UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

UM PROCESSO PARA EXTRAÇÃO DE ESQUEMAS CONCEITUAIS EM FONTES DE DADOS JSON BASEADO EM TÉCNICAS DE SIMILARIDADE DE TEXTO

DISSERTAÇÃO DE MESTRADO

Fhabiana Thieli dos Santos Machado

Santa Maria, RS, Brasil

2017

UM PROCESSO PARA EXTRAÇÃO DE ESQUEMAS CONCEITUAIS EM FONTES DE DADOS JSON BASEADO EM TÉCNICAS DE SIMILARIDADE DE TEXTO

Fhabiana Thieli dos Santos Machado

Dissertação apresentada ao Curso de Mestrado do Programa de Pós-Graduação em Ciência da Computação (PPGCC), Área de Concentração em Computação, da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para obtenção do grau de
Mestre em Ciência da Computação

Orientadora: Profª. Drª. Deise de Brum Saccol

Santa Maria, RS, Brasil

2017

Machado, Fhabiana Thieli dos Santos

Um processo para extração de esquemas conceituais em fontes de dados JSON baseado em técnicas de similaridade de texto / por Fhabiana Thieli dos Santos Machado. – 2017.
99 f.: il.; 30 cm.

Orientadora: Deise de Brum Saccol
Dissertação (Mestrado) - Universidade Federal de Santa Maria, Centro de Tecnologia, Programa de Pós-Graduação em Ciência da Computação, RS, 2017.

1. NoSQL orientados a documentos. 2. Extração de esquema. 3. Similaridade de texto. I. Saccol, Deise de Brum. II. Título.

E-mail: fsantos@inf.ufsm.br

Universidade Federal de Santa Maria
Centro de Tecnologia
Programa de Pós-Graduação em Ciência da Computação

A Comissão Examinadora, abaixo assinada,
aprova a Dissertação de Mestrado

UM PROCESSO PARA EXTRAÇÃO DE ESQUEMAS CONCEITUAIS EM FONTES DE DADOS JSON BASEADO EM TÉCNICAS DE SIMILARIDADE DE TEXTO

elaborada por
Fhabiana Thieli dos Santos Machado

como requisito parcial para obtenção do grau de
Mestre em Ciência da Computação

COMISSÃO EXAMINADORA:

Deise de Brum Saccol, Drª.
(Presidente/Orientadora)

Sergio Luis Sardi Mergen, Dr. (UFSM)

Ronaldo dos Santos Mello, Dr. (UFSC)

Santa Maria, 29 de junho de 2017.

AGRADECIMENTOS

Primeiramente, eu gostaria de agradecer àquele que me deu a vida, força, coragem e oportunidade de estar escrevendo isso hoje: Deus. Foram dois anos de muitos sacrifícios, mas de experiências e oportunidades incríveis.

Agradeço ao apoio incondicional da minha família, base de tudo, que por vezes nem conseguiam entender como eu daria conta de tudo e ficavam na expectativa comigo. Nem eu entendo, mas com fé tudo se encaixou em seu devido lugar e deu certo.

À minha mãe, Marli Machado, hoje professora aposentada do estado, que muito acompanhei em sala de aula quando criança, obrigada pela força e orações. Ao meu pai, Valdoi Machado, que fica todo bobo ao falar que tem uma filha estudada, obrigada pelas orações. E aos meus maninhos gêmeos, Jamer e Jader, corram atrás porque o futuro só depende de vocês.

Por fim, ao apoio, orientações e conselhos de minha orientadora, Dr^{a}. Deise de Brum Saccol, que mesmo no período de gestação e da novidade de ser mãe, não deixou de me auxiliar, responder emails e corrigir textos :). Obrigada pela compreensão e apoio especialmente quando iniciei como professora substituta. As lições aprendidas durante o mestrado levarei para toda minha vida acadêmica.

"Assim diz o Senhor: "Não se glorie o sábio em sua sabedoria nem o forte em sua força nem o rico em sua riqueza, mas quem se gloriar, glorie-se nisto: em compreender-me e conhecer-me, pois eu sou o Senhor, e ajo com lealdade, com justiça e com retidão sobre a terra, pois é dessas coisas que me agrado."Jeremias 9:23,24"

RESUMO

Dissertação de Mestrado
Programa de Pós-Graduação em Ciência da Computação
Universidade Federal de Santa Maria

UM PROCESSO PARA EXTRAÇÃO DE ESQUEMAS CONCEITUAIS EM FONTES DE DADOS JSON BASEADO EM TÉCNICAS DE SIMILARIDADE DE TEXTO

AUTORA: FHABIANA THIELI DOS SANTOS MACHADO
ORIENTADORA: DEISE DE BRUM SACCOL
Local da Defesa e Data: Santa Maria, 29 de junho de 2017.

Os modelos de dados NoSQL (*Not Only SQL*) vêm se destacando devido à sua promessa de flexibilidade de esquemas e escalabilidade frente ao grande volume de dados gerados atualmente. Sua flexibilidade permite, por exemplo, que documentos dentro da mesma coleção possuam campos distintos. Este fato se torna um problema no momento que é preciso acessar o banco de dados de forma unificada, ou de modo automatizado através de rotinas de programação, pois não há uma padronização em sua estrutura. Nesse sentido o trabalho apresenta um processo para extração de esquema em fontes de dados JSON (*JavaScript Object Notation*).

Esta proposta diferencia-se por analisar campos que representam a mesma informação, mas que estejam escritos de modo diferente. No contexto deste trabalho, diferença de escrita diz respeito ao tratamento de sinônimos, grafia similar e mesmo radical de palavra. Para tal, são utilizadas técnicas como funções de similaridade baseadas em caractere e sinônimos, assim como extrator de radicais. Portanto, o objetivo do trabalho é extrair o esquema implícito presente nessas fontes de dados aplicando diferentes técnicas de equivalência textual em nomes de campos, bem como produzir um esquema conceitual e os respectivos mapeamentos para os termos equivalentes.

Palavras-chave: NoSQL orientados a documentos. Extração de esquema. Similaridade de texto.

ABSTRACT

Master's Dissertation
Post-Graduate Program in Computer Science
Federal University of Santa Maria

A PROCESS FOR CONCEPTUAL SCHEMA EXTRACTION IN DATASETS JSON BASED ON TEXT SIMILARITY TECHNIQUES

AUTHOR: FHABIANA THIELI DOS SANTOS MACHADO
ADVISOR: DEISE DE BRUM SACCOL
Defense Place and Date: Santa Maria, June 29th, 2017.

NoSQL (Not Only SQL) data models have been notable for their promise of schema flexibility and scalability considering the large volume of data. Their flexibility allows, for example, that documents within the same collection have different attributes. This fact becomes a problem when there is the need to access the database in a unified way, or in an automated way through programming, since there is no standard structure. In this sense, this work presents a process for schema extraction in datasets in JSON (JavaScript Object Notation) data sources.

This proposal differs by analyzing attributes that represent the same information, but are differently written. In the context of this work, writing difference concerns the treatment of synonyms, similar spelling and identical word radical. To achieve this goal, we use techniques such as character based similarity functions and synonyms, as well as stemming extractor. Therefore, this work aims to extract the implicit schema in these datasets by applying different textual equivalence techniques in attribute names, as well as to produce a conceptual schema and the respective mappings for the equivalent terms.

Keywords: Document-oriented database. structure extraction. text similarity.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE LISTAGENS

LISTA DE ABREVIATURAS E SIGLAS

API	*Application Programming Interface*
AHP	*Analytic Hierarchy Process*
BSON	*Binary JSON*
CSV	*Comma Separated Values*
DER	*Diagrama Entidade-Relacionamento*
DTD	*Document Type Definition*
EER	*Entidade-Relacionamento Estendido*
ER	*Entidade-Relacionamento*
GB	*Gigabyte*
GUI	*Graphical User Interface*
HD	*Hard Disk*
IC	*Information Content*
IDEF1X	*Integrated DEFinition for Information Modelling*
JSON	*JavaScript Object Notation*
LCS	*Lowest Common Subsumer*
NOSQL	*Not Only Structured Query Language*
PHP	*Hypertext Preprocessor*
RAM	*Random Access Memory*
RSLP	*Removedor de Sufixos da Língua Portuguesa*
SGBD	*Sistema Gerenciador de Banco de Dados*
XML	*eXtensible Markup Language*

SUMÁRIO

1 INTRODUÇÃO

Devido ao crescente volume de dados gerado por diversas aplicações hoje em dia, algumas questões começaram a surgir, como por exemplo, suporte à escalabilidade. Um único servidor pode não comportar a quantidade de dados gerada em certo espaço de tempo. Os bancos de dados relacionais, até então consolidados no mercado, não foram projetados para executar em um sistema de dados distribuído (VIEIRA et al., 2012). Deste modo, começaram a ser desenvolvidas algumas soluções específicas para este problema, como é o caso do pioneiro *BigTable* da *Google Inc*. (CHANG et al., 2008). Esses novos modelos de banco de dados foram nomeados pelo termo NoSQL (*Not Only SQL*) e caracterizam-se por ter ausência de esquema, executar de forma distribuída e não possuir relacionamentos gerenciados pelo SGBD (Sistema Gerenciador de Banco de Dados). No geral, possuem código aberto com origem em projetos de software livre (S.D, 2011).

Um dos pontos que mais se destacam em NoSQL é a flexibilidade de esquema (BRITO, 2010). Porém é um equívoco afirmar que ele não existe. Ao projetar um banco de dados, se faz necessária uma aplicação que o acesse. Para que esta seja programável, assume-se que determinados campos estão presentes, bem como que contenha dados com determinado significado e com um tipo. Dessa forma, há um esquema implícito no banco de dados, ou seja, um conjunto de suposições sobre a estrutura dos dados no código que o manipula.

Os bancos de dados NoSQLs podem ser classificados quanto ao modelo de dados em chave-valor, documento, colunar e grafos (SADALAGE; FOWLER, 2012). Conforme ranking do mês de maio de 2017 do *DB-Engines Ranking*[1] que mede a popularidade dos sistemas gerenciadores de banco de dados, o NoSQL *MongoDB* (orientado a documentos) destaca-se em 5º lugar atrás dos SGBD*s PostgreSQL, Oracle e MySQL*. Portanto este trabalho é direcionado a esta categorização.

Alguns trabalhos existentes na literatura tratam da extração de esquemas em bancos de dados NoSQL. Com o objetivo de obter este esquema implícito (IZQUIERDO; CABOT, 2013) propõe um processo composto de três fases para dados armazenados no formato JSON (*JavaScript Object Notation*). De mesmo modo, (KLETTKE et al., 2015) propõe um processo de extração de esquema que infere *outliers* estruturais.

No entanto, tal flexibilidade de esquemas neste modelo permite que um campo possa

[1] http://db-engines.com/en/ranking

estar presente em alguns documentos e em outros não, ou ainda, que existam campos com nomes distintos, inclusive entre documentos pertencentes à mesma coleção. Desta forma, podem existir campos de um mesmo domínio que representam a mesma informação, mas que estão descritos com nomes diferentes. Um exemplo é demonstrado na Listagem 1.1. Este exemplo apresenta três documentos JSON pertencentes ao mesmo domínio.

```
1      Documento A      |     Documento B     |   Documento C
2  {                    |{                    |{
3    "Grades":{         |"Grades":{           |"Grades":{
4      "_id": {         |  "_id": {           |  "_id": {
5       "$oid":"50b59"  |   "$oid":"50bed"    |   "$oid": "6522c3"
6      },               |  },                 |   },
7      "student": 0 ,   |  "students": 0,     |  "learner": 0,
8      "class_id": 2,   |  "id_class": 2,     |  "class_id": 2,
9      "scores":[{      |  "scores":[{        |  "scores":[{
10      "type": "exam",|    "type":"exam",    |   "type":"exam",
11      "score": 21.24}|    "average":21.25}|   "grade": 21.23}
12      ]               |   ]                 |    ]
13     }                |  }                  |   }
14    }                 | }                   |  }
```

Listagem 1.1 – Exemplo de documentos com campos que representam a mesma informação

Os exemplos apontam algumas informações com mesmo significado, porém representadas de maneira diferente: (i) *student* (linha 7A) e *students* (linha 7B) possivelmente se referem a estudantes, pois apresentam apenas uma diferença de plural. Este tipo de alteração pode ocorrer com demais sufixos e prefixos de linguagem; (ii) *class_id* (linha 8A e 8C) e *id_class* (linha 8B) possivelmente se referem à identificação da turma, porém os termos estão escritos em ordem trocada, ou seja, uma com o termo "id" no início e outra no final da palavra; (iii) *score* (linha 11A), *average* (linha 11B) e *grade* (linha 11C) possivelmente se referem às notas de alunos, entretanto com grafias diferentes, ou seja, com palavras sinônimas.

Esta questão se torna de grande importância no momento em que é preciso acessar o banco de dados de forma unificada, ou ainda, de modo automatizado, através de rotinas de programação, uma vez que não há uma padronização na estrutura dos documentos. Da mesma forma, na *web*, também podem existir documentos de fontes diferentes, que se queira acessar de modo unificado. Os trabalhos encontrados na revisão de literatura não abordam a questão de grafias diferentes para campos equivalentes.

Neste contexto, o objetivo desta dissertação é extrair o esquema implícito em fontes de dados JSON, identificando equivalências entre campos que estejam escritos de forma diferente, mas que representam a mesma informação. Nesta proposta, a identificação de equivalências

trata de palavras que apresentam grafia similar, mesmo radical de origem e sinônimos. Dessa forma, possibilita-se o acesso de modo unificado a essas fontes de dados ainda que não haja uma padronização nos seus campos, como por exemplo, em bancos de dados NoSQL orientados a documentos.

1.1 Objetivos Específicos

Os objetivos específicos desta dissertação são os seguintes:

- Definir um processo para a extração de esquemas em fontes de dados JSON, bem como, em bancos de dados NoSQL orientados a documentos;
- Definir um mecanismo de equivalência entre campos dos documentos, com base em análise de similaridade;
- Definir um mecanismo de geração de esquema conceitual consolidado a partir de diversos documentos, mantendo os mapeamentos entre os campos equivalentes;
- Validar a proposta de extração de esquemas em um conjunto de dados real, analisando medidas de revocação e precisão.

O processo proposto usa técnicas de similaridade baseadas em caractere e em conhecimento, isto é, sinônimos juntamente com um extrator de radicais para identificar equivalências. Este processo é estruturado em quatro etapas: (1) pré-processamento; (2) análise de similaridade; (3) identificação de equivalências e (4) representação da estrutura. Como entrada, é prevista uma coleção pertencente ao mesmo domínio armazenada no formato JSON em um banco de dados NoSQL orientado a documentos. Como saída, produz-se uma representação em nível conceitual do banco de dados NoSQL. Para os campos que foram unificados na saída gerada, o processo ainda armazena os mapeamentos feitos em formato JSONPath. A validação da proposta foi feita através de estudo de caso no domínio de publicações.

1.2 Organização do Texto

Este trabalho é organizado da seguinte maneira.

O Capítulo 2 é dedicado à **Fundamentação Teórica**. Este capítulo apresenta o modelo de dados NoSQL orientado a documentos, relata seus principais formatos de armazenamento:

JSON e XML (*Extensible Markup Language*), bem como alguns apontamentos sobre esquemas e representação conceitual para esta categorização. Do mesmo modo, expõe as técnicas aplicadas para identificação de equivalência textual, dentre elas: funções de similaridade baseadas em caractere e conhecimento, a técnica de extração de radicais e explica o método de tomada de decisões AHP (Processo Analítico Hierárquico) para auxiliar na inferência de equivalências. Por fim, são descritos os trabalhos relacionados.

O capítulo 3 define o **Processo Proposto**. Inicialmente este capítulo apresenta uma visão geral e a seguir detalha todo o processo definido. Este é composto pelas etapas de pré-processamento, análise de similaridade, identificação de equivalências e representação da estrutura. Para cada etapa é descrito o objetivo, entrada e saída. Neste processo cada atividade principal é especificada em subatividades que são detalhadas e exemplificadas. Estas subatividades são descritas por algoritmos.

O capítulo 4 é o **Estudo de Caso**. Este capítulo expressa a aplicação do processo com estudo de caso do domínio de publicações acadêmicas. São apresentadas as ferramentas utilizadas e duas formas de teste: em blocos ou tratando todo documento como um único bloco. Por se tratar de um mecanismo semiautomático, por vezes, um especialista do domínio necessita intervir em algumas decisões. O capítulo encerra com as discussões.

O capítulo 5 se refere à **Conclusões**. Neste capítulo são apontados os trabalhos futuros, demais considerações e contribuições referentes a esta dissertação.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo são descritos os conceitos, técnicas e implementações necessárias ao desenvolvimento do processo. Para tal são relatados:

- os principais conceitos ligados a bancos de dados NoSQL orientados a documentos: seus principais formatos de armazenamento; a sintaxe JSONPath; uma noção sobre como são definidos esquemas e a representação conceitual para este modelo;
- as técnicas para identificação da equivalência: funções de análise de similaridade textual baseadas em caractere e em conhecimento conforme categorização de (GOMAA; FAHMY, 2013); o algoritmo extrator de radicais e um método de apoio à tomada de decisões para auxiliar no processo de calcular a correspondência dos termos;
- os trabalhos relacionados com o processo de extração proposto.

2.1 NoSQL Orientados a Documentos

Segundo (SADALAGE; FOWLER, 2012) o termo NoSQL é aplicado a "bancos de dados não relacionais que abraçam um esquema livre, executam em sistemas distribuídos e abrem mão da consistência por outras propriedades úteis".

Cada banco de dados criado utiliza um modelo diferente. Estes podem ser classificados em quatro categorias distintas: chave-valor, orientado a documentos, orientado a colunas e orientado a grafos. Com exceção do modelo de grafos, os dados são armazenados por agregado. Este termo representa um conceito semelhante a um registro que suporta uma lista de valores aninhados ou conjunto de objetos.

O modelo orientado a documentos se caracteriza por armazenar coleções. Uma coleção é definida por lotes de agregados, como os representados no trecho de código seguinte das linhas 1 a 9. Cada registro possui um identificador que é usado para obter o conjunto de dados. Geralmente este identificador é gerado automaticamente pelo banco de dados ou pode ser atribuído manualmente na inserção. A categoria de documentos é semelhante ao chave-valor, mas diferencia-se pelo fato de permitir consultas nos campos aninhados internos do registro. Seu principal conceito é o de documento que consiste em um objeto com código único e um conjunto de campos que podem ser do tipo XML, JSON, BSON (*Binary JSON*), dentre outros.

Complementa (SADALAGE; FOWLER, 2012) que um documento é uma estrutura de dados de árvore hierárquica que pode consistir de mapas, coleções e valores escalares. Acrescenta que eles são semelhantes entre si, mas não precisam ter exatamente a mesma estrutura, ou seja, o esquema dos dados pode diferir entre documentos, mas estes ainda podem pertencer à mesma coleção. Nesta categoria, os bancos de dados NoSQL mais populares são: MongoDB[2], CouchDB[3], Terrastore[4] e OrientDB[5].

Na Listagem 2.1 são apresentados dois exemplos de agregados (BOAGLIO, 2015):

```
1 {" clienteId": 1,                  //Objeto cliente
2   "nome ": "Martin ",
3   "endCobranca": [{"cidade":"Porto Alegre"}]
4 }
5 {"pedidoId": 99,                   //Objeto pedido
6  "clienteId": 1,
7   "formaPgto": "cartao",
8    "item": [{"produto": "arroz"}]
9 }
```

Listagem 2.1 – Exemplo de agregados

Estes objetos poderiam ser registros no banco de dados: um representando o cliente e outro o pedido. O cliente, por exemplo, possui um campo "*endCobranca*" (linha 3), que é uma lista de valores, assim como o campo item (linha 8) do pedido. No segundo objeto está o campo "*clienteId*" (linha 6) indicando a qual cliente pertence as informações do pedido.

Apesar de não suportar transações, a atualização de um agregado é atômica. Dessa forma, o modelo orientado a documentos destaca-se por possuir uma estrutura implícita e permitir consulta em seus campos aninhados.

2.1.1 JavaScript Object Notation

O principal formato de arquivo adotado por um banco de dados NoSQL orientado a documentos, como em MongoDB e CouchDB, é o *JavaScript Object Notation*. Este é um formato padrão para representação de dados que surgiu no ano de 2001 e é largamente utilizado para representar dados semiestruturados. De acordo com (INTERNATIONAL, 2013), JSON é um formato baseado em texto que possibilita a troca de dados independente da linguagem.

A importância dessa tecnologia é evidente na troca de informações entre sistemas, pois

[2] https://www.mongodb.com/
[3] http://couchdb.apache.org/
[4] http://highscalability.com/blog/2009/12/30/terrastore-scalable-elastic-consistent-document-store.html
[5] http://orientdb.com/orientdb/

se tornou padrão de formato para diversas aplicações e também é largamente empregado na comunicação de respostas Ajax, como em demais sistemas.

A estrutura deste formato é flexível e permite trabalhar com dados sem ter que definir um esquema rígido a *priori*. No entanto, ele possui um pequeno conjunto de regras estruturantes para sua representação. No geral, um arquivo JSON consiste em uma sequência de *tokens* do padrão *Unicode* de acordo com sua gramática. Sua *sintaxe* inclui as estruturas: colchetes ([ ]), chaves ({}), dois pontos e vírgula, além de *true, false* e *null*. Um exemplo de arquivo JSON é demonstrado no Capítulo 1 com os documentos A, B e C representados na Listagem 1.1.

Os tipos de valores aceitos são: *object, array, number, string, true, false e null*. A gramática para um objeto é representada na Figura 2.1. Conforme se observa, ela consiste de um par chave-valor delimitado por chaves ({}). Este conceito de chave-valor é também fundamento básico de qualquer armazenamento em banco de dados NoSQL.

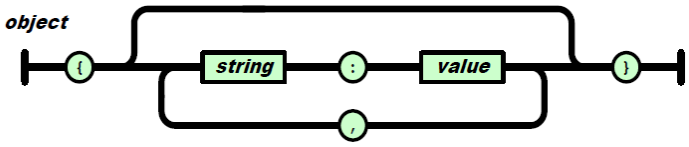

Figura 2.1 – Gramática JSON para objetos - Fonte: (INTERNATIONAL, 2013)

A gramática deste formato é pontual para o trabalho, visto que esse tipo de arquivo será processado ao longo da extração do esquema. Logo, todo documento de entrada deve ser válido, pois a gramática é utilizada para gerar regras da representação conceitual. Deste modo, com base no exposto, são listadas algumas definições:

- ***Documento***. Um documento pode ser composto de objetos e *arrays*;
- ***Objeto***. Cada objeto é representado por um par $string : value$ separados por vírgula e delimitado por colchetes;
- ***Arrays***. Cada *array* é composto por vários objetos ou apenas valores separados por vírgula;
- ***Campos***. Parte $string$ do par $string : value$ cujo objeto é composto.

Na proposta são realizadas tarefas como processar ou percorrer um documento JSON. Para tal, pode-se empregar algumas bibliotecas existentes ou efetuar este processo através de

um *parser*. Para a linguagem Java, por exemplo, há as seguintes implementações: o projeto JSR 353: *JavaTM API (Application Programming Interface) for JSON Processing*[6] que disponibiliza um arquivo de extensão .jar para importação e permite percorrer o documento através de eventos. Outra biblioteca disponível é a *json.org*[7].

2.1.1.1 JSONPath

Um dos objetivos específicos do trabalho se refere a armazenar os mapeamentos dos termos unificados no esquema conceitual para seus correspondentes documentos de origem. A linguagem JSONPath é utilizada para este propósito.

Um mapeamento pode ser definido como o conjunto de informações necessárias para acessar os dados interpretados como correspondentes em documentos diferentes. Por sua vez, essas informações podem ser representadas através de expressões de caminho.

Neste intuito, para realizar essa representação com o formato JSON existem algumas abordagens que se assemelham a *XPath* (XPATH, 2013). XPath é uma linguagem formada por um conjunto de regras e expressões para referenciar partes de um documento XML. Além deste método, há a possibilidade de mapear o JSON utilizando a mesma *sintaxe* do XPath[8]. Contudo, (GOESSNER, 2007) apresenta uma proposta mais apropriada denominada JSONPath que adapta expressões e regras da versão XML.

A principal diferença está na notação com ponto (*dot-notation*) ou com colchetes (*bracket-notation*). Um exemplo comparativo pode ser observado na Tabela 2.1 que ilustra ambas representações para um mesmo resultado.

Tabela 2.1 – Comparativo da sintaxe entre XPath e JSONPath e seus resultados - Fonte: (GOESSNER, 2007)

XPath	JSONPath	Resultado
/store/book/author	$.store.book[*].author	autores de todos os livros armazenados
//author	$..author	todos os autores
//book[3]	$..book[2]	terceiro livro
//book[price<10]	$..book[?(@.price<10)]	filtra todos os livros com preço menor que 10

Conforme exemplificado, a intenção é que as expressões sejam aplicadas de modo mais semelhante possível ao definido em XPath. Além da notação de ponto, como JSON não tem

[6] https://www.jcp.org/en/jsr/detail?id=353
[7] http://stleary.github.io/JSON-java/index.html
[8] https://www.npmjs.com/package/JPath

necessariamente um objeto raiz, a *sintaxe* assume o símbolo $ a um objeto de nível externo. Para tal, são fornecidas algumas implementações em linguagens como Java[9] e PHP[10] (*Hypertext Preprocessor*), bem como ferramentas *online* para validar expressões[11].

Nesta proposta, as expressões em JSONPath registram os campos consolidados no processo de extração, isto é, identificados como equivalentes à mesma informação.

2.1.2 Extensible Markup Language

Um banco de dados NoSQL orientado a documentos tem dois formatos principais de armazenamento: JSON e XML. Por outro lado, o processo apresentado tem por base apenas o formato JSON, por ser mais recente e mais explorado em sistemas NoSQL como MongoDB e CouchDB. Contudo, é conveniente que este processo de extração se aplique a ambos.

O XML é um dos principais formatos de portabilidade para diferentes aplicações. Seus documentos possuem a estrutura embutida entre *tags* e podem ser acompanhados de um DTD (*Document Type Definition*) (MOH; LIM; NG, 2000) que formaliza e define sua estrutura, mas isto não é obrigatório.

De modo que o trabalho permita o uso do formato XML, são apresentadas algumas alternativas. Uma das formas de tratamento apropriada para esta questão é a conversão para JSON antes do início do processo. Existem algumas ferramentas disponíveis *online*[12] ou ainda a partir de aplicações proprietárias como *Altova XMLSpy*. Do mesmo modo, é possível realizar a conversão diretamente através de linguagens de programação. Java, por exemplo, fornece a biblioteca *JSON-Java*[13] apropriada para tal tarefa.

2.1.3 Esquema em Modelos NoSQL

Como o trabalho envolve extração de esquemas em bancos de dados NoSQL orientado a documentos, os conceitos de entidade, atributo e relacionamento necessitam ser explorados, pois seguem algumas especificidades das definições do modelo relacional. Para tal, são revisados alguns trabalhos que se preocupam com o projeto e gerenciamento de esquemas em modelos NoSQL.

[9] https://github.com/jayway/JsonPath
[10] https://code.google.com/archive/p/jsonpath/
[11] https://jsonpath.curiousconcept.com/
[12] https://www.freeformatter.com/, http://www.utilities-online.info/xmltojson/#.WCjAdforLIU
[13] https://github.com/stleary/JSON-java

Entidade. Para (VARGA; JÁNOSI-RANCZ; KÁLMÁN, 2016), cada entidade está em um contexto multivalorado e cada objeto deste contexto pode ser modelado como entidade. Isto é, entidade tem o mesmo sentido do conceito relacional, apenas difere-se em poder corresponder a um documento ou objeto em JSON, etc.

Atributo. Semelhante ao modelo relacional, de acordo com (STÖRL et al., 2015), atributos são definidos como propriedades representantes de objetos do mundo real. Por exemplo, *arrays* podem representar atributos multivalorados.

Relacionamento. Esta é a principal diferença, no contexto deste trabalho, pois um banco de dados NoSQL não é estruturado com base em relações. Para fins de flexibilidade, um banco de dados NoSQL geralmente possui um modelo com dados aninhados que inferem relacionamentos implícitos. Para exemplificação dos conceitos é utilizado o trecho de código referente ao documento A na Listagem 1.1. Deste modo, "*_id*" (linha 4) pode ser considerado uma entidade que possui atributo "*$oid*" (linha 5), assim como "*scores*" (linha 9) também é entidade com atributos "*type e score*" (linhas 10 e 11). Ambos possuem um relacionamento implícito por pertencerem ao mesmo documento.

Indica (BOAGLIO, 2015) que o princípio de um modelo de dados NoSQL é agregar dados que fazem parte do mesmo contexto na mesma coleção. Dessa forma, flexibilizando a estrutura em favor de maior eficiência em leituras e escritas, isto é, para melhor acesso e recuperação.

Fazendo uma analogia ao modelo relacional, em um relacionamento do tipo um-para-um uma entidade é aninhada em outra. Em relacionamentos um-para-muitos, conforme (VARGA; JÁNOSI-RANCZ; KÁLMÁN, 2016), geralmente o lado "N"do relacionamento é aninhado. Já para o caso de ocorrências do tipo muitos-para-muitos, uma possibilidade é a utilização de referências bidirecionais, apesar de o SGBD não oferecer atualizações atômicas neste caso. Isto é, utilizando o exemplo dos agregados *cliente e pedido* apresentados na Seção 2.1, seria semelhante a criar uma chave de pedido em cliente e de cliente em pedido.

2.1.4 Representação Conceitual em NoSQL

Quanto à modelagem conceitual, em termos de modelo relacional, a representação utilizada é geralmente o Diagrama Entidade Relacionamento (DER). Esta é uma abordagem criada em 1976 por Peter Chen e sua notação é considerada como padrão.

Por outro lado, quanto à representação conceitual em NoSQL existem duas aborda-

gens principais. A primeira consiste em utilizar a notação Entidade-Relacionamento Estendido (EER) (LIMA, 2016), porém este modelo é próprio dos bancos de dados relacionais e não trata das especificidades de um modelo para agregados. A segunda é encontrada nos trabalhos de (BENSON, 2014) e (JOVANOVIC; BENSON, 2013) que trata de modelagem para bancos de dados NoSQL.

Na abordagem de (JOVANOVIC; BENSON, 2013) a representação é feita através da notação denominada IDEF1X (*Integrated DEFinition for Information Modelling*) que foi desenvolvida na década de 70. Ela possui adaptações para caracterizar a modelagem em NoSQL através de regras para representação de relacionamentos, entidades, etc. Todavia, seu pré-requisito é que em um modelo de agregados as outras entidades derivem de uma raiz, ou seja, uma única entidade padrão designadora de dependências com responsabilidade de atualizações para cada uma das entidades dependentes.

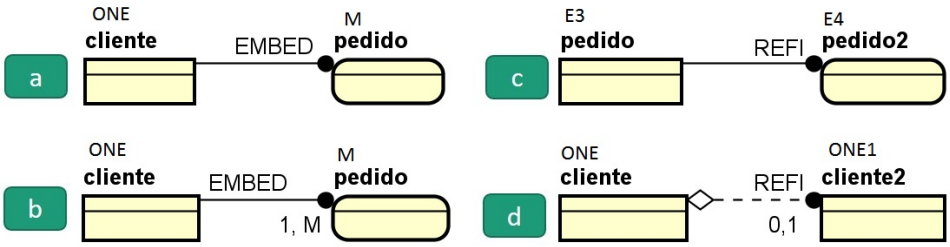

Figura 2.2 – Exemplo da notação proposta para modelo de dados orientados a agregados - Fonte: adaptado de (JOVANOVIC; BENSON, 2013)

Um exemplo pode ser visualizado na Figura 2.2 que apresenta algumas notações possíveis para relacionamentos com cardinalidade do tipo $M : 1$ nas Figuras *a e b*, ou seja, aponta um agregado embutido. O rótulo “EMBED” deve ser colocado ao lado da entidade que será incorporada. As notações das Figuras *c e d* ilustram relacionamentos de cardinalidade $1 : 1$ e $0 : 1$, respectivamente. O operador “REFI” é posicionado na entidade onde somente um valor de referência será "copiado", isto é, referenciado na outra entidade, assim gerando uma espécie de conexão entre as classes.

2.2 Técnicas para Identificação de Equivalência Textual

As técnicas para auxílio na identificação de equivalência entre os campos dos documentos utilizadas nesta proposta são descritas nesta seção. No contexto deste trabalho, ser equivalente abrange semelhança entre caracteres, sinônimos e mesmo radical da palavra.

Segundo (GOMAA; FAHMY, 2013), as técnicas de similaridade textual são categorizadas em baseadas em caractere e conhecimento, permitindo assim uma análise da escrita e do significado da palavra, respectivamente. As técnicas categorizadas como baseadas em conhecimento indicam o grau de semelhança ou o grau de correspondência para sinônimos. Entre outras técnicas, faz parte desta categoria o Wordnet (MILLER, 1995), que é um banco de dados léxico da língua inglesa.

Outra técnica aplicada neste sentido é a de análise morfológica da palavra que consiste em extrair seu radical, eliminando sufixos e prefixos. Esta é aplicada como similaridade ao analisar os valores de saída: um para equivalente ou zero para diferente.

Esses critérios (semelhança, sinônimos e radical), combinados inferem a equivalência. Para auxílio neste cálculo é empregada uma técnica de apoio a tomada de decisões para definição de pesos.

2.2.1 Similaridade de Texto

Esta seção trata das técnicas consideradas na análise da similaridade de texto que são aplicadas aos campos do documento para definir a unificação dos termos. Segundo (DORNELES, 2010) o principal objetivo de similaridade de dados é mostrar computacionalmente, através de um valor, o quanto dois objetos são semelhantes entre si. Este conceito pode englobar tanto texto simples (palavras), textos longos (como por exemplo, em documentos estruturados) ou com estruturas de dados mais complexas (árvores, grafos, imagens) etc.

As medidas existentes de cálculo de similaridade são bastante distintas. Dependem do objetivo, objeto de entrada e seu algoritmo de funcionamento, como por exemplo, Levenshtein[14] (LEVENSHTEIN, 1966), Jaro (JARO, 1989), Jaro-Winkler (WINKLER, 1990), etc. Como regra geral, aponta (C. F. DORNELES, 2008) que dada uma função de similaridade $fs\left(a_1, a_2\right) \rightarrow s$, esta calcula um escore s no intervalo $[0, 1]$ para um par de valores $a1$ e $a2$.

Em termos de aplicação, estas funções podem ser utilizadas, por exemplo, em integração de dados para reconhecer representações de dados originadas em diferentes fontes que representam o mesmo objeto do mundo real. Assim, pode-se dizer que dois objetos são considerados similares se o escore gerado por uma função f for maior do que um determinado limiar, ou seja, ponto de corte.

Quanto à categorização há diferentes classificações. Para (DORNELES, 2010), a simila-

[14] Também encontrada na literatura como Levenstein

ridade textual pode ser dividida em valores atômicos e agregados, ou seja, em nível de palavras ou de toda estrutura do documento, por exemplo.

De outro modo, segundo (GOMAA; FAHMY, 2013), as palavras podem ser similares de duas formas: lexicalmente e semanticamente. No primeiro item, as palavras são similares se possuem uma sequência de caracteres semelhantes. No segundo ponto, se elas têm o mesmo sentido, empregadas da mesma forma, no mesmo contexto, com grafia diferente. Ainda, as funções de análise léxica podem ser categorizadas como baseadas em: (*i*) *string*, com subcategorias para estudo caractere a caractere e para análise do termo como parte da palavra. De outro modo, (*ii*) as de análise semântica se categorizam em baseadas no *corpus* e conhecimento.

Para este trabalho são analisadas palavras constantes nos campos dos documentos JSON (valores atômicos), porém utilizando a classificação de (GOMAA; FAHMY, 2013) detalhada a seguir.

2.2.1.1 Baseada em Cadeias de Caracteres

As medidas de similaridade baseadas em *string* operam sobre sequências e composição de caracteres e são subdivididas em análise baseadas em caractere e em termo. Neste trabalho apenas a primeira abordagem é utilizada. Dessa forma, as medidas baseadas em cadeia de caractere comparam a distância entre dois textos por aproximação e comparação de igualdade. Um exemplo de implementação destas medidas em linguagem Java é o *SimMetrics*[15].

Algumas das principais funções baseadas em caractere são listadas a seguir:

- *Longest Common SubString* - considera a similaridade baseado no tamanho da maior troca contínua de caracteres existentes em ambas *strings*;
- *Levenshtein* - conta o número mínimo de operações necessárias para transformar uma palavra em outra. Uma operação pode ser definida por inserção, remoção ou substituição;
- *Jaro* - se baseia na posição e número de caracteres em comum entre palavras;
- *N-gram* - este conceito é uma subsequência de n itens para uma dada sequência de texto. O algoritmo compara esses *n-grams* para cada caractere ou palavras em duas *strings*. O resultado é gerado dividindo o número de *n-grams* similares pelo número máximo de *grams*.

[15] https://github.com/Simmetrics/simmetrics

Neste trabalho, como a análise de texto é feita sobre palavras curtas, a medida que melhor se aplica é a de Levenshtein (LEVENSHTEIN, 1966). Esta fornece um valor de distância calculada entre duas *strings* através do número de operações necessárias para transformar uma palavra na outra.

Como o resultado de Levenshtein é um número de operações necessárias para se chegar a outra *string*, é necessário aplicar a Equação (2.1) dada por (DORNELES, 2010) para gerar um escore de similaridade entre 0 e 1, normalizando, assim, o valor da distância, onde 0 (zero) representa totalmente diferente e 1 (um) significa idêntico.

A Equação (2.1) consiste em pegar o valor resultante da função $Levenshtein$ em *string* $s1$ e *string* $s2$ dividido pelo maior tamanho dentre as duas palavras $s1$ e $s2$. Por fim, o resultado é subtraído de 1.

$$\mathrm{SimLev} = 1 - \frac{Levenshtein\,(\mathrm{s1}, \mathrm{s2})}{max\,(size\,(s1)\,, size\,(s2))} \tag{2.1}$$

Um exemplo de aplicação da Equação 2.1 é apresentada no capítulo seguinte.

2.2.1.2 Baseada em Conhecimento

Esta é uma medida semântica que se baseia no grau de similaridade conforme o significado da informação em uma rede semântica (GOMAA; FAHMY, 2013). Uma das ferramentas mais populares neste sentido é o *Wordnet* (MILLER, 1995). Este é um vasto banco de dados léxico da língua inglesa com nomes, pronomes, verbos, adjetivos agrupados em conjuntos de sinônimos cognitivos. Um exemplo de implementação em linguagem Java é o WS4J[16].

Existem diferentes medidas que calculam a similaridade semântica. Um exemplo de medida é (*i*) *Resnik* (RESNIK, 1995) que, baseado em uma estrutura de hierarquia, utiliza o conteúdo informacional do ancestral comum mais informativo. Essa medida não leva em consideração a distância do termo até seu ancestral comum. Outras medidas, como (*ii*) *Lin* e (*iii*) *Jiang e Conrath* utilizam o conteúdo informacional do ancestral comum com o termo comparado.

Para estas medidas é aplicado a concepção de *Lowest Common Subsumer* (LCS) dos nós. Por exemplo, dados dois nós de conceito C1 e C2 em uma hierarquia do tipo "é-um", o LCS é definido como o nó mais específico que ambos compartilham como um antepassado. Por exemplo, se C1 era “carro” e C2 era “barco”, então o LCS seria “veículo” (FERNANDO;

[16] https://github.com/Sciss/ws4j

STEVENSON, 2008), isto é, o nó antepassado mais próximo que ambos possuem em comum.

Resnik. A medida de *Resnik* usa o conteúdo de informação (IC - *Information Content*) do LCS em dois conceitos, dada pela Equação (2.2).

$$\text{SimRes} = IC(LCS(C1, C2)) \tag{2.2}$$

O conteúdo de informação de um nó é uma estimativa de quão informativo é este. Frequentemente, os conceitos são considerados de baixo conteúdo informativo e são raras as ocorrências de alto conteúdo informativo.

Lin. A medida de Lin (LIN, 1998) baseia-se na medida Resnik, porém o normalizando através da aplicação do conteúdo de informação dos dois nós conforme Equação (2.3).

$$\text{SimLin} = \frac{2 * IC(LCS(C1, C2))}{IC(C1) + IC(C2)} \tag{2.3}$$

Como exemplo, ao testar os termos "car" e "boat" na implementação WS4J[17], o LCS encontrado é o termo "vehicle", gerando os valores $5,5313$ para Resnik e $0,7198$ para Lin. No site disponibilizado no rodapé também é possível visualizar o detalhamento do cálculo.

Estas são as principais medidas de similaridade semântica que se baseiam na informação do conteúdo, sendo que a medida de Lin é utilizada por este trabalho seguindo a implementação do *WordNet*.

2.2.1.3 Extração de Radicais

Esta técnica é aplicada também no processo de identificação de equivalências. Por exemplo, duas palavras que apresentam o mesmo radical são mapeadas para um mesmo conceito. Neste contexto, é considerada como técnica para similaridade de texto ao aplicar o valor 1 (um), para aquelas que possuírem radicais idênticos, e 0 (zero) para os radicais diferentes.

A área de recuperação de informação também utiliza várias técnicas. Um exemplo de técnica consolidada é o *stemming*. Esta pode ser categorizada em removedores de afixos e consiste em reduzir as palavras ao seu radical.

Segundo (COELHO, 2007), *Stemmers* são definidos por algoritmos ou programas que tem por objetivo reduzir as palavras a uma forma comum de representação através do processo de combinar formas morfologicamente variantes de um termo. Não precisa ser uma palavra

[17] http://ws4jdemo.appspot.com/?mode=w&s1=&w1=car&s2=&w2=boat

válida, apenas captar o significado desta. Como se trata de uma análise gramatical, ele depende do idioma para o qual é desenvolvida.

Para o português, por exemplo, existe o RSLP (Removedor de Sufixos da Língua Portuguesa) (ORENGO; HUYCK, 2001). Este é composto de uma sequência de passos que objetivam reduzir a palavra ao seu radical (*stem*). Dentre esses passos pode-se citar: redução de plural, advérbio, feminino, aumentativo, substantivo, verbos e acentos. Neste caso, há algumas dificuldades geradas pelo idioma como exceções, palavras com mesma grafia e significados diferentes, nomes próprios, etc.

De modo semelhante, no inglês destaca-se *Porter Stemmer Algorithm* (PORTER, 2006). Este também efetua a remoção de sufixos, prefixos e baseia-se em regras que são aplicadas se determinadas condições forem satisfeitas. Existem diversas implementações, como por exemplo, em linguagem Java[18].

O uso desta técnica neste trabalho visa comparar a semelhança de palavras através da equivalência de seu radical. No processo proposto, esta técnica será aplicada em conjunto com as demais para identificar a similaridade.

2.2.2 Processo de Tomada de Decisão

Para auxiliar no processo de tomada de decisão quanto à equivalência a partir das diferentes técnicas de similaridade, foi adotado o método denominado AHP (em português, Processo Analítico Hierárquico). Este tem por objetivo encontrar uma solução adequada com base em várias alternativas para um dado problema. Neste trabalho, o método é executado para ponderar as principais técnicas de equivalência aplicadas, definindo uma hierarquia de relevância.

Conforme (SAATY, 2008) o método AHP é composto pelos seguintes passos:

1. Definir o problema e os atributos.

   Para exemplificação, neste contexto, os atributos são Sinônimos (A), Radical (B) e Similaridade de caractere (C).

2. Estruturar a hierarquia de decisão definindo os pesos que cada critério deve ter em relação aos demais. Para este passo, é dada a Tabela 2.2.

   Dado os critérios A, B e C, pode-se então definir uma ordem de importância:

[18] http://tartarus.org/martin/PorterStemmer/java.txt

Tabela 2.2 – Valores numéricos para cada importância relativa - Fonte: (SAATY, 2008)

1	Mesma importância
2	Levemente mais importante
3	Fracamente mais importante
4	Fracamente a moderadamente mais importante
5	Moderadamente mais importante
6	Moderadamente a fortemente mais importante
7	Fortemente mais importante
8	Muito mais importante
9	Absolutamente mais importante

- Sinônimos (A) é levemente mais importante que radical (B) (valor 2);
- Similaridade de caractere (C) é fracamente mais importante que sinônimos (A) (valor 3);
- Similaridade de caractere (C) é fracamente a moderadamente mais importante que radical (B) (valor 4);

3. Organizar os critérios em uma matriz.

   Seguindo o mesmo exemplo, dado os itens A, B e C, com três atributos, a matriz será do tipo 3x3, onde o valor das linhas e colunas é a relação numérica dada a Tabela 2.2. Uma demonstração é feita na Figura 2.3.

$$M = \begin{matrix} \overbrace{\phantom{\begin{bmatrix}1&2&1/3\end{bmatrix}}}^{A\quad B\quad C} \\ \begin{bmatrix} 1 & 2 & 1/3 \\ 1/2 & 1 & 1/4 \\ 3 & 4 & 1 \end{bmatrix} \end{matrix} \begin{matrix} A \\ B \\ C \end{matrix}$$

Figura 2.3 – Matriz de exemplo de critérios - Fonte: adaptado de SAATY(2008)

4. Gerar o vetor que irá determinar o peso de cada critério.

   Para tal, calcula-se a matriz quadrada dada a matriz anterior, soma-se as linhas e divide-se o total da soma de todas as linhas pela soma de cada linha.

   Dessa forma, é gerado o vetor que normaliza o peso de cada critério como exemplificado na Figura 2.4. Esta etapa repete-se até que a iteração corrente apresente o mesmo resultado que a iteração anterior com precisão de quatro dígitos, por exemplo.

$$M = \overset{\text{Matriz quadrada}}{\begin{bmatrix} 3.0000 & 5.3333 & 1.1666 \\ 1.7500 & 3.0000 & 1.7500 \\ 8.0000 & 14.000 & 3.0000 \end{bmatrix}} = \overset{\text{Soma das linhas}}{\begin{bmatrix} 9.4999 \\ 6.5000 \\ 25.0000 \end{bmatrix}} = \overset{\text{Normalizado}}{\begin{bmatrix} 4.3158 \\ 6.3076 \\ 1.6399 \end{bmatrix}}$$

$$\text{TOTAL} = 40.9999$$

Figura 2.4 – Gerando o vetor normalizado - Fonte: adaptado de SAATY(2008)

5. Montar a matriz que gera o *ranking* dos critérios.

   Nesta etapa, multiplica-se a matriz M da Figura 2.3 pelo vetor normalizado da Figura 2.4 conforme demonstrado na Figura 2.5.

$$M = \overset{\text{Matriz inicial}}{\begin{bmatrix} 1 & 2 & 1/3 \\ 1/2 & 1 & 1/4 \\ 3 & 4 & 1 \end{bmatrix}} X \overset{\text{Normalizado}}{\begin{bmatrix} 4.3158 \\ 6.3076 \\ 1.6399 \end{bmatrix}} = \overset{\text{Ranking}}{\begin{bmatrix} 17.4776 \\ 8.8754 \\ 39.8177 \end{bmatrix}}$$

Figura 2.5 – Gerando ranking dos critérios - Fonte: adaptado de SAATY(2008)

Conforme representado no *ranking* ao final da Figura 2.5, o critério de similaridade de caractere (C) é o mais relevante pois possui valor maior que os demais. A seguir o critério de sinônimos (A) como segundo mais relevante e por fim o critério de radical (B) é o menos relevante.

Este resultado é utilizado nos estudos de caso realizados no Capítulo 4 onde é aplicado o método AHP. Dessa forma, aplicando o peso de valor um para a técnica de radical, peso de valor dois para técnica de sinônimos e peso de valor três para similaridade de caractere.

2.3 Trabalhos Relacionados

Desde o surgimento do paradigma NoSQL em 2009 (SADALAGE; FOWLER, 2012), a forma de pensar sobre esquema mudou. Estes novos modelos de dados propõem agilidade ao desenvolvimento pela escalabilidade horizontal e flexibilidade de esquema. Isto gerou uma oportunidade de pesquisa sobre a organização desses bancos de dados. Pensar sobre esquemas pode ser uma contradição em NoSQL (ATZENI, 2015), porém mesmo que não haja uma estrutura explícita, existe um esquema implícito e dinâmico que acompanha a evolução do banco de dados.

Dentre as oportunidades de pesquisa destacam-se três tipos de abordagens: (*i*) que se

preocupam com a modelagem conceitual desse novo paradigma (BUGIOTTI et al., 2014), (VARGA; JÁNOSI-RANCZ; KÁLMÁN, 2016), (*ii*) que procuram gerenciar a evolução do esquema (SCHERZINGER; CERQUEUS; ALMEIDA, 2015) e (*iii*) que visam extrair esta estrutura a partir dos dados ou da base de dados.

Esta última abordagem é apropriada para a flexibilidade de esquemas do modelo NoSQL, pois extrai o esquema presente nos dados em determinado momento sem se preocupar necessariamente com modelagem ou gerenciamento do mesmo. O presente trabalho se concentra na abordagem (*iii*), que é detalhada a seguir.

Alguns trabalhos trazem uma proposta mais geral com relação ao NoSQL com foco em um sistema de persistência poliglota (CASTREJÓN et al., 2013), (SHARP et al., 2013), isto é, com mais de um modelo de persistência ou para dados semiestruturados (NESTOROV; ABITEBOUL; MOTWANI, 1997). Contudo, a dissertação tem foco em fontes de dados no formato JSON, bem como as informações armazenados em NoSQLs orientados a documentos. Portanto, são consideradas também, abordagens que visam extrair uma estrutura diretamente de arquivos em formato JSON (IZQUIERDO; CABOT, 2013), (IZQUIERDO; CABOT, 2016), (MITSCH et al., 2012). Dessa forma os trabalhos diretamente relacionados são aqueles que visam a extração de esquemas em um NoSQL orientado a documentos e/ou apenas armazenados em JSON (KLETTKE et al., 2015), (RUIZ; MORALES; MOLINA, 2015) e (GARG; KAUR, 2015).

Em (KLETTKE et al., 2015) é proposto um algoritmo que utiliza medidas de similaridade para apresentar um conceito semelhante a "cardinalidade ". Esta é do tipo $n : Z$ (onde n é o número de ocorrências e Z o total de documentos). Através desta medida são inferidas propriedades obrigatórias e opcionais.

De outra forma (RUIZ; MORALES; MOLINA, 2015) utiliza um processo de engenharia reversa com base em modelo dirigido a dados para inferir versões de esquemas. Já (GARG; KAUR, 2015) desenvolve uma GUI (*Graphical User Interface*) integrada com *MongoDB Inspector* para visualização do esquema.

Com foco apenas em dados JSON (IZQUIERDO; CABOT, 2013) apresenta um processo composto de 3 fases: (1) pré-descoberta, (2) descoberta de serviço-único e (3) descoberta multi-serviço. A implementação deste pode ser vista em *JSONDiscover* (IZQUIERDO; CABOT, 2016), porém voltado para *API Web Services*.

Esta dissertação apresenta um processo para extração de esquemas em banco de dados

NoSQL orientado a documentos armazenados em formato JSON e diferencia-se por tratar campos que possuam escrita diferente, mas que representam a mesma informação no esquema. Isto é realizado através de técnicas que identificam equivalência em nomes de campos no documento, abordagem que não é tratada pelos demais trabalhos.

Dessa forma, o processo tem como entrada uma coleção de documentos de mesmo domínio e gera como saída uma representação conceitual unificada do esquema implícito presente no banco de dados.

De modo a melhor visualizar o que foi exposto, a Tabela 2.3 mostra um comparativo dos principais trabalhos relacionados à extração de esquema que utilizam o formato JSON, seja armazenado em NoSQL ou não como na proposta de (IZQUIERDO; CABOT, 2013).

Tabela 2.3 – Comparativo dos trabalhos relacionados

	Izquierdo & Cabot, 2013	**Klettke, Störl, Scherzinger, & Regensburg, 2015**	**Ruiz, Morales, & Molina, 2015**	**Proposta**
Origem	APIs de serviço *web*	Dataset do MongoDB	MongoDb, CouchDB e Hbase	Dataset NoSQL documento
Metodologia	Processo em 3 fases	Algoritmo	Engenharia reversa	Processo em 4 fases
Particularidade	Aplicado a serviços web	Detecta outliers em documentos, como campos obrigatórios e opcionais	Utiliza versões	Similaridade de texto
Modelo de saída	Modelo de domínio da Aplicação (ECORE)	JSON Schema	Meta-modelo de esquema NoSQL	Esquema conceitual

De modo geral, todos apresentam origem em JSON, com metodologia composta por várias fases, seja por processo, algoritmo ou aplicando engenharia reversa. Este trabalho difere-se dos demais ao utilizar técnicas de similaridade de texto, possibilitando uma representação unificada para conjuntos de documentos do mesmo domínio que porventura apresentem termos sinônimos ou palavras semelhantes em caractere e radical. Como saída destaca-se por apresentar um esquema conceitual em notação apropriada ao modelo NoSQL, diferente dos demais.

2.4 Considerações Finais

Este capítulo apresentou alguns conceitos e técnicas ligadas ao desenvolvimento do processo de extração de esquemas proposto. Foi apresentada a categoria de modelos NoSQL orientados a documento. Seus principais formatos de armazenamento são XML e JSON.

Neste processo, a gramática JSON tem papel importante. É através dela que serão validados os documentos para entrada, assim como geradas as regras para conversão do modelo conceitual. Uma alternativa para a representação dos mapeamentos resultantes dessa consolidação dos campos é o uso de expressões JSONPath.

Apesar da proposta utilizar apenas um formato, existem ferramentas de conversão de XML para JSON que podem ser aplicadas antes do início do processo. Dessa forma a proposta se torna abrangente aos dois principais formatos.

Foram citados também alguns conceitos ligados à esquemas em NoSQL, assim como formas para representação visual de bancos de dados orientados a agregados. Para este último destaca-se (JOVANOVIC; BENSON, 2013) que utiliza uma adaptação da notação IDEF1X baseada em um elemento raiz a partir do qual se constrói o modelo.

A seguir foram descritas as técnicas de identificação de equivalência aplicadas neste processo. Inicialmente, foram relatadas algumas medidas das funções de similaridade de texto que podem ser classificadas quanto a análise caractere por caractere ou baseada no significado que dada palavra representa.

Outra técnica a agregar na identificação de equivalência é a *Stemming*. Ela analisa a palavra morfologicamente, extraindo seu radical através da remoção de afixos de determinado idioma.

Para que se possa definir uma hierarquia de relevância das técnicas aplicadas, foi utilizado o método de apoio a tomada de decisões AHP. Para cada caso a ser aplicada a proposta, determinada técnica pode ter mais ou menos influência em relação as demais. Assim são definidas relações iniciais e o método gera um *ranking* como saída.

Dentre os trabalhos relacionados destacam-se aqueles que visam extrair uma estrutura implícita de dados em formato JSON armazenados em um banco de dados NoSQL orientado a documentos. A proposta de (IZQUIERDO; CABOT, 2016) trabalha com o mesmo formato, porém, voltada para *web services*. Já (GARG; KAUR, 2015) apresenta uma abordagem específica para a solução MongoDB. No trabalho de (RUIZ; MORALES; MOLINA, 2015) inferem-se ver-

sões de esquemas. Por fim, (KLETTKE et al., 2015) apresenta uma solução mais completa de extração envolvendo NoSQL com detecção de possíveis *outliers*, isto é, campos que aparecem apenas em alguns poucos documentos.

Esta proposta se diferencia no sentido de apresentar um processo de extração que abrange análise de similaridade de texto para possíveis diferenças de escrita em nomes de campos que indiquem a mesma informação. Esta também consolida o modelo em uma representação conceitual armazenando os mapeamentos em expressões JSONPath.

3 PROCESSO DE EXTRAÇÃO DE ESQUEMAS CONCEITUAIS

Neste capítulo é descrito o processo proposto para extração do esquema implícito em bancos de dados NoSQL orientados a documento. Seu objetivo é, a partir de uma base de dados existente no formato JSON, extrair um esquema unificado. Deste modo, são aplicadas técnicas para equivalência de texto para eliminar possíveis diferenças de escrita em campos que se referem à mesma informação.

Este processo é composto por quatro etapas principais que são descritas no decorrer do capítulo: (1) pré-processamento, (2) análise de similaridade, (3) identificação de equivalências e (4) representação da estrutura. Como saída, ele gera um esquema conceitual representado em uma notação adaptada aos bancos de dados NoSQL juntamente com os mapeamentos dos termos consolidados para seus equivalentes.

3.1 Visão Geral

Um banco de dados NoSQL orientado a documentos é composto por várias coleções. Para cada coleção podem existir vários documentos. Cada documento é um registro em formato JSON que pode conter campos aninhados como objetos e *arrays*. Cada registro possui um identificador único que é gerado pelo SGBD ou que pode ser atribuído via código no momento da criação.

O objetivo geral deste trabalho é extrair o esquema implícito presente no banco de dados, gerando uma representação conceitual. Devido à flexibilidade de esquema, campos que representam a mesma informação podem estar escritos de forma diferente. Para consolidar o esquema, estas diferenças devem ser eliminadas.

Desse modo, como entrada o processo recebe documentos pertencentes à uma coleção de mesmo domínio, armazenada em NoSQL no formato JSON. Como saída gera a representação conceitual de um esquema unificado. Para tratar os campos, são aplicadas técnicas para identificação de similaridade baseadas em caractere, conhecimento e extrator de radicais. Estas técnicas combinadas apontam a equivalência, sendo que quando um campo é consolidado, seu mapeamento da estrutura unificada para o documento original é armazenado.

Para exemplificar o processo proposto é utilizado o conjunto de dados disponibilizado para testes[19] no formato JSON. Esse é extraído através da API do *Twitter* que produz infor-

[19] https://dev.twitter.com/rest/reference/get/statuses/user_timeline

mações sobre usuários e seus *Tweets*. A Listagem 3.1 ilustra dois documentos com pequenas adaptações em sua estrutura que serão empregados para demonstração ao longo deste capítulo. Observa-se que os arquivos de entrada necessitam ser válidos pela gramática.

```
[{
        "coordinates": null,
        "favorited": false,
        "truncated": false,
        "created_at": "Wed Aug 29 17:12:58 +0000 2012",
        "id_str": "240859602684612608",
        "entity": {
                "urls": [{
                        "expanded_url": "https://dev.twitter.com/
                           blog/twitter-certified-products",
                        "url": "https://t.co/MjJ8xAnT",
                        "indices": [
                                52,
                                73
                        ],
                        "display_url": "dev.twitter.com/blog/
                           twitter-c"
                }],
                "hashtags": []
        },
        "in_reply_to_user_id_str": null,
        "text": "Introducing the Twitter Certified Products
           Program: https://t.co/MjJ8xAnT",
        "retweet_count": 121,
        "in_reply_to_status_id_str": null,
        "id": 240859602684612608,
        "in_reply_to_user_id": null,
        "place": null
}, {
        "coordinates": null,
        "favorited": false,
        "truncated": false,
        "at_created": "Sat Aug 25 17:26:51 +0000 2012",
        "id_str": "239413543487819778",
        "entities": {
                "urls": [{
                        "expanded_url": "https://dev.twitter.com/
                           issues/485",
                        "url": "https://t.co/p5bOzH0k",
                        "indices": [
                                97,
                                118
                                ],
                        "display_url": "dev.twitter.com/issues/485
                           "
                }],
                "hashtags": [],
```

```
43                  "user_mentions": []
44          },
45          "in_reply_to_user_id_str": null,
46          "text": "We are working to resolve issues with application
                management & logging in to the dev portal: https://t.
               co/p5bOzH0k ^TS",
47          "retweet_count": 105,
48          "in_reply_to_status_id_str": null,
49          "id": 239413543487819778,
50          "in_reply_to_user_id": null,
51          "spot": null
52  }]
```

Listagem 3.1 – Exemplo de documentos de entrada do processo

Ao analisar o código, observa-se que os documentos (linhas 1 a 26 e 26 a 52) possuem estruturas semelhantes, porém não idênticas. Identificam-se pequenas diferenças de escrita que visam ser tratadas pelas técnicas propostas, como, por exemplo, "*entity*" (linha 7) e "*entities*" (linha 32) indicam diferença de plural.

Para melhor compreensão e desenvolvimento deste trabalho são apontadas algumas definições a seguir.

Definição 1. **Estrutura**. Estrutura se refere aos campos do documento JSON, no sentido de diferenciar o que não são dados.

Como exemplo, dado o Documento 1,"coordinates" (linha 2) e "favorited" (linha 3) representam parte da estrutura.

Definição 2. **Esquema**. Esquema tem um sentido mais abrangente do que estrutura, pois está geralmente relacionado a entidades, relacionamentos e atributos.

Como exemplo, uma descrição textual do esquema presente no intervalo das linhas 8 a 18 do Documento 1 é mostrada a seguir:

```
1 urls (expanded_url, url, display_url)
2 indices (num)
3 hashtags ()
```

Definição 3. **Bloco**. Um bloco é um conjunto de atributos dentro de símbolos delimitadores. Um bloco pode estar aninhado em outros.

Como exemplo, no Documento 1 um bloco é o intervalo das linhas 11 a 14 delimitadas por colchetes.

Definição 4. **Delimitadores**. Conjunto de símbolos da gramática JSON que marcam o início e fim de um objeto ou *array*.

Como exemplo, os objetos são indicados por chaves {} e *arrays* são indicados por col-

chetes []. De mesmo modo, um delimitador pode indicar o fim do nome de um campo, indicado por ponto-e-vírgula ';'.

Apresentadas estas definições, a seção a seguir descreve o processo proposto neste trabalho.

3.2 Processo Proposto

O processo de extração de esquema tem início dada uma única coleção e por fim gera uma representação conceitual própria a bancos de dados NoSQL. Para que este processo seja realizado, este é divido em 4 etapas principais, conforme exibido na Figura 3.1.

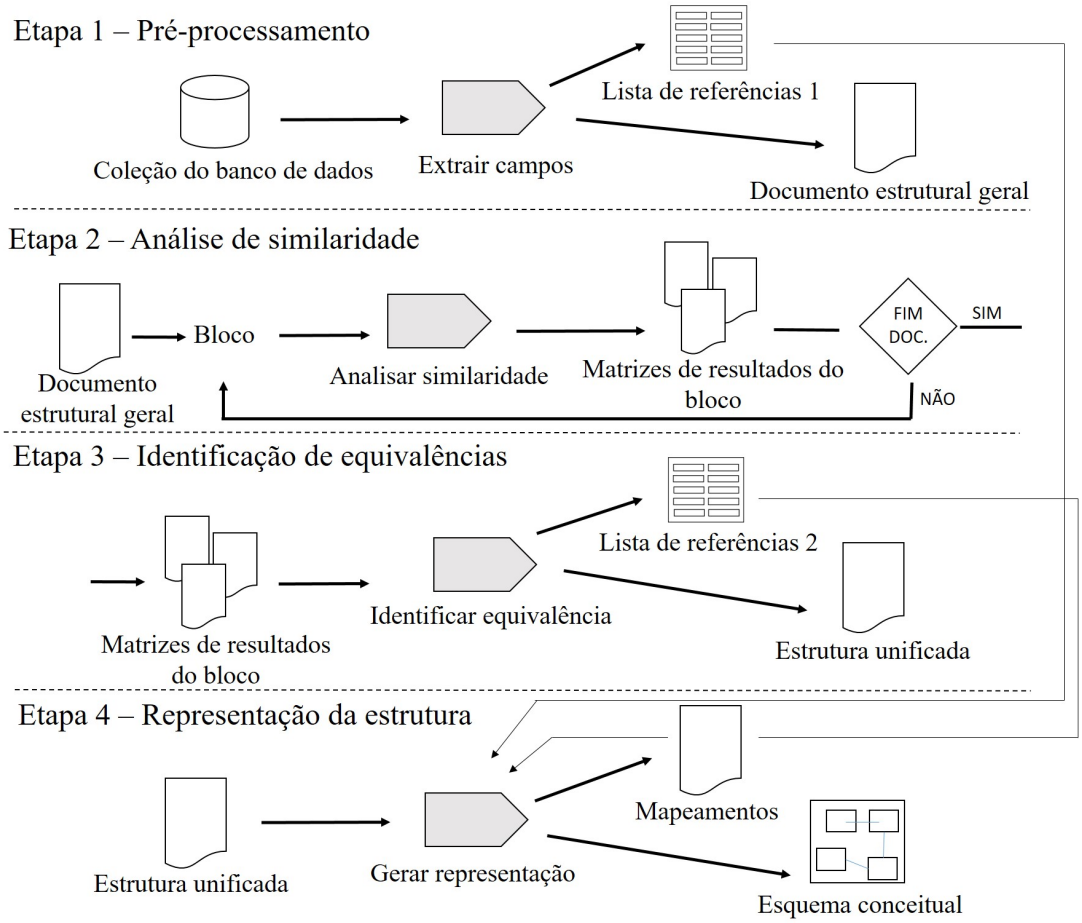

Figura 3.1 – Processo geral proposto

Estas etapas são brevemente descritas a seguir:

1. Pré-processamento. Esta etapa é executada entre os documentos da coleção. Tem por objetivo percorrê-los, transformando-os em um único documento. Neste processo são extraídos apenas os campos, armazenando-os para posterior tratamento. Para aqueles que apresentam

campos distintos do documento que está sendo unificado, é armazenada uma lista guardando a referência para seus respectivos documentos de origem.

Entrada: coleção do banco de dados NoSQL em formato JSON válido pela gramática.

Saída: apresenta dois artefatos, um documento único estrutural geral que agrega todos os campos distintos em um arquivo e uma lista de referências que guarda a origem dos campos.

2. Análise de similaridade. Esta etapa deve ser efetuada para cada bloco do documento. São necessárias iterações até atingir o fim do arquivo. Para cada bloco, são aplicadas as técnicas de similaridade baseadas em cadeias de caracteres e baseada em conhecimento, além do extrator de radicais para identificar equivalências de campos com escrita diferente, mas que representam a mesma informação.

Entrada: um bloco por iteração pertencente ao documento estrutural geral.

Saída: três matrizes de resultados por iteração, ou seja, uma para cada técnica aplicada nas análises par-a-par das palavras presentes no bloco.

3. Identificação de equivalências. Esta etapa é realizada por blocos. Este processo é responsável pela definição de equivalência ou não. Para tal, são cruzados os dados das três matrizes resultantes e calculado o resultado. Um por vez, todos os blocos são lidos até o fim do documento. Nesta atividade principal, a estrutura unificada é gerada junto a uma segunda lista de referências com equivalências em mais de um campo.

Entrada: três matrizes de resultados por cada bloco.

Saída: arquivo com a estrutura unificada e a segunda lista de referências.

4. Representação da estrutura. Esta etapa define algumas regras que percorrem a estrutura unificada com o intuito de inferir entidades, atributos e relacionamentos. Conforme o que foi identificado, é gerada uma representação conceitual do banco de dados NoSQL. Ao consultar as listas de referências geradas no processo, os mapeamentos são definidos.

Entrada: arquivo com estrutura unificada.

Saída: esquema conceitual da coleção presente no banco de dados NoSQL e os mapeamentos dos campos consolidados com seus respectivos correspondentes.

Cada uma das fases descritas é composta por uma atividade geral. Estas são detalhadas em subatividades para atingir o objetivo maior. As seguintes subseções detalham cada etapa e explicam as subatividades.

3.3 Pré-processamento

A etapa de pré-processamento, representada na Figura 3.2, tem por objetivo preparar os documentos para as etapas seguintes. Para tal, inicialmente todos os documentos são percorridos juntando todos os campos em um único, de modo que permaneçam apenas os campos distintos. Este processo visa diminuir o número de comparações a serem feitas de modo a melhorar a eficiência.

Dessa forma, a partir da coleção, cada documento é percorrido separando apenas os campos, ou seja, a parte $String$ do par $String : value$ JSON, reiterando que os dados são desconsiderados no contexto deste trabalho. Conforme forem surgindo termos distintos do documento unificado, estes são armazenados em uma lista de referências para possibilitar os mapeamentos futuramente.

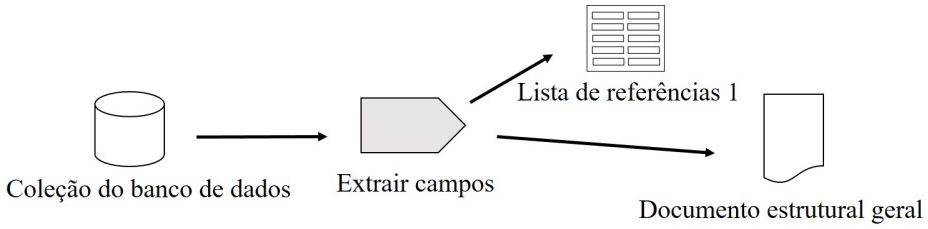

Figura 3.2 – Etapa de pré-processamento

Entrada. Um coleção existente armazenada em um banco de dados NoSQL orientado a documentos pertencente ao mesmo domínio. Cada documento deve ser validado pela gramática.

Atividade Extrair Campos. Esta atividade tem por finalidade percorrer os documentos separando apenas os campos distintos, mesclando em um único arquivo e guardando uma lista de referências. Esta atividade é detalhada em outras subatividades.

Saída. O primeiro artefato de saída é um arquivo no formato de texto contendo um único documento representando a coleção, chamado de documento estrutural geral. Este deve possuir, além dos delimitadores que auxiliarão nas etapas futuras, os campos com nomes distintos, mantida a hierarquia, isto é, o bloco correspondente. O segundo artefato é uma lista que contenha o termo e a referência a quais documentos originais o contém.

Exemplo. Considerando como entradas o Documento 1 e 2 apresentados na Listagem 3.1, a saída desta etapa é representada pela Listagem 3.2:

```
1 {
2         coordinates, favorited, truncated, created_at, at_created,
              id_str,
```

```
3           entity, entities {
4                   urls [{ expanded_url, url, indices [], display_url
                        }]
5                   hashtags[],
6                   user_mentions[]
7           }
8           in_reply_to_user_id_str, text, retweet_count,
9           in_reply_to_status_id_str, id, in_reply_to_user_id,
10          place, spot
11  }
```

Listagem 3.2 – Exemplo de saída da etapa pré-processamento

Dessa forma, são mantidos os campos e delimitadores de modo a eliminar os repetidos e guardar os distintos, procurando manter o bloco de origem. Parte da lista de referências para este exemplo, é observada na Tabela 3.1.

Tabela 3.1 – Exemplo parcial de saída da lista de referências da etapa pré-processamento

Termo	**Origem**
at_created	Doc2
created_at	Doc1
entities	Doc2
place	Doc1
spot	Doc2

Na Tabela 3.1 é representada parte da saída, porém, devem ser armazenados todos os termos distintos (neste caso, considera-se os Documentos 1 e 2 da Listagem 3.1) com a referência ao documento de origem.

A atividade principal é **extrair campos**. Esta é dividida em duas subatividades, conforme observado na Figura 3.3, especificadas nas subseções seguintes.

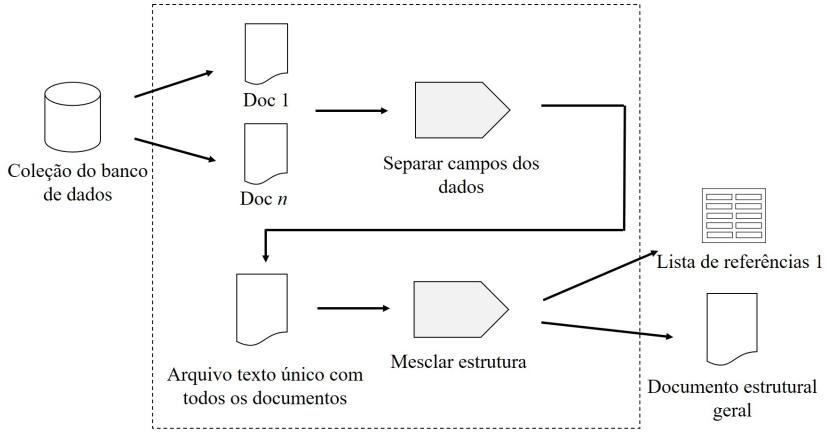

Figura 3.3 – Detalhamento da atividade extrair campos

Separar campos dos dados. Consiste em executar um processamento nos documentos mantendo apenas os nomes dos campos e os símbolos delimitadores. Isto é feito para cada documento, gerando um arquivo texto único. Nesta fase nenhuma comparação é realizada.

Mesclar estrutura. Nesta etapa, o intuito é de evitar a redundância dos elementos, de modo a reduzir o número de entradas para análise de similaridade textual na próxima etapa do processo. Assim sendo, ela executa testes e comparações para mesclar e manter apenas os campos distintos correspondentes a cada bloco. Neste processo, guarda-se a referência de origem dos campos distintos.

3.3.1 Separar Campos dos Dados

Esta atividade tem como objetivo percorrer cada documento JSON validado mantendo apenas os campos e delimitadores. A validação da gramática pode ser realizada de diferentes formas, como por exemplo, *online* com JSONLint[20].

Para esta tarefa há diferentes possibilidades:

- (*i*) Utilizar uma API para processamento do JSON. Como exemplo, pode-se citar a API Java JSR 353[21], que disponibiliza um *parser* para captar eventos e tratá-los, como *JSONObject* ou *JSONArray*.

- (*ii*) Utilizar uma ferramenta proprietária, como *Altova XMLSpy* que, a partir de um documento JSON, possibilita além de validar a gramática, e gerar o esquema deste. Ela possibilita também a visualização no formato de *grid* ou *schema*.

Devido a questões de licença, a abordagem (*ii*) não é empregada como recurso principal. No entanto, permanece como apoio para verificar e validar o processo realizado via programação.

Dessa forma, o método (*i*) é utilizado para a atividade visto que a biblioteca é de código aberto e acessível. Ela fornece um *parser* que trata eventos. O algoritmo desta tarefa consiste em percorrer o documento e quando os eventos de início e fim de objeto/*array* acontecerem, assim como eventos de nome do campo, guardá-los em um arquivo texto.

Entrada. Documentos pertencentes à mesma coleção em formato JSON.

Atividade. Descrita pelo Algoritmo 1.

[20] http://jsonlint.com/
[21] https://json-processing-spec.java.net/

Saída. Arquivo texto contendo todos os documentos e seus respectivos campos.

A atividade pode ser sintetizada no Algoritmo 1 que foi implementado em linguagem Java. Este processa o arquivo e, ao encontrar determinados eventos, mantém apenas o delimitador, apagando a parte que contém os dados.

Algoritmo 1: Atividade separar campos dos dados

Entrada: Documentos d pertencentes à mesma coleção
Saída: Arquivo .txt com todos os campos e delimitadores de cada d

```
1  início
2      para cada documento da coleção faça
3          selecione evento faça
4              caso inicio_array faça  grava [ ;
5              caso fim_array faça  grava ] ;
6              caso inicio_obj faça  grava { ;
7              caso fim_obj faça  grava } ;
8              caso nome faça  grava nome;
9          fim
10     fim
11     Grava arquivo texto;
12 fim
```

Exemplo. Como esta subatividade é realizada para cada documento, é demonstrado um exemplo com base no Documento 1 e 2 da Listagem 3.1. Sua saída correspondente é o seguinte arquivo texto representado pela Listagem 3.3:

```
1  [
2  {coordinates; favorited; truncated; created_at; id_str; entity;
3  {urls;
4  [
5  {expanded_url; url; indices;
6  []display_url; }]hashtags;
7  []}in_reply_to_user_id_str; text; retweet_count;
       in_reply_to_status_id_str; id; in_reply_to_user_id; place; }
8  {coordinates; favorited; truncated; at_created; id_str; entities;
9  {urls;
10 [
11 {expanded_url; url; indices;
12 []display_url; }]hashtags;
13 []user_mentions;
14 []}in_reply_to_user_id_str; text; retweet_count;
       in_reply_to_status_id_str; id; in_reply_to_user_id; spot; }
15 ]
```

Listagem 3.3 – Exemplo de arquivo texto único com todos os documentos

Dessa forma, esta saída representa a estrutura do JSON em um arquivo texto com os campos e delimitadores para os documentos de exemplo da Seção 3.1 (linhas 1 a 7 do Docu-

mento 1 e linhas 8 a 15 do Documento 2). Este artefato é encaminhado à atividade seguinte para mesclar os documentos em um único.

3.3.2 Mesclar Estrutura

Esta atividade tem por objetivo processar e reduzir os documentos a um único documento, eliminando aqueles campos duplicados. Ela é descrita como um procedimento programável em qualquer linguagem de programação. O objetivo é apenas percorrer um arquivo de texto mantendo os elementos distintos e os delimitadores.

Neste ponto é preciso escolher um modo de execução para o processo, no caso (1) é necessária a intervenção de um usuário com conhecimento do domínio, para definir qual bloco é correspondente a outro, em termos de estrutura hierárquica, ou (2) tratar tudo como se fosse um único bloco eliminando os delimitadores. Na abordagem (1), todo o processo, testes e comparações são executados para cada bloco em particular, apenas juntando-os novamente ao final. No modo (2), as análises e comparações são realizadas no documento como um todo, ignorando possíveis diferenças de níveis hierárquicos dentro dos documentos. O processo permanece o mesmo para ambos, apenas divergindo no modo como será executado.

Outro ponto a destacar diz respeito ao tratamento de palavras homônimas, isto é, palavras com a mesma grafia e pronúncia mas com significados diferentes, pois podem existir palavras com mesma escrita em blocos diferentes. Por exemplo, caso o termo “url” (linha 1) também estivesse singular como “url” (linha 2).

```
1 url [{
2          expanded_url, url, indices [], display_url
3          }]
```

Neste caso, o termo pode não indicar a mesma informação por estar em um nível distinto dentro do arquivo JSON. Uma abordagem simplista para evitar unificar palavras que não indiquem a mesma informação é unificar apenas aquelas que possuam um mesmo nó pai, fato que é garantido na abordagem (1) ao executar o processo apenas a cada bloco interno, mantendo a hierarquia.

Contudo, o escopo do trabalho é limitado a uma coleção do banco de dados por vez. Conforme indicado na Seção 2.1.3, cada coleção deve ser organizada de modo a conter os dados que devem ser acessados juntos, ou seja, do mesmo contexto, favorecendo escritas e leituras. Portanto, considera-se, na abordagem (2), que todos os campos podem ser equivalentes não sendo tratadas ocorrências de homônimos.

Entrada. Arquivo texto contendo todos os documentos e seus respectivos campos.

Atividade. Descrita pelo Algoritmo 2.

Saída. Documento estrutural geral e lista de referências 1.

Este documento estrutural geral é a mescla de campos distintos de todos os documentos pertencentes à coleção de entrada. Ele tem formato de texto e não é um arquivo JSON válido.

Algoritmo 2: Atividade mesclar estrutura

Entrada: Arquivo .txt com todos os campos e delimitadores de cada d
Saída: Arquivo .txt com único documento d contendo os campos distintos, lista de referências 1

```
1  início
2      Carrega arquivo texto;
3      Considera doc1 como consolidado;
4      Adiciona doc1 na lista de referências;
5      enquanto houver próximo documento faça
6          doc_atual = documento;
7          para cada campo_doc_atual faça
8              para cada campo_doc_consolidado faça
9                  se campo_doc_atual é igual campo_doc_consolidado então
10                     Adiciona referência do doc_atual na lista de referências para o
                        campo_consolidado
11                     Próximo campo_doc_atual
12                 fim
13                 senão
14                     se doc_consolidado chegou ao fim então
15                         Adiciona campo_doc_atual ao doc_consolidado
16                         Adiciona referência do novo campo consolidado na lista de
                            referências
17                     fim
18                 fim
19             fim
20         fim
21     fim
22 fim
```

O Algoritmo 2 apresenta um laço de comparações em que é testado cada elemento atual com aqueles já presentes na estrutura única, isto é, cada campo do documento atual é comparado com os campos do documento consolidado. Os testes são realizados dentro do mesmo bloco com comparações do tipo "todos com todos". Caso o campo já exista no arquivo consolidado, nada acontece e passa-se para o próximo campo. Se até o fim do documento não for encontrada nenhuma ocorrência, este campo é adicionado ao arquivo consolidado e sua referência ao documento de origem é guardada em uma lista de referências.

Exemplo. Dando sequência ao processo a partir do exemplo da Listagem 3.1, a saída gerada ao final desta atividade é dada pela Listagem 3.4.

```
1  {
2          coordinates, favorited, truncated, created_at, at_created,
               id_str,
3          entity, entities {
4            urls [{ expanded_url, url, indices [], display_url }]
5                          hashtags[],
6                          user_mentions[]
7                  }
8          in_reply_to_user_id_str, text, retweet_count,
9          in_reply_to_status_id_str, id, in_reply_to_user_id,
10         place, spot
11 }
```

Listagem 3.4 – Exemplo de documento estrutural geral

Este indica um único arquivo, denominado documento estrutural geral, que contém campos distintos mantendo o mesmo bloco. Ele é representado no formato texto e não indica um JSON válido conforme se observa nas linhas 1 a 11. A saída da lista de referências para este exemplo já é demonstrada na Tabela 3.1.

3.4 Análise de Similaridade

A etapa de análise de similaridade tem por objetivo, a partir dos campos distintos extraídos dos documentos JSON, aplicar diferentes técnicas de análise de similaridade de texto visando identificar palavras que representam a mesma informação, mas que por motivos diversos, como a falta de padronização, estejam com grafias diferentes.

Para isto as análises são feitas em cada bloco com três técnicas distintas aplicadas a cada par de palavras que o compõe. Este processo é repetido até o fim do documento geral. Cada análise gera três matrizes de resultados, conforme apontado na Figura 3.4.

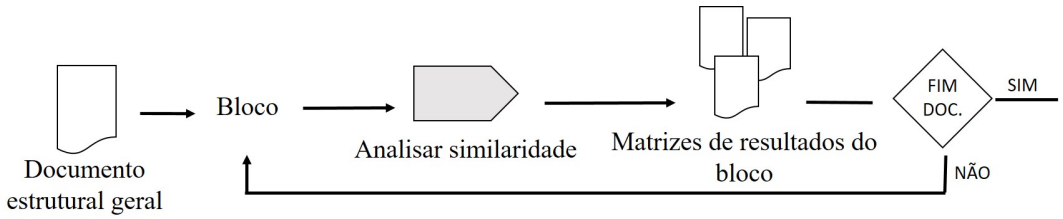

Figura 3.4 – Etapa de análise de similaridade

Entrada. Documento estrutural geral que é processado por blocos.

Atividade Analisar Similaridade. Nesta atividade, o arquivo de entrada é copiado e serve de artefato inicial para aplicação das três técnicas diferentes. Estas são executadas em paralelo. A entrada passa por cada técnica e gera matrizes de resultados com valores indicando a similaridade por bloco. Esta atividade é decomposta em outras subatividades.

Saída. Para cada bloco de entrada são geradas três matrizes de resultados.

Para esta etapa são dadas algumas observações:

- Dentro de um mesmo bloco, o pressuposto é de que todas as palavras possam ser consideradas equivalentes.
- As comparações são realizadas na forma todos com todos dentro de mesmo bloco.
- Se a palavra é única no bloco não há necessidade de se testar equivalência. Logo, este campo é consolidado.
- Os valores resultantes estão no intervalo de 0 (zero) a 1 (um), onde 0 indica totalmente diferente e 1 indica totalmente semelhante.

Exemplo. Considere o trecho de código a seguir retirado da saída da etapa de pré-processamento. A linha 1 apresenta um bloco delimitado pelos símbolos [{ e }].

```
1 urls [{ expanded_url, url, indices [], display_url }]
```

Aplicando, por exemplo, a técnica de extrator de radicais sobre este trecho de código os quatro radicais extraídos são:

```
1 expand_url, url, indic, displai_url
```

Logo, identifica-se que nenhum é equivalente e a saída é representada através de uma matriz mostrada na Tabela 3.2.

Tabela 3.2 – Exemplo de uma matriz de saída da etapa análise de similaridade

	expanded_url	url	indices	display_url
expanded_url	1	0	0	0
url	0	1	0	0
indices	0	0	1	0
display_url	0	0	0	1

O resultado é exibido em uma matriz de resultados do bloco. Esta contém informações sobre o resultado de determinada função de similaridade para cada par de palavras. Segundo

formalizado por (FERNANDO; STEVENSON, 2008), dada uma matriz de similaridade W, um elemento $w_{ij} \in W$ representa a similaridade das palavras p_i e p_j de acordo com alguma medida. Se $w_{ij} == w_{ji}$ a matriz é também simétrica.

A saída correspondente é representada em uma matriz cuja diagonal principal será sempre 1 (um). Esta também apresenta a característica de ser simétrica tanto abaixo quanto acima da diagonal principal. No caso da técnica de extrator de radicais, os valores resultantes são sempre inteiros (0 ou 1), ou seja, se possui o mesmo radical (um) ou não (zero).

A atividade principal desta etapa é **analisar similaridade**. Esta é dividida em subatividades conforme detalhada na Figura 3.5. Nestas são aplicadas três técnicas diferentes executadas de modo paralelo:

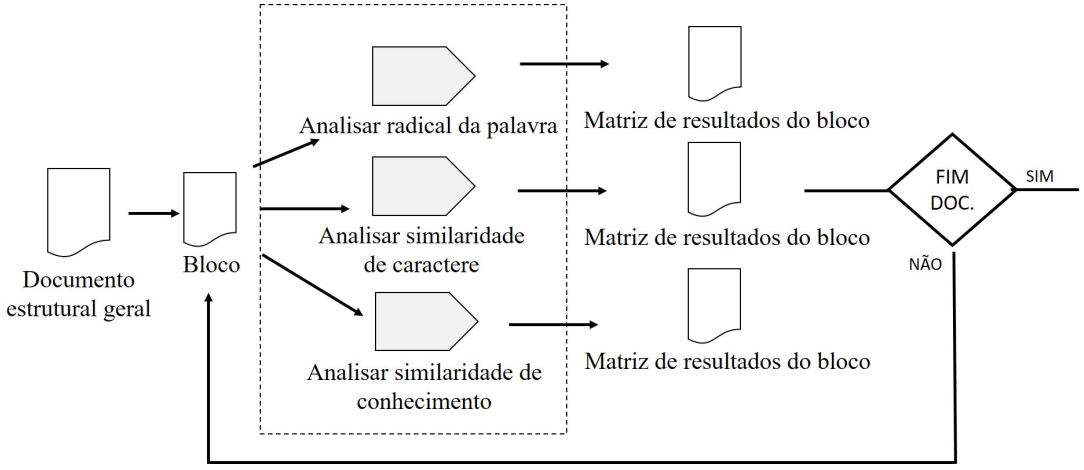

Figura 3.5 – Detalhamento da atividade analisar similaridade

Analisar radical da palavra. Esta subatividade diz respeito à técnica de *Stemming* (extrator de radicais). É executada com o algoritmo de (PORTER, 2006). Ao aplicar regras de remoção de sufixos e prefixos da língua inglesa, o resultado é um radical. Este não necessariamente é uma palavra e pode não possuir significado.

Analisar similaridade baseada em caractere. Consiste em aplicar alguma técnica descrita nesta categorização. Para testes foi escolhida a função de Levenshtein (LEVENSHTEIN, 1966) por se aplicar a textos curtos com base no número mínimo de operações necessárias para transformar uma *string* em outra. Essa função também é aplicada em verificadores ortográficos.

Analisar similaridade baseada em conhecimento. Visa o foco semântico. Para testes é utilizada a implementação *Wordnet Similarity for Java* por consistir em um dos maiores dicionários semânticos.

3.4.1 Analisar Radical da Palavra

Esta atividade tem por objetivo considerar o radical da palavra através do processo de *Stemming* que consiste em reduzir as palavras derivadas à sua base (*stem*). Para que esta técnica seja aplicada, é necessário que o nome do campo seja uma palavra válida na língua inglesa. Caso contrário não terá efeito. A implementação disponível em java[22] é utilizada para testes.

Entrada. Um bloco do documento estrutural geral.

Atividade Analisar Radical da Palavra. Esta atividade consiste em aplicar uma implementação de *stemming*, no caso o algoritmo de Porter. Após são realizadas as comparações para definir os valores. Esta subatividade é descrita pelo Algoritmo 3.

Saída. Matriz de resultados do bloco.

Algoritmo 3: Atividade analisar radical da palavra

Entrada: Arquivo .txt com campos e delimitadores de cada bloco b
Saída: Matriz de resultados de b

```
1  início
2      Carrega arquivo texto com apenas as palavras de b
3      Executa extrator de radicais para todas as palavras ∈ b
4      para cada radical ∈ b faça
5          Compara este com todos os outros;
6          se radical_a é igual radical_b então Matriz_ab = 1;
7          senão Matriz_ab = 0;
8      fim
9      Grava Matriz;
10 fim
```

Dessa forma, o Algoritmo 3 utiliza apenas a parte texto do arquivo, isto é, sem os delimitadores, para aplicar o extrator de radicais. Ao resultado gerado são efetuadas comparações dentro do mesmo bloco para ver se algum radical equivale. Caso sim, o valor gerado para aquele par de palavras é 1 (um), indicando total similaridade. Caso não, o valor gerado é 0 (zero), apontando nenhuma similaridade.

Exemplo. Considere o seguinte conjunto de palavras que é parte do documento estrutural geral representado pela Listagem 3.4.

```
1 coordinates; favorited; truncated; entity; entities;
```

A Figura 3.6 representa o resultado ao executar a implementação de Porter sobre este conjunto de palavras, isto é, as reduz à sua base gramatical.

[22] https://tartarus.org/martin/PorterStemmer/java.txt

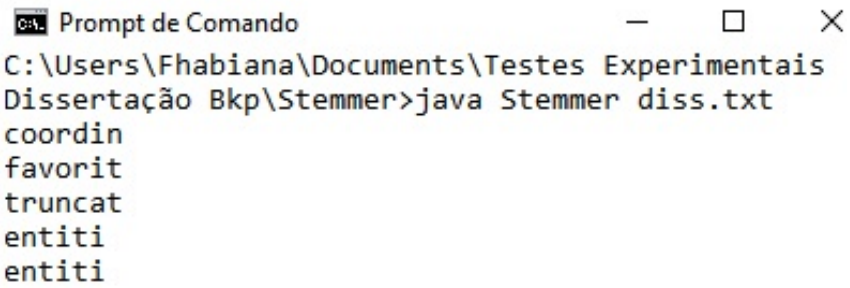

Figura 3.6 – Exemplo de saída ao aplicar o algoritmo de Porter

Outra questão diz respeito a que algumas palavras não são entradas válidas na língua inglesa. Assim, o radical permanece igual ao elemento de entrada, ou seja, não tem alterações ou ainda, são reduzidas partes de palavras existentes.

O próximo passo é realizar uma série de comparações que irá definir o valor para cada par de elementos Wij da matriz como equivalente (um) ou não (zero).

Tabela 3.3 – Exemplo de matriz de resultados ao aplicar Stemmer

	coordinates	favorited	truncated	entity	entities
coordinates	1	0	0	0	0
favorited	0	1	0	0	0
truncated	0	0	1	0	0
entity	0	0	0	1	1
entities	0	0	0	1	1

O resultado é exibido através da Tabela 3.3 que representa a matriz. Pode-se identificar que apenas os campos correspondentes às palavras "*entity*" e "*entities*" são equivalentes pois apresentam valor 1 (um).

3.4.2 Analisar Similaridade de Caractere

O objetivo desta atividade é analisar a palavra com o enfoque sintático, isto é, identificar pequenas variações na escrita ou caracteres trocados que correspondam a mesma palavra. De modo diferente da técnica anterior, para este procedimento não é necessário que a palavra seja uma palavra válida na língua inglesa.

A função aplicada é a de Levenshtein (LEVENSHTEIN, 1966), uma das funções categorizadas como Distância de Edição. Esta escolha se justifica por ser uma função de similaridade para textos curtos e atômicos baseada em caractere que testa o número de operações necessárias para transformar um palavra em outra. Para testes é utilizada a implementação em java do

pacote SimMetrics[23].

Entrada. Um bloco do documento estrutural geral.

Atividade Analisar Similaridade Baseada em Caractere. Esta atividade é executada a cada par de palavras e consiste em aplicar uma implementação de função de similaridade, no caso, o algoritmo de Levenshtein. Como esta atividade gera um valor não padronizado, o próximo passo é aplicar uma equação para gerar um valor no intervalo de 0 a 1 que indique a semelhança. Esta subatividade é descrita pelo Algoritmo 4.

Saída. Matriz de resultados do bloco.

Algoritmo 4: Atividade analisar similaridade baseada em caractere

```
Entrada: Arquivo .txt com campos e delimitadores de cada bloco b
Saída: Matriz de resultados de b
1  início
2      Carrega arquivo texto com apenas as palavras de b em um vetor
3      para cada linha a da Matriz faça
4          para cada coluna b da Matriz faça
5              Executa Levenshtein para par de palavras a, b do vetor;
6              Executa equação para gerar valor de similaridade sim;
7              Matriz_ab = sim;
8          fim
9      fim
10     Grava Matriz
11 fim
```

O Algoritmo 4 detalha a sequência de passos necessárias à atividade. As palavras devem ser testadas "todas com todas" e para cada par é aplicada a medida seguida de uma equação para padronizar o resultado. Este valor resultante representa o grau de similaridade entre estas.

Exemplo. Considere o seguinte conjunto de palavras que é parte do documento estrutural geral representado pela Listagem 3.4.

```
1 created_at; at_created; id_str; id;
```

Como demonstração, ao aplicar a função presente no pacote SimMetrics em "*created_at*" e "*at_created*" a saída produzida é exibida na Figura 3.7.

A Figura 3.7 representa o código-fonte em linguagem Java, com duas *strings* de entrada executando a função de Levenshtein implementada pelo pacote *SimMetrics*. A Tabela 3.4 sintetiza o resultado ao aplicar a medida de Levenshtein no conjunto de palavras de exemplo.

[23] https://github.com/Simmetrics/simmetrics

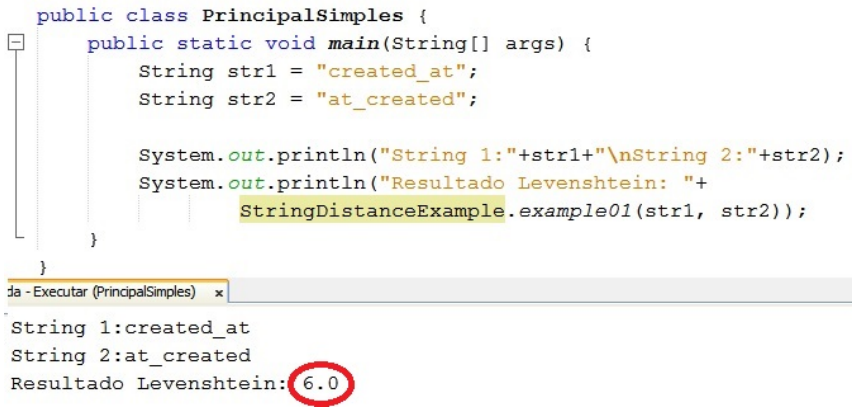

Figura 3.7 – Exemplo de saída ao aplicar o algoritmo de Levenshtein

Tabela 3.4 – Exemplo dos resultados ao aplicar o algoritmo de Levenshtein

	created_at	at_created	id	id_str
created_at	1	6.0	8.0	9.0
at_created	6.0	1	9.0	8.0
id	8.0	9.0	1	4.0
id_str	9.0	8.0	4.0	1

De acordo com a Tabela 3.4, para o par de palavras "*created_at*" e "*at_created*" o resultado de Levenshtein é igual a 6. Este valor está associado ao número de operações necessárias para transformar uma *String* em outra. Portando, o próximo passo é aplicar a fórmula para obter o valor de similaridade. Utilizando o exemplo da Figura 3.7, cujo resultado de $Levenshtein\,(\mathrm{s1}, \mathrm{s2}) = 6$, dadas as palavras "*created_at*" e "*at_created*" de tamanho $s1 = 10$ e $s2 = 10$, aplica-se a Equação 3.1:

$$\mathrm{SimLev} = 1 - \frac{6}{max\,(10, 10)} = 1 - \frac{6}{10} = 0.4 \tag{3.1}$$

Dessa forma, o valor de similaridade para estas duas *strings* é de 0.4. Esta normalização, proposta por (DORNELES, 2010), é aplicada de mesmo modo ao conjunto de exemplo, resultando na Tabela 3.5.

Tabela 3.5 – Exemplo de matriz de resultados ao aplicar equação SimLev

	created_at	at_created	id	id_str
created_at	1	0.4	0.2	0.1
at_created	0.4	1	0.1	0.2
id	0.2	0.1	1	0.4
id_str	0.1	0.2	0.4	1

Dessa forma, observa-se que os resultados apresentam valores baixos, isto é, indicam baixo grau de similaridade, pois quanto mais próximos de 1 (um) mais semelhantes.

3.4.3 Analisar Similaridade de Conhecimento

O objetivo desta atividade é identificar uma similaridade semântica entre as palavras, ou seja, aquelas que, mesmo com grafias diferentes podem ser consideradas sinônimas. Um ponto importante é de que a palavra deve ser válida na língua inglesa, pois se trata de um dicionário semântico.

A implementação utilizada para testes é o Wordnet:Similarity[24] que possui versão *web*[25] incluindo várias medidas. Algumas destas medidas pertencem à categoria que tem por base o conteúdo de informação (GOMAA; FAHMY, 2013), desse modo, diferenciando-se das demais que analisam o tamanho do caminho ou o grau de relação entre as palavras. Nesta categoria destacam-se: Resnik, Lin, Jiang e Conrath.

Para esta análise foi adotada a medida de Lin (LIN, 1998). A medida considera o conteúdo de informação de A e B analisando também o ancestral comum. Esta possui uma correlação maior com o juízo humano do que (RESNIK, 1995) e (WU; PALMER, 1994), além de seu resultado estar no domínio dos inteiros com intervalo de 0 e 1.

Entrada. Um bloco do documento estrutural geral.

Atividade Analisar Similaridade Baseada em Conhecimento. Esta atividade constitui-se em aplicar o Wordnet: Similarity para descobrir sinônimos. Para tal, é captado apenas o resultado de *lin*, tendo por base sua fórmula, que leva em consideração o ancestral comum e por já possuir um valor no intervalo de 0 a 1 que auxilia para posterior análise. Esta subatividade é descrita pelo Algoritmo 5.

Saída. Matriz de resultados do bloco.

O Algoritmo 5 detalha a sequência de passos para a subatividade analisar similaridade baseada em conhecimento. Ao executar esta implementação do *WordNet*, várias medidas são apresentadas. Nesse contexto aplica-se apenas o valor de Lin para montar a matriz de resultados.

Exemplo. Considere o seguinte conjunto de palavras que é parte do documento estrutural geral representado pela Listagem 3.4.

```
1 entity; entities; place; spot;
```

[24] https://github.com/Sciss/ws4j
[25] http://ws4jdemo.appspot.com/

Algoritmo 5: Atividade analisar similaridade baseada em conhecimento

Entrada: Arquivo .txt com campos e delimitadores de cada bloco b
Saída: Matriz de resultados de b

```
1 início
2     Carrega arquivo texto com apenas as palavras de b
3     para cada par de palavras faça
4         Executa Wordnet para as palavras a e b;
5         Matriz_ab = valor resultante de Lin;
6     fim
7     Grava Matriz
8 fim
```

Como demonstração é dada a Figura 3.8 na implementação do WordNet:Similarity.

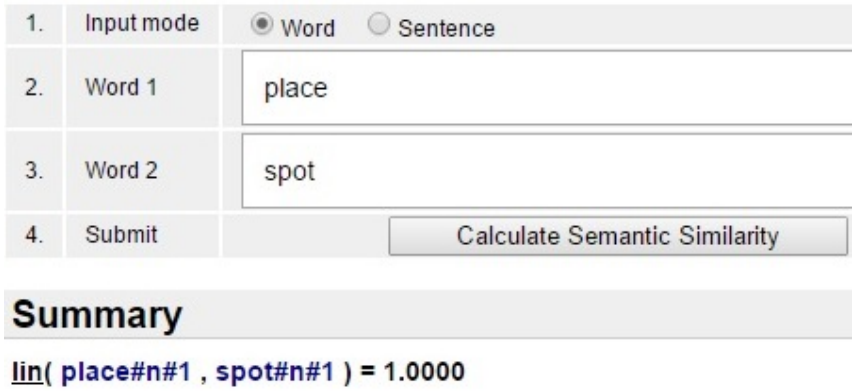

Figura 3.8 – Exemplo de saída ao aplicar o WordNet:Similarity

A Figura 3.8 apresenta uma tela do teste executado *online* com o par de palavras *place e spot*. Ambas são válidas na língua inglesa e conforme observado o valor resultante para medida de Lin é 1, indicando que são totalmente equivalentes. A Tabela 3.6 sintetiza o resultado ao aplicar a medida de Lin no conjunto de palavras de exemplo.

Tabela 3.6 – Exemplo dos resultados ao aplicar a medida de Lin

	entity	entities	place	spot
entity	1	-	0	0
entities	-	1	-	-
place	0	-	1	1
spot	0	-	1	1

Como Wordnet se trata de uma base de dados semântica, nem todas as palavras são encontradas, como no caso dos elementos representados com um traço (-) na Tabela 3.6. Para as palavras "*place*" e "*spot*" o valor 1 identifica total equivalência semântica.

3.5 Identificação de Equivalências

A etapa de identificação de equivalências demonstrada na Figura 3.9 é decisiva no processo de unificação do esquema. Seu objetivo é realizar testes e comparações a fim de definir a equivalência entre os campos. Para tal, utiliza os valores resultantes de cada medida aplicada na etapa anterior para efetuar o cálculo por bloco.

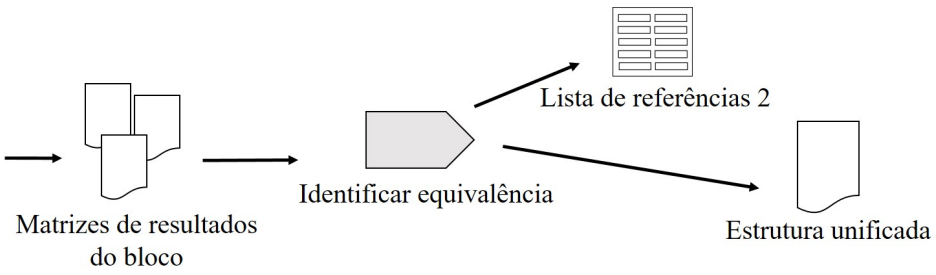

Figura 3.9 – Etapa de identificação de equivalência

Após identificar a equivalência dois artefatos são gerados: a estrutura unificada e uma segunda lista de referências. O primeiro se refere à um documento em formato texto contendo as informações agrupadas de todos os blocos pertencentes à coleção de início. O segundo diz respeito às informações de campos equivalentes entre si. A lista de referências é utilizada posteriormente para gerar os mapeamentos.

Entrada. Matrizes de resultados das técnicas aplicadas agrupados por bloco.

Atividade Identificar Equivalência. Na atividade identificar equivalência são executados cálculos e testes por bloco. Após, esses resultados são sintetizados em uma estrutura unificada. Neste mesmo processo são armazenados os campos consolidados e a quais outros correspondem. Esta atividade é decomposta em outras subatividades.

Saída. Esta atividade possui duas saídas: um arquivo texto com a estrutura unificada e uma segunda lista de referências.

Exemplo. Para exemplificação são utilizados um conjunto de palavras pertencentes aos documentos de entrada exibidos na Listagem 3.1. Estas são próprias para os casos de teste das três técnicas de análise diferentes aplicadas e descritas a seguir:

```
1 created_at; at_created; entity; entities; place; spot;
```

A síntese das três matrizes de resultados para este bloco é exibida na Figura 3.10, onde a letra (A) representa os valores da técnica de extração de radicais, a (B) da função de similaridade baseada em caractere e (C) análise de sinônimo através da função *Lin* do *WordNet*. Estas

indicam a entrada da atividade e como a matriz é simétrica pode-se utilizar apenas os elementos acima da diagonal principal.

Exemplo de matriz de resultados ao aplicar o algoritmo de Porter

	created_at	at_created	entity	entities	place	spot
created_at	1	0	0	0	0	0
at_created		1	0	0	0	0
entity			1	1	0	0
entities				1	0	0
place					1	0
spot						1

(A)

Exemplo de matriz de resultados ao aplicar a técnica de *Levenshtein*

	created_at	at_created	entity	entities	place	spot
created_at	1	0.4	0.2	0.2	0.2	0.1
at_created		1	0.1	0.1	0.2	0.1
entity			1	0.625	0	0.166
entities				1	0.125	0.125
place					1	0
spot						1

(B)

Exemplo de matriz de resultados ao aplicar a métrica de *Lin*

	created_at	at_created	entity	entities	place	spot
created_at	1	-	-	-	-	-
at_created		1	-	-	-	-
entity			1	-	0	0
entities				1	-	-
place					1	1
spot						1

(C)

Exemplo de matriz única resultante do bloco

	created_at	at_created	entity	entities	place	spot
created_at	1	0	0	0	0	0
at_created		1	0	0	0	0
entity			1	1	0	0
entities				1	0	0
place					1	1
spot						1

(D)

Figura 3.10 – Exemplo das três matrizes de resultados para um bloco

O próximo passo é efetuar o cálculo para cada elemento. O resultado é dado como 0 ou 1, ou seja, $\{x \in Z | 0 \leq x \leq 1\}$. Aplicando o cálculo, apresentado na subatividade a seguir, sobre esses números, obtém-se a matriz de resultados (D) apresentada na Figura 3.10.

Observa-se que os valores da diagonal principal serão sempre iguais a 1. Os elementos que apresentam o valor 1, localizados fora da diagonal principal, são considerados equivalentes. Neste exemplo, é o caso do par de palavras *entity* e *entities* assim como *place* e *spot*.

Portanto o artefato de saída estrutura unificada, é exibido na Listagem 3.5:

```
1 {
2         created_at
3         at_created
4         entity
5         place
6 }
```

Listagem 3.5 – Exemplo de saída etapa identificar equivalências

O intervalo das linhas 1 a 6, juntamente com os delimitadores, indicam a estrutura. Posteriormente, este arquivo será processado para gerar a representação conceitual.

A Tabela 3.7 exibe uma lista de referência das palavras que foram consolidadas para suas outras formas de ocorrência.

Tabela 3.7 – Exemplo da segunda lista de referências

Consolidado	**Termo equivalente**
entity	entities
place	spot

A atividade principal desta etapa é **identificar equivalências**. Esta é dividida em subatividades, segundo a Figura 3.11, que são detalhadas nas subseções seguintes.

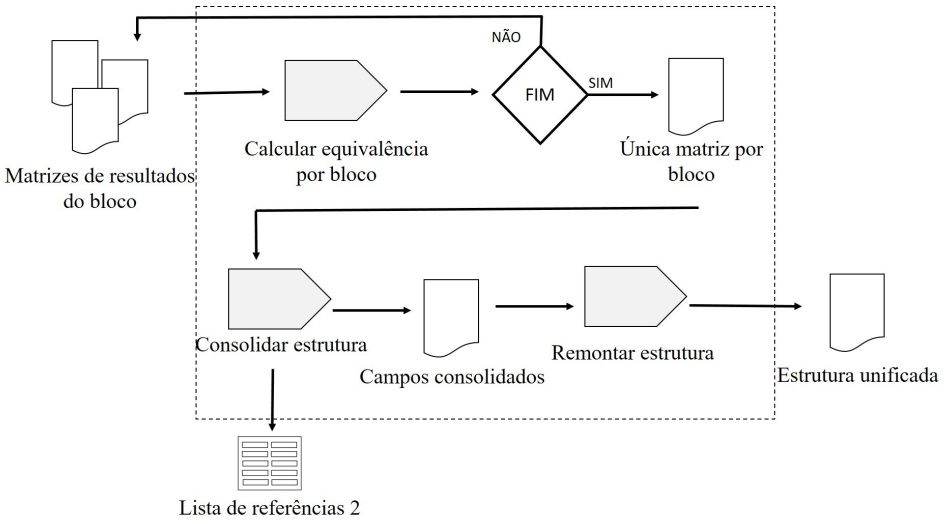

Figura 3.11 – Detalhamento da atividade identificar equivalência

Calcular equivalência por bloco. Esta atividade é executada em blocos. Ela possui como entrada três matrizes de resultados, referentes a cada técnica aplicada. São efetuados testes, comparações e cálculos que geram uma única matriz com valores de 0 ou 1 para aquele bloco específico. Este processo é executado até o fim do documento.

Consolidar estrutura. Possui como entrada uma única matriz de resultados por bloco executando testes para unificar os campos. Os blocos são executados preferencialmente em ordem, sendo que, ao final, montam um arquivo único com os campos consolidados. A atividade também gera uma segunda lista de referências com os termos e seus equivalentes.

Remontar estrutura. Esta etapa reorganiza os blocos em um único documento e/ou adiciona novamente os delimitadores para caso o documento tenha sido executado como um todo.

3.5.1 Calcular Equivalência por Bloco

Calcular equivalência é um dos processos mais importantes neste trabalho. Seu objetivo é gerar uma matriz de elementos referentes aos pares de palavras, com resultados que indicam equivalência ou não entre estas. Esta subatividade possui alguns pontos a serem destacados:

Dos valores de entrada. Cada medida gera um valor padronizado dentro do mesmo intervalo. A técnica de extrator de radicais fornece o valor 0 ou 1, isto é, $\{x \in Z | 0 \leqslant x \leqslant 1\}$. De outro modo, as técnicas de análise de similaridade baseadas em caracteres ou conhecimento fornecem um valor real no intervalo de 0 e 1, isto é, $\{x \in R | 0 \leqslant x \leqslant 1\}$.

Cabe ressaltar que o valor 0 (zero) e a ausência de valor como entrada não significam a mesma coisa, pois o valor zero indica dessemelhança enquanto o outro indica ausência de informação ou impossibilidade de aplicação de determinada técnica. Assim sendo, este processo recebe como entrada três matrizes que se referem ao resultado do extrator de radicais, similaridade de caractere e similaridade baseada em conhecimento.

Dos testes e observações. Para desenvolvimento dos testes e comparações apenas a parte superior da diagonal principal é usada visto que a matriz é simétrica, o que diminui o processamento. Outro ponto a destacar é que, se a palavra em questão não for válida na língua inglesa, possivelmente apenas a técnica de similaridade baseada em caractere poderá ser aplicada. Quanto aos testes, são expostos alguns casos:

- Se pelo menos em uma das medidas o valor for igual a 1, isto é, se o radical for o mesmo, ou se for similar em caractere ou se for considerado sinônimo, então considera-se equivalente;

- Se todos os valores das medidas apresentarem 0 ou ausência de valor então considera-se diferente;

- Se apenas uma das técnicas apresentar valor no intervalo $\{x \in R | 0 < x < 1\}$, então aplica-se o ponto de corte definido;
- Caso não se aplique nenhum caso de teste anterior, utiliza-se o método de cálculo mostrado a seguir.

Do cálculo. No caso de mais de uma técnica apresentar valores reais, este procedimento é aplicado. Inicialmente, deve-se definir pesos para as técnicas de acordo com o domínio ou contexto em questão. Para auxiliar na tomada de decisão é definida a aplicação do método AHP descrito na fundamentação teórica. Este indica um peso associado a cada técnica: similaridade (implementação *SimMetrics*), sinônimos (implementação *WS4J*) e radical (implementação *Porter*) que devem ser atribuídas aos pesos 1, 2 e 3 conforme ordem gerada no método.

Com base neste resultado, aplica-se uma média ponderada demonstrada pela Equação 3.2, reiterando que, cada teste ou cálculo é feito para cada elemento da matriz de entrada.

$$Resultado_{ij} = \frac{(pesoA * radical + pesoB * caractere + pesoC * sinonimo)}{6} \tag{3.2}$$

Assim, se o valor for maior que o ponto de corte, é considerado equivalente e o valor do elemento é definido como um, caso contrário zero.

Do valor do ponto de corte. Um limiar (*do inglês threshold*) é o fator que separa os elementos relevantes dos irrelevantes. Defini-lo não é uma tarefa trivial e na literatura existem trabalhos se preocupam exclusivamente com este ponto (SANTOS et al., 2011), (STASIU, 2007) e (DA SILVA et al., 2007).

Para (CHRISTEN, 2012) selecionar um ponto de corte em que os resultados produzem alta qualidade pode tanto ser feito manualmente ou, se um conjunto de treino estiver disponível, através de aprendizado de máquina.

Destaca também (STASIU, 2007) que há abordagens baseadas em função de recompensa e em análise estatística, contudo, um limiar ótimo implica em selecionar um ou mais limiares que resultam em valores máximos de revocação e precisão. Neste trabalho, é adotada uma abordagem simplista onde este é elaborado manualmente com base em testes realizados neste sentido.

Da atividade calcular equivalência por bloco. Esta atividade é efetuada em todos os blocos do documento. A cada iteração entram três matrizes que se transformam em uma única contendo o resultado definitivo que irá consolidar os campos posteriormente.

Entrada. Três matrizes de resultados cujos elementos representam os pares de palavras analisadas e os valores definem o grau de similaridade entre estas.

Atividade Calcular Equivalência por bloco. Esta atividade efetua alguns testes e calcula o resultado para cada elemento. Gera uma matriz única de resultados por bloco e assim sucessivamente até o fim do documento. Esta subatividade é descrita pelo Algoritmo 6.

Saída. Única matriz de resultado por bloco.

O Algoritmo 6 exibe os casos de testes necessários para identificação da equivalência. O processo é executado em cada bloco aplicando a sequência para cada elemento da matriz.

Algoritmo 6: Atividade calcular equivalência por bloco

```
   Entrada: Três matrizes de resultados das tecnicas aplicadas
   Saída: Unica matriz com resultado definitivo do bloco
   Dados: Rad[][], Lev[][], Lin[][]
 1 início
 2     enquanto houver proximo bloco faça
 3         para cada elemento da matriz faça
 4             se Rad ou Lev ou Lin for igual a 1 então
 5                 Considera equivalente
 6             fim
 7             senão se Rad e Lev e Lin for igual a 0 então
 8                 Considera diferente
 9             fim
10             senão se apenas uma medida for diferente de 0 então
11                 se elemento > ponto de corte 1 então  Considera equivalente;
12                 senão Considera diferente;
13             fim
14             senão
15                 Aplica media ponderada com pesos definidos pelo método AHP
16                 se resultado > ponto de corte 2 então  Considera equivalente;
17                 senão  Considera diferente;
18             fim
19         fim
20     fim
21 fim
```

Exemplo. Para exemplificação são apresentados alguns casos de cálculo para determinado elemento da matriz. Sejam as três matrizes de nomes Rad, Lev e Lin com mesmo elemento de linha i e coluna j a ser analisado.

1º caso: Seja Rad[i][j] = 1, Lev[i][j] = 0 e Lin[i][j] = 0. O resultado para este elemento será igual a um, isto é, matriz Resultado[i][j] = 1. Este caso é válido para qualquer das matrizes de entrada com valor 1.

2º caso: Seja Rad[i][j] = 0, Lev[i][j] = 0 e Lin[i][j] = 0. O resultado para este elemento será igual a zero, isto é, matriz Resultado [i][j] = 0. Neste caso, nenhum dos elementos indica equivalência, logo não é necessário realizar mais nenhum teste.

3º caso: Seja Rad[i][j] = 0, Lev[i][j] = 0.8 e Lin[i][j] = 0. Então, considera-se apenas a medida que possui valor. Neste caso, a matriz Resultado [i][j] = 1 se o valor do elemento for maior que o ponto de corte, isto é, Lev[i][j] > ponto de corte. Para testes, neste caso o ponto de corte adotado é de 0.7.

4º caso: Seja Rad[i][j] = 0, Lev[i][j] = 0.2 e Lin[i][j] = 0.8. Então, aplica-se a definição de pesos e a média ponderada.

Considerando que, para este último caso, a medida que indica similaridade do radical será sempre zero, utiliza-se, para exemplificação, o valor 0.7. Assim, é apresentada a Tabela 3.8 com a combinação de distribuição de pesos possíveis e o resultado em cada caso, sendo que este é feito a partir da Equação 3.2.

Tabela 3.8 – Matriz de exemplo de diferentes pesos no cálculo de equivalência

(PesoA * Rad + PesoB * Lev + PesoC * Lin) / 6	Resultado
(**1***0+**2***0.7+**3***0.7) / 6	0.583
(**1***0+**3***0.7+**2***0.7) / 6	0.583
(**2***0+**1***0.7+**3***0.7) / 6	0.466
(**2***0+**3***0.7+**1***0.7) / 6	0.466
(**3***0+**1***0.7+**2***0.7) / 6	0.35
(**3***0+**2***0.7+**1***0.7) / 6	0.35

Desse modo, o valor resultante tende a ser baixo. Assim, observa-se que apenas quando o peso atribuído ao radical é o menor, o resultado tende a ser maior que 0.5. Nas demais ponderações é inferior a este valor. Logo, no 4º caso, o limiar de corte definido é de 0.5, pois, se o peso atribuído à técnica de radical for o maior, mesmo considerando o valor de teste 0.9, o valor máximo obtido será aproximadamente 0.45, ou seja, $\{x \in R | x \leq 0.45\}$. O que não indicará equivalência.

3.5.2 Consolidar Estrutura

O processo de consolidar estrutura tem como objetivo ler as matrizes definitivas dos blocos e selecionar os campos consolidados guardando a equivalência em um segunda lista de referências. Nesta etapa os elementos W_{ij} possuem valores inteiros, isto é, zero ou um.

Um bloco por vez é executado compondo novamente os campos dos documentos retirando aqueles considerados equivalentes do arquivo. Esta saída é gerada no formato texto, contendo apenas os campos consolidados sem delimitadores que serão adicionados em atividade seguintes. Durante esta subatividade, para aqueles campos que apresentam uma forma correspondente, é montada uma segunda lista de referências.

Entrada. Única matriz de resultado por bloco.

Atividade consolidar estrutura. Esta fase inclui todos os blocos da coleção e é detalhada no Algoritmo 7. Ao consolidar, a atividade apaga as demais ocorrências do campo que estão escritas de forma distinta e as guarda na lista de referências.

Saída. Campos consolidados e o artefato lista de referências 2.

Algoritmo 7: Atividade consolidar estrutura

```
   Entrada: Única matriz por bloco
   Saída: Campos consolidados; Lista de referências 2
 1 início
 2 |  Carrega matriz de resultados do bloco;
 3 |  Consolida todas as palavras da coluna;
 4 |  para cada elemento acima da diagonal principal faça
 5 |  |  se elemento da Matriz_ab é igual a 1 então
 6 |  |  |  Mantém a primeira ocorrência do termo (palavra da linha)
 7 |  |  |  Apaga a palavra da coluna dos campos consolidados
 8 |  |  |  Grava equivalência na Lista de referências 2
 9 |  |  fim
10 |  fim
11 fim
```

Exemplo. Para exemplificação deste processo é dada a Tabela 3.9 com o resultado do bloco para o conjunto de palavras:

```
1 coordinates; favorited; entity; entities;
```

Tabela 3.9 – Matriz de exemplo de resultados definitivo dos bloco

	coordinates	favorited	entity	entities
coordinates	1	0	0	0
favorited		1	0	0
entity			1	1
entities				1

Ao executar o algoritmo 7 sobre estes dados é gerado um arquivo texto com os campos consolidados conforme trecho abaixo.

```
1 { coordinates; favorited; entity; }
```

A segunda lista de referências é exemplificada através da Tabela 3.7. Para este exemplo, apenas o termo "*entity*" irá corresponder a "*entities*".

3.5.3 Remontar Estrutura

O objetivo desta atividade é apenas reorganizar e remontar o documento completo adicionando os delimitadores corretamente. Caso o processo tenha sido executado em blocos, a atividade os organiza em ordem. Caso tenha sido executado como um único documento, os delimitadores que foram desconsiderados para a análise são recolocados nesta etapa.

Entrada. Campos consolidados.

Atividade remontar estrutura. Esta atividade, expressa pelo Algoritmo 8, remonta os blocos em ordem ou monta um novo documento com base no documento de origem com maior número de blocos.

Saída. Estrutura unificada, pronta para ser processada na próxima etapa de representação da estrutura.

Algoritmo 8: Atividade remontar estrutura

```
   Entrada: Campos consolidados, lista de referências 1 e lista de referências 2
   Saída: Estrutura consolidada
 1 início
 2     Carrega campos consolidados e listas
 3     selecione modo de execução faça
 4         caso único bloco faça
 5             Escolhe o documento de origem para referência (com maior número de
                blocos)
 6             para cada campo do documento de referência faça
 7                 se campo está consolidado então mantém;
 8                 senão verifica os correspondentes que estão consolidados e os
                    substitui;
 9             fim
10             Os campos consolidados que não estão no documento referência, são
                incluídos em um novo objeto delimitado por chaves
11         fim
12         caso por blocos faça
13             Remonta os blocos em ordem mantendo símbolos delimitadores de
                origem
14         fim
15     fim
16 fim
```

Exemplo. Para exemplificação é apresentado no trecho de código a seguir (linhas 1 e 2) como os blocos 1 e 2 eram originalmente, isto é, segundo saída da atividade mesclar estrutura, representada pela Listagem 3.4.

```
1 Bloco 1: {coordinates, favorited, truncated, at_created,
2          created_at, id_str, entity, entities}
3 Bloco 2: [{expanded_url, url, indices[], display_url}]
```

Assim também, são dados os campos consolidados (linhas 1 e 2) após o processamento desta etapa:

```
1 Bloco 1: coordinates, favorited, truncated, at_created,
2          created_at, id_str, entity
3 Bloco 2: expanded_url, url, indices, display_url
```

Desse modo, esta atividade apenas reorganiza e adiciona os delimitadores corretamente, gerando o segundo artefato da atividade principal, a estrutura unificada. Esta é exibida na Listagem 3.6 e apresenta os blocos do elemento raiz, "*entity*" e "*urls*".

```
1 {
2         coordinates, favorited, truncated, created_at, id_str,
3         entity {
4                 urls [{ expanded_url, url, indices [], display_url
                      }]
5         hashtags[],
6         user_mentions[]
7         }
8 }
```

Listagem 3.6 – Exemplo de estrutura unificada

Este resultado não representa um JSON válido, mas um arquivo texto com delimitadores aplicados segundo a gramática, para que seja possível processá-lo na etapa seguinte ao gerar a representação da estrutura.

3.6 Representação da Estrutura

A etapa de representação da estrutura, apontada pela Figura 3.12, tem por objetivo, a partir do artefato estrutura unificada, aplicar a conversão desta para uma notação visual, bem como gerar os mapeamentos dos termos consolidados para seus equivalentes. A atividade principal desta etapa é gerar representação.

O arquivo de entrada apresenta os campos juntamente com os delimitadores que auxiliam tanto no processo de montar o esquema conceitual como para representar os mapeamentos com a notação JSONPath.

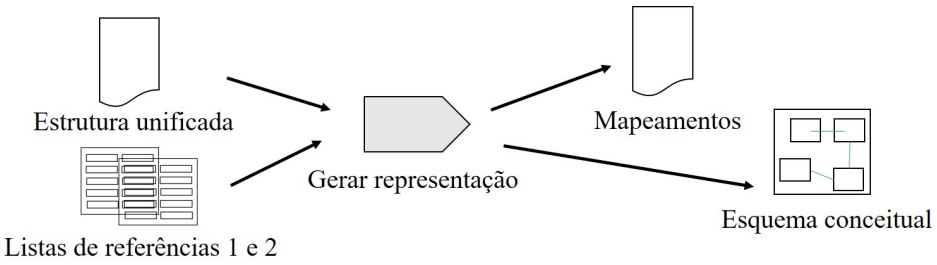

Figura 3.12 – Etapa de representação da estrutura

Entrada. Estrutura unificada.

Atividade gerar representação. A atividade gerar representação consiste em produzir os dois artefatos finais do processo: (1) o esquema conceitual de modo visual através da aplicação de regras de conversão e (2) os mapeamentos dos termos consolidados para os equivalentes identificados no processo. Esta atividade é especificada em outras subatividades.

Saída. Esquema conceitual e mapeamentos.

Exemplo. Para exemplificação dessa atividade, é utilizada a estrutura unificada representada na Listagem 3.6. Este trecho de código possui os campos consolidados e delimitadores que indicam início e fim do bloco. A Figura 3.13 representa a saída gerada ao aplicar algumas regras de conversão que são detalhadas nas subseções seguintes.

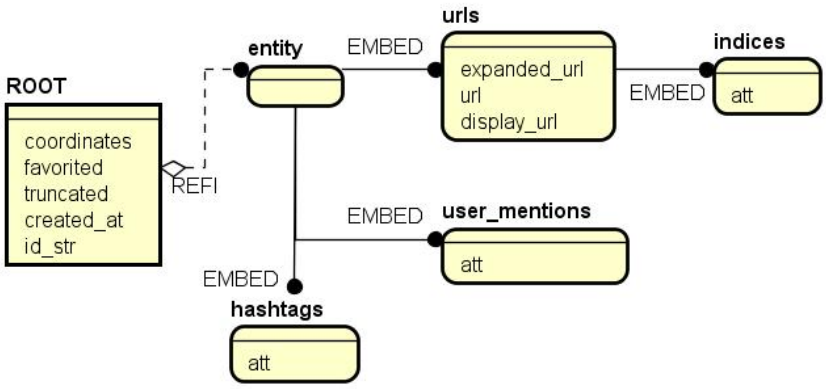

Figura 3.13 – Exemplo de saída da etapa representação da estrutura

Neste exemplo, conforme observado a raiz do documento não possui nome, pois não há palavra antes da primeira chave de abertura. Todavia, a classe é indicada pelo rótulo *ROOT*. A entidade ROOT possui relacionamento com *entity*, que por sua vez, se relaciona com *hashtags*, *user_mentions* e *urls*. Esta última também possui relação com *indices*. As entidades que serão aninhadas à entidade raiz, recebem o rótulo *EMBED*.

Considerando o esquema conceitual da Figura 3.13, a única palavra que possui termos

correspondentes é *entity*, logo, a Tabela 3.10 demonstra o segundo artefato de saída.

Tabela 3.10 – Exemplo de mapeamento gerado na etapa representação da estrutura

Consolidado	**Termo**	**Origem**
$.Twitter.entity	$.Twitter.entities	Doc 2

Portanto, este termo corresponde a *entities* que se encontra no documento 2 de origem.

A atividade principal é **gerar representação**. Esta é dividida em duas subatividades, conforme Figura 3.14 e especificado nas subseções seguintes.

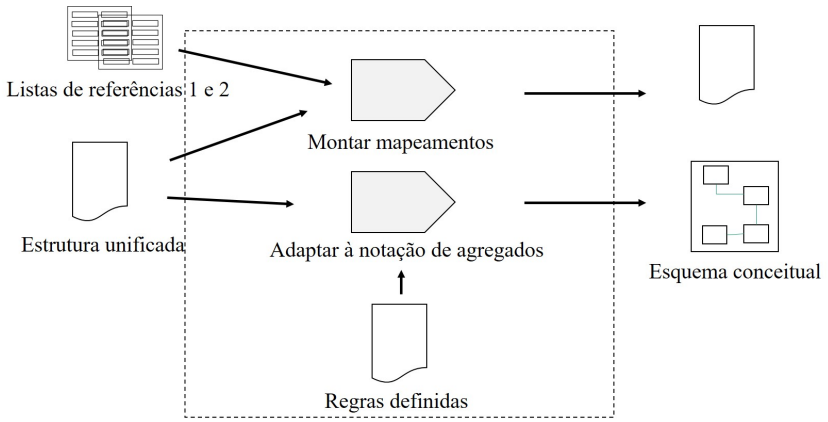

Figura 3.14 – Detalhamento da atividade gerar representação

Listas de referências 1 e 2. Estes artefatos são os mesmos gerados ao longo do processo de extração. A primeira lista é gerada na atividade principal extrair campos da Seção 3.3 e a segunda é saída da atividade principal identificar equivalência da Seção 3.5.

Atividade montar mapeamentos. Esta atividade tem por objetivo consultar ambas as listas para inferir novas equivalências entre os campos, montando uma tabela final com o caminho do termo consolidado representado pela notação JSONPath.

Regras definidas. Este novo artefato apresenta as regras definidas. Estas são empregadas na identificação dos blocos em objetos ou *arrays* que auxiliam na definição de entidades, atributos e relacionamentos do esquema conceitual.

Atividade adaptar à notação de agregados. Esta atividade tem por objetivo montar a representação através da notação IDEF1X adaptada ao modelo NoSQL proposta por (JOVANOVIC; BENSON, 2013) aplicando as regras definidas.

3.6.1 Montar Mapeamentos

Mapeamentos. Como a proposta deste trabalho abrange a consolidação de um esquema, para que seja possível realizar futuras consultas, entre outras tarefas, é necessário guardar os mapeamentos. Estas informações relacionam um termo consolidado aos seus correspondentes em outros documentos pertencentes à mesma coleção, para que ao acessar determinado termo, tenha-se conhecimento de quais outros estão relacionados.

Dessa maneira, a atividade tem por objetivo, com base nas listas de referências e na estrutura unificada, identificar possíveis equivalências internas e montar uma representação em formato de tabela com as respectivas informações: termo consolidado, seu correspondente e documento de origem do correspondente. Para representação do termo consolidado é utilizada a notação JSONPath que especifica caminhos dentro de um documento.

Entrada. Estrutura unificada, lista de referências 1 e lista de referências 2.

Atividade montar mapeamentos. Esta atividade é demonstrada pelo Algoritmo 9 e consiste em, com base nos elementos de entrada, verificar novas equivalências entre os campos e montar uma tabela de saída.

Saída. Mapeamentos.

Algoritmo 9: Atividade montar mapeamentos

```
Entrada: Lista de referências(1), Lista de referências(2) e estrutura consolidada
Saída: Mapeamentos
1  início
2      Carrega ambas listas de referências e estrutura unificada
3      para cada campo consolidado ∈ estrutura unificada faça
4          Procura-o na estrutura unificada montando a referência JSONPath;
5          Verifica na lista de referências 2 quais termos são equivalentes;
6          Verifica equivalências do tipo A=B e B=C => A=C;
7          para cada termo equivalente faça
8              Verifica documento de origem na lista de referências 1;
9          fim
10     fim
11     Grava mapeamentos;
12 fim
```

Exemplo. Para exemplificação de verificar equivalências é dado o seguinte trecho de código representando a segunda lista de referências:

```
1 authors = source
2 source = author
```

Assim sendo, conforme linhas 1 e 2 do código anterior, pode-se inferir que *authors* será correspondente a *author*, como detalhada na linha 3 do Algoritmo 9.

3.6.2 Regras Definidas

Este artefato tem por objetivo apresentar as regras definidas para conversão da estrutura unificada em um esquema conceitual. Estas são adaptadas da proposta de (IZQUIERDO; CABOT, 2013) cujas regras se baseiam no conjunto de símbolos da gramática JSON para auxiliar no processamento.

De modo geral, este artefato visa identificar entidades, atributos e relacionamentos para esquemas em NoSQL. No contexto deste trabalho, uma entidade pode corresponder a um bloco e os relacionamentos identificados são do tipo um para um $1:1$ e um para muitos $1:n$.

Nesta proposta, por exemplo, ao encontrar o início de um objeto ou *array* infere-se uma nova classe de relacionamento $1:n$.

Regras. As regras para construção do esquema conceitual são descritas e numeradas por Cn onde n aponta o número da regra. Inicialmente é identificado o delimitador que marca o início do documento d, que deve ser encerrado ao encontrar o mesmo símbolo de fechamento. As regras definidas são as seguintes:

- C1 – Cada '{' de nome $String$ gera uma nova classe C de nome $String$ e cria um relacionamento do tipo $0:1$. A classe C só é fechada quando encontrar '}' e o número de chaves abertas corresponda ao número de chaves fechadas.

- C2 – Cada campo cujo $value$ = 'tipo primitivo qualquer' gera um atributo att na classe C. Se este já existe nada acontece, senão o atributo é criado;

- C3 – Cada '[' de nome $String$ gera uma nova classe C de nome $String$ com atributo de nome att e cria um relacionamento $0:N$. A classe C só é fechada quando encontrar ']' e o número de colchetes abertos corresponda ao número de colchetes fechados;

- C4 – Cada '[{' gera uma nova classe c de nome $String$ e cria um relacionamento $0:N$. Ela aplica C2 novamente. Quando encontrar '}', volta ao início da classe C para testar novamente os atributos att. Se o atributo já existe nada acontece, senão ele é criado dentro da classe C; A classe só encerra quando encontrar ']';

Conforme observado, a regra C1 é aplicada para gerar uma nova classe cujo nome é a $String$ que antecede a abertura da chave. Esta regra ocorre ao identificar o início de um objeto. Este relacionamento é do tipo $0:1$.

A regra C2 transforma em atributos os campos nas classes correspondentes. Ela é executada após a identificação de início de uma classe.

A regra C3 ocorre quando é aberto um *array* com enumeração de itens dentro, gerando uma classe nova com um único atributo de nome att.

A regra C4 também gera uma nova classe cujo nome é a $String$ que antecede a abertura do colchete. Este relacionamento é do tipo $1:n$. Este caso, porém, diferencia-se primeiramente por conter objetos aninhados. Assim os atributos desta classe são criados por iterações nos objetos, isto é, aplica C2 novamente a cada vez que encerrar o objeto interno. Desse modo, o processo retorna ao início do objeto para verificar se o atributo att já existe. Caso seja diferente, ele é acrescentado no esquema conceitual.

3.6.3 Adaptar à Notação de Agregados

Esta atividade tem por objetivo montar a representação do esquema conceitual, artefato final do processo proposto neste trabalho. Para tal, a atividade aplica as regras de conversão definidas sobre o arquivo de entrada. Os relacionamentos entre coleções, isto é, do tipo muitos para muitos $n:n$, não estão no escopo deste, pois o trabalho trata apenas das relações entre documentos e não entre coleções.

O arquivo de entrada apresenta os campos, juntamente com os delimitadores como chaves e colchetes, que auxiliam a identificar entidades, atributos e relacionamentos, gerando uma representação na notação IDEF1X própria ao modelo NOSQL de acordo com (JOVANOVIC; BENSON, 2013). Esta notação parte de um nó raiz sendo os demais aninhados nesta.

Entrada. Estrutura unificada e as regras de conversão definidas.

Atividade adaptar à notação de agregados. Consiste em percorrer o arquivo texto aplicando as regras definidas a cada caso encontrado. Por fim, gera uma representação do esquema conceitual unificado. Esta atividade é representada pelo Algoritmo 10.

Saída. Esquema conceitual que consiste no artefato final do processo deste trabalho.

O Algoritmo 10 percorre o documento. Ao identificar um evento do tipo delimitador, verifica qual é o caso e aplica a regra. De modo geral, se o evento é apenas uma chave "{", a regra infere um relacionamento do tipo $0:1$. Para os demais eventos, ele infere relacionamento

Algoritmo 10: Atividade adaptar à notação de agregados

Entrada: Estrutura unificada; Regras definidas
Saída: Esquema conceitual

```
1  início
2      Carrega arquivo com a estrutura unificada;
3      Primeiro objeto é considerada entidade raiz;
4      selecione evento faça
5          caso { faça Aplica regra C1;
6          caso value faça Aplica regra C2;
7          caso [ faça Aplica regra C3;
8          caso [{ faça Aplica regra C4;
9      fim
10     Monta representação visual
11 fim
```

do tipo $1:n$. Ao criar uma nova entidade, seu nome será a *string* anterior. Caso seja um atributo, apenas mantém seu nome. Se este atributo já existe nada é feito, caso contrário adiciona-se um novo. Após a atividade monta uma representação na notação proposta.

Exemplo. Para exemplificação é dado o seguinte trecho de código na Listagem 3.7 que representa a estrutura unificada, como parte do documento apresentado na Listagem 3.1.

```
1 entity{
2         urls [{
3                 expanded_url; url; indices [] display_url;
4         }]
5         hashtags[]
6 }
```

Listagem 3.7 – Exemplo de entrada da atividade adaptar notação a agregados

A partir deste trecho de código, a Figura 3.15 demonstra a aplicação para cada caso de regra apresentado.

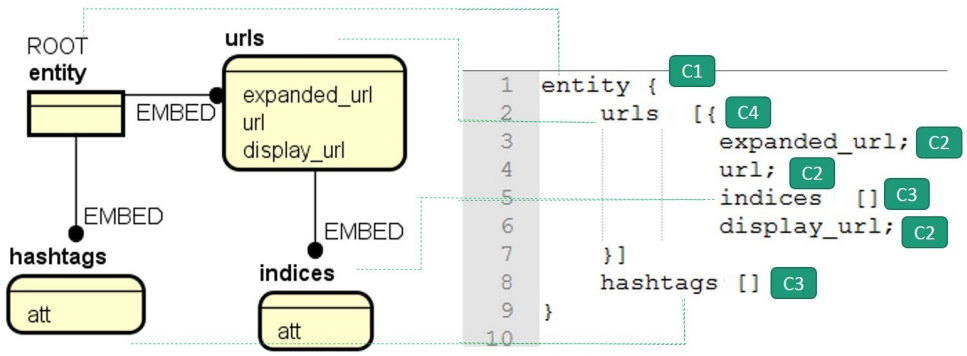

Figura 3.15 – Exemplo de aplicação das regras na atividade adaptar notação à agregados

A regra C1 é aplicada na linha 1 ao identificar o início de um objeto, criando uma enti-

dade de nome "*entity*" considerada raiz. A regra C2 é aplicada nas linhas 3, 4 e 6 transformando-os em atributos. A regra C3 é utilizada nas linhas 5 e 8 ao encontrar um *array*, ainda que esteja vazio. Por fim, a regra C4 é apontada na linha 2 ao identificar o início de um *array* seguido de objeto, criando uma entidade de nome "*urls*". Para relacionamentos que são aninhados é utilizada a notação EMBED.

3.7 Considerações Finais

Este capítulo apresentou o processo proposto para extração de esquemas em bancos de dados NoSQL orientados a documentos. Diferencia-se principalmente por considerar que pela falta de padronização ou pela flexibilidade de esquema existam campos com nomes diferentes que se refiram à mesma informação entre documentos da coleção e entre cada bloco interno.

Para tal são aplicadas diferentes técnicas de análise textual aos campos extraídos para identificar equivalências com base na análise do radical da palavra, na sua sintaxe e em sua semântica. Este processo gera ao final uma representação de um esquema conceitual em uma notação adaptada ao paradigma NoSQL.

Este processo consiste basicamente em quatro etapas:

Pré-processamento: é realizado entre documentos, extraindo apenas os campos dos documentos JSON e unificando-os em um único arquivo texto para análise posterior. Deste modo, recebe como entrada a coleção do banco de dados NoSQL orientado a documentos e gera um arquivo chamado documento estrutural geral, juntamente com uma lista da referências que contém os termos distintos e seus documentos de origem.

Análise de similaridade: neste ponto são executadas três técnicas distintas que fazem a análise par a par de cada palavra. Esta etapa é realizada por blocos até o fim do documento. Possui como entrada o documento estrutural geral e produz três matrizes por bloco com os resultados das análises. Estes resultados são valores do intervalo de 0 a 1, ou seja, $\{x \in R | 0 \leqslant x \leqslant 1\}$ onde 0 indica total dessemelhança e 1 totalmente equivalente.

Identificação de equivalências: esta parte é fundamental no processo. Realiza uma série de testes e comparações de modo a apontar quais elementos são equivalentes. Analisa as matrizes resultantes da etapa anterior, elemento por elemento. Caso ao menos um resultado seja igual a 1, infere-se equivalência. Caso todos os resultados sejam igual a 0, infere-se que são distintos. Caso apenas uma medida apresente valor, desde que diferente de zero e um, esta é utilizada como base para ponto de corte. Caso estas apresentem números do intervalo real de 0

a 1 então é necessário definir prioridades entre as técnicas. Para tal, é utilizado o método AHP no auxílio de tomada de decisão. Ao definir peso para as técnicas é feita uma média ponderada e acima de determinado ponto de corte é considerado equivalente. Ainda nesta etapa, é gerada uma segunda lista de referências com as equivalências entre os termos e a estrutura unificada.

Representação da estrutura: esta etapa produz os dois artefatos finais do processo: os mapeamentos e o esquema conceitual. Consiste de duas atividades principais que utilizam a estrutura unificada como entrada. A primeira monta uma tabela com mapeamentos dos termos consolidados para seus correspondentes utilizando ambas listas de referências. A segunda aplica regras de conversão baseadas na gramática JSON sobre a estrutura unificada para gerar uma representação visual do esquema conceitual em notação adaptada ao modelo NoSQL.

Dessa forma é realizado o processo de extração do esquema implícito em bancos de dados NoSQL orientados a documentos. Tem por base uma coleção de documentos e gera uma representação unificada de seus elementos. Através dos mapeamentos armazenados possibilita a realização de futuras consultas no esquema.

4 ESTUDO DE CASO

Este capítulo apresenta a aplicação do processo de extração proposto em um estudo de caso. O processo tem início a partir de uma coleção de documentos em formato JSON e gera um esquema conceitual representado de modo visual e uma tabela com os mapeamentos.

O capítulo é introduzido com uma visão geral dos testes realizados, apresentando definições como, por exemplo, de domínio e ponto de corte. As ferramentas utilizadas são detalhadas juntamente com os algoritmos que fazem parte do processo. Por conseguinte, são demonstrados dois exemplos de aplicação a partir do domínio de publicações científicas e, por fim, são relatadas as discussões.

4.1 Visão Geral

Neste estudo de caso são expostas duas formas de aplicação. A primeira corresponde à execução por blocos, sendo necessária a intervenção de um usuário do domínio para especificar quais blocos são equivalentes. A segunda trata todos os blocos como um único documento, não sendo necessária a intervenção de um especialista, executando o processo nos campos como um todo.

Quanto aos arquivos de entrada, uma das características é que esta também se aplica a outros formatos, desde que seja possível converter para JSON antes do início do processo. Assim, a proposta também aceita arquivos XML e CSV (*Comma Separated Values*), desde que previamente convertidos.

Domínio de aplicação. O domínio de aplicação de testes escolhido é relacionado a publicações científicas. Assim, os arquivos de entrada são referências exportadas de artigos e demais trabalhos científicos de bibliotecas digitais como DBLP[26], Scopus[27], PubMed[28] e Bibsonomy[29]. Estas fontes de dados foram escolhidas devido ao fato de permitirem exportação das referências nos formatos XML ou CSV, possibilitando a conversão para JSON.

Os arquivos de entrada foram escolhidos de modo aleatório, visto que, no contexto deste trabalho, os dados são desconsiderados no processo de extração de esquema. Apenas para fins de avaliação de precisão e revocação os dados são analisados, isto é, para confirmar se o

[26] http://dblp.uni-trier.de/
[27] https://www.scopus.com/home.uri
[28] https://www.ncbi.nlm.nih.gov/pubmed
[29] https://www.bibsonomy.org/

resultado inferido está adequado ou incorreto.

Ponto de corte. Quanto à definição do ponto de corte, foi realizada uma série de testes com intuito de maximizar os índices de revocação e precisão conforme apontado por (STA-SIU, 2007). Dessa forma, dado um determinado caso de testes disponível na pasta pública do OneDrive[30], é apresentada a Tabela 4.1 com as variações de revocação e precisão conforme a variação do ponto de corte.

Tabela 4.1 – Valores de revocação e precisão para variações no ponto de corte

Caso 1	Caso AHP	Recuperados	Rec. Relevantes	Relevantes	Revocação	Precisão
0,75	0,5	10	7	11	63,63%	70%
0,7	0,5	14	10	11	90,9%	71,42%
0,7	0,45	15	10	11	90,9%	66,66%
0,6	0,5	22	11	11	100%	59,09%

As duas primeiras colunas da Tabela 4.1 indicam os dois casos do processo em que é necessário aplicar um ponto de corte. O primeiro caso é quando apenas uma das medidas utilizadas apresenta valor (este fato ocorre quando uma palavra não é válida na língua inglesa), o segundo caso se aplica quando for executada a média ponderada com definição de pesos pelo método AHP. Tendo em vista que cada caso possui suas particularidades, são necessários limiares diferentes. As demais colunas demonstram os documentos recuperados, recuperados relevantes, relevantes e os resultados para revocação e precisão.

De acordo com o exemplo de testes desta tabela, há no conjunto 11 equivalências relevantes. Ao modificar os valores de ponto de corte são apresentados os resultados com objetivo de obter melhores índices de revocação e precisão. Desse modo, optou-se para aplicação no estudo de caso, o valor $0,7$ para caso apenas uma medida apresente valor e $0,5$ para caso seja aplicada a média ponderada e o AHP. Neste último caso, o valor baixo justifica-se pelo fato de que neste caso sempre a medida que indica o radical terá valor zero.

Testes. Conforme organização do processo, este permite aplicação tanto em arquivos armazenados em uma coleção do banco de dados NoSQL, como para arquivos em formato JSON de diferentes fontes de dados que se queira acessar de modo unificado. Este último também pode ser armazenado em um banco de dados orientado a documentos.

Sendo assim, o processo pode ser executado de dois modos: (1) testando por blocos ou (2) todos os campos como um único documento. O processo (1) é semiautomático, pois,

[30] https://1drv.ms/f/s!AkZi7x6fPkh7xzxEXLfNvOF29VUS

necessita da intervenção de um usuário com conhecimento do domínio que identifique qual bloco é correspondente a qual, em termos de nível de hierarquia. O processo (2) ignora possíveis blocos e considera que todos os campos podem ser equivalentes entre si.

Algumas atividades foram implementadas e outras foram executadas manualmente, sendo que ambos os estudos de caso realizados, testes e implementações encontram-se disponíveis em uma pasta pública do OneDrive[31] acessíveis pelo endereço no rodapé. Assim sendo, o capítulo encontra-se estruturado em dois estudos de caso, pertencentes ao mesmo domínio. O primeiro é executado em blocos, representando documentos de uma coleção e um segundo cujos campos são testados como um todo, onde os blocos são abstraídos, investigando equivalências em todo documento.

Avaliação. Para fins de avaliação são utilizadas as medidas de revocação e precisão. Segundo (DORNELES, 2010) a revocação é a proporção do total de pares similares existentes que aparecem no resultado final (Equação 4.1). Por outro lado, a precisão indica a proporção dos pares de valores corretamente identificados como similares que aparecem no resultado (Equação 4.2).

$$\text{Revocação} = \frac{\text{Recuperados Relevantes}}{\text{Relevantes}} \tag{4.1}$$

$$\text{Precisão} = \frac{\text{Recuperados Relevantes}}{\text{Recuperados}} \tag{4.2}$$

Desse modo, a revocação é definida pela divisão dos documentos recuperados relevantes pelos relevantes, enquanto a precisão é a divisão dos documentos recuperados relevantes sobre os recuperados.

4.2 Ferramentas

Esta seção descreve as ferramentas utilizadas para executar o processo de extração de esquema. Alguns algoritmos foram implementados e o restante é executado manualmente.

O processo tem início com os arquivos em formato JSON e gera, ao final, um esquema conceitual e uma tabela com mapeamentos. Desse modo, ele consiste em executar a sequência de algoritmos apresentados nas atividades das 4 etapas: pré-processamento, análise de similaridade, identificação de equivalências e representação da estrutura. Todos os testes são execu-

[31] https://1drv.ms/f/s!AkZi7x6fPkh7xzxEXLfNvOF29VUS

tados em um computador com Windows 10 Home, processador Intel(R) Core(TM) i5, 8GB de memória RAM e 1TB de HD.

Antes de dar início ao processo, o arquivo precisa estar em formato JSON e ser válido pela gramática. Nesse sentido, para validação do esquema e conversão do formato XML para JSON é utilizada a ferramenta Altova XML Spy 2016[32]. De mesmo modo, é realizada a conversão do formato CSV para JSON[33] de modo *online*;

A seguir são listados cada algoritmo e sua forma de execução:

- O Algoritmo 1 representa a atividade separar campos dos dados. Este foi implementado em linguagem Java e utiliza a API JSR353 para processar um arquivo JSON.

- O Algoritmo 2 representa a atividade mesclar estrutura. Este foi implementado em linguagem PHP executado através do navegador com o servidor local XAMPP[34].

- O Algoritmo 3 representa a atividade analisar radical da palavra. Este não foi implementado. Consiste em aplicar a implementação de Porter Stemmer a um documento do tipo texto contendo as palavras para extrair seu radical. Este processo é feito via *prompt* de comando. A matriz de saída é montada manualmente.

- O Algoritmo 4 representa a atividade analisar similaridade baseada em caractere. Este utiliza a API SimMetrics que fornece o método da medida Levenshtein. Dessa forma, o algoritmo foi implementado em linguagem Java com o objetivo de montar a matriz de saída. O método principal recebe um vetor de *strings* como entrada e aplica a medida seguida da equação de similaridade para cada par de palavras.

- O Algoritmo 5 representa a atividade analisar similaridade baseada em conhecimento. Este é um processo executado *online*, através do *link* comparação de sentenças, que permite entrar com uma cadeia de palavras, gerando uma matriz de comparações par a par extraindo o valor da medida de Lin. O site utilizado[35] é uma implementação do Wordnet para Java.

- O Algoritmo 6 representa a atividade calcular equivalência por bloco. Este tem como entrada três matrizes de resultados geradas anteriormente. Assim sendo, é

[32] https://www.altova.com/download/xmlspy.html
[33] http://www.csvjson.com/csv2json
[34] https://www.apachefriends.org/pt_br/download.html
[35] http://ws4jdemo.appspot.com/

executado em planilhas do Microsoft Excel 2013, onde cada caso é testado através da aplicação de fórmulas e testes lógicos, resultando em uma matriz com valores zero e um.

- O Algoritmo 7 representa a atividade consolidar estrutura. Este é executado manualmente ao verificar a planilha resultante, gerando um arquivo com campos consolidados e uma nova lista de referências.

- O Algoritmo 8 representa a atividade remontar estrutura. Este também é executado manualmente, adicionando os delimitadores de origem, ou reorganizando os blocos, para gerar a estrutura consolidada.

- O Algoritmo 9 representa a atividade montar mapeamentos. Este é executado manualmente e tem pro objetivo gerar os mapeamentos, representando com notação JSONPath o termo consolidado.

- Por fim, o Algoritmo 10 representa a atividade adaptar à notação de agregados. Este consiste em manualmente aplicar as regras definidas para gerar uma representação visual. Para visualização é utilizada a ferramenta de modelagem Astah Professional[36], cujo diagrama representa um modelo ER (entidade-relacionamento) na notação IDEF1X proposta.

Nas próximas seções são demonstrados os casos de testes propostos para cada forma de aplicação.

4.3 Caso 1 - Executado por blocos

Neste caso de teste, os documentos são separados em blocos marcados através dos delimitadores. Nesta abordagem, é necessária a intervenção de um especialista do domínio que indique quais blocos irão corresponder a outros, isto é, indicando equivalências entre eles. O intuito da investigação é encontrar no esquema campos que possam se referir à mesma informação, mesmo que escritos de modo diferente.

De outro modo, este experimento objetiva também demonstrar a aplicação em dados que já estejam presentes no banco de dados NoSQL. Este caso diferencia-se pelo fato de que, estando em uma mesma base de dados, que possivelmente é acessível através de programação,

[36] http://astah.net/editions/professional

estes tendem a possuir uma estrutura bem semelhante aos outros documentos pertencentes à coleção, ainda que o modelo permita a flexibilidade.

Ainda, ao inferir uma estrutura mais semelhante, a abordagem de separar em blocos para testes é adequada, pois facilita identificar a correspondência de hierarquia dos blocos entre documentos. Considerando que estes estão em um mesmo nível, ou seja, eles apresentam o mesmo pai ou equivalente, isto poderia evitar identificar incorretamente palavras homônimas.

Para cada caso, são utilizados quatro documentos de entrada que são representados a seguir. A Figura 4.1 se refere a arquivos da biblioteca DBLP extraídos em formato XML.

Documento 1


```
{
"dblp": {
"article": {
      "@key": "journals/ijgi/QiuZDWF17",
      "@mdate": "2017-02-21",
      "author": [
          "Linyao Qiu",
          "Qing Zhu",
          "Zhiqiang Du",
          "Meng Wang",
          "Yida Fan"
      ],
      "title": "\nAn On-Demand Retrieval Method
Based on  Hybrid NoSQL for Multi-Layer Image Tiles
in Disaster Reduction Visualization.\n",
      "pages": 8,
      "year": 2017,
      "volume": 6,
      "journal": "ISPRS Int. J. Geo-Information",
      "number": 1,
      "ee":
"http://dx.doi.org/10.3390/ijgi6010008",
      "url":
"db/journals/ijgi/ijgi6.html#QiuZDWF17"
     }
   }
}
```


Fonte:http://dblp.uni-trier.de/rec/xml/journals/ijgi/QiuZDWF17.xml

Documento 2


```
{
"dblp": {
      "proceedings": {
      "@key": "conf/naa/2016",
      "@mdate": "2017-04-13",
      "editor": [
          "Ivan Dimov","Istvan Farago", "Lubin G. Vulkov"
      ],
      "title": "\nNumerical Analysis and Its Applications
- 6th International Conference, NAA 2016, Lozenetz,
Bulgaria, June 15-22, 2016, Revised Selected Papers\n",
      "booktitle": "NAA",
      "year": 2017,
      "series": {
          "@href": "db/journals/lncs.html",
          "$": "Lecture Notes in Computer Science"
      },
      "volume": 10187,
      "isbn": [
          "978-3-319-57098-3", "978-3-319-57099-0"
      ],
      "ee": "http://dx.doi.org/10.1007/978-3-319-57099",
      "url": "db/conf/naa/naa2016.html"
      }
   }
}
```


Fonte: http://dblp.uni-trier.de/rec/xml/conf/naa/2016.xml

Figura 4.1 – Documentos de entrada 1 e 2 para o estudo de caso 1

Conforme observado o documento 1 corresponde a um artigo de periódico e o documento 2 a uma publicação de evento. De modo semelhante, a Figura 4.2 apresenta o documento 3 referente a uma publicação de conferência e o documento 4 como uma referência da biblioteca Scopus extraído em formato CSV e convertido para JSON.

A partir destes documentos de entrada, um especialista do domínio define quais blocos são correspondentes e o processo de extração é iniciado.

Na etapa de **pré-processamento** os dados são descartados mantendo apenas os campos. As atividades separar campos dos dados e mesclar estrutura são aplicadas. O processo completo está disponível na pasta pública e por questões de espaço apenas as saídas principais são expostas neste documento. Esta etapa mantém apenas a estrutura dos arquivos JSON e percorre todos

Documento 3


```
{
"dblp": {
      "inproceedings": {
      "@key": "conf/ACMse/ParkerPV13",
      "@mdate": "2015-05-07",
      "author": [
          "Zachary Parker",
          "Scott Poe",
          "Susan V. Vrbsky"
      ],
      "title": "Comparing NoSQL MongoDB to an SQL
DB.",
      "pages": "5:1-5:6",
      "year": 2013,
      "booktitle": "ACM Southeast Regional
Conference",
      "ee":
"http://doi.acm.org/10.1145/2498328.2500047",
      "crossref": "conf/ACMse/2013",
      "url":
"db/conf/ACMse/ACMse2013.html#ParkerPV13"
      }
   }
}
```


Fonte: http://dblp.uni-trier.de/rec/xml/conf/ACMse/ParkerPV13.xml

Documento 4


```
{
      "Authors": "Gonzalez-Aparicio M.T., Ogunyadeka
A., Younas M., Tuya J., Casado R.",
      "Title": "Transaction processing in
consistency-aware users applications deployed on
NoSQL databases",
      "Year": 2017,
      "Source title": "Human-centric Computing and
Information Sciences",
      "Volume": 7,
      "Issue": 1,
      "Art. No.": 7,
      "Page start": "",
      "Page end": "",
      "Page count": "",
      "Cited by": "",
      "DOI": "10.1186/s13673-017-0088-3",
      "Link":
"https://www.scopus.com/inward/record.uri?eid=2-s2.0-
85017092398&doi=10.1186%2fs13673-017-0088-
3&partnerID=40&md5=428fd7f4ac78b0e500e305981b9d17a5",
      "EID": "2-s2.0-85017092398"
}
```


Fonte: https://goo.gl/WjBCwT

Figura 4.2 – Documentos de entrada 3 e 4 para o estudo de caso 1

os documentos mesclando-os em um único arquivo. Portanto, os artefatos de saída são o documento estrutural geral, representado pela Listagem 4.1, e a lista de referências 1 apresentada na Tabela 4.2.

```
1 Bloco 1: dblp;
2 Bloco 2: article; proceedings; inproceedings;
3 Bloco 3: @key; @mdate; author; title; pages; year; volume;
4 journal; number; ee; url; editor; booktitle; series; isbn;
5 crossref; authors; source title; issue; art no; Page start;
6 Page end; Page count; Cited by; DOI; Link; EID;
7 Bloco 4: @href; $;
```

Listagem 4.1 – Documento estrutural geral para o estudo de caso 1

A saída da Listagem 4.1 indica cada bloco encontrado entre os documentos com seus respectivos campos. No bloco 3 (linha 3), por exemplo, foram mesclados todos os campos distintos pertencentes a este bloco nos documentos de entrada 1, 2, 3 e 4.

Tabela 4.2 – Lista de referências 1 para o caso 1

Article – doc1	Pages – doc1, 3	editor – doc 2	Issue – doc 4
Proceedings – doc 2	Year – doc1, 2, 3, 4	Booktitle – doc 2,3	Art no – doc 4
Inproceedings – doc 1	Volume – doc1, 2,4	Series – doc 2	Page start - doc4
@key – doc1, 2, 3	Journal – doc 1	Isbn – doc 2	Page end – doc 4
@mdate – doc 1, 2, 3	Number – doc1	Crossref – doc 3	Page count – doc 4
author – doc 1, 3	Ee – doc1, 2, 3	Authors – doc4	Cited by – doc 4
Title – doc1, 2, 3, 4	url – doc1 ,2, 3	Source title – doc4	DOI – doc 4
Link – doc 4	EID - doc 4		

Por sua vez a lista de referências da Tabela 4.2 armazena todos os termos distintos e seus respectivos documentos de origem, pois ao mesclar os campos em um único se perderia a referência de origem destes. Por exemplo, o termo "*article*" ocorre apenas no documento de entrada 1.

A partir de então, são abstraídos os delimitadores e cada palavra é comparada com as demais pertencentes ao mesmo bloco. O passo seguinte é a etapa de **análise de similaridade** que consiste em aplicar as implementações das medidas propostas, sendo que para cada uma das três é gerada uma planilha em Excel representando uma matriz resultante.

A técnica de radicais produz apenas números inteiros zero ou um e as demais geram números reais neste intervalo. Portanto, a saída desta etapa é composta de três matrizes, conforme exemplificado na Figura 4.3, que representam: análise de radical (letra A), caractere (letra B) e conhecimento (letra C) aplicadas pelas implementações de *Porter Stemmer*, *Levenshtein* e *Lin*, respectivamente.

Porter Stemmer

	author	authors	number	issue	series
author	1	**1**	0	0	0
authors		1	0	0	0
number			1	0	0
issue				1	0
series					1

(A)

Levenshtein

	author	authors	number	issue	series
author	1	**0,875**	0,4286	0,1429	0,1429
authors		1	0,375	0,125	0,25
number			1	0,2857	0,2857
issue				1	0,2857
series					1

(B)

Wordnet: Lin

	author	authors	number	issue	series
author	1	**1**	0,1636	0,2906	0,1636
authors		1	0,1636	0,2906	0,1636
number			1	**1**	**0,8855**
issue				1	**0,8855**
series					1

(C)

Figura 4.3 – Exemplo de matrizes resultantes da análise de similaridade para o estudo de caso 1

A próxima etapa é fundamental no trabalho e diz respeito à **identificar equivalências**. A primeira atividade consiste em calcular equivalência por bloco, cujo resultado é uma única matriz do bloco, com os valores zero, indicando diferença e um indicando total similaridade.

Utilizando como exemplo a Figura 4.3, o par de palavras "*author*" e "*authors*", como possui valor 1 em uma das medidas é considerado equivalente. Isto também se aplica ao par "*number*" e "*issue*" que apresenta valor 1 para *Wordnet*. Por outro lado, para o caso dos pares "*number*" e "*series*" é aplicado a média ponderada, com os pesos do método AHP definidos na Seção 2.2.2, sobre os valores 0, 0.2857 e 0.8855, respectivamente. Ao aplicar o ponto de corte é gerada a matriz única do bloco.

A atividade seguinte, consolidar estrutura, consiste em verificar na planilha quais palavras foram detectadas como equivalentes. Estas compõem a segunda lista de referências, primeiro artefato desta etapa, representados pela Listagem 4.2.

```
Bloco 2: proceedings = inproceedings
Bloco 3: author = authors
Bloco 3: number = issue
```

Listagem 4.2 – Lista de referências 2 para o estudo de caso 1

O segundo artefato de saída desta etapa é gerado pela atividade remontar estrutura que reorganiza os blocos. Desse modo, a estrutura unificada completa, que representa apenas os campos com os delimitadores correspondentes ao bloco, é demonstrada na Listagem 4.3.

```
{
  dblp {
    article; proceedings; {
                @key; @mdate; author[]; title; pages; year;
                volume; journal; number; ee; url; editor[];
                booktitle; series {
                        @href; $;
                }
                isbn[]; crossref; source title; issue; art no;
                Page start; Page end; Page count; Cited by;
                DOI; Link; EID;
        }
  }
}
```

Listagem 4.3 – Estrutura unificada para o estudo de caso 1

Por fim, a partir da estrutura unificada e de ambas listas de referência, tem início a etapa de **representação da estrutura**.

Para tal, são aplicadas regras definidas com base no processamento de um arquivo JSON para gerar o esquema conceitual na notação de (JOVANOVIC; BENSON, 2013) e a tabela com os mapeamentos consolidados. O esquema extraído pelo processo é representado pela Figura 4.4 que possui como raiz a entidade originada pelo termo *dblp* e as demais entidades são aninhadas (do termo *EMBED*) nesta principal.

O segundo artefato gerado pelo processo são os mapeamentos da Tabela 4.3. Esta representa alguns mapeamentos do termo consolidado para seus correspondentes e documento de origem. O termo consolidado é expresso pela notação JSONPath que indica seu endereço no documento.

Considerando que, para esta amostra, os valores que poderiam ser considerados equivalentes foram recuperados, a Tabela 4.4 indica os índices de revocação e precisão.

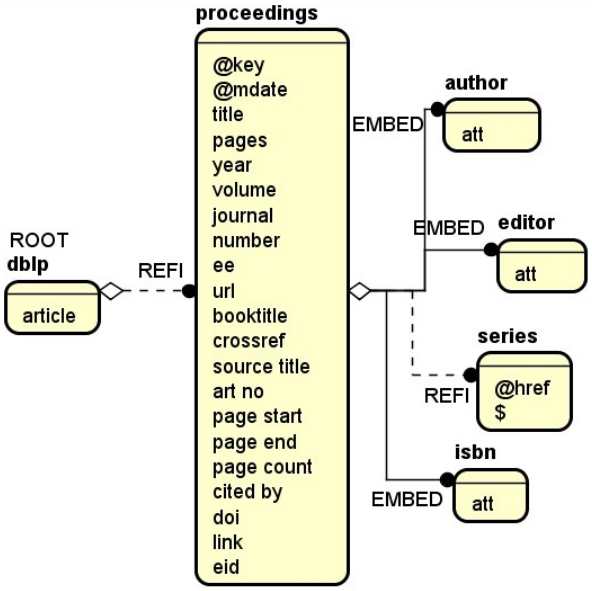

Figura 4.4 – Esquema conceitual extraído pelo processo para o estudo de caso 1

Tabela 4.3 – Mapeamentos para o estudo de caso 1

Consolidado	**Termo**	**Origem**
$.dblp.proceedings.author	$.authors	Doc 4
$.dblp.proceedings.number	$.issue	Doc 4
$.dblp.proceedings	$.dblp.inproceedings	Doc 3

Tabela 4.4 – Valores de revocação e precisão para o estudo de caso 1

Recuperados	Rec. Relevantes	Relevantes	Revocação	Precisão
3	3	3	100%	100%

Como a amostra é pequena, não em termos de números de documentos, mas sim com relação a variação da estrutura, apenas 3 campos poderiam ser inferidos como equivalentes. Tendo por consideração que estes foram recuperados, os índices expressam 100%.

Portanto, ao utilizar a abordagem de execução por blocos podem ser evitados problemas com relação a identificar homônimos incorretamente, porém restringe-se a aplicação do processo, no sentido de que, ao reduzir o escopo ao bloco não se pode comparar possíveis palavras que também sejam equivalentes, mas que estejam em outro nível de hierarquia dentro do JSON. De mesmo modo, outro fato que contribui para uma pequena amostra é de que estes documentos se referem quase todos a uma mesma base de dados com pequenas variações.

Os documentos utilizados no caso de teste foram adicionados ao banco de dados NoSQL orientado a documentos MongoDB, em uma única coleção, sendo realizado um backup do

banco de dados em formato JSON, disponível também na pasta do OneDrive.

4.4 Caso 2 - Executado como um único bloco

Neste segundo caso de teste, o processo de extração é realizado nos campos do documento como um todo, isto é, representando um único bloco. Para tal, considera-se que os campos de modo geral podem ser equivalentes a qualquer outro, independente do nível de hierarquia que se encontram e que representam informações de um mesmo contexto.

De forma a se obter maiores variações no esquema, este experimento é realizado com arquivos de mesmo domínio, mas pertencentes a fontes diferentes, cujo propósito é acessar de modo unificado, exemplificando uma outra aplicação do processo que se adapta a arquivos JSON de modo geral da *web*. Todavia estes podem ser armazenados em um banco de dados NoSQL orientado a documentos de mesmo modo, conforme foi realizado. O *backup* do banco de dados MongoDB desta coleção encontra-se disponível na pasta pública.

Para este caso de teste também são utilizados quatro documentos de entrada representados a seguir. A Figura 4.5 demonstra os dois primeiros documentos. O documento 1 se refere a uma publicação da base Bibsonomy. Por sua vez, o documento 2 apresenta uma referência a um artigo da biblioteca DBLP, ambos extraídos em formato XML.

Documento 1


```
{
  "Source": {
      "Author": {
         "Author": {
          "NameList": {
              "Person": { "First": "Rick",
                          "Last": "Cattell"  }
           }
          }
      },
      "BIBTEX_KeyWords": "cloud-computing, nosql",
      "City": "New York, NY, USA",
      "JournalName": "SIGMOD Rec.",
      "Month": "May",
      "Pages": "12-27",
      "Publisher": "ACM",
      "SourceType": "JournalArticle",
      "StandardNumber": " ISSN: 0163-5808 DOI:
http://doi.acm.org/10.1145/1978915.1978919",
      "Tag": "Cattell:2011:SSN:1978915.1978919",
      "Title": "Scalable SQL and NoSQL data stores",
      "URL":
"http://doi.acm.org/10.1145/1978915.1978919",
      "Volume": 39.0,
      "Year": 2011.0
   }
}
```


Fonte: https://www.bibsonomy.org/bibtex/2717bc3cdbc21c56c65e356f237be0203/hidders

Documento 2


```
{
  "dblp": {
    "article": {
      "@key": "journals/ijbpim/AlmeidaBF15",
      "@mdate": "2016-01-05",
      "author": [
        "Rafael Almeida",
        "Jorge Bernardino",
        "Pedro Furtado"
      ],
      "title": "\nTesting SQL and NoSQL
approaches for big data warehouse systems.\n",
      "pages": "322-334",
      "year": 2015,
      "volume": 7,
      "journal": "IJBPIM",
      "number": 4,
      "ee":
"http://dx.doi.org/10.1504/IJBPIM.2015.073656",
      "url":
"db/journals/ijbpim/ijbpim7.html#AlmeidaBF15"
    }
  }
}
```


Fonte: http://dblp.uni-trier.de/rec/xml/journals/ijbpim/AlmeidaBF15.xml

Figura 4.5 – Documentos de entrada 1 e 2 para o estudo de caso 2

A Figura 4.6 ilustra apenas um trecho do documento 3, por se tratar de um arquivo

grande, que se refere a uma publicação da base PubMedArticle. As reticências indicam continuidade. Assim, também o documento 4 representa uma publicação da biblioteca Scopus. O documento 3 foi extraído em formato XML e o documento 4 em CSV.

Documento 3


```
{
  "PubmedArticle": {
    "MedlineCitation": {
      "@Status": "In-Process",
      "@Owner": "NLM",
      "PMID": { ... },
      "DateCreated": { ... },
      "DateRevised": { ... },
      "Article": {
        "@PubModel": "Electronic-eCollection",
        "Journal": {
          "ISSN": {
            "@IssnType": "Electronic",
            "$": "1932-6203"
          },
          "JournalIssue": {
            "@CitedMedium": "Internet",
            "Volume": 11,
            "Issue": 12,
            "PubDate": { ... }
          },
          "Title": "PloS one",
          "ISOAbbreviation": "PLoS ONE"
        },
        "ArticleTitle": "A Scalable Data Access Layer
to Manage Structured Heterogeneous Biomedical Data.",
        ...
```


Fonte: https://www.ncbi.nlm.nih.gov/pubmed/?term=A+Scalable+Data+Access+Layer+to+Manage+Structured+Heterogeneous&report=xml&format=text

Documento 4


```
{
    "Authors": "Reniers V., Rafique A., Van Landuyt
D., Joosen W.",
    "Title": "Object-NoSQL Database Mappers: a
benchmark study on the performance overhead",
    "Year": 2017,
    "Source title": "Journal of Internet Services and
Applications",
    "Volume": 8,
    "Issue": 1,
    "Art. No.": 1,
    "Page start": "",
    "Page end": "",
    "Page count": "",
    "Cited by": "",
    "DOI": "10.1186/s13174-016-0052-x",
    "Link": " ... ",
    "Affiliations": " ... ",
    "Authors with affiliations": " ... ",
    "Correspondence Address": " ... ",
    "Editors": "",
    "Publisher": "Springer London",
    "ISSN": 18674828,
    "ISBN": "",
    "CODEN": "",
    "PubMed ID": "",
    "Language of Original Document": "English",
}
```


Fonte: https://goo.gl/TZ3dN5

Figura 4.6 – Documentos de entrada 3 e 4 para o estudo de caso 2

Na etapa de **pré-processamento** os campos são separados e condensados todos em um único arquivo, chamado documento estrutural geral, contendo apenas aqueles campos distintos. Nesta etapa os delimitadores são desprezados. A Listagem 4.4 representa a saída da etapa de pré-processamento.

```
1  source; author; namelist; person; first; last; bibtex_keywords;
2  city; journalname; month; pages; publisher; sourcetype;
3  standardnumber; tag; title; url; volume; year; dblp; article;
4  @key; @mdate; journal; number; ee; pubmedarticle; medlinecitation;
5  @status; @owner; pmid; @version; $; datecreated; day; daterevised;
6  @pubmodel; issn; @issntype; journalissue; @citedmedium; issue;
7  pubdate; isoabbreviation; articletitle; pagination; medlinepgn;
8  elocationid; @eidtype; @validyn; abstract; abstracttext;
9  authorlist; @completeyn; lastname; forename; initials;
10 identifier; @source; affiliationinfo; affiliation; language;
11 publicationtypelist; publicationtype; @ui; articledate; @datetype;
12 medlinejournalinfo; country; medlineta; nlmuniqueid; issnlinking;
13 authors; source title; art. no.; page start; page end; page count;
14 cited by; doi; link; affiliations; authors with affiliations;
15 correspondence address; editors; isbn; coden; pubmed id;
```

```
16 language of original document; abbreviated source title; eid;
```

Listagem 4.4 – Documento estrutural geral para o estudo de caso 2

O segundo artefato da etapa de pré-processamento é a lista de referências 1 que contém os campos e a referência de quais documentos ocorrem. Para tal, é demonstrada apenas parte desta na Tabela 4.5.

Tabela 4.5 – Lista de referências 1 para o caso 2

person - doc 1	first - doc 1	city - doc 1	language - doc 3
source - doc 1	month - doc 1, 3	affiliation - doc 3	affiliations - doc 4
author - doc 1, 2 e 3	pages - doc 1, 2	authors - doc 4	year - doc 1, 2, 3 e 4
namelist - doc 1	publisher - doc 1, 4	article - doc 2, 3	title - doc 1, 2 e 4

Assim, na etapa de **análise de similaridade**, são geradas matrizes gravadas em Excel de tamanho 92x92 para cada análise para a par. Esta etapa resulta em três matrizes com os resultados de cada técnica de similaridade textual aplicada, conforme exemplo de saída, na Figura 4.7.

De acordo com a Figura 4.7, a técnica de radicais identificou como equivalentes os termos "*affiliation*" e "*affiliations*". Por sua vez, apresentam valores altos de similaridade os termos "*authors*" e "*authorlist*" na técnica de Levenshtein, assim como "*volume*" e "*journal*" na técnica do Wordnet.

A próxima etapa do processo é **identificar equivalências**. A primeira atividade consiste em calcular equivalências, neste caso, tratando todas as palavras como pertencentes a um único bloco. O primeiro resultado é uma matriz com valores zero ou um indicando a similaridade ou não.

A atividade seguinte, realizada pelo Algoritmo 7, consiste em consolidar a estrutura, isto é, apagar os termos equivalentes mantendo a primeira ocorrência. Esta mesma atividade gera a lista de referências 2, conforme a Listagem 4.5, contendo os termos que foram considerados equivalentes.

```
1 source = author
2 number = issue
3 @source = source
4 affiliationinfo = affiliation
5 publicationtypelist = publicationtype
6 articletitle = articledate
7 country = city
8 medlineta = medlinepgn
9 source = authors
```

Porter Stemmer

	affiliationInfo	Affiliation	affiliations	authors	authorlist	journal	volume
affiliationInfo	1	0	0	0	0	0	0
Affiliation		1	**1**	0	0	0	0
affiliations			1	0	0	0	0
authors				1	0	0	0
authorlist					1	0	0
journal						1	0
volume							1

(A)

Levenshtein

	affiliationInfo	Affiliation	affiliations	authors	authorlist	journal	volume
affiliationInfo	1	**0,75**	**0,75**	0,25	0,25	0,1875	0,125
Affiliation		1	**0,9231**	0,25	0,25	0,1667	0,1667
affiliations			1	0,3846	0,2308	0,1538	0,1538
authors				1	**0,7273**	0,125	0,125
authorlist					1	0,1818	0,2727
journal						1	0,25
volume							1

(B)

Wordnet: Lin

	affiliationInfo	Affiliation	affiliations	authors	authorlist	journal	volume
affiliationInfo	1	-	-	-	-	-	-
Affiliation		1	1	0	-	0,0780	0,2558
affiliations			1	0	-	0,0780	0,2558
authors				1	-	0,1492	0,1810
authorlist					1	-	-
journal						1	**0,8415**
volume							1

(C)

Figura 4.7 – Exemplo de matrizes resultantes da análise de similaridade para o estudo de caso 2

```
10 author = authors
11 authorlist = authors
12 affiliationinfo = affiliations
13 affiliation = affiliations
14 isbn = issn
```

Listagem 4.5 – Lista de referências 2 para o estudo de caso 2

O próximo passo é o Algoritmo 8, responsável por remontar a estrutura reinserindo os delimitadores e blocos. Para tal, é escolhido o documento de origem que contenha o maior número de blocos para referência. Após, são executados testes para verificar se o termo está consolidado ou, caso contrário, é substituído por seu correspondente consolidado e este é apagado. Os campos que não forem substituídos na estrutura original do documento de referência escolhido são considerados como um bloco separado ligado na mesma raiz.

Deste modo, a atividade remontar estrutura gera como saída a estrutura unificada que consiste em um arquivo texto com campos e delimitadores prontos para a etapa seguinte. Para este exemplo, a saída é demonstrada na Figura 4.8.

```
1  {
2      pubmedarticle; {
3          medlinecitation; {
4              @status; @owner;
5              pmid; { @version; $; }
6              datecreated; { year; month; day; }
7              daterevised; article; {
8                  @pubmodel;
9                  journal; {
10                     isbn; { @issntype; }
11                     journalissue; { @citedmedium; volume; number; pubdate; }
12                     title;
13                     isoabbreviation;
14                 }
15                 articletitle;
16                 pagination; { medlineta;}
17                 elocationid; { @eidtype; @validyn; }
18                 abstract; { abstracttext; }
19                 authorlist; {
20                     @completeyn;
21                     source; [{
22                         lastname; forename; initials; identifier; { affiliation;}
23                     }]
24                 }
25                 language;
26                 publicationtypelist; { @ui;}
27                 {
28                     @datetype;
29                 }
30             }
31             medlinejournalinfo; { country; nlmuniqueid; issnlinking;}
32         }
33     }
34     {
35         namelist;
36         ...
```

Figura 4.8 – Exemplo parcial da estrutura unificada para o estudo de caso 2

O documento 3 de entrada da Figura 4.6 foi escolhido como documento de referência para aplicar esta atividade. Neste documento os campos que possuem equivalentes são substituídos e os demais são mantidos. Os termos que não se enquadram nesta situação são adicionados a um bloco que tem início na linha 34 da Figura 4.8.

A partir da estrutura consolidada e de ambas as listas de referência, tem início a etapa de **representação da estrutura**. Esta objetiva gerar os dois artefatos finais do processo de extração: os mapeamentos e o esquema conceitual.

A partir da segunda lista de referências, representada pela Listagem 4.5, é identificado que um termo pode corresponder a mais de um outro, assim como, mais de um termo consolidado pode se referir a mesma palavra, como por exemplo, tanto "*source*" como "authorlist" equivalem a "*authors*". O próximo passo é montar os mapeamentos, demonstrados na Figura 4.9, com base nas listas de referências 1 e 2 e na estrutura unificada. Desse modo, o termo consolidado é expresso em JSONPath e são montados os termos correspondentes com seus documentos de origem. Conforme apontado, *source* tem mais de um termo correspondente, bem como o termo *authorlist* também se refere a *authors*.

Consolidado	Termo	Origem
$..pubmedarticle.medlinecitation.article.authorlist.source	$.Source.author	Doc 1
	$.dblp.article.author	Doc 2
	$.pubmedarticle.medlinecitation.article.authorlist.author	Doc 3
	$.pubmedarticle.medlinecitation.article.authorlist.author.identifier.@source	Doc 3
	$.authors	Doc 4
$..pubmedarticle.medlinecitation.article.journal.journalissue.number	$.pubmedarticle.medlinecitation.article.journal.journalissue.issue	Doc 3
	$.issue	Doc 4
$..pubmedarticle.medlinecitation.article.authorlist.source.identifier.affiliation	$.pubmedarticle.medlinecitation.article.authorlist.author.affiliationInfo	Doc 3
	$.Affiliations	Doc 4
$..pubmedarticle.medlinecitation.article.publicationtypelist	$.pubmedarticle.medlinecitation.article.publicationtypelist.publicationtype	Doc 3
$..pubmedarticle.medlinecitation.medlinejournalinfo.country	$.Source.city	Doc 1
$..pubmedarticle.medlinecitation.article.pagination.medlineta	$.pubmedarticle.medlinecitation.article.pagination.Medlinepgn	Doc 3
$..pubmedarticle.medlinecitation.article.authorlist	$.Authors	Doc 4
$..pubmedarticle.e1.isbn	$.pubmedarticle.medlinecitation.article.journal.Issn	Doc 3
	$.issn	Doc 4

Figura 4.9 – Mapeamentos para o estudo de caso 2

Por fim, o esquema conceitual é gerado com base nas regras definidas partindo de um elemento raiz, no caso, o termo *pubmedarticle* de acordo com a Figura 4.10.

Neste exemplo, encontram-se várias entidades aninhadas em outras. Estas são representadas pelos termos *EMBED* para relacionamentos do tipo $1:n$ e *REFI* para relacionamentos do tipo $0:1$. Para alguns casos em que a entidade não possuía nome, foi atribuído *e0* e *e1*. O documento utilizado de referência para remontar a estrutura foi o documento 3 do PubMedArticle da Figura 4.6. Assim, os campos que restaram foram adicionados à entidade *e1* ligadas a raíz.

Para fins de avaliação, é apresentada a Tabela 4.6 com os índices de revocação e precisão.

Tabela 4.6 – Valores de revocação e precisão para o estudo de caso 2

Rec.	Rec. Rel.	Rel.	Revocação	Precisão
14	12	15	80%	85,71%

Ao utilizar uma abordagem que procura equivalência no documento como um todo, o número de documentos recuperados aumentou. Neste caso de teste, 15 campos poderiam ser considerados como relevantes, isto é, que representam a mesma informação, mas estão escritos de modo distinto. Foram recuperados 14 ao total, sendo que os termos faltantes poderiam ser *url* e *link*. Destes, 2 foram identificados incorretamente, são estes: $isbn = issn$ e $articledate = articletitle$. Este fato ocorreu devido ao índice da medida de Levenshtein, que compara o

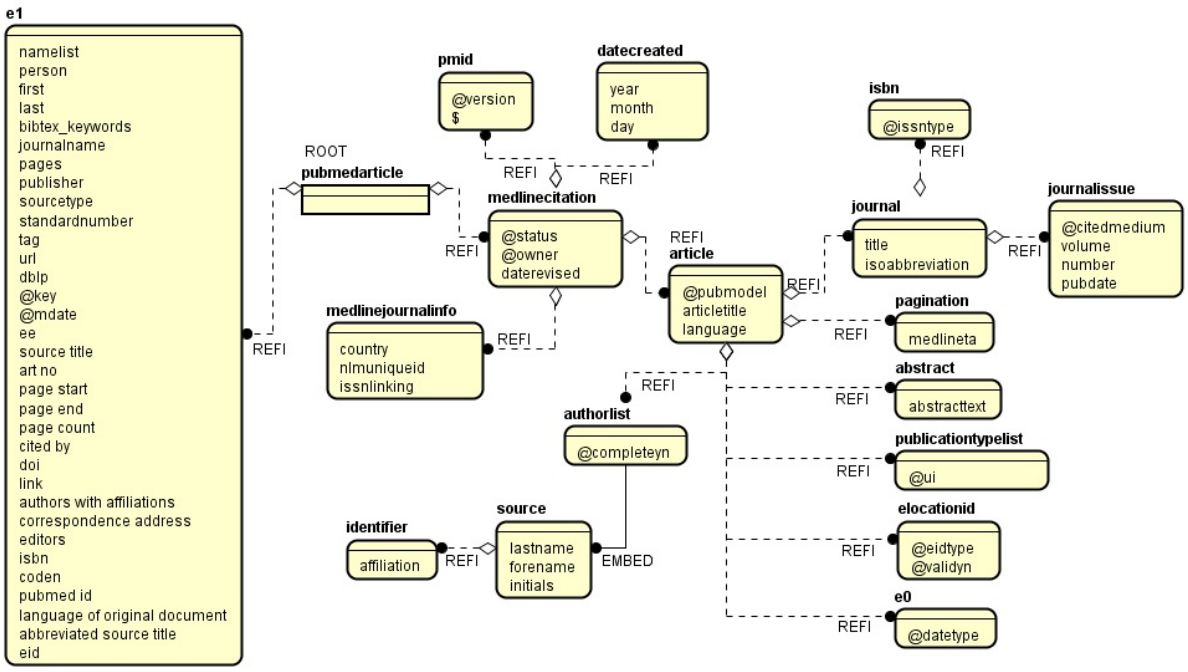

Figura 4.10 – Esquema conceitual extraído pelo processo para o estudo de caso 2

número de operações necessárias para se chegar de uma *String* a outra.

Deste modo, considera-se que o processo obteve bom desempenho, errando apenas em uma desvantagem da medida utilizada para identificar similaridade de caracteres. Os índices de precisão e revocação não são maiores devido ao tamanho da amostra. Também deve-se considerar que o processo de extração analisa apenas estruturas diferentes, o que apesar da flexibilidade de esquema permitida, não é usual nos conjuntos de dados armazenados em bancos de dados NoSQL pesquisados.

4.5 Considerações Finais

Este capítulo apresentou o estudo de caso realizado para o processo de extração proposto. Este propõe como entrada uma coleção de arquivos em formato JSON armazenados em um banco de dados NoSQL orientado a documentos, mas também pode ser aplicada a arquivos com o mesmo formato na *web* em geral, desde que pertencentes ao mesmo domínio de aplicação.

Todas as ferramentas utilizadas são descritas, sendo que o processo consiste na execução dos algoritmos definidos nas atividades. Algumas destas foram implementadas, outras utilizam implementações de terceiros, enquanto outras são executadas manualmente. Por exemplo, na etapa **pré-processamento** ambos algoritmos 1 e 2 foram implementados, em linguagem Java e

PHP respectivamente. Estes separam os campos dos dados e mesclam todos os documentos da coleção em um arquivo único, armazenando uma referência de em quais documentos de origem cada campo se encontrava.

Na etapa de **análise de similaridade** são utilizadas implementações existentes para técnica de radicais (*Porter Stemmer*), análise de caractere (*Levenshtein*) e análise de sinônimos (medida de Lin da implementação do Wordnet). Algumas implementações adicionais foram feitas para gerar uma matriz com resultados para cada par de palavras analisada, salvando-as em Excel.

Para **identificar equivalências** a atividade do algoritmo 6, calcular equivalência por bloco, é fundamental. Nela são realizados testes e aplicados o ponto de corte definido para caso apenas uma técnica apresente valor, e o outro ponto de corte para caso se aplique a média ponderada com definição de pesos através do método AHP. As demais atividades montam uma estrutura consolidada e geram uma lista de referências que guarda as equivalências inferidas.

A última etapa, **representar estrutura** consiste em percorrer o arquivo consolidado aplicando regras de conversão definidas com base na gramática de um JSON. Este processo é realizado manualmente, mas é gerada uma visualização do esquema conceitual através de um modelo ER na notação IDEF1X.

A seguir foram detalhados dois casos de teste. O primeiro, executado por blocos, é mais adequado a documentos com esquemas semelhantes, pois facilita na identificação de equivalência entre os níveis de blocos de cada documento. Para este processo é necessária a intervenção de um especialista do domínio. Este possibilita evitar inconsistências, do tipo palavras homônimas, ao analisar apenas blocos equivalentes. Por outro lado, este modo de execução restringe a aplicação do processo.

O segundo é executado como um único bloco sem necessitar intervenção de um especialista de domínio. Deste modo é necessário que todos os campos do documento possam ser tidos como equivalentes com outros em qualquer nível. Possivelmente, esta proposta de execução se adapte melhor a arquivos de formato JSON em geral, pertencentes ao mesmo domínio, que se queira acessar de modo unificado. Como nestes arquivos a estrutura tende a ter maiores variações, o processo de extração pode ser melhor explorado, encontrando uma amostra maior.

Dessa forma o processo para extração de esquemas conceituais em bancos de dados NoSQL orientados a documentos permite duas formas de execução. Esta explora a flexibilidade de esquemas deste modelo, ao identificar palavras correspondentes em sinônimos, grafia similar

e mesmo radical. Também aplica-se a demais formatos de arquivos de entrada que seja possível converter para JSON. Para a avaliação foram utilizados os índices de revocação e precisão que apontam bons resultados, sendo os erros gerados por exemplo, em palavras muito parecidas em grafia, mas com significados distintos ou que segundo o dicionário é um sinônimo, mas não se aplica no domínio testado.

5 CONCLUSÃO E TRABALHOS FUTUROS

Esta dissertação tem como propósito extrair esquemas conceituais em fontes de dados JSON, com foco em banco de dados NoSQL orientado a documentos. Esta categoria de banco de dados ganhou destaque nos últimos anos com as soluções MongoDB e CouchDB, por exemplo. Uma das principais características de um modelo NoSQL diz respeito à flexibilidade de esquemas. No intuito de explorá-la, o processo de extração visa identificar equivalências nos campos de documentos JSON que estejam escritos de modo diferente, seja por falta de padronização ou por equívoco, mas que representem a mesma informação. Dessa forma, utiliza técnicas de similaridade que abrangem grafia similar, sinônimos e equivalência de radical da palavra.

A principal contribuição diz respeito ao processo para extração de esquemas conceituais que tem como entrada uma coleção de documentos em formato JSON. O processo aplica-se entre documentos e dentro do documento, gerando relacionamentos do tipo $1 : n$ ou $0 : n$, tendo em vista que no modelo NoSQL estes indicam que determinada entidade está aninhada em um elemento raiz. Apesar de ser executada com base na gramática JSON, ela permite ser aplicada a outros formatos como XML e CSV, que devem ser convertidos antes do início do processo.

O processo consiste em executar uma sequência de dez algoritmos descritos pelas atividades e subatividades das etapas de (1) pré-processamento, (2) análise de similaridade, (3) identificação de equivalência e (4) representação da estrutura. Alguns destes foram implementados, outros aplicados com implementações de terceiros e outros executados manualmente.

Em (1) são descartados os dados e mesclados todos os documentos em um único arquivo. A partir de então, em (2) são aplicadas em paralelo as três técnicas propostas para cada par de palavras, gerando matrizes com valores de similaridade no intervalo de zero e um, isto é, $\{x \in R | 0 \leq x \leq 1\}$. Desse modo, em (3), com base em determinado ponto de corte, são inferidas equivalências entre pares de palavras, produzindo uma estrutura unificada.

Outra contribuição do trabalho diz respeito a definição de regras de conversão de esquema JSON unificado em um esquema conceitual. Estas regras foram adaptadas de (IZQUIERDO; CABOT, 2013) e tem como base a gramática JSON. Nesse sentido, blocos correspondem a entidades e seus campos internos a atributos. Conforme o símbolo delimitador do bloco, é inferido o relacionamento que, nesta proposta, são sempre aninhados a uma outra entidade.

Assim sendo, na etapa (4) são aplicadas estas regras definidas gerando uma representação visual do esquema conceitual extraído, bem como mapeamentos dos campos consolidados para seus termos correspondentes. A extração de um esquema unificado pode ser útil para, em um trabalho futuro, possibilitar a submissão de consultas sobre ele, pois um mapeamento indica a quais outros termos aquele termo consolidado se refere e aponta os respectivos documentos de origem dos termos correspondentes.

O processo de extração proposto foi desenvolvido para coleções de documentos JSON armazenados em NoSQL, mas pode ser aplicado a arquivos neste formato da *web* em geral que se pretenda acessar de modo unificado, desde que pertencentes ao mesmo domínio. Similarmente, o processo permite duas formas de execução: (a) separando por blocos, que possivelmente se adapta melhor a casos em que a estrutura é mais semelhante, pelo fato de identificar equivalência entre blocos de diferentes documentos, e (b) tratando todos os campos como pertencentes a um único bloco. Este último é mais adequado para coleções com maior variação na estrutura desde que permita encontrar equivalências no documento como um todo.

Os testes realizados demonstram que o processo possui mais abrangência conforme maior número de variações, mantendo bons índices de revocação e precisão, equivocando-se apenas em casos de palavras muito semelhantes mas com significados distintos, ou alguma inconsistência do dicionário de sinônimos.

Conforme o exposto, o trabalho possui como premissa que as palavras pertençam ao mesmo contexto e, no caso de execução em blocos, considera que elas possuam um mesmo pai. No entanto, pode haver casos de palavras que apresentam a mesma grafia e possuem sentidos diferentes, chamados de homônimos. Uma solução futura poderia adicionar um tratamento para estas exceções.

Outro ponto é de que esta se aplica a uma coleção do banco de dados por vez, assim não tratando possíveis formas de relacionamento entre coleções. Isto não é usual em NoSQL, mas, por exemplo, dada uma coleção cliente e outra produto, o administrador pode adicionar um campo referência de cliente em produto e vice-versa. Este "relacionamento" poderia ser considerado do tipo $n : n$ e explorado em uma melhoria do processo.

Com o crescimento do volume de dados e da popularização de dados semiestruturados como JSON, é necessário preocupar-se com esquemas para que se possa desenvolver aplicações que os acessem de modo coerente. Um próximo passo do trabalho seria a implementação dos demais algoritmos propostos para desenvolvimento de uma ferramenta.

O processo utiliza pontos de corte de modo a maximizar os índices de revocação e precisão, contudo, a corretude do modelo não é verificada. Esta é uma possibilidade a ser explorada futuramente. A validação pode ocorrer de diferentes abordagens, como por exemplo, (*i*) quanto às equivalências detectadas e (*ii*) quanto à notação utilizada. O ponto (*i*) objetiva verificar se as palavras identificadas como equivalentes ao final do processo estão corretas. O aspecto (*ii*) visa analisar se a notação gerada encontra-se correta e adequada ao domínio de aplicação. A notação de modelo conceitual aplicada neste trabalho, objetiva explorar as características particulares dos modelos orientados a agregados.

Dessa forma esta dissertação apresentou um processo para extração de esquema conceitual em fontes de dados JSON, com foco em bancos de dados NoSQL orientados a documentos, baseada em técnicas de similaridade de texto. Outros trabalhos relacionados como (IZQUIERDO; CABOT, 2013), (KLETTKE et al., 2015), (RUIZ; MORALES; MOLINA, 2015) também extraem esquema para dados JSON armazenados em um banco de dados NoSQL orientado a documentos ou não, porém, esta proposta diferencia-se ao explorar a flexibilidade de esquemas, identificando campos equivalentes em termos de sinônimo, radical da palavra e caractere gerando um esquema unificado.

REFERÊNCIAS

ATZENI, P. Models for NoSQL Databases: a contradiction? In: INTERNATIONAL CONFERENCE ON CONCEPTUAL MODELING. **Anais...** [S.l.: s.n.], 2015. p.133–133.

BENSON, S. R. **Polymorphic Data Modeling**. 2014. Dissertação (Mestrado em Ciência da Computação) — Georgia Southern University.

BOAGLIO, F. **MongoDB**: construa novas aplicações com novas tecnologias. [S.l.]: Editora Casa do Código, 2015.

BRITO, R. W. Bancos de dados NoSQL x SGBDs relacionais: análise comparativa. **Faculdade Farias Brito e Universidade de Fortaleza**, [S.l.], 2010.

BUGIOTTI, F. et al. Database design for NoSQL systems. In: INTERNATIONAL CONFERENCE ON CONCEPTUAL MODELING. **Anais...** Springer, 2014. p.223–231.

C. F. DORNELES, R. M. G. **Aplicação de Funções de Similaridade e Detecção de Diferenças em Grandes Volumes de Dados Distribuídos**. [S.l.]: Rio de Janeiro: Editora PUC-Rio, v. 1, p. 233-272, 2008. Jornadas de Atualização em Informática - JAI.

CASTREJÓN, J. C. et al. ExSchema: discovering and maintaining schemas from polyglot persistence applications. In: ICSM. **Anais...** [S.l.: s.n.], 2013. p.496–499.

CHANG, F. et al. Bigtable: a distributed storage system for structured data. **ACM Transactions on Computer Systems (TOCS)**, [S.l.], v.26, n.2, p.4, 2008.

CHRISTEN, P. **Data matching**: concepts and techniques for record linkage, entity resolution, and duplicate detection. [S.l.]: Springer Science & Business Media, 2012.

COELHO, A. R. **Stemming para a língua portuguesa**: estudo, análise e melhoria do algoritmo rslp. 2007. 69f. Monografia (Graduação em Ciência da Computação), Universidade Federal do Rio Grande do Sul, Porto Alegre.

DA SILVA, R. et al. Measuring quality of similarity functions in approximate data matching. **Journal of Informetrics**, [S.l.], v.1, n.1, p.35–46, 2007.

DORNELES, C. F. **Similaridade de dados**. Acessado em outubro 2016, http://www.inf.ufsc.br/ r.mello/ine5454/Similaridade.pdf.

FERNANDO, S.; STEVENSON, M. A semantic similarity approach to paraphrase detection. In: ANNUAL RESEARCH COLLOQUIUM OF THE UK SPECIAL INTEREST GROUP FOR COMPUTATIONAL LINGUISTICS, 11. **Proceedings...** [S.l.: s.n.], 2008. p.45–52.

GARG, B.; KAUR, K. G. **Schema Extraction of Document Database-MongoDB**. 2015. Tese (Doutorado em Ciência da Computação) — Thapar University.

GOESSNER, S. **JSONPath - XPath for JSON**. Acessado em novembro 2016, http://goessner.net/articles/JsonPath/.

GOMAA, W. H.; FAHMY, A. A. A survey of text similarity approaches. **International Journal of Computer Applications**, [S.l.], v.68, n.13, 2013.

INTERNATIONAL, E. **The JSON Data Interchange Format**. Acessado em novembro 2016, http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf.

IZQUIERDO, J. L. C.; CABOT, J. Discovering implicit schemas in json data. In: INTERNATIONAL CONFERENCE ON WEB ENGINEERING. **Anais...** [S.l.: s.n.], 2013. p.68–83.

IZQUIERDO, J. L. C.; CABOT, J. JSONDiscoverer: visualizing the schema lurking behind json documents. **Knowledge-Based Systems**, [S.l.], v.103, p.52–55, 2016.

JARO, M. A. Advances in record-linkage methodology as applied to matching the 1985 census of Tampa, Florida. **Journal of the American Statistical Association**, [S.l.], v.84, n.406, p.414–420, 1989.

JOVANOVIC, V.; BENSON, S. Aggregate Data Modeling Style. **SAIS 2013**, [S.l.], p.70–75, 2013.

KLETTKE, M. et al. Schema Extraction and Structural Outlier Detection for JSON-based NoSQL Data Stores. In: BTW. **Anais...** [S.l.: s.n.], 2015. v.2105, p.425–444.

LEVENSHTEIN, V. I. Binary codes capable of correcting deletions, insertions and reversals. In: SOVIET PHYSICS DOKLADY. **Anais...** [S.l.: s.n.], 1966. v.10, p.707.

LIMA, C. de. **Projeto lógico de bancos de dados NOSQL documento a partir de esquemas conceituais entidade-relacionamento estendido (EER)**. 2016. Dissertação (Mestrado em Ciência da Computação) — Universidade Federal de Santa Catarina.

LIN, D. An information-theoretic definition of similarity. In: ICML. **Anais...** [S.l.: s.n.], 1998. v.98, p.296–304.

MILLER, G. A. WordNet: a lexical database for english. **Communications of the ACM**, [S.l.], v.38, n.11, p.39–41, 1995.

MITSCH, S. et al. User profile integration made easy: model-driven extraction and transformation of social network schemas. In: INTERNATIONAL CONFERENCE ON WORLD WIDE WEB, 21. **Proceedings...** [S.l.: s.n.], 2012. p.939–948.

MOH, C.-H.; LIM, E.-P.; NG, W.-K. DTD-Miner: a tool for mining dtd from xml documents. In: ADVANCED ISSUES OF E-COMMERCE AND WEB-BASED INFORMATION SYSTEMS, 2000. WECWIS 2000. SECOND INTERNATIONAL WORKSHOP ON. **Anais...** [S.l.: s.n.], 2000. p.144–151.

NESTOROV, S.; ABITEBOUL, S.; MOTWANI, R. Inferring structure in semistructured data. **ACM SIGMOD Record**, [S.l.], v.26, n.4, p.39–43, 1997.

ORENGO, V. M.; HUYCK, C. R. A Stemming Algorithmm for the Portuguese Language. In: SPIRE. **Anais...** [S.l.: s.n.], 2001. v.8, p.186–193.

PORTER, M. **The Porter Stemming Algorithm**. Acessado em novembro 2016, http://tartarus.org/martin/PorterStemmer/.

RESNIK, P. Using information content to evaluate semantic similarity in a taxonomy. **arXiv preprint cmp-lg/9511007**, [S.l.], 1995.

RUIZ, D. S.; MORALES, S. F.; MOLINA, J. G. Inferring versioned schemas from nosql databases and its applications. In: INTERNATIONAL CONFERENCE ON CONCEPTUAL MODELING. **Anais...** [S.l.: s.n.], 2015. p.467–480.

SAATY, T. L. Decision making with the analytic hierarchy process. **International journal of services sciences**, [S.l.], v.1, n.1, p.83–98, 2008.

SADALAGE, P. J.; FOWLER, M. **NoSQL distilled**: a brief guide to the emerging world of polyglot persistence. [S.l.]: Pearson Education, 2012.

SANTOS, J. B. dos et al. Automatic threshold estimation for data matching applications. **Information Sciences**, [S.l.], v.181, n.13, p.2685–2699, 2011.

SCHERZINGER, S.; CERQUEUS, T.; ALMEIDA, E. C. de. ControVol: a framework for controlled schema evolution in nosql application development. In: IEEE 31ST INTERNATIONAL CONFERENCE ON DATA ENGINEERING, 2015. **Anais...** [S.l.: s.n.], 2015. p.1464–1467.

S.D. **NoSQL-Database.Org**. Acesso em maio de 2017, http://nosql-database.org/index.html.

SHARP, J. et al. Data Access for Highly-Scalable Solutions: using sql, nosql, and polyglot persistence. **Microsoft patterns & practices**, [S.l.], 2013.

STASIU, R. K. **Avaliação da qualidade de funções de similaridade no contexto de consultas por abrangência**. 2007. Tese (Doutorado em Ciência da Computação) — Universidade Federal do Rio Grande do Sul.

STÖRL, U. et al. Schemaless NoSQL Data Stores-Object-NoSQL Mappers to the Rescue? In: BTW. **Anais...** [S.l.: s.n.], 2015. p.579–599.

VARGA, V.; JÁNOSI-RANCZ, K. T.; KÁLMÁN, B. Conceptual Design of Document NoSQL Database with Formal Concept Analysis. **Acta Polytechnica Hungarica**, [S.l.], v.13, n.2, 2016.

VIEIRA, M. R. et al. Bancos de Dados NoSQL: conceitos, ferramentas, linguagens e estudos de casos no contexto de big data. **Simpósio Brasileiro de Bancos de Dados**, [S.l.], 2012. Minicurso.

WINKLER, W. E. String Comparator Metrics and Enhanced Decision Rules in the Fellegi-Sunter Model of Record Linkage, [S.l.], 1990.

WU, Z.; PALMER, M. Verbs semantics and lexical selection. In: ASSOCIATION FOR COMPUTATIONAL LINGUISTICS, 32. **Proceedings...** [S.l.: s.n.], 1994. p.133–138.

XPATH. **XQuery 1.0 and XPath 2.0 Data Model (XDM)**. Acessado em novembro 2016, https://www.w3.org/TR/xpath-datamodel/.