 12 e 13 foram atribuídas aos ciclos de transformação 03 e 04, respectivamente, dado que a transformação envolvia a conexão com um conjunto de dados remoto diferente, melhorando assim o desempenho da transformação e da exploração de dados.

O Apêndice B contém a especificação das regras de transformação e consultas SPARQL definidas nesta prova de conceito. Adicionalmente, todos os arquivos relacionados a prova de conceito encontram-se disponíveis online[7].

5.4.1 Ciclo de transformação 01

5.4.1.1 Especificação das regras de transformação

A especificação das regras de transformação para o primeiro ciclo de transformação foi trivial, pois envolveu a definição de regras simples relacionadas aos arquivos SSD de origem IDF e SDRF. A Listagem 2 apresenta a especificação de regras de transformação definidas neste ciclo.

O elemento de configuração (linhas 1–5) especifica o IRI base para a criação dos nós RDF (`http://example.org/onto/individual/`), a sintaxe de exportação do conjunto DAL (`N-Triples`) e um *namespace* para a ontologia OBI usada nas regras de transformação (`obi`). Para este ciclo de transformação foram criadas três regras, sendo uma baseada em coluna, identificada como `investigation`, e duas baseadas em linhas, identificadas como `organismTransformation` e `sampleTransformation`.

A regra `investigation` produz uma instância da classe `investigation` (`obi:investigation`) (linha 7) para representar o experimento como um todo. Essa instância é utilizada como sujeito das triplas criadas durante o processamento do corpo de regra. O

[7] https://github.com/gcsgpp/SSD2LOD_CaseStudy

```
1  config_element {
2    "default_baseIRI" = "http://example.org/onto/individual/",
3    "export_syntax" = "N-Triples",
4    "namespace" = "obi" refers_to "http://purl.obolibrary.org/obo/OBI"
5  }
6
7  column_based_rule investigation [ "Comment[ArrayExpressAccession]" /BASEIRI("https://www.
       ebi.ac.uk/arrayexpress/experiments/", "ebi-ae") is_equivalent_to "obi:investigation"
       ]{
8    links_to "Person First Name" /; "Person Last Name" using "Contributor",
9    links_to "Comment[AEExperimentType]" using "experiment data type",
10   links_to "Protocol Hardware" using "measurement device",
11   links_to organismTransformation using "has participant"
12 }
13
14 row_based_rule organismTransformation [ "Characteristics[organism]" is_equivalent_to "obi
       :organism"]{
15   links_to "Characteristics[organism]" using "label",
16   links_to sampleTransformation using "has sample",
17 }
18
19 row_based_rule sampleTransformation ["Source Name" is_equivalent_to "obi:sample from
       organism"]{
20   links_to "Characteristics[organism part]" using "organism part"
21 }
```

Listagem 2 – Regras de transformação para o ciclo de transformação 01.

corpo de regra possui quatro declarações que utilizam os predicados `Contributor` (linha 8), `experiment data type` (linha 9), `measurement device` (linha 10) e `has participant` (linha 11). A declaração da linha 11 é utilizada para ligar o sujeito da regra `investigation` às triplas criadas pela regra `organismTransformation`.

A regra `organismTransformation` produz uma instância da classe `organism` (`obi:organism`) (linha 14) para representar o organismo utilizado no experimento. Essa instância é utilizada como sujeito das triplas criadas durante o processamento do corpo de regra. O corpo de regra possui duas declarações que utilizam os predicados `label` (linha 15) e `has sample` (linha 16). A declaração é utilizada para ligar o sujeito da regra `organismTransformation` às triplas criadas pela regra `sampleTransformation`.

A regra `sampleTransformation` cria instâncias da classe `sample from organism` (`obi:sample from organism`) (linha 19) para representar cada amostra do organismo

utilizada no experimento. Essas instâncias são utilizadas como sujeitos das triplas criadas durante o processamento do corpo de regra. O corpo de regra possui uma declaração contendo o predicado `organism part` (linha 20), que liga o(s) sujeito(s) a uma string representando o nome da amostra.

5.4.1.2 Transformação e exploração dos dados

Após a especificação das regras de transformação, utilizamos a ferramenta SSD2LOD Web para executar o processo de transformação propriamente dito. Para isso, submetemos os arquivos SSD (IDF e SDRF), a ontologia de suporte, a especificação de regras de transformação e iniciamos o processo de transformação, o qual resultou em um conjunto DAL contendo 53 triplas. Posteriormente, realizamos a etapa de exploração de dados. Definimos um conjunto de consultas SPARQL e as executamos usando a ferramenta SSD2LOD Web para obter as respostas para as questões de competência envolvidas.

A definição das consultas SPARQL para este ciclo de transformação 01 foi direta, pois envolvia a simples recuperação de informações do conjunto DAL produzido pela atividade de transformação de dados correspondente. A Listagem 3 apresenta a consulta SPARQL para a obtenção da resposta para a questão *"Quais são os nomes dos pesquisadores envolvidos no experimento?"* (QC 01).

```
1 PREFIX dcterms: <http://purl.org/dc/elements/1.1/>
2 SELECT DISTINCT ?researcher_name WHERE {
3
4   ?s dcterms:contributor ?researcher_name .
5
6 }
```

Listagem 3 – Consulta SPARQL para a questão de competência 01.

A linha 1 especifica um *namespace* para o vocabulário `dcterms` a fim de facilitar sua referência dentro da consulta. A linha 2 seleciona como resposta da consulta os valores armazenados na variável `researcher_name`. A linha 04 seleciona todos os sujeitos e objetos que sejam ligados entre si por meio do predicado `dcterms:contributor`. Esses valores são armazenados, respectivamente, nas variáveis `s` e `researcher_name`. A resposta desta consulta teve como retorno o valor `Sabine Spath`. A Figura 29 apresenta os resultados de cada consulta SPARQL utilizada para responder as questões de competência 01 a 05.

Figura 29 – Resultados das consultas SPARQL do ciclo de transformação 01.

CQ 1 Query Result	CQ 2 Query Result	CQ 3 Query Result	CQ 4 Query Result	CQ 5 Query Result
researcher_name	data_type	platform	organism_name	sample_name
"Sabine Spath"	"RNA-seq of coding RNA"	"Illumina HiSeq 4000" "BD FACS Aria III"	"Mus musculus"	"lung" "kidney" "spleen" "central nervous system" "liver"

Fonte: Autoria própria.

5.4.2 Ciclo de transformação 02

5.4.2.1 Especificação das regras de transformação

A especificação das regras de transformação para o segundo ciclo de transformação também foi simples. Porém, a especificação destas regras envolveu o uso de elementos condicionais para identificar genes regulados positivamente e genes regulados negativamente em três diferentes arquivos contendo os dados da análise diferencial. A identificação de um gene diferencialmente expresso tomou como base os mesmos critérios definidos em Spath et al. (2017), isto é, um gene é regulado positivamente se possuir valor de FDR menor ou igual a 0.05 e valor de log2ratio maior ou igual a 1. De forma análoga, um gene é regulado negativamente se possuir valor de FDR menor ou igual a 0.05 e valor de log2ratio menor ou igual a -1.

As questões de competência associadas a este ciclo de transformação estavam relacionadas a três diferentes arquivos de análise diferencial: *CNSOverKidney.txt*, *CNSOverLiver.txt* e *CNSOverLung.txt*. Portanto, criamos um conjunto de regras de transformação e elementos de condição para cada arquivo de análise a ser transformado. Cada conjunto conteve três (3) regras de transformação e dois (2) elementos de condição. A diferença entre cada conjunto consiste apenas no nome do arquivo de análise de transformação a ser processado. Desta forma, ao final da especificação das regras de transformação, o documento resultante conteve nove (9) regras de transformação e seis (6) elementos de condição. A Listagem 4 apresenta um trecho da especificação das regras de transformação definidas para o segundo ciclo de transformação.

As três regras de transformação apresentada na Listagem 4 visam transformar os genes diferencialmente expressos definidos no conjunto SSD *CNSOverKidney.txt*. Tais regras foram identificadas como `CNSOverKidney` (linha 6), `upRegulatedKidney` (linha 11) e

```
config_element {
  "default_baseIRI" = "http://example.org/onto/individual/",
  "export_syntax" = "N-Triples",
  "namespace" = "uniprot" refers_to "http://purl.uniprot.org/core/"
}
row_based_rule CNSOverKidney [ "" /DefaultValue("CNSOverKidney") is_equivalent_to "Gene
    expression comparison"]{
  links_to upRegulatedKidney using "identified gene",
  links_to downRegulatedKidney using "identified gene",
  links_to "" /DefaultValue("CNSOverKidney") using "label"
}
row_based_rule upRegulatedKidney ["" /COL("gene_id", "result_CNS_over_kidney.tsv") /; ""
    /DefaultValue("CNSOverKidney") /CE(ConditionUpRegulatedCNSOverKidney)
    is_equivalent_to "uniprot:Gene"]{
  links_to "" /COL("gene_name", "result_CNS_over_kidney.tsv") using "label",
  links_to "" /COL("fdr", "result_CNS_over_kidney.tsv") /DT("double") using "fdr value",
  links_to "" /COL("log2 Ratio", "result_CNS_over_kidney.tsv") /DT("double") using "log 2
     ratio value"
}
row_based_rule downRegulatedKidney ["" /COL("gene_id", "result_CNS_over_kidney.tsv")  /;
    "" /DefaultValue("CNSOverKidney") /CE(ConditionDownRegulatedCNSOverKidney)
    is_equivalent_to "uniprot:Gene"]{
  links_to "" /COL("gene_name", "result_CNS_over_kidney.tsv") using "label",
  links_to "" /COL("fdr", "result_CNS_over_kidney.tsv") /DT("double") using "fdr value",
  links_to "" /COL("log2 Ratio", "result_CNS_over_kidney.tsv") /DT("double") using "log 2
     ratio value"
}
condition_element ConditionUpRegulatedCNSOverKidney{
  "" /COL("fdr", "result_CNS_over_kidney.tsv") <= "0.05",
  "" /COL("log2 Ratio", "result_CNS_over_kidney.tsv") >= "1"
}
condition_element ConditionDownRegulatedCNSOverKidney{
  "" /COL("fdr", "result_CNS_over_kidney.tsv") <= "0.05",
  "" /COL("log2 Ratio", "result_CNS_over_kidney.tsv") <= "-1"
}
```

Listagem 4 – Regras de transformação para o ciclo de transformação 02.

`downRegulatedKidney` (linha 16). A regra `CNSOverKidney` é responsável por agregar os nós RDFs representando os genes regulados positivamente e regulados negativamente identificados na análise diferencial `CNSOverKidney`. A regra `upRegulatedKidney` é responsável por transformar os genes regulados positivamente enquanto que a regra `downRegulatedKidney` é responsável por transformar os genes regulados negativamente. Os dois elementos condicionais especificados foram identificados como `ConditionUpRegulatedCNSOverKidney` (linha 21), contendo os critérios para determinar se o gene é regulado positivamente, e `ConditionDownRegulatedCNSOverKidney` (linha 25), contendo os critérios para determinar se o gene é regulado negativamente.

A regra `CNSOverKidney` produz uma instância da classe `Gene expression comparison`, utilizada para ligar os nós representando os genes regulados positivamente e regulados negativamente produzidos pelas regras `upRegulatedKidney` e `downRegulatedKidney`. Portanto, a regra `CNSOverKidney` possui no corpo de regra três declarações. As duas primeiras declarações (linhas 7 e 8) utilizam o predicado `identified gene` para encadear as regras de transformação, `upRegulatedKidney` e `downRegulatedKidney`. A terceira declaração (linha 9) utiliza o predicado `label` para armazenar uma string representando o nome da análise (`CNSOverKidney`).

A regra `upRegulatedKidney` produz instâncias da classe `Gene`, as quais são utilizadas como sujeitos das triplas criadas durante o processamento do corpo de regra. No entanto, a criação dessas instâncias é vinculada ao elemento de condição `ConditionUpRegulatedCNSOverKidney` (linha 11). Assim, somente os dados que atendem todos os critérios definidos nesse elemento de condição são utilizados para a criação das instâncias da classe `Gene`. Já o corpo de regra possui três declarações que utilizam os predicados `label` (linha 12), para representar o nome do gene, `fdr value` (linha 13), para representar o valor FDR do gene e, `log 2 ratio value` (linha 14), para representar o valor de expressão do gene.

O elemento de condição `ConditionUpRegulatedCNSOverKidney` possui duas instruções lógicas para assegurar que o gene sendo transformado seja considerado regulado positivamente. A primeira instrução lógica assegura que o valor `FDR` é menor ou igual (<=) a 0.05 (linha 22), enquanto que a segunda instrução lógica assegura que o valor de `log2 Ratio` seja maior ou igual (>=) a 1 (linha 23). Já o elemento de condição `ConditionDownRegulatedCNSOverKidney` também possui duas instruções lógicas, porém, desta vez, para assegurar que o gene sendo transformado seja considerado regulado ne-

```
1  PREFIX onto: <http://example.org/onto/exp5412#>
2  PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
3  PREFIX uniprot: <http://purl.uniprot.org/core/>
4
5  SELECT DISTINCT ?genes_iri (?genes AS ?upRegulated) WHERE {
6
7    ?comparison rdfs:label "CNSOverKidney" .
8    ?comparison onto:identified_gene ?genes_iri .
9
10   ?genes_iri a uniprot:Gene.
11   ?genes_iri rdfs:label ?genes .
12   ?genes_iri onto:log2ratio_value ?log2 .
13
14   FILTER (?log2 >= 1) .
15 }
```

Listagem 5 – Consulta SPARQL para a questão de competência 06.

gativamente. A primeira instrução lógica assegura que o valor `FDR` seja menor ou igual (<=) a 0.05 (linha 26), enquanto que a segunda instrução lógica assegura que o valor de `log2 Ratio` seja menor ou igual (<=) a -1 (linha 27). Desta forma, somente os genes diferencialmente expressos são transformados em nós RDF e estarão presentes no conjunto DAL para posterior exploração.

5.4.2.2 Transformação e exploração dos dados

Após a especificação das regras de transformação, executamos a transformação dos dados propriamente dita. Para isso, submetemos os arquivos SSD (*CNSOverKidney.txt*, *CNSOverLiver.txt* e *CNSOverLung.txt*), a ontologia de suporte, a especificação de regras de transformação e iniciamos o processo de transformação, o qual resultou em um conjunto DAL contendo 49.536 triplas.

Posteriormente, foi iniciada a etapa de exploração de dados. Definimos e executamos um conjunto de consultas SPARQL a fim de obter as respostas para as questões de competência relacionadas. Assim como no primeiro ciclo de transformação, a definição das consultas SPARQL para o segundo ciclo de transformação foi direta, pois envolvia a simples recuperação de informações do conjunto DAL. A Listagem 5 apresenta a consulta SPARQL para a obtenção da resposta para a questão *"Quais genes são regulados positivamente na amostra SNC versus fígado?"* (QC 06).

Primeiramente definidos os *namespaces* das ontologias a serem usados na consulta (linhas 1–3). Posteriormente, definimos que o retorno da consulta deveria conter os IRIs dos genes (variável `genes_iri`) e o nome do respectivo gene (variável `genes`) (linha 5). Em seguida, recuperamos do conjunto DAL o nó RDF que representa a análise diferencial `CNSOverKidney` (linha 7) e recuperamos os genes ligados ao nó RDF da análise diferencial (linha 8). Posteriormente, garantimos que os genes são de fato instâncias da classe `uniprot:Gene` (linha 10) e recuperamos seus respectivos `labels` (linha 11) e valores de log2ratio (linha 12). Finalmente, filtramos os genes de modo a retornar apenas os que possuem o valor de log2ratio maior ou igual a 1 (linha 14).

As consultas SPARQL associadas às questões de competência 07 a 11 são bastante semelhantes à consulta apresentada na Listagem 5. As principais diferenças entre as consultas estão na análise diferencial filtrada (linha 7) e no valor de log2ratio filtrado (linha 14).

A Tabela 6 apresenta um trecho do resultado da consulta SPARQL definida para responder a questão de competência 06. O trecho apresentado pela Tabela 6 possui uma coluna contendo o IRI do nó que representa um dado gene regulado positivamente (`genes_iri`) e uma coluna contendo o respectivo nome do gene (`upRegulated`). Todos os IRIs são compostos pelo IRI Base definido no elemento de configuração, seguido do identificador único do gene extraído do SSD, seguido do nome da análise diferencial. O IRI comum entre cada nó foi apresentado por meio do *namespace* `insta` (de *instance*) por motivo de concisão.

5.4.3 Ciclo de transformação 03

5.4.3.1 Especificação das regras de transformação

A especificação das regras de transformação para o terceiro ciclo de transformação foi mais desafiadora do que a especificação das regras dos ciclos anteriores, dado que envolveu a recuperação remota de conjuntos de dados. Assim, a especificação dessas transformações exigiu primeiramente o estudo da estrutura geral desses conjuntos de dados remotos para que fosse possível recuperar os nós RDFs desejados por meio de consultas SPARQL. Além disso, uma vez que não estávamos interessados em identificar separadamente genes

Tabela 6 – Resultado da consulta SPARQL para a questão de competência 06.
insta: http://example.org/onto/individual/

Genes_IRI	Up Regulated
<insta:ENSMUSG00000036545_CNSOverKidney>	Adamts2
<insta:ENSMUSG00000050271_CNSOverKidney>	D8Ertd82e
<insta:ENSMUSG00000041420_CNSOverKidney>	Meis3
<insta:ENSMUSG00000034640_CNSOverKidney>	Tiparp
<insta:ENSMUSG00000023960_CNSOverKidney>	Enpp5
<insta:ENSMUSG00000030731_CNSOverKidney>	Syt3
<insta:ENSMUSG00000025701_CNSOverKidney>	Alox5
<insta:ENSMUSG00000034858_CNSOverKidney>	Fam214a
<insta:ENSMUSG00000021477_CNSOverKidney>	Ctsl
<insta:ENSMUSG00000038806_CNSOverKidney>	Sde2
<insta:ENSMUSG00000054612_CNSOverKidney>	Mgmt
<insta:ENSMUSG00000072812_CNSOverKidney>	Ahnak2
<insta:ENSMUSG00000045268_CNSOverKidney>	Zfp691
<insta:ENSMUSG00000068874_CNSOverKidney>	Selenbp1
<insta:ENSMUSG00000033917_CNSOverKidney>	Gde1

regulados negativamente e genes regulados positivamente nesses conjuntos de dados, mas apenas as vias metabólicas que o conjunto comum de genes diferencialmente expressos estavam envolvidos, usamos diretamente o arquivo de dados contendo os resultados da interseção entre as três comparações diferentes (*CNSOverKidney.txt*, *CNSOverLiver.txt* e *CNSOverLung.txt*) como conjunto SSD de origem para a transformação. A Listagem 6 apresenta a regra de transformação especificada para este ciclo de transformação.

Para a recuperação de vias metabólicas do conjunto de dados externos foi necessário especificar um elemento de pesquisa (linhas 13–28). Nesse elemento definimos um identificador único (`searchPathway`) e o *endpoint* a ser utilizado na pesquisa das vias metabólicas de interesse (linha 13). Posteriormente, definimos um conjunto de *namespaces* (linhas 14–16) para facilitar a legibilidade da consulta e definimos a informação que deveria ser retornada como resposta da consulta (`pathway`, linha 17).

Em seguida pesquisamos todos os nós RDF que: i) fossem instâncias da classe `GeneProduct` (linha 18); ii) tivessem um nome associado por meio do predicado `label` (linha 19); e iii) tivessem nós RDF associados por meio do predicado `isPartOf` (linha 20). As instâncias da classe `GeneProduct` foram vinculadas à variável `geneProduct`. Os nomes associados por meio do predicado `label` foram vinculados à variável `label_raw`. Os nós RDF associados por meio do predicado `isPartOf` foram associados à variável `pathway`. Adicionalmente, garantimos que cada nó RDF da variável `pathway` fosse instância da classe `Pathway` (linha 21).

```
1  config_element {
2    "default_baseIRI" = "http://example.org/onto/individual/",
3    "export_syntax" = "N-Triples",
4    "namespace" = "ro" refers_to "http://purl.obolibrary.org/obo/RO",
5    "namespace" = "uniprot" refers_to "http://purl.uniprot.org/core/"
6  }
7
8  row_based_rule differentiallyExpressedGene ["gene_id.Set1" is_equivalent_to "uniprot:Gene
       "]{
9    links_to "gene_name.Set1" using "label",
10   links_to "gene_name.Set1" /SE(searchPathway, ?pathway) /NODE("Pathway") using "ro:
       involved in"
11 }
12
13 search_element searchPathway["http://sparql.wikipathways.org/"]{$
14   PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
15   PREFIX wp: <http://vocabularies.wikipathways.org/wp#>
16   PREFIX dcterms: <http://purl.org/dc/terms/>
17   SELECT DISTINCT ?pathway WHERE {
18     ?geneProduct a wp:GeneProduct .
19     ?geneProduct rdfs:label ?label_raw .
20     ?geneProduct dcterms:isPartOf ?pathway .
21     ?pathway a wp:Pathway .
22
23     FILTER REGEX(UCASE(STR(?label_raw)), UCASE(?tsvData)).
24
25     ?pathway wp:organismName ?organism .
26     FILTER REGEX(UCASE(STR(?organism )), UCASE("mus musculus")).
27   }
28 $}
```

Listagem 6 – Regras de transformação para o ciclo de transformação 03.

Posteriormente, filtramos a lista de nomes de genes (variável `label_raw`) pelo valor da variável `tsvData` (linha 23). O valor da variável `tsvData` consiste no item de dados da coluna SSD definida na declaração de regra. Por fim, recuperamos e filtramos os nomes dos organismos de cada via metabólica pelo nome "mus musculus" (linhas 25 e 26). O nome "mus musculus" é utilizado para que não sejam retornadas vias metabólicas pertencentes a outros organismos.

Especificamos a regra de transformação `differentiallyExpressedGene` (linhas 8–11) a fim de criar, para cada gene extraído da coluna SSD associada (`gene_id.Set1`), um nó RDF (como instância da classe `Gene`). Cada um desses nós RDF é então utilizado

como sujeito das triplas do corpo de regra. O corpo de regra possui duas declarações: a primeira utiliza o predicado `label` (linha 9) para armazenar o nome do gene, enquanto a segunda declaração utiliza o predicado `involved in` (linha 10) para ligar o sujeito a um conjunto de nós RDF de vias metabólicas que este gene está envolvido.

O conjunto de nós RDF de vias metabólicas é obtido em duas etapas: i) a busca de uma lista de identificadores únicos das vias metabólicas por meio do elemento de pesquisa `searchPathway`, o qual utiliza como parâmetro de pesquisa o item de dado presente na coluna SSD `gene_name.Set1`; e ii) a criação de um nó RDF (como instância da classe `Pathway`) para cada item da lista de identificadores únicos.

5.4.3.2 Transformação e exploração dos dados

Após a especificação das regras de transformação, executamos a transformação dos dados propriamente dita. Para isso, submetemos o arquivo SSD de interseção da análise diferencial, a ontologia de suporte, a especificação de regras de transformação e iniciamos o processo de transformação, o qual resultou em um conjunto DAL contendo 5.782 triplas.

Posteriormente, foi realizada a etapa de exploração de dados. A definição da consulta SPARQL para o ciclo de transformação 03 foi mais desafiadora, pois envolveu a recuperação de informações do conjunto DAL produzido pela atividade de transformação e também o acesso a um *endpoint* SPARQL remoto.

A fim de responder a questão de competência 12, foi preciso obter a razão entre o número de genes diferencialmente expressos que estavam envolvidos em uma dada via metabólica e o número total conhecido (existente) de genes que estão envolvidos nessa mesma via metabólica. Obtivemos o número de genes diferencialmente expressos envolvidos em uma determinada via, consultando o conjunto DAL produzido pela transformação de dados, enquanto que obtivemos o número total (conhecido) de genes envolvidos na via metabólica, consultando o *endpoint* SPARQL do portal WikiPathways.

A Listagem 7 apresenta a consulta SPARQL utilizada na obtenção da resposta para a questão *"Quais são as principais vias metabólicas associadas a genes diferencialmente expressos das amostras de SNC versus pulmão, fígado e rim?"* (QC 12).

Desta forma, essa consulta pode ser dividida em quatro partes. A primeira parte (linhas 1–7) define vários *namespaces*. A segunda parte (linhas 9–13) retorna

```
1  PREFIX onto:  <http://example.org/onto/exp5412#>
2  PREFIX obo:   <http://purl.obolibrary.org/obo/>
3  PREFIX uniprot:    <http://purl.uniprot.org/core/>
4  PREFIX rdfs:    <http://www.w3.org/2000/01/rdf-schema#>
5  PREFIX dc:    <http://purl.org/dc/terms/>
6  PREFIX dcterms: <http://purl.org/dc/elements/1.1/>
7  PREFIX wp:    <http://vocabularies.wikipathways.org/wp#>
8
9  SELECT DISTINCT ?pathway_label
10         ?pathway
11         ?genes_count
12         (count(?geneProduct) AS ?genes_known)
13         ((?genes_count / ?genes_known) AS ?ratio) WHERE {
14
15   {
16     SELECT DISTINCT ?pathway (count(?gene_label) AS ?genes_count) WHERE {
17       ?genes_iri a uniprot:Gene.
18       ?genes_iri rdfs:label ?gene_label .
19
20       ?genes_iri obo:RO_0002331 ?pathway
21     } GROUP BY ?pathway ORDER BY DESC(?genes_count)
22   }
23
24   SERVICE <http://sparql.wikipathways.org/> {
25     ?geneProduct a wp:GeneProduct .
26     ?geneProduct dc:isPartOf ?pathway .
27
28     ?pathway dcterms:title ?pathway_label_with_lang .
29     BIND(STR(?pathway_label_with_lang) AS ?pathway_label)
30   }
31 } GROUP BY ?pathway ?genes_count ?pathway_label
32 ORDER BY DESC(?ratio)
```

Listagem 7 – Consulta SPARQL para a questão de competência 12.

as vias metabólicas, a quantidade de genes do conjunto DAL que estão envolvidos na via metabólica, a quantidade de genes conhecidos que estão envolvidos na respectiva via metabólica e, finalmente, a razão entre as duas quantidades de genes. A terceira parte (linhas 15–22) visa recuperar os nós RDF das vias metabólicas em que os genes diferencialmente expressos estão envolvidos (`involved in` representado na ontologia por `RO_0002331`) e a quantidade de genes envolvidos em cada via metabólica. Finalmente, a quarta parte (linhas 24–30) visa recuperar remotamente os genes que estão envolvidos em uma determinada via metabólica.

A Tabela 7 apresenta as dez (10) vias metabólicas mais relevantes como parte do resultado da consulta SPARQL utilizada para responder a questão de competência 12. Na tabela são apresentadas cinco colunas, a saber: `pathway_label`, contendo os nomes das vias metabólicas que um ou mais genes diferencialmente expressos estão envolvidos; `pathway`, contendo o IRI do nó RDF que representa a via metabólica no *endpoint* SPARQL do portal WikiPathways; `genes_count`, contendo a quantidade de genes diferencialmente expressos dos SSD que estão envolvidos na via metabólica; `genes_known`, contendo a quantidade total de genes conhecidos que estão envolvidos na via metabólica e, finalmente, `ratio`, contendo a razão entre os genes identificados no SSD e a quantidade total de genes conhecidos que estão envolvidos na via metabólica.

Tabela 7 – Vias metabólicas associadas aos genes diferencialmente expressos. Wikipathways: http://identifiers.org/wikipathways/

Pathway Label	Pathway	Genes Count	Known Genes	Ratio
Eicosanoid Lipid Synthesis Map	<Wikipathways:WP4335_r98939>	5	6	0.833
Macrophage markers	<Wikipathways:WP2271_r69962>	6	10	0.600
ApoE and miR-146 in inflammation and atherosclerosis	<Wikipathways:WP3592_r94329>	5	9	0.556
Nucleotide GPCRs	<Wikipathways:WP207_r69172>	6	11	0.545
Eicosanoid Synthesis	<Wikipathways:WP318_r89525>	10	19	0.526
Circulating monocytes and cardiac macrophages in diastolic dysfunction	<Wikipathways:WP4474_r102094>	2	4	0.500
Statin Pathway	<Wikipathways:WP1_r103264>	9	20	0.450
Osteoclast	<Wikipathways:WP454_r81173>	4	9	0.444
Eicosanoid metabolism via Cyclo Oxygenases (COX)	<Wikipathways:WP4347_r102928>	10	23	0.435
Eicosanoid metabolism via Lipo Oxygenases (LOX)	<Wikipathways:WP4348_r102924>	9	22	0.409
Cholesterol Biosynthesis	<Wikipathways:WP103_r101818>	6	15	0.400
Triacylglyceride Synthesis	<Wikipathways:WP386_r77423>	9	23	0.391
Cholesterol metabolism (includes both Bloch and Kandutsch-Russell pathways)	<Wikipathways:WP4346_r102711>	16	42	0.381
Ptf1a related regulatory pathway	<Wikipathways:WP201_r69113>	4	11	0.364
Selenium Micronutrient Network	<Wikipathways:WP1272_r95973>	9	25	0.360

5.4.4 Ciclo de transformação 04

5.4.4.1 Especificação das regras de transformação

Assim como no ciclo de transformação anterior, a especificação das regras de transformação para o quarto ciclo de transformação também foi desafiadora porque novamente envolveu a recuperação de conjuntos de dados remotos. A Listagem 8 apresenta as regras de transformação especificadas para este ciclo de transformação.

Para a recuperação dos processos biológicos do conjunto de dados externos especifi-

```
1  config_element {
2    "default_baseIRI" = "http://example.org/onto/individual/",
3    "export_syntax" = "N-Triples",
4    "namespace" = "ro" refers_to "http://purl.obolibrary.org/obo/RO",
5    "namespace" = "uniprot" refers_to "http://purl.uniprot.org/core/",
6    "namespace" = "go" refers_to "http://purl.obolibrary.org/obo/GO",
7  }
8
9  row_based_rule differentiallyExpressedGene ["gene_id.Set1" is_equivalent_to "uniprot:Gene
       "]{
10   links_to "GO BP.Set1" /SP("; ") /NODE("go:biological_process") /SE(searchBP, ?bpIRI)
       using "ro:participates in"
11 }
12
13 search_element searchBP["http://rdf.geneontology.org/blazegraph/namespace/kb/sparql"]{$
14   PREFIX obo: <http://www.geneontology.org/formats/oboInOwl#>
15   SELECT DISTINCT ?bpIRI WHERE{
16     ?bpIRI obo:id ?id .
17     BIND(str(?id) as ?idString).
18     VALUES ?idString { ?tsvData }
19   }
20 $}
```

Listagem 8 – Regras de transformação para o ciclo de transformação 04.

camos um elemento de pesquisa (linhas 13–20). Nesse elemento definimos um identificador único (`searchBP`) e o *endpoint* a ser utilizado na pesquisa dos processos biológicos de interesse (linha 13). Posteriormente, definimos um *namespace* (linha 14) para facilitar a legibilidade da consulta e também qual informação deveria ser retornada como resposta da consulta (`bp_IRI`, linha 15). Em seguida, pesquisamos os identificadores dos nós RDF e atribuímos esses identificadores à variável `id` (linha 16). Posteriormente, transformamos os valores contidos na variável `id` em strings e atribuímos esses valores à variável `idString` (linha 17). Essa transformação para strings é necessária para que esses identificadores possam ser comparados com outra string, contida na variável `tsvData` (linha 18), que, consequentemente, descarta qualquer valor diferente do valor comparado. O valor da variável `tsvData` consiste no item de dado da coluna SSD definida na declaração de regra.

Especificamos a regra de transformação `differentiallyExpressedGene` (linhas 9–11) a fim de criar, para cada gene extraído da coluna SSD associada (`gene_id.Set1`), um nó RDF (instância da classe `Gene`) ligado a uma lista de processos biológicos em que este gene participa. A lista de processos biológicos é extraída da coluna SSD associada

(`GO BP.Set1`) e usada como entrada para pesquisar remotamente o identificador único do processo biológico correspondente. Tal pesquisa é realizada por meio do elemento de pesquisa `searchBP`. Para cada identificador resultante, criamos um nó RDF correspondente (instância da classe `biological_process`) e usamos o predicado `participates in` para vincular ambos os nós.

5.4.4.2 Transformação e exploração dos dados

Após a especificação das regras de transformação, foi executada a transformação dos dados propriamente dita. Para isso, submetemos o arquivo SSD de interseção da análise diferencial, a ontologia de suporte, a especificação de regras de transformação e iniciamos o processo de transformação, o qual gerou um conjunto DAL contendo 21.523 triplas.

Posteriormente, realizamos a etapa de exploração de dados. A definição da consulta SPARQL para este ciclo de transformação envolveu a recuperação de informações do conjunto DAL produzido pela atividade de transformação correspondente e também de um *endpoint* SPARQL remoto. A Listagem 9 apresenta a consulta SPARQL para a obtenção da resposta para a questão de competência *"Quais são os principais processos biológicos associados a genes diferencialmente expressos das amostras de SNC versus pulmão, fígado e rim?"* (QC 13).

Seguimos uma abordagem análoga à abordagem utilizada para responder a questão de competência 12. Para cada processo biológico, precisávamos obter a razão entre o número de genes diferencialmente expressos que participavam desse processo e o número total de genes do organismo `mus musculus` que direta e indiretamente participavam desse mesmo processo. Um gene participa diretamente de um processo biológico caso o nó RDF do gene esteja ligado diretamente ao nó de um dado processo biológico. Um gene participa indiretamente de um processo biológico caso participe de um de seus subprocessos ou participe de um processo que seja parte de um de seus (sub)processos. Obtivemos o número de genes diferencialmente expressos que participam de um determinado processo biológico por meio da consulta ao conjunto DAL produzido pela transformação de dados, enquanto que obtivemos o número total de genes que participam do processo biológico por meio da consulta ao *endpoint* Uniprot SPARQL.

Desta forma, essa consulta pode ser dividida em quatro partes. A primeira parte

```
1 PREFIX obo:<http://purl.obolibrary.org/obo/>
2 PREFIX rdfs:<http://www.w3.org/2000/01/rdf-schema#>
3 PREFIX uniprot:<http://purl.uniprot.org/core/>
4 PREFIX up:<http://purl.uniprot.org/core/>
5 PREFIX owl:<http://www.w3.org/2002/07/owl#>
6
7 SELECT DISTINCT ?biological_process ?bp_iri ?expr_genes (COUNT(distinct ?protein)
8   AS ?total_genes) ((?expr_genes / ?total_genes) AS ?ratio) WHERE {
9
10  { SELECT DISTINCT ?bp_iri (COUNT(?bp_iri) AS ?expr_genes) WHERE {
11      ?genes_iri a uniprot:Gene.
12      ?genes_iri obo:RO_0000056 ?bp_iri.
13    } GROUP BY ?bp_iri ORDER BY DESC(?expr_genes)
14  }
15
16  SERVICE SILENT <https://sparql.uniprot.org/sparql/> {
17    ?bp_iri rdfs:label ?biological_process.
18    ?sub_bp (rdfs:subClassOf|owl:someValuesFrom)* ?bp_iri.
19    ?protein up:classifiedWith ?sub_bp.
20    ?protein up:organism <http://purl.uniprot.org/taxonomy/10090>.
21  }
22 } GROUP BY ?biological_process ?bp_iri ?expr_genes ?total_genes
23 ORDER BY DESC(?ratio)
```

Listagem 9 – Consulta SPARQL para a questão de competência 13.

(linhas 1–5) define vários *namespaces* para facilitar a legibilidade da consulta. A segunda parte (linhas 7 e 8) retorna os nomes dos processos biológicos, seus respectivos identificadores únicos, a quantidade total de genes diferencialmente expressos que participam do processo biológico no conjunto DAL, a quantidade de genes que (recursivamente) participam de um dado processo biológico e, finalmente, a razão entre as duas quantidades de genes. A terceira parte (linhas 10–14) recupera os nós RDF dos processos biológicos em que genes diferencialmente expressos participam (`participates in` identificado na ontologia por `RO_0000056`) e a quantidade de genes participantes dos respectivos processos biológicos. Finalmente, a quarta parte (linhas 16–21) recupera remotamente os genes do organismo mus musculus (representado por `http://purl.uniprot.org/taxonomy/10090`) que (recursivamente) participam de um determinado processo biológico.

A execução dessa consulta no *endpoint* remoto SPARQL revelou um problema de desempenho inesperado sempre que tentávamos recuperar os genes associados a um processo biológico de finalidade geral. Como pretendíamos identificar (recursivamente)

os genes que indiretamente participam de um dado processo biológico, sempre que uma consulta para identificar os genes agregados que participam de um processo biológico de propósito geral era realizada, tínhamos um erro de esgotamento de temporização devido ao longo tempo de processamento.

Resolvemos esse problema por meio da identificação e eliminação da pesquisa dos processos biológicos que estavam excedendo o tempo limite de uma consulta (não descrito na Listagem 9 por motivo de concisão). No entanto, essa eliminação não influencia em última análise o resultado dessa atividade de exploração de dados, pois qualquer processo biológico de finalidade geral que tenha um número alto de genes que participam direta ou indiretamente desse processo tem como consequência uma baixa relevância neste contexto.

A Tabela 8 apresenta os 10 principais processos biológicos (maior razão computada) identificados como resposta para a questão de competência 13. Como resultado da execução da consulta SPARQL definida para a questão de competência 13, obtivemos uma tabela contendo um total de 4.366 processos biológicos. Para selecionar os principais processos biológicos, filtramos esses registros eliminando-se qualquer processo biológico contendo um pequeno número de genes participantes diferencialmente expressos, assim como qualquer processo biológico contendo um pequeno número de genes participantes conhecidos totais. Como critério para a filtragem definimos o mínimo de 5 genes participantes diferencialmente expressos no experimento e mínimo de 25 genes participantes totais conhecidos.

Tabela 8 – Processos biológicos associados aos genes diferencialmente expressos. OBO: http://purl.obolibrary.org/obo/

Biological Process	Biological Process IRI	Expressed Genes	Total Genes	Ratio
Brown fat cell differentiation	<OBO:GO_0050873>	15	39	0.3846
Positive regulation of defense response to virus by host	<OBO:GO_0002230>	15	43	0.3488
Positive regulation of B cell proliferation	<OBO:GO_0030890>	17	53	0.3207
Peptidyl-tyrosine autophosphorylation	<OBO:GO_0038083>	12	39	0.3076
Bone resorption	<OBO:GO_0045453>	8	27	0.2962
Lymph node development	<OBO:GO_0048535>	9	32	0.2812
Negative regulation of peptidyl-serine phosphorylation	<OBO:GO_0033137>	9	33	0.2727
Positive regulation of vascular endothelial growth factor production	<OBO:GO_0010575>	7	27	0.2592
Hair follicle morphogenesis	<OBO:GO_0031069>	8	31	0.2580
Substrate adhesion-dependent cell spreading	<OBO:GO_0034446>	13	53	0.2452

5.5 Discussão

Neste capítulo demonstramos a aplicação da abordagem de transformação SSD2LOD Transformation Approach, assim como o potencial de descoberta de conhecimento a partir de um conjunto de dados transformado. Desta forma, descrevemos os principais aspectos da aplicação da abordagem de transformação em um experimento de análise diferencial disponibilizado no ArrayExpress. Por meio deste exercício definimos um conjunto de perguntas de interesse, cujas respostas deveriam ser obtidas ao final do processo de transformação. Na sequência, especificamos regras de transformação utilizando elementos condicionais e de busca de dados remotos, obtivemos o conjunto de dados transformado e, finalmente, exploramos este conjunto de dados para obter as respostas desejadas.

A aplicação da nossa abordagem de transformação facilitou a descoberta de conhecimento em duas situações específicas. No primeiro caso, a abordagem de transformação permitiu a identificação das principais vias metabólicas que os genes diferencialmente expressos estão envolvidos. Para tanto, recuperamos, para cada via metabólica identificada, a quantidade de genes diferencialmente expressos associados à via. Em seguida, buscamos no portal WikiPathways a quantidade total conhecida de genes envolvidos em cada via metabólica. Com base nessas duas informações, calculamos a razão entre as mesmas. Finalmente, ordenamos a lista de vias metabólicas em ordem decrescente de razão computada. A identificação das principais vias metabólicas não foi apresentada por Spath et al. (2017). Portanto, não pudemos comparar nossos resultados com os resultados obtidos no estudo de expressão gênica original.

No segundo caso, a abordagem de transformação permitiu a identificação dos principais processos biológicos que os genes diferencialmente expressos participam. Para tanto, obtivemos, para cada process biológico em que genes diferencialmente expressos participam, a quantidade de genes participantes. Posteriormente, buscamos no portal UniProt os genes que participam direta ou indiretamente de cada processo biológico identificado, obtendo dessa forma, a quantidade total de genes conhecidos que participam de cada processo biológico. Na sequência, calculamos a razão entre essas duas quantidades. Finalmente, ordenamos a lista de processos biológicos em ordem decrescente de razão computada.

Os dez processos biológicos mais relevantes (maior razão computada) que identificamos neste estudo foram então confrontados com as informações contidas em Spath et al. (2017), por meio da Figura S7-E. Essa figura foi gerada usando a ferramenta ReViGo (SUPEK et al., 2011), a partir dos dados de enriquecimento funcional (não disponíveis no artigo). Essa figura apresenta, em um formato *TreeMap*, os processos biológicos agrupados por grau de similaridade associados aos genes diferencialmente expressos. Para confrontar nossos resultados com os dados contidos na figura, buscamos verificar se os processos biológicos identificados em nossa abordagem estavam presentes diretamente ou indiretamente na figura.

Processos biológicos presentes diretamente são aqueles que estavam destacados explicitamente na figura. Processos biológicos presentes indiretamente são especializações de um processo destacado explicitamente na figura ou podem ser relacionados de alguma forma (*part_of*, *regulates*) a um destes processos biológicos ou a especializações destes. A identificação dos processos biológicos presentes diretamente foi simples. Já a identificação dos processos biológicos presentes indiretamente envolveu a pesquisa manual no portal QuickGO[8] para identificar a estrutura de cada (sub)processo biológico. Desta forma, considerando-se os dez (10) processos biológicos mais relevantes apresentados na Tabela 8, conseguimos identificar a presença direta ou indiretamente de oito (8) desses processos em Spath et al. (2017).

Esses resultados demonstram a capacidade de descoberta de conhecimento apenas explorando o conjunto DAL, sem a necessidade da execução de análises adicionais de bioinformática que muitas vezes são realizadas com recursos proprietários. Desta forma, esta prova de conceito demonstrou que a abordagem SSD2LOD Transformation Approach é capaz de realizar a transformação dos dados, agregar valor semântico aos mesmos e, como consequência, permitir a descoberta de novos conhecimentos.

[8] https://www.ebi.ac.uk/QuickGO/

6

Conclusão

Este trabalho teve por objetivo propor uma abordagem de geração de DAL a partir de dados em formato texto semiestruturado de bioinformática e o desenvolvimento de ferramentas de suporte a esta abordagem. Este capítulo apresenta as principais contribuições e limitações do trabalho, além de realizar uma comparação com abordagens de transformação relacionadas.

O capítulo está estruturado da seguinte forma: a seção 6.1 apresenta as principais contribuições do trabalho; a seção 6.2 discute a abordagem SSD2LOD Transformation Approach, comparando-a com trabalhos relacionados; por fim, a seção 6.3 apresenta os trabalhos futuros.

6.1 Principais contribuições

As principais contribuições deste trabalho estão relacionadas com o desenvolvimento de uma abordagem de transformação de dados semiestruturados de bioinformática em DAL e um conjunto de ferramentas de suporte a esta abordagem. Esta abordagem consiste de quatro atividades: i) definição de Questões de Competências (QC); ii) especificação de regras de transformação; iii) transformação de dados; e iv) exploração dos dados transformados.

A primeira atividade visa a elaboração de um conjunto de perguntas para as quais desejamos ter respostas ao final do processo de transformação. Essas questões são importantes para guiar o desenvolvimento da especificação das regras de transformação e posteriormente a exploração dos dados transformados.

A segunda atividade visa a especificação de regras de transformação usando a linguagem SSD2LOD Transformation Language associadas às questões de competência identificadas. Essa linguagem é utilizada para vincular itens de dados a termos de uma ontologia e possui um conjunto de recursos que facilitam esta atividade e a posterior transformação dos dados. Dentre os principais recursos destacam-se o suporte à especificação de manipulação de dados para facilitar a quebra e subsequente transformação, a especificação de condicionais para permitir a execução de uma transformação, e finalmente, a especificação de busca de nós RDF em conjuntos de dados remotos.

A terceira atividade consiste na execução da transformação dos dados propriamente dita, considerando-se o conjunto SSD de entrada, a especificação das regras de transformação e a(s) ontologia(s) utilizada(s). Já a quarta atividade consiste na exploração dos dados transformados. Nesta atividade, o conjunto DAL resultante é utilizado para responder as questões de competência elaboradas na primeira atividade. Para apoiar as atividades de transformação e exploração dos dados desenvolvemos um conjunto de três ferramentas de suporte: SSD2LOD Transformation Tool, SSD2LOD Web Service e SSD2LOD Web.

SSD2LOD Transformation Tool consiste em uma ferramenta de linha de comando que permite a entrada do(s) arquivo(s) SSD de origem, da(s) ontologia(s) de origem e das regras de transformação. Esses arquivos são então utilizados para executar a transformação dos dados e retornar o conjunto DAL resultante da transformação. SSD2LOD Web Service permite o acesso remoto ao SSD2LOD Transformation Tool por meio de requisições HTTP, facilitando sua integração com outras ferramentas para a execução e monitoramento de uma transformação, assim como para a recuperação do conjunto DAL resultante da transformação. Finalmente, SSD2LOD Web consiste em um sistema web que consome as operações do SSD2LOD Web Service para fornecer uma interface web de acesso à transformação. Portanto, essa ferramenta permite o uso da ferramenta de transformação de forma simples e acessível para qualquer tipo de usuário.

A abordagem SSD2LOD Transformation Approach e seu conjunto de ferramentas de suporte representam uma solução simples, sistemática e flexível para a transformação dos dados de bioinformática em DAL. Tal solução permite que dados semiestruturados sejam transformados em DAL, fazendo uso, sempre que pertinente, de informações já existentes em outros conjuntos DAL externos. Já para a área de web semântica de maneira geral, tal solução facilita a criação de novos conjuntos de dados abertos ligados, o que

pode futuramente resultar em um aumento da quantidade de dados ligados publicamente disponíveis, consequentemente facilitando ainda mais a descoberta de novos conhecimentos.

6.2 Discussão

Neste trabalho desenvolvemos uma abordagem sistemática para a transformação de dados semiestruturados de bioinformática em DAL e um conjunto de ferramentas de suporte associadas. Adicionalmente, ilustramos a abordagem por meio do desenvolvimento de uma prova de conceito envolvendo a transformação em DAL de um conjunto de dados processados, oriundos de um estudo de genômica funcional publicado na literatura.

A prova de conceito desenvolvida permitiu a transformação de um conjunto de dados processados de genômica funcional em um conjunto DAL resultante com posterior descoberta de conhecimento. Nesta prova de conceito conseguimos identificar um conjunto de vias metabólicas e processos biológicos relevantes, resultado este que é particularmente significativo uma vez que, em geral, a identificação de vias metabólicas e processos biológicos de interesse na genômica funcional é normalmente obtida por meio de diferentes atividades de análise, como, por exemplo, a análise de enriquecimento funcional e a análise de vias metabólicas, frequentemente realizadas com o auxílio de ferramentas (recursos) proprietárias.

Diferentes abordagens e ferramentas de apoio associadas têm sido propostas na literatura para a transformação de dados biomédicos semiestruturados em DAL. No entanto, essas abordagens geralmente não possuem a simplicidade, a sistematicidade e a flexibilidade para serem usadas no domínio da bioinformática. Por exemplo, a abordagem desenvolvida por Belleau et al. (2008) permite a integração baseada em RDF de múltiplas fontes de dados de bioinformática, tanto estruturados quanto semiestruturados. Apesar do aparente amplo suporte para a transformação de diferentes tipos de dados, essa abordagem tem duas limitações principais. Primeiro, a abordagem depende de uma única (e fixa) ontologia para integrar as diferentes fontes de dados. Em segundo lugar, a abordagem requer a criação de um software personalizado para cada fonte de dados a ser transformada. Assim, as regras de transformação são codificadas neste software. Por sua vez, a abordagem SSD2LOD Transformation Approach suporta tanto o uso de diferentes ontologias quanto o uso da linguagem SSD2LOD Transformation Language para a especificação de regras de

transformação sem a necessidade de desenvolvimento de softwares específicos.

A plataforma *eXframe* desenvolvida por Merrill et al. (2014) é direcionada para a transformação de experimentos biomédicos usando mapeamentos pré-definidos de campos de formulário para classes de uma ontologia de experimentos biológicos. Como esta plataforma foi desenvolvida para resolver um problema específico, ela não pode ser facilmente adaptada a outros propósitos. Em contraste, a abordagem SSD2LOD Transformation Approach, de propósito geral, pode ser usada em diferentes cenários.

A ferramenta *InstanceLoaderDB* desenvolvida por Kaalia e Ghosh (2016) suporta a transformação de conjuntos de dados SSD relacionados a diabetes em instâncias de classes de uma ontologia específica da doença. No entanto, essa ferramenta não suporta a definição explícita de mapeamentos entre itens de dados e instâncias de classes de uma ontologia usando uma linguagem de especificação de regra de transformação. A abordagem SSD2LOD Transformation Approach permite a especificação de regras de transformação, as quais são usadas como entrada para o processo de transformação.

A abordagem proposta por Jovanovik e Trajanov (2017) assemelha-se em muitos pontos à nossa própria abordagem. Contudo, as atividades da abordagem de transformação proposta pelos autores podem ser consideradas demasiadamente genéricas, deixando dúvidas quanto à aplicação concreta das mesmas. A abordagem SSD2LOD Transformation Approach apresenta atividades bem definidas, elaboradas para serem executadas sequencialmente de forma sistemática, auxiliando o usuário na obtenção dos conhecimentos desejados por meio de questões de competência. Adicionalmente, a abordagem proposta por Jovanovik e Trajanov utiliza a ferramenta de transformação OpenRefine (VERBORGH; WILDE, 2013). Esta ferramenta possui um mecanismo simples de busca remota de IRI, restrito à busca de uma dada propriedade. Tal característica limita, portanto, as possibilidades de recuperação de dados. Por outro lado, nossa ferramenta de suporte permite a execução de um *script* SPARQL completo, conferindo maior flexibilidade na busca por IRIs remotos.

A abordagem de Bernabé-Díaz et al. (2019) também pode ser considerada bastante semelhante a nossa abordagem de transformação e ferramentas associadas. A abordagem dos autores utiliza regras de transformação, permite a transformação de dados tabulares e permite a busca externa de IRIs. No entanto, essa abordagem possui limitações em

relação a nossa abordagem. A primeira limitação consiste no suporte apenas indireto para a transformação de dados semiestruturados em DAL. Segundo esta abordagem, dados semiestruturados são primeiramente transformados em um documento XML para posteriormente serem transformados em DAL, o que torna o processo de transformação mais complexo. Outra desvantagem consiste na utilização de apenas um arquivo de dados de entrada e de apenas uma ontologia de origem, ao contrário da nossa abordagem que permite a utilização de múltiplos arquivos de dados e múltiplas ontologias.

Sernadela, González-Castro e Oliveira (2017) desenvolveram a ferramenta `Scaleus` para prover suporte à transformação de arquivos tabulares e planilhas em RDF. Por um lado, como `Scaleus` permite apenas a especificação direta do IRI de um predicado (a importação de uma ontologia OWL não é suportada), a representação semântica das transformações de dados é mais complexa e propensa a erros. Por outro lado, a abordagem SSD2LOD Transformation Approach não apenas provê suporte à importação de classes e propriedades de ontologias OWL, mas também faz uso de uma linguagem que utiliza diretamente esses conceitos e relacionamentos para a especificação das transformações semânticas.

Apesar dos benefícios da SSD2LOD Transformation Approach, devemos reconhecer algumas limitações ao nosso estudo. Primeiro, nossa abordagem não foi aplicada na transformação de grandes conjuntos de dados. No entanto, como conjuntos extremamente grandes de dados processados de bioinformática são bastante incomuns, acreditamos que as ferramentas de suporte desenvolvidas podem ser utilizadas nos cenários de transformação de mais comuns desta área. Além disso, observamos durante a realidade de nossa prova de conceito que o gargalo da transformação está na integração com conjuntos de dados remotos durante a execução de uma consulta SPARQL. Enfrentar tal limitação está fora do escopo desta pesquisa.

Até onde sabemos, nenhuma abordagem abrangente foi definida para apoiar a transformação de dados semiestruturados de bioinformática em DAL. Acreditamos que a abordagem SSD2LOD Transformation Approach possa ser aplicada na transformação de dados semiestruturados em DAL não apenas no domínio da bioinformática, mas em qualquer outro domínio de conhecimento com requisitos de transformação semelhantes.

Desta forma, a abordagem SSD2LOD Transformation Approach e suas ferramentas

de suporte facilitam a transformação e a exploração de dados em bioinformática, permitindo que seus usuários utilizem informações semânticas para a descoberta de novos conhecimentos no domínio. A disponibilidade de soluções simples e eficazes para a transformação de diferentes tipos de dados em DAL favorecerá o uso de tecnologias semânticas, contribuindo assim para o desenvolvimento da web semântica como um todo.

6.3 Trabalhos futuros

A abordagem de transformação SSD2LOD Transformation Approach e ferramentas de suporte associadas foram concebidas para a transformação de dados semiestruturados de bioinformática em DAL. Desta forma, pretendemos inicialmente investigar a aplicação da abordagem de transformação e infraestrutura de suporte em outros domínios de conhecimento, possivelmente envolvendo conjuntos maiores de dados, como, por exemplo, na área governamental. Tal experiência nos permitirá identificar oportunidades de melhoria, tanto na abordagem de transformação quanto nas ferramentas de suporte, de modo a torná-las ainda mais genéricas.

Adicionalmente, pretendemos desenvolver um editor com suporte à sintaxe da liguagem de transformação. Com esse editor será possível implementar um mecanismo de *drag-n-drop* dos elementos da especificação de regras de transformação integrado a um assistente passo a passo de especificação para guiar na montagem da especificação. Adicionalmente, será possível também implementar funções de auxílio, como, por exemplo, autocompletar termos e símbolos para a alteração manual de uma regra de transformação.

Esse editor também poderá ser usado como base para o desenvolvimento de uma plataforma de transformação de SSD para DAL mais abrangente. Tal plataforma poderá conter recursos adicionais, como publicação de conjuntos DAL em um dado *endpoint* SPARQL, visualizações do conjunto DAL transformado e suporte aprimorado para a integração com conjuntos de dados remotos.

Referências

ABEYSINGHE, R. et al. Quality assurance of nci thesaurus by mining structural-lexical patterns. *AMIA Annual Symp Proc*, American Medical Informatics Association, v. 2017, p. 364–373, 2018. Disponível em: <https://www.ncbi.nlm.nih.gov/pubmed/29854100>.

ABITEBOUL, S. Querying Semi-Structured Data. In: *Proceedings of the 6th International Conference on Database Theory (ICDT'97)*. [S.l.]: Springer-Verlag, 1997. p. 1–18.

ABITEBOUL, S.; BUNEMAN, P.; SUCIU, D. *Data on the Web: From Relations to Semistructured Data and XML*. [S.l.: s.n.], 2000. 258 p. ISBN 1-55860-622-X.

Aduna Software. *The SeRQL query language*. 2017. Disponível em: <http://archive.rdf4j.org/users/ch11.html#section-introduction>.

ANDREAS, H.; KATJA, H.; RALF, S. *Linked Data Management*. CRC Press, 2014. Disponível em: <https://www.crcpress.com/Linked-Data-Management/Harth-Hose-Schenkel/p/book/9781466582408>.

ANTONIOU, G. et al. *A Semantic Web Primer*. [S.l.]: MIT Press, 2012. 270 p. ISBN 978-0-262-01828-9.

ANTONIOU, G.; HARMELEN, F. V. *A Semantic Web Primer*. [S.l.]: Massachusetts Institute of Technology, 2008. 264 p. ISBN 9780262012423.

Apache Software Foundation. *ARQ - A SPARQL Processor for Jena*. 2017. Disponível em: <https://jena.apache.org/documentation/query/>.

Apache Software Foundation. *TDB*. 2017. Disponível em: <https://jena.apache.org/documentation/tdb/index.html>.

ARP, R.; SMITH, B.; SPEAR, A. D. *Building ontologies with Basic Formal Ontology*. Cambridge, USA: MIT Press, 2015.

ASHBURNER, M. et al. Gene ontology: tool for the unification of biology. The Gene Ontology Consortium. *Nature Genetics*, v. 25, n. 1, p. 25–29, 2000.

AUER, S. et al. Triplify: Light-Weight Linked Data Publication from Relational Databases. *Proceedings of the 18th International Conference on World Wide Web, Madrid*, p. 621–630, 2009.

BANDROWSKI, A. et al. The Ontology for Biomedical Investigations. *PLOS ONE*, v. 11, n. 4, p. 1–19, 2016.

BARRETT, T. et al. NCBI GEO: mining tens of millions of expression profiles–database and tools update. *Nucleic Acids Research*, v. 35, p. D760–D765, 2007.

BELLEAU, F. et al. Bio2rdf: Towards a mashup to build bioinformatics knowledge systems. *Journal of Biomedical Informatics*, v. 41, n. 5, p. 706–716, 2008. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1532046408000415>.

BERNABÉ-DÍAZ, J. A. et al. Efficient, semantics-rich transformation and integration of large datasets. *Expert Systems with Applications*, v. 133, p. 198 – 214, 2019. ISSN 0957-4174.

BERNERS-LEE, T. W3 Future Directions. *Plenary talk at the First International World Wide Web Conference*, p. 1–4, 1994. Disponível em: <https://www.w3.org/Talks/WWW94Tim/>.

BERNERS-LEE, T. *Linked Data - Design Issues*. 2006. Disponível em: <https://www.w3.org/DesignIssues/LinkedData.html>.

BERNERS-LEE, T.; HENDLER, J.; LASSILA, O. The Semantic Web will enable machines to. *Scientific American*, v. 21, 2001.

BIZER, C. The emerging web of linked data. *IEEE Intelligent Systems*, 2009.

BIZER, C.; HEATH, T.; BERNERS-LEE, T. Linked data-the story so far. *International Journal on Semantic Web and Information Systems*, 2009. ISSN 1552-6283.

BOSAK, J.; BRAY, T. Xml and the second-generation web. *Scientific American - SCI AMER*, p. 89–93, 1999.

BRAZMA, A. et al. ArrayExpress—a public repository for microarray gene expression data at the EBI. *Nucleic Acids Research*, v. 31, n. 1, p. 68–71, 01 2003. ISSN 0305-1048.

BROWN, G. R. et al. Gene: a gene-centered information resource at ncbi. *Nucleic Acids Research*, v. 43, n. D1, p. D36–D42, 2015. Disponível em: <http://dx.doi.org/10.1093/nar/gku1055>.

CHRISTENSEN, H. *Learning Materials in Biosciences Introduction to Bioinformatics in Microbiology*. [S.l.: s.n.], 2018. 219 p. ISBN 978-3-319-99280-8.

CORLOSQUET, S. et al. Produce and consume linked data with drupal! In: BERNSTEIN, A. et al. (Ed.). *The Semantic Web - ISWC 2009*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009. p. 763–778. ISBN 978-3-642-04930-9.

CURÉ, O.; BLIN, G. *RDF Database Systems: Triples Storage and SPARQL Query Processing*. [S.l.: s.n.], 2014.

DACONTA, M. C.; OBRST, L. J.; SMITH, K. T. *The Semantic Web: A Guide to the Future of XML, Web Services, and Knowledge Management*. [S.l.]: Wiley Publishing, 2003. 281 p. ISBN 0-471-43257-1.

DECKER, S. et al. The semantic web: the roles of xml and rdf. *IEEE Internet Computing*, v. 4, n. 5, p. 63–73, Sep. 2000. ISSN 1089-7801.

DESSIMOZ, C.; ŠKUNCA, N. (Ed.). *The Gene Ontology Handbook*. [S.l.]: Springer New York, 2017.

DUCHARME, B. *Learning SPARQL*. [S.l.]: O'Reilly Media, 2013. 366 p. ISBN 978-1-449-37143-2.

DUMONTIER, M. et al. Bio2rdf release 3: A larger connected network of linked data for the life sciences. In: *Proceedings of the 2014 International Conference on Posters & Demonstrations Track - Volume 1272*. CEUR-WS.org, 2014. (ISWC-PD'14), p. 401–404. Disponível em: <http://dl.acm.org/citation.cfm?id=2878453.2878554>.

European Bioinformatics Institute. *European Bioinformatics Institute*. 2017. Disponível em: <http://www.ebi.ac.uk/>.

FALBO, R. D. A. SABiO: Systematic approach for building ontologies. *CEUR Workshop Proceedings*, 2014.

FARIAS, C. R. G. de. *Architectural Design of Groupware Systems: a Component-Based Approach*. Tese (Doutorado) — University of Twente, 2002.

FERNÁNDEZ-LÓPEZ, M.; GÓMEZ-PÉREZ, A.; JURISTO, N. Methontology: From ontological art towards ontological engineering. In: *Proceedings of the Ontological Engineering AAAI-97 Spring Symposium Series*. American Asociation for Artificial Intelligence, 1997. Ontology Engineering Group - OEG. Disponível em: <http://oa.upm.es/5484/>.

Gene Ontology. *GO-CAM Documentation*. 2019. Disponível em: <https://geneontology.cloud/docs>.

GENESERETH, M. R.; NILSSON, N. *Logical Foundations of Artificial Intelligence*. [S.l.: s.n.], 1987. 406 p. ISBN 978-0-934613-31-6.

GEROIMENK, V.; CHEN, C. *Visualizing the Semantic Web*. [S.l.]: Springer-Verlag London, 2003. ISBN 1-85233-976-4.

GRUBER, T. R. Toward principles for the design of ontologies used for knowledge sharing? *International Journal of Human-Computer Studies*, v. 43, n. 5-6, p. 907–928, 1995. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1071581985710816>.

GRÜNINGER, M.; FOX, M. S. Methodology for the design and evaluation of ontologies. In: . [s.n.], 1995. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.44.8723>.

GUARDIA, G. D. A. *Suporte ao desenvolvimento e à composição de serviços web semânticos para a análise de expressão gênica*. 429 p. Tese (Doutorado) — University of São Paulo, 2016. Disponível em: <http://www.teses.usp.br/teses/disponiveis/95/95131/tde-28102016-101702>.

HUANG, H. et al. i proclass: an integrated database of protein family, function and structure information. *Nucleic Acids Research*, v. 31, n. 1, p. 390–392, 2003. Disponível em: <http://dx.doi.org/10.1093/nar/gkg044>.

ISOTANI, S.; BITTENCOURT, I. I. *Dados Abertos Conectados*. [s.n.], 2015. 175 p. ISBN 978-85-7522-449-6. Disponível em: <http://ceweb.br/livros/dados-abertos-conectados/http://ceweb.br/publicacao/livro-dados-abertos/>.

JOVANOVIK, M.; TRAJANOV, D. Consolidating drug data on a global scale using Linked Data. *Journal of Biomedical Semantics*, Journal of Biomedical Semantics, v. 8, p. 1–24, 2017. ISSN 20411480.

JUPP, S. et al. The EBI RDF platform: Linked open data for the life sciences. *Bioinformatics*, v. 30, n. 9, p. 1338–1339, 2014. ISSN 14602059.

KAALIA, R.; GHOSH, I. Semantics based approach for analyzing disease-target associations. *Journal of Biomedical Informatics*, Elsevier Inc., v. 62, p. 125–135, 2016. ISSN 15320464.

LEE, D.; CHU, W. W. Comparative Analysis of Six XML Schema. *SIGMOD Rec.*, p. 1–24, 2000.

LEGAZ-GARCÍA, M. d. C. et al. Generation of open biomedical datasets through ontology-driven transformation and integration processes. *Journal of Biomedical Semantics*, v. 7, n. 1, p. 32, Jun 2016. ISSN 2041-1480. Disponível em: <https://doi.org/10.1186/s13326-016-0075-z>.

Linking Open Data Project. *Linking Open Data Project*. 2017. Disponível em: <https://www.w3.org/wiki/SweoIG/TaskForces/CommunityProjects/LinkingOpenData>.

MCCRAE, J. et al. *The Linking Open Data cloud diagram*. 2019. Disponível em: <http://lod-cloud.net/>.

MERRILL, E. et al. Semantic Web repositories for genomics data using the eXframe platform. *Journal of biomedical semantics*, v. 5, n. Suppl 1, p. S3, 2014. ISSN 2041-1480.

National Center for Biotechnology Information. *PubMed Help*. Bethesda (MD): [s.n.], 2016. 155 p. Disponível em: <https://www.ncbi.nlm.nih.gov/books/NBK3830/pdf/Bookshelf_NBK3830.pdf>.

National Center for Biotechnology Information. *Gene Expression Omnibus*. 2017. Disponível em: <https://www.ncbi.nlm.nih.gov/geo/>.

NICOLA, A. D.; MISSIKOFF, M.; NAVIGLI, R. A Proposal for a Unified Process for Ontology Building: UPON. *DEXA 2005: Proceedings of the 16th International Conference on Database and Expert Systems Applications*, 2005.

NOY, N. F.; MCGUINNESS, D. L. Ontology Development 101: A Guide to Creating Your First Ontology. *Stanford Knowledge Systems Laboratory*, 2001.

ORACLE. *JavaServer Pages Standard Tag Library*. 2017. Disponível em: <oracle.com/technetwork/java/jstl-137486.html>.

ORACLE. *JavaServer Pages Technology*. 2017. Disponível em: <http://www.oracle.com/technetwork/java/javaee/jsp/index.html>.

PEVSNER, J. *Bioinformatics and Functional Genomics*. 3. ed. [S.l.]: Wiley-Blackwell, 2015. ISBN 1118581784.

RAYNER, T. F. et al. A simple spreadsheet-based, MIAME-supportive format for microarray data: MAGE-TAB. *BMC Bioinformatics*, v. 7, n. 1, p. 489, 2006.

RIGDEN, D. J.; FERNANDEZ-SUAREZ, X. M.; GALPERIN, M. Y. The 2016 database issue of nucleic acids research and an updated molecular biology database collection. *Nucleic Acids Research*, v. 44, n. D1, p. D1–D6, 2016. Disponível em: <http://dx.doi.org/10.1093/nar/gkv1356>.

RUSU, O. et al. Converting unstructured and semi-structured data into knowledge. In: *2013 11th RoEduNet International Conference*. [S.l.]: IEEE, 2013. p. 1–4. ISBN 978-1-4673-6116-3. ISSN 20681038.

SAMBREKAR, K.; RAJPUROHIT, V. S.; JOSHI, J. A Proposed Technique for Conversion of Unstructured Agro-Data to Semi-Structured or Structured Data. In: *2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)*. IEEE, 2018. p. 1–5. ISBN 978-1-5386-5257-2. Disponível em: <https://ieeexplore.ieee.org/document/8697432/>.

Semantic Web Education and Outreach Interest Group. *Semantic Web Education and Outreach Interest Group*. 2017. Disponível em: <www.w3.org/wiki/SweoIG>.

SERNADELA, P.; GONZÁLEZ-CASTRO, L.; OLIVEIRA, J. L. SCALEUS: Semantic Web Services Integration for Biomedical Applications. *Journal of Medical Systems*, Journal of Medical Systems, v. 41, n. 4, 2017. ISSN 1573689X.

SIMPERL, E.; BU, Q.; LI, Y. Using microtasks to crowdsource DBpedia entity classification : A study in workflow design. *Semantic Web Journal*, 2015.

SINT, R. et al. Combining Unstructured, Fully Structured and Semi-Structured Information in Semantic Wikis. In: *CEUR Workshop Proceedings*. Heraklion: [s.n.], 2009. v. 464, p. 73–87.

SIOUTOS, N. et al. NCI Thesaurus: A semantic model integrating cancer-related clinical and molecular information. *Journal of Biomedical Informatics*, v. 40, n. 1, p. 30–43, 2007.

SLENTER, D. N. et al. Wikipathways: a multifaceted pathway database bridging metabolomics to other omics research. *Nucleic Acids Research*, v. 46, n. D1, p. D661–D667, 11 2017. ISSN 0305-1048.

SMITH, B. et al. Relations in biomedical ontologies. *Genome Biology*, v. 6, n. 5, p. R46.1–R46.15, 2005.

SPATH, S. et al. Dysregulation of the cytokine gm-csf induces spontaneous phagocyte invasion and immunopathology in the central nervous system. *Immunity*, v. 46, n. 2, p. 245–260, 2017.

SUPEK, F. et al. Revigo summarizes and visualizes long lists of gene ontology terms. *PLOS ONE*, Public Library of Science, v. 6, n. 7, p. 1–9, 07 2011. Disponível em: <https://doi.org/10.1371/journal.pone.0021800>.

The Internet Society. *RFC 3987 Internationalized Resource Identifiers (IRIs)*. 2005. Disponível em: <https://tools.ietf.org/html/rfc3987>.

The Internet Society. *Uniform Resource Identifier (URI): Generic Syntax*. 2005. Disponível em: <https://tools.ietf.org/pdf/rfc3986.pdf>.

THOMPSON, R. et al. Rd-connect: An integrated platform connecting databases, registries, biobanks and clinical bioinformatics for rare disease research. *Journal of General Internal Medicine*, v. 29, n. 3, p. 780–787, 2014. ISSN 1525-1497.

TREVINO, V.; FALCIANI, F.; BARRERA-SALDAÑA, H. A. DNA Microarrays: a Powerful Genomic Tool for Biomedical and Clinical Research. *Molecular Medicine*, v. 13, n. October, p. 527–541, 2007.

TUCKEY, P. *UrlRewriteFilter*. 2017. Disponível em: <http://tuckey.org/urlrewrite/>.

VERBORGH, R.; WILDE, M. D. *Using OpenRefine*. [S.l.]: Packt Publishing, 2013. ISBN 9781783289080.

W3C. *Extensible Markup Language (XML) 1.0 (Fifth Edition)*. 2008. Disponível em: <http://www.w3.org/TR/xml/>.

W3C. *OWL 2 Web Ontology Language - Structural Specification and Functional-Style Syntax (Second Edition)*. 2012. Disponível em: <https://www.w3.org/2007/OWL/draft/ED-owl2-syntax-20090914/all.pdf>.

W3C. *OWL 2 Web Ontology Language Direct Semantics (Second Edition)*. 2012. Disponível em: <http://www.w3.org/TR/2012/REC?owl2?direct?semantics?20121211/>.

W3C. *OWL 2 Web Ontology Language Primer (Second Edition)*. 2012. 1–34 p. Disponível em: <https://www.w3.org/TR/2012/REC-owl2-primer-20121211/>.

W3C. *OWL 2 Web Ontology Language Profiles (Second Edition)*. 2012. Disponível em: <http://www.w3.org/TR/2012/REC-owl2-profiles-20121211/>.

W3C. OWL 2 Web Ontology Language RDF-Based Semantics (Second Edition). 2012. Disponível em: <http://www.w3.org/TR/2012/REC-owl2-rdf-based-semantics-20121211/>.

W3C. *W3C XML Schema Definition Language (XSD) 1.1 Part 1: Structures*. [S.l.], 2012. Disponível em: <https://www.w3.org/XML/Schema>.

W3C. *SPARQL 1.1 Overview*. 2013. Disponível em: <http://www.w3.org/TR/2013/REC?sparql11?overview?20130321>.

W3C. *RDF 1.1 Concepts and Abstract Syntax*. 2014. Disponível em: <http://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/>.

W3C. *RDF Schema 1.1*. 2014. 1–16 p. Disponível em: <http://www.w3.org/TR/2014/REC-rdf-schema-20140225/>.

WAAGMEESTER, A. et al. Using the semantic web for rapid integration of wikipathways with other biological online data resources. *PLOS Computational Biology*, Public Library of Science, v. 12, n. 6, p. 1–11, 06 2016.

WANG, Z.; GERSTEIN, M.; SNYDER, M. RNA-Seq: a revolutionary tool for transcriptomics. *Nature reviews Genetics*, v. 10, n. 1, p. 57–63, 2009.

WOOD, D. et al. *Linked Data - Structured data on the web*. [S.l.: s.n.], 2013. 336 p. ISBN 9781617290398.

World Health Organization. *ATC - Structure and principles*. 2017. Disponível em: <https://www.whocc.no/atc/structure_and_principles/>.

Apêndices

A

ESPECIFICAÇÃO DE REGRA DE TRANSFORMAÇÃO

Este documento apresenta a Forma Backus-Naur Estendida (EBNF) da linguagem de especificação de regras de transformação SSD2LOD Transformation Language.

```
<transformation_specification> ::= <configuration_element> <transformation_rule>+
<condition_element>* <search_element>*

<configuration_element> ::= 'config_element ' <config_body>
<config_body> ::= '{' <body_specification> '}'

<body_specification> ::= <baseIRI_spec> (',' <syntax_spec>)? (',' <header_spec>)?
    (',' <namespace_spec>)

<baseIRI_spec> ::= '"default_baseIRI" = "' string '"'

<syntax_spec> ::= '"export_syntax" =' <syntax>
<syntax> ::= '"Turtle"' | '"RDF/XML"' | '"N-Triples"'

<header_spec> ::= '"has_non_processable_header" = ' <boolean>
<boolean> ::= '"true"' | '"false"'

<namespace_spec> ::= '"namespace" = "' <namespace> '" refers_to ' <IRI>
<namespace> ::= string
<IRI> ::= '"' string '"'
<transformation_rule> ::= <rule_header> '{' <rule_body> '}'

<rule_header> ::= <rule_type>' '<rule_identifier>' '<subject_type>
<rule_type> ::= 'column_based_rule' | 'row_based_rule'
<rule_identifier> ::= string

<subject_type> ::= '[' ( (<SSD_column_header> <flag>*) |
    (<SSD_column_header> <flag>* '/;') )+ 'is_equivalent_to' <ontology_class> ']'
```

```
<SSD_column_header> ::= '"' string '"'
<ontology_class> ::= ('"' string '"') | ('"' <namespace> ':' <string> '"')

<rule_body> ::= <rule_statement> (',' <rule_statement> )*
<rule_statement> ::= 'links_to ' <triple_object> ('/;' <triple_object>)*
    ' using ' <triple_predicate>
<triple_object> ::= (<SSD_column_header> | <rule_identifier>) <flag>*
<triple_predicate> ::= '"' string '"'

<condition_element> ::= 'condition_element ' <condition_element_identifier>
    <condition_body>
<condition_element_identifier> ::= string

<condition_body> ::= '{' <condition_statement> (',' <condition_statement>)* '}'
<condition_statement> ::= <SSD_column_header> <operator> <value>
<operator> ::= '==' | '!=' | '<' | '<=' | '>' | '>='
<value> ::= '"' string '"'

<search_element> ::= <search_header> '{$' <sparql_query> '$}'

<search_header> ::= 'search_element' <search_element_identifier> <search_endpoint>
<search_element_identifier> ::= string
<search_endpoint> ::= '["' <URI> '"]'
<URI> ::= '"' string '"'

<sparql_query> :: string

<flag> ::= <defined_flag> (' ' <defined_flag>)+
<defined_flag> ::= <NM_flag> | <SP_flag> | <BaseIRI_flag> | <DefaultValue_flag> | <
    DT_flag> | <CustomID_flag> | <Col_flag> | <SE_flag> | <Condition_flag> |
    <NODE_flag>

<NM_flag> ::= '/NM'

<SP_flag> ::= '/SP(' <character> (',' <position>)? ')'
<character> ::= '"' string '"'
<position> ::= integer

<BaseIRI_flag> ::= '/BaseIRI(' <IRI> ',' <namespace> ')'
<namespace> ::= '"' string '"'

<DefaultValue_flag> ::= '/DefaultValue(' <content> ')'
<content> ::= '"' string '"'

<DT_flag> ::= '/DT(' <XSD_datatype_name> ')'
<XSD_datatype_name> ::= '"' string '"'

<CustomID_flag> ::= '/ID("' string '")'

```

```
77 <Col_flag> ::= '/COL(' (<column_number>|<column_name>) ',' <filename> ')'
78 <filename> ::= '"' string '"'
79 <column_number> ::= integer
80 <column_name> ::= '"' string '"'
81
82 <SE_flag> ::= '/SE(' <search_element_identifier>, <SPARQL_variable> ')'
83 <SPARQL_variable> ::= string
84
85 <Condition_flag> ::= '/CE(' <condition_element_identifier> ')'
86
87 <NODE_flag> ::= '/NODE(' <ontology_class> ')'
```

Listagem 10 – EBNF da linguagem SSD2LOD Transformation Language.

B

CICLOS DE TRANSFORMAÇÃO

Este apêndice apresenta as regras de transformação dos ciclos de transformação e as consultas SPARQL para as questões de competência.

B.1 Ciclo de transformação 01

```
config_element {
  "default_baseIRI" = "http://example.org/onto/individual/",
  "export_syntax" = "N-Triples",
  "namespace" = "obi" refers_to "http://purl.obolibrary.org/obo/OBI"
}
column_based_rule investigation ["Comment[ArrayExpressAccession]" /BASEIRI("https://www.ebi.ac.uk/arrayexpress/experiments/", "ebi-ae") is_equivalent_to "obi:investigation"]{
  links_to "Person First Name" /; "Person Last Name" using "Contributor",
  links_to "Comment[AEExperimentType]" using "experiment data type",
  links_to "Protocol Hardware" using "measurement device",
  links_to organismTransformation using "has participant"
}
row_based_rule organismTransformation [ "Characteristics[organism]" is_equivalent_to "obi:organism"]{
  links_to "Characteristics[organism]" using "label",
  links_to sampleTransformation using "has sample",
}

row_based_rule sampleTransformation ["Source Name" is_equivalent_to "obi:sample from organism"]{
  links_to "Characteristics[organism part]" using "organism part"
}
```

Listagem 11 – Regras de transformação para o ciclo de transformação 01.

```
PREFIX dcterms: <http://purl.org/dc/elements/1.1/>
SELECT DISTINCT ?researcher_name WHERE {

  ?investigation dcterms:contributor ?researcher_name .

}
```

Listagem 12 – Consulta SPARQL para a questão de competência 01.

```
PREFIX onto: <http://example.org/onto/exp5412#>
SELECT DISTINCT ?data_type WHERE {

  ?investigation onto:experiment_data_type ?data_type

}
```

Listagem 13 – Consulta SPARQL para a questão de competência 02.

```
PREFIX onto: <http://example.org/onto/exp5412#>
SELECT DISTINCT ?platform WHERE {

  ?investigation onto:measurement_device ?platform

}
```

Listagem 14 – Consulta SPARQL para a questão de competência 03.

```
PREFIX obo: <http://purl.obolibrary.org/obo/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT DISTINCT ?organism_name WHERE {

  ?organism a obo:OBI_0100026 .
  ?organism rdfs:label ?organism_name

}
```

Listagem 15 – Consulta SPARQL para a questão de competência 04.

```
PREFIX obo: <http://purl.obolibrary.org/obo/>
PREFIX onto: <http://example.org/onto/exp5412#>
SELECT DISTINCT ?sample_name WHERE {

  ?sample a obo:OBI_0000671 .
  ?sample onto:organism_part ?sample_name

}
```

Listagem 16 – Consulta SPARQL para a questão de competência 05.

B.2 Ciclo de transformação 02

```
config_element {
  "default_baseIRI" = "http://example.org/onto/individual/",
  "export_syntax" = "N-Triples",
  "namespace" = "uniprot" refers_to "http://purl.uniprot.org/core/"
}

row_based_rule CNSOverKidney [ "" /DefaultValue("CNSOverKidney") is_equivalent_to "Gene expression comparison"]{
  links_to upRegulatedKidney using "identified gene",
  links_to downRegulatedKidney using "identified gene",
  links_to "" /DefaultValue("CNSOverKidney") using "label"
}

row_based_rule upRegulatedKidney ["" /COL("gene_id", "Processed SSD - CNSOverKidney.txt") /; "" /DefaultValue("CNSOverKidney") /CE(ConditionUpRegulatedCNSOverKidney) is_equivalent_to "uniprot:Gene"]{
  links_to ""/COL("gene_name", "Processed SSD - CNSOverKidney.txt") using "label",
  links_to "" /COL("fdr", "Processed SSD - CNSOverKidney.txt") /DT("double") using "fdr value",
  links_to "" /COL("log2 Ratio", "Processed SSD - CNSOverKidney.txt") /DT("double") using "log 2 ratio value"
}

row_based_rule downRegulatedKidney ["" /COL("gene_id", "Processed SSD - CNSOverKidney.txt") /; "" /DefaultValue("CNSOverKidney") /CE(ConditionDownRegulatedCNSOverKidney) is_equivalent_to "uniprot:Gene"]{
  links_to ""/COL("gene_name", "Processed SSD - CNSOverKidney.txt") using "label",
  links_to "" /COL("fdr", "Processed SSD - CNSOverKidney.txt") /DT("double") using "fdr value",
  links_to "" /COL("log2 Ratio", "Processed SSD - CNSOverKidney.txt") /DT("double") using "log 2 ratio value"
}

condition_element ConditionUpRegulatedCNSOverKidney{
  "" /COL("fdr", "Processed SSD - CNSOverKidney.txt") <= "0.05",
  "" /COL("log2 Ratio", "Processed SSD - CNSOverKidney.txt") >= "1"
}

condition_element ConditionDownRegulatedCNSOverKidney{
  "" /COL("fdr", "Processed SSD - CNSOverKidney.txt") <= "0.05",
  "" /COL("log2 Ratio", "Processed SSD - CNSOverKidney.txt") <= "-1"
}

row_based_rule CNSOverLiver [ "" /DefaultValue("CNSOverLiver") is_equivalent_to "Gene expression comparison"]{
  links_to upRegulatedLiver using "identified gene",
```

```
37   links_to downRegulatedLiver using "identified gene",
38   links_to "" /DefaultValue("CNSOverLiver") using "label"
39 }
40
41 row_based_rule upRegulatedLiver ["" /COL("gene_id", "Processed SSD - CNSOverLiver.txt")
       /; "" /DefaultValue("CNSOverLiver") /CE(ConditionUpRegulatedCNSOverLiver)
       is_equivalent_to "uniprot:Gene"]{
42   links_to "" /COL("gene_name", "Processed SSD - CNSOverLiver.txt") using "label",
43   links_to "" /COL("fdr", "Processed SSD - CNSOverLiver.txt") /DT("double") using "fdr
       value",
44   links_to "" /COL("log2 Ratio", "Processed SSD - CNSOverLiver.txt") /DT("double") using
       "log 2 ratio value"
45 }
46
47 row_based_rule downRegulatedLiver ["" /COL("gene_id", "Processed SSD - CNSOverLiver.txt")
        /; "" /DefaultValue("CNSOverLiver") /CE(ConditionDownRegulatedCNSOverLiver)
       is_equivalent_to "uniprot:Gene"]{
48   links_to "" /COL("gene_name", "Processed SSD - CNSOverLiver.txt") using "label",
49   links_to "" /COL("fdr", "Processed SSD - CNSOverLiver.txt") /DT("double") using "fdr
       value",
50   links_to "" /COL("log2 Ratio", "Processed SSD - CNSOverLiver.txt") /DT("double") using
       "log 2 ratio value"
51 }
52
53 condition_element ConditionUpRegulatedCNSOverLiver{
54   "" /COL("fdr", "Processed SSD - CNSOverLiver.txt") <= "0.05",
55   "" /COL("log2 Ratio", "Processed SSD - CNSOverLiver.txt") >= "1"
56 }
57
58 condition_element ConditionDownRegulatedCNSOverLiver{
59   "" /COL("fdr", "Processed SSD - CNSOverLiver.txt") <= "0.05",
60   "" /COL("log2 Ratio", "Processed SSD - CNSOverLiver.txt") <= "-1"
61 }
62
63 row_based_rule CNSOverLung [ "" /DefaultValue("CNSOverLung") is_equivalent_to "Gene
       expression comparison"]{
64   links_to upRegulatedLung using "identified gene",
65   links_to downRegulatedLung using "identified gene",
66   links_to "" /DefaultValue("CNSOverLung") using "label"
67 }
68
69 row_based_rule upRegulatedLung ["" /COL("gene_id", "Processed SSD - CNSOverLung.txt") /;
       "" /DefaultValue("CNSOverLung") /CE(ConditionUpRegulatedCNSOverLung) is_equivalent_to
        "uniprot:Gene"]{
70   links_to "" /COL("gene_name", "Processed SSD - CNSOverLung.txt") using "label",
71   links_to "" /COL("fdr", "Processed SSD - CNSOverLung.txt") /DT("double") using "fdr
       value",
72   links_to "" /COL("log2 Ratio", "Processed SSD - CNSOverLung.txt") /DT("double") using "
       log 2 ratio value"
```

```
73 }
74
75 row_based_rule downRegulatedLung ["" /COL("gene_id", "Processed SSD - CNSOverLung.txt")
       /; "" /DefaultValue("CNSOverLung") /CE(ConditionDownRegulatedCNSOverLung)
       is_equivalent_to "uniprot:Gene"]{
76   links_to "" /COL("gene_name", "Processed SSD - CNSOverLung.txt") using "label",
77   links_to "" /COL("fdr", "Processed SSD - CNSOverLung.txt") /DT("double") using "fdr
       value",
78   links_to "" /COL("log2 Ratio", "Processed SSD - CNSOverLung.txt") /DT("double") using "
       log 2 ratio value"
79 }
80 condition_element ConditionUpRegulatedCNSOverLung{
81   "" /COL("fdr", "Processed SSD - CNSOverLung.txt") <= "0.05",
82   "" /COL("log2 Ratio", "Processed SSD - CNSOverLung.txt") >= "1"
83 }
84 condition_element ConditionDownRegulatedCNSOverLung{
85   "" /COL("fdr", "Processed SSD - CNSOverLung.txt") <= "0.05",
86   "" /COL("log2 Ratio", "Processed SSD - CNSOverLung.txt") <= "-1"
87 }
```

Listagem 17 – Regras de transformação para o ciclo de transformação 02

```
PREFIX onto: <http://example.org/onto/exp5412#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX uniprot: <http://purl.uniprot.org/core/>

SELECT DISTINCT ?genes_iri (?genes as ?upRegulated) WHERE {

  ?comparison rdfs:label "CNSOverKidney" .
  ?comparison onto:identified_gene ?genes_iri .

  ?genes_iri a uniprot:Gene.
  ?genes_iri rdfs:label ?genes .
  ?genes_iri onto:log2ratio_value ?log2 .

  FILTER (?log2 >= 1) .
}
```

Listagem 18 – Consulta SPARQL para a questão de competência 06.

```
PREFIX onto: <http://example.org/onto/exp5412#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX uniprot: <http://purl.uniprot.org/core/>

SELECT DISTINCT ?genes_iri (?genes as ?downRegulated) WHERE {

  ?genes_iri a uniprot:Gene.
  ?genes_iri rdfs:label ?genes .
  ?genes_iri onto:log2ratio_value ?log2 .

  ?comparison onto:identified_gene ?genes_iri .
  ?comparison rdfs:label "CNSOverKidney" .

  FILTER (?log2 <= -1) .
}
```

Listagem 19 – Consulta SPARQL para a questão de competência 07.

```
PREFIX onto: <http://example.org/onto/exp5412#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX uniprot: <http://purl.uniprot.org/core/>

SELECT DISTINCT ?genes_iri (?genes as ?upRegulated) WHERE {

  ?genes_iri a uniprot:Gene.
  ?genes_iri rdfs:label ?genes .
  ?genes_iri onto:log2ratio_value ?log2 .

  ?comparison onto:identified_gene ?genes_iri .
  ?comparison rdfs:label "CNSOverLiver" .

  FILTER (?log2 >= 1) .
}
```

Listagem 20 – Consulta SPARQL para a questão de competência 08.

```
PREFIX onto: <http://example.org/onto/exp5412#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX uniprot: <http://purl.uniprot.org/core/>

SELECT DISTINCT ?genes_iri (?genes as ?downRegulated) WHERE {

  ?genes_iri a uniprot:Gene.
  ?genes_iri rdfs:label ?genes .
  ?genes_iri onto:log2ratio_value ?log2 .

  ?comparison onto:identified_gene ?genes_iri .
  ?comparison rdfs:label "CNSOverLiver" .

  FILTER (?log2 <= -1) .
}
```

Listagem 21 – Consulta SPARQL para a questão de competência 09.

```
PREFIX onto: <http://example.org/onto/exp5412#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX uniprot: <http://purl.uniprot.org/core/>

SELECT DISTINCT ?genes_iri (?genes as ?upRegulated) WHERE {

  ?genes_iri a uniprot:Gene.
  ?genes_iri rdfs:label ?genes .
  ?genes_iri onto:log2ratio_value ?log2 .

  ?comparison onto:identified_gene ?genes_iri .
  ?comparison rdfs:label "CNSOverLung" .

  FILTER (?log2 >= 1) .
}
```

Listagem 22 – Consulta SPARQL para a questão de competência 10.

```
PREFIX onto: <http://example.org/onto/exp5412#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX uniprot: <http://purl.uniprot.org/core/>

SELECT DISTINCT ?genes_iri (?genes as ?downRegulated) WHERE {

  ?genes_iri a uniprot:Gene.
  ?genes_iri rdfs:label ?genes .
  ?genes_iri onto:log2ratio_value ?log2 .

  ?comparison onto:identified_gene ?genes_iri .
  ?comparison rdfs:label "CNSOverLung" .

  FILTER (?log2 <= -1) .
}
```

Listagem 23 – Consulta SPARQL para a questão de competência 11.

B.3 Ciclo de transformação 3

```
config_element {
  "default_baseIRI" = "http://example.org/onto/individual/",
  "export_syntax" = "N-Triples",
  "namespace" = "ro" refers_to "http://purl.obolibrary.org/obo/RO",
  "namespace" = "uniprot" refers_to "http://purl.uniprot.org/core/"
}

row_based_rule differentiallyExpressedGene ["gene_id.Set1" is_equivalent_to "uniprot:Gene
    "]{
  links_to "gene_name.Set1" using "label",
  links_to "gene_name.Set1" /SE(searchPathway, ?pathway) /NODE("Pathway") using "ro:
    involved in"
}

search_element searchPathway["http://sparql.wikipathways.org/"]{$
  PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
  PREFIX wp: <http://vocabularies.wikipathways.org/wp#>
  PREFIX dcterms: <http://purl.org/dc/terms/>
  SELECT DISTINCT ?pathway WHERE {
    ?geneProduct a wp:GeneProduct .
    ?geneProduct rdfs:label ?label_raw .
    ?geneProduct dcterms:isPartOf ?pathway .
    ?pathway a wp:Pathway .

    FILTER regex(ucase(str(?label_raw)), ucase(?tsvData)).

    ?pathway wp:organismName ?organism .
    FILTER regex(ucase(str(?organism )), ucase("mus musculus")).
  }
$}
```

Listagem 24 – Regras de transformação para o ciclo de transformação 03

```
PREFIX onto: <http://example.org/onto/exp5412#>
PREFIX obo: <http://purl.obolibrary.org/obo/>
PREFIX uniprot: <http://purl.uniprot.org/core/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dc: <http://purl.org/dc/terms/>
PREFIX dcterms: <http://purl.org/dc/elements/1.1/>
PREFIX wp: <http://vocabularies.wikipathways.org/wp#>

SELECT DISTINCT ?pathway_label
        ?pathway
        ?genes_count
        (count(?geneProduct) as ?genes_known)
        ((?genes_count / ?genes_known) as ?ratio) WHERE {

  {
    SELECT DISTINCT ?pathway (count(?gene_label) as ?genes_count) WHERE {
      ?genes_iri a uniprot:Gene.
      ?genes_iri rdfs:label ?gene_label .

      ?genes_iri obo:RO_0002331 ?pathway
    } GROUP BY ?pathway ORDER BY DESC(?genes_count)

  }

  SERVICE <http://sparql.wikipathways.org/> {
    ?geneProduct a wp:GeneProduct .
    ?geneProduct dc:isPartOf ?pathway .

    ?pathway dcterms:title ?pathway_label_with_lang .
    BIND(str(?pathway_label_with_lang) as ?pathway_label)
  }

} GROUP BY ?pathway ?genes_count ?pathway_label
ORDER BY DESC(?ratio)
```

Listagem 25 – Consulta SPARQL para a questão de competência 12.

B.4 Ciclo de transformação 4

```
config_element {
  "default_baseIRI" = "http://example.org/onto/individual/",
  "export_syntax" = "N-Triples",
  "namespace" = "ro" refers_to "http://purl.obolibrary.org/obo/RO",
  "namespace" = "uniprot" refers_to "http://purl.uniprot.org/core/",
  "namespace" = "go" refers_to "http://purl.obolibrary.org/obo/GO",
}

row_based_rule differentiallyExpressedGene ["gene_id.Set1" is_equivalent_to "uniprot:Gene
    "]{
  links_to "GO BP.Set1" /SP("; ") /NODE("go:biological_process") /SE(searchBP, ?bpIRI)
    using "ro:participates in"
}

search_element searchBP["http://rdf.geneontology.org/blazegraph/namespace/kb/sparql"]{$
  PREFIX obo: <http://www.geneontology.org/formats/oboInOwl#>
  SELECT DISTINCT ?bpIRI WHERE {
    ?bpIRI obo:id ?id .
    BIND(str(?id) as ?idString).
    VALUES ?idString { ?tsvData }
  }
$}
```

Listagem 26 – Regras de transformação para o ciclo de transformação 04.

```
PREFIX obo: <http://purl.obolibrary.org/obo/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX uniprot: <http://purl.uniprot.org/core/>
PREFIX up: <http://purl.uniprot.org/core/>
PREFIX owl: <http://www.w3.org/2002/07/owl#>

SELECT DISTINCT ?biological_process ?bp_iri ?expr_genes (COUNT(DISTINCT ?protein) as ?
    total_genes) ((?expr_genes / ?total_genes) as ?ratio) WHERE {

  {
    SELECT DISTINCT ?bp_iri (COUNT(?bp_iri) as ?expr_genes) WHERE {
      ?genes_iri a uniprot:Gene .
      ?genes_iri obo:RO_0000056 ?bp_iri .

      FILTER (?bp_iri not IN (<http://purl.obolibrary.org/obo/GO_0008150>,
                  <http://purl.obolibrary.org/obo/GO_0006810>,
                  <http://purl.obolibrary.org/obo/GO_0008152>,
                  <http://purl.obolibrary.org/obo/GO_0009058>,
                  <http://purl.obolibrary.org/obo/GO_0050896>,
                  <http://purl.obolibrary.org/obo/GO_0009607>,
                  <http://purl.obolibrary.org/obo/GO_0006139>,
                  <http://purl.obolibrary.org/obo/GO_0006807>,
                  <http://purl.obolibrary.org/obo/GO_0055085>,
                  <http://purl.obolibrary.org/obo/GO_0019222>,
                  <http://purl.obolibrary.org/obo/GO_0002376>,
                  <http://purl.obolibrary.org/obo/GO_0006811>,
                  <http://purl.obolibrary.org/obo/GO_0006725>,
                  <http://purl.obolibrary.org/obo/GO_0006796>,
                  <http://purl.obolibrary.org/obo/GO_0051234>,
                  <http://purl.obolibrary.org/obo/GO_0007275>,
                  <http://purl.obolibrary.org/obo/GO_0030154>,
                  <http://purl.obolibrary.org/obo/GO_0050790>,
                  <http://purl.obolibrary.org/obo/GO_0044237>,
                  <http://purl.obolibrary.org/obo/GO_0050794>,
                  <http://purl.obolibrary.org/obo/GO_0032502>,
                  <http://purl.obolibrary.org/obo/GO_1901360>
                  )
          )

    } GROUP BY ?bp_iri ORDER BY DESC(?expr_genes)

  }

  SERVICE SILENT <https://sparql.uniprot.org/sparql/> {
    ?bp_iri rdfs:label ?biological_process .
    ?sub_bp (rdfs:subClassOf|owl:someValuesFrom)* ?bp_iri .
    ?protein up:classifiedWith ?sub_bp.
    ?protein up:organism <http://purl.uniprot.org/taxonomy/10090> .
  }
```

```
49
50 } GROUP BY ?biological_process ?bp_iri ?expr_genes ?total_genes
51 ORDER BY DESC(?ratio)
```

Listagem 27 – Consulta SPARQL para a questão de competência 13.