Ferramenta para Construção de Ontologia a Partir de Dados Não Estruturados

Allan Renato Sabino[1], Roberto Heinzle[2]

[1]Curso de Ciência da Computação
Universidade Regional de Blumenau (FURB) – Blumenau, SC – Brasil

[2]Departamento de Sistemas e Computação
Universidade Regional de Blumenau (FURB) – Blumenau, SC – Brasil

allanrenatosabino@gmail.com, heinzle@furb.br

Abstract. *This paper presents the specification and implementation of a tool to generate, from unstructured text, knowledge bases in the form of a Web Ontology Language (OWL) ontology. The process is done in two stages: knowledge discovery and construction of the knowledge base. In the first stage, are used Text Mining (TM) technologies and Natural Language Processing (NLP) to discover knowledge from a collection of textual documents. Finally, it is created an ontology based on the OWL tags specification from the Protégé tool. The ontology created was then edited by the Protégé tool, thereby demonstrating that it is compatible with standard OWL labels used by it.*

Resumo. *Este trabalho apresenta a especificação e implementação de uma ferramenta para a criação de bases de conhecimento na forma de ontologia na linguagem Web Ontology Language (OWL) a partir de textos não estruturados. O processo se divide em duas etapas: descoberta do conhecimento e construção da base de conhecimento. Na primeira etapa são utilizadas as tecnologias da Mineração de Texto (MT) e do Processamento de Linguagem Natural (PLN) para descobrir conhecimento a partir de uma coleção de documentos textuais. Por fim, é utilizada a especificação de tags OWL da ferramenta Protégé para criar uma ontologia. A ontologia criada foi então editada pela ferramenta Protégé, demonstrando assim que ela é compatível com o padrão de tags OWL utilizado pela mesma.*

1. Introdução

De acordo com Kuechler (2007, p. 86), no período de 2003 a 2010 a quantidade de informações no universo digital passou de cinco hexabytes (aproximadamente cinco bilhões de gigabytes) para 988 hexabytes. Segundo o autor, cerca de 80% destes dados estão em formato não estruturado, dos quais uma parte significativa são textos. Marcacini, Moura e Rezende (2011, p. 7) afirmam que "a organização inteligente dessas coleções textuais é de grande interesse para a maioria das organizações, pois agiliza os processos de busca e recuperação da informação".

Marcacini, Moura e Rezende (2011, p. 7) afirmam que a transformação de dados brutos e informação em conhecimento facilita sua análise e compreensão pelo gestor da organização. Assim sendo, busca-se criar soluções para automatizar este processo, diminuindo a intervenção humana para a descoberta (também conhecida como extração) do conhecimento. Um método utilizado para esse propósito é a Mineração de Texto (MT), que é definida como um conjunto de técnicas usadas para navegar, organizar e

descobrir conhecimento em bases de texto. A MT, por sua vez, pode ser apoiada, de acordo com Aranha et al. (2004, p. 105), pelo Processamento de Linguagem Natural (PLN). Utilizando conhecimento da área de linguística, o PLN permite aproveitar ao máximo o conteúdo do texto, extraindo entidades e seus relacionamentos, detectando sinônimos, corrigindo palavras escritas de forma incorreta e ainda desambiguizando-as.

A simples descoberta e extração do conhecimento, entretanto, pode não ser suficiente para resolver a questão da transformação de dados brutos em conhecimento, podendo ser interessante estruturá-lo na forma de uma Base de Conhecimento (BC). A construção de uma BC é viabilizada com os mecanismos de Representação do Conhecimento (RC). Esta é descrita, por Heinzle (2011, p. 93), como um conjunto de sentenças em uma linguagem formal, para a qual são definidas uma semântica e um conjunto de regras de inferência capazes de gerar novas sentenças, a partir das sentenças disponíveis. Um exemplo de formalismo que tem como propósito representar conhecimento é uma ontologia escrita com a linguagem OWL.

Diante desse cenário, o trabalho investigativo desenvolvido e ora apresentado neste artigo teve como objetivo a construção de uma ferramenta para a criação automática de BC, na forma de uma ontologia OWL, a partir de textos não estruturados, através das tecnologias MT e PLN.

2. Fundamentação Teórica

Nesta seção são apresentados os principais ferramentais teóricos estudados para o desenvolvimento do referido trabalho. A seção 2.1 trata a respeito da representação do conhecimento, abordando ontologias e suas linguagens de representação. A seção seguinte (2.2) discursa acerca da MT, apontando e descrevendo as suas etapas. Por fim, na seção 2.3 é explanado sobre PLN, conceituando-a e descrevendo sua utilização na ferramenta desenvolvida.

2.1. Representação do conhecimento

Existem múltiplos formalismos e ferramentas para representar computacionalmente o conhecimento. Todos têm como desafio central a construção de dispositivos computacionais capazes de simular, na máquina, algumas características da inteligência humana. Diferente dos dados, que são armazenados em bancos de dados, o conhecimento necessita de estruturas mais complexas para seu armazenamento. Estas estruturas são chamadas de BC e são criadas fazendo-se uso de formalismos próprios, dentre os quais destaca-se a ontologia.

A palavra ontologia é de origem grega e significa *ontos* (ser) + *logos* (conhecimento sobre), tendo sido criada entre os séculos XVII e XVIII por filósofos alemães para denominar o ramo da filosofia que trata da natureza e organização do ser. Heinzle (2011, p. 106) descreve que, a partir do início da década de 90 do século passado, o termo foi adotado pelas áreas da Engenharia do Conhecimento, dos Sistemas de Informação e da Ciência da Computação, onde recebeu adaptações em sua definição e passou a ter outra interpretação. Para Chandrasekaran e Josephson (1999, p. 22), o termo ontologia na Engenharia do Conhecimento e na Ciência da Computação, refere-se à representação de um vocabulário relacionado a um certo domínio, onde a qualificação não está no vocabulário mas sim nos conceitos expostos por ele.

Heinzle (2011, p. 107) explana sobre os componentes básicos das ontologias. Segundo ele, as classes são as unidades básicas de toda ontologia. Elas representam coleções de elementos que possuem atributos iguais e formam conceitos que definem

um determinado objeto. Os conceitos representam todas as coisas relacionadas ao domínio que se pretende modelar. As ligações entre estes conceitos ocorrem com base nos relacionamentos ou relações. Estes descrevem as interações entre os conceitos, as quais representam os relacionamentos semânticos envolvidos no domínio. As relações são descritas como propriedades binárias que descrevem características e relacionamentos entre classes ou indivíduos por meio dos quais é possível afirmar fatos sobre eles. Relacionadas às propriedades existem declarações para impor a elas restrições e caracterizações. *Range* (escopo) e *domain* (domínio) são os elementos utilizados para definir os conjuntos de elementos válidos envolvidos numa relação. O escopo é a classe de que parte a propriedade, e o domínio é a classe em que ela chega. As propriedades conectam indivíduos de um escopo a indivíduos de um domínio Os axiomas são regras relativas às relações que devem obrigatoriamente serem cumpridas pelos elementos de uma ontologia, sendo restrições. As instâncias representam os elementos ou objetos da ontologia, ou seja, são exemplares individuais das classes.

Quanto à representação de uma ontologia, existem linguagens próprias para denota-las. Entre elas está a OWL, que é uma linguagem que integra as tecnologias recomendadas pelo consórcio W3C desde fevereiro de 2004 (W3C, 2004). É baseada na sintaxe do *eXtensive Markup Language* (XML) e *Resource Description Framework Schema* (RDFS), tratando-se de uma revisão da linguagem DAML+OIL. Heinzle (2011, p. 115) registra que a OWL mantém, em comum com as linguagens predecessoras, a sintaxe e a lógica formal descritiva como mecanismo para expressar semântica por axiomas lógicos.

A estrutura padrão de uma ontologia OWL, escrita com a ferramenta Protégé (PROTÉGÉ, 2013), é composta por duas tags: `<Ontology>` e `</Ontology>`. A primeira *tag* indica o início da ontologia, enquanto a segunda a finaliza. Após isso, a ontologia é descrita como um conjunto de declarações, cada qual indicando um ferramental utilizado para representar o conhecimento (classe ou relacionamento). As declarações são representadas através de blocos contidos entre as tags: `<Declaration>` e `</Declaration>`.

Para se definir uma classe dentro de um arquivo OWL utiliza-se a *tag* `<Class IRI="#nomeDaClasse"/>`, onde `nomeDaClasse` é definida como uma *Internationalized Resource Identifier* (IRI) com o nome que se deseja denominar a classe. Uma *Object Property* denota a relação entre classes. Para descrevê-la faz-se uso da *tag* `<ObjectProperty IRI="#nomeDaPropriedade"/>`, onde o `nomeDaPropriedade` é denotado através de uma IRI. A *Object Property* contém um domínio e um escopo. A definição do domínio ocorre colocando uma *Object Property* e uma classe entre as *tags* `<ObjectPropertyDomain>` e `</ObjectPropertyDomain>`. A definição de um escopo é envolta pelas *tags* `<ObjectPropertyRange>` e `</ObjectPropertyRange>`.

No Quadro 1 é exposta uma ontologia OWL escrita no Protégé. Nas linhas 3 e 6 são criadas as classes `Comprador` e `Carro` respectivamente. Na linha 9 é criada a *Object Property* `serDono`. O código entre as linhas 11 e 14 atribui a classe `Comprador` como domínio da propriedade `serDono`. Por fim, entre as linhas 15 e 18 é atribuída a classe `Carro` como escopo da propriedade `serDono`.

Quadro 1. Estrutura de um arquivo OWL escrito com o Protégé

```
1  <Ontology>
2       <Declaration>
3            <Class IRI="#Comprador"/>
4       </Declaration>
5       <Declaration>
```

```
6            <Class IRI="#Carro"/>
7       </Declaration>
8       <Declaration>
9            <ObjectProperty IRI="#serDono"/>
10      </Declaration>
11      <ObjectPropertyDomain>
12           <ObjectProperty IRI="#serDono"/>
13           <Class IRI="#Comprador"/>
14      </ObjectPropertyDomain>
15      <ObjectPropertyRange>
16           <ObjectProperty IRI="#serDono"/>
17           <Class IRI="#Carro"/>
18      </ObjectPropertyRange>
19 </Ontology>
```

2.2. Descoberta de conhecimento com mineração de texto

Segundo Wives (2000), entre os métodos para a descoberta de conhecimento em bases textuais está a MT. Aranha (2007, p. 40), classifica o processo de MT em cinco macro etapas: coleta, pré-processamento, indexação, mineração e análise da informação.

Konchady (2006, p. 70) afirma que a primeira etapa do processo de MT tem como função formar a base textual de trabalho. Soares (2013, p. 50) registra que é importante que os documentos sejam relevantes ao domínio da aplicação do conhecimento a ser extraído, pois a seleção de documentos irrelevantes pode prejudicar o processo de MT, aumentando a dimensionalidade dos dados desnecessariamente. O pré-processamento, no entendimento de Soares (2013, p. 51), é responsável por criar uma representação mais estruturada do texto, capaz de alimentar algoritmos de descoberta de conhecimento. Aranha (2007, p. 42) descreve o pré-processamento como um conjunto de transformações realizadas sobre alguma coleção de textos com o objetivo de fazer com que esses passem a ser estruturados.

Para a transformação do texto não estruturado num modelo de representação estruturado, várias etapas são realizadas. Segundo Soares (2013, p. 52), o primeiro passo do processo de pré-processamento é a tokenização ou atomização, tendo como objetivo dividir um documento textual em unidades mínimas, mas que exprimam a mesma semântica do documento original. O autor utiliza o termo *token* para designar estas unidades que, geralmente, correspondem a uma única palavra do texto. A abordagem baseada em palavras possui certas vantagens pois estas, geralmente, possuem expressividade semântica, motivo pelo qual foi adotada neste trabalho. Conforme Soares (2013, p. 54), uma vez realizada a tokenização, o passo seguinte é a identificação do que pode ser desconsiderado nos passos posteriores do processamento dos textos, pois nem todas as palavras dos documentos devem ser adicionadas na estrutura de índice. Bastos (2006, p. 37) denomina esses *tokens* de *stopwords* e descreve que os mesmos fazem parte de uma lista chamada *stoplist*.

A terceira etapa da MT é a indexação. Manning, Raghavan e Schutze (2007, p. 140) conceituam a indexação como sendo a fase responsável por criar estruturas de dados denominadas índices, capazes de permitir a consulta a uma informação gerada com base na etapa de pré-processamento, sem que seja necessário analisar toda a base textual. O resultado do processo de indexação é o arquivo de índices. Konchady (2006, p. 50) afirma que existem diversas técnicas para este processo, sendo que a mais utilizada é a dos índices invertidos. Trata-se de uma estrutura de dados composta de uma lista ordenada, geralmente denominada vocábulo ou vocabulário, que armazena todas as palavras distintas encontradas nos textos e os documentos nos quais elas ocorrem.

A etapa seguinte da MT é denominada de mineração, e é nela que ocorre a busca efetiva por conhecimento a partir dos dados. Segundo Soares (2013, p. 65), a mineração compreende a aplicação de algoritmos sobre os dados de forma a extrair o conhecimento implícito neles. No presente trabalho foram aplicados os chamados métodos linguísticos para implementação desta etapa. Foi utilizada a classificação gramatical das palavras para descobrir padrões e, assim, conhecimento. Para tal, dois padrões foram adotados: todas as palavras que forem classificadas morfologicamente como substantivo ou nome próprio são consideradas classes da ontologia e verbos são considerados relacionamentos. Caso um verbo em uma sentença esteja entre duas classes ele é considerado um relacionamento, caso contrário ele é desconsiderado.

A última etapa da MT é a Análise da Informação. Para Zhu e Davidson (2007, p. 150), esta etapa abrange o tratamento do conhecimento obtido na etapa de mineração, através da análise, visualização e interpretação deste. Segundo os autores, esse tratamento tem como objetivo avaliar a utilidade do conhecimento descoberto.

2.3. Processamento de linguagem natural

Com o intuito de que o computador interaja com um ser humano através da linguagem natural, surgiu a área de pesquisa denominada de PLN. Segundo Müller (2003, p. 2), o processo de PLN é dividido em quatro etapas: análise morfológica, análise sintática, análise semântica e análise pragmática. A análise morfológica estuda a construção das palavras, com seus radicais e afixos, que correspondem às partes estáticas e variantes da palavra, além das classes gramaticais, com suas inflexões verbais. A análise sintática diz respeito ao estudo das relações formais entre as palavras em uma sentença. A análise semântica é um processo de mapeamento de sentenças de uma linguagem visando a representação de seu significado, baseado nas construções obtidas na análise sintática. Por fim, a análise pragmática diz respeito ao processamento da forma que a linguagem é utilizada para comunicar.

Duas técnicas de PLN foram utilizadas no presente trabalho. A primeira delas é a etiquetagem de classes gramaticais, que faz parte da análise morfológica. Ela objetiva, de acordo com Soares (2013, p. 56), classificar todas as palavras contidas no documento textual em sua classe gramatical. A identificação da classe das palavras presentes em uma sentença facilita o entendimento desta e muitas vezes soluciona alguns problemas simples de ambiguidade.

Qualquer documento textual, no entanto, apresenta palavras flexionadas nas mais diversas formas. Na língua portuguesa, um substantivo pode ser flexionado em número e apresentar o mesmo valor semântico. Para Cegalla (2005, p. 40) o processo de formação de palavras é, na maior parte das vezes, realizado pela derivação de radicais, resultando na criação de palavras que também exprimem o mesmo significado. Para resolver este problema, foi usada a segunda técnica de PLN, denominada de lematização ou *stemming*. Lematização é o processo de reduzir ao radical original palavras derivadas ou flexionadas deste. O principal objetivo do processo de lematização é reduzir a grande dimensionalidade dos documentos durante o processo de MT.

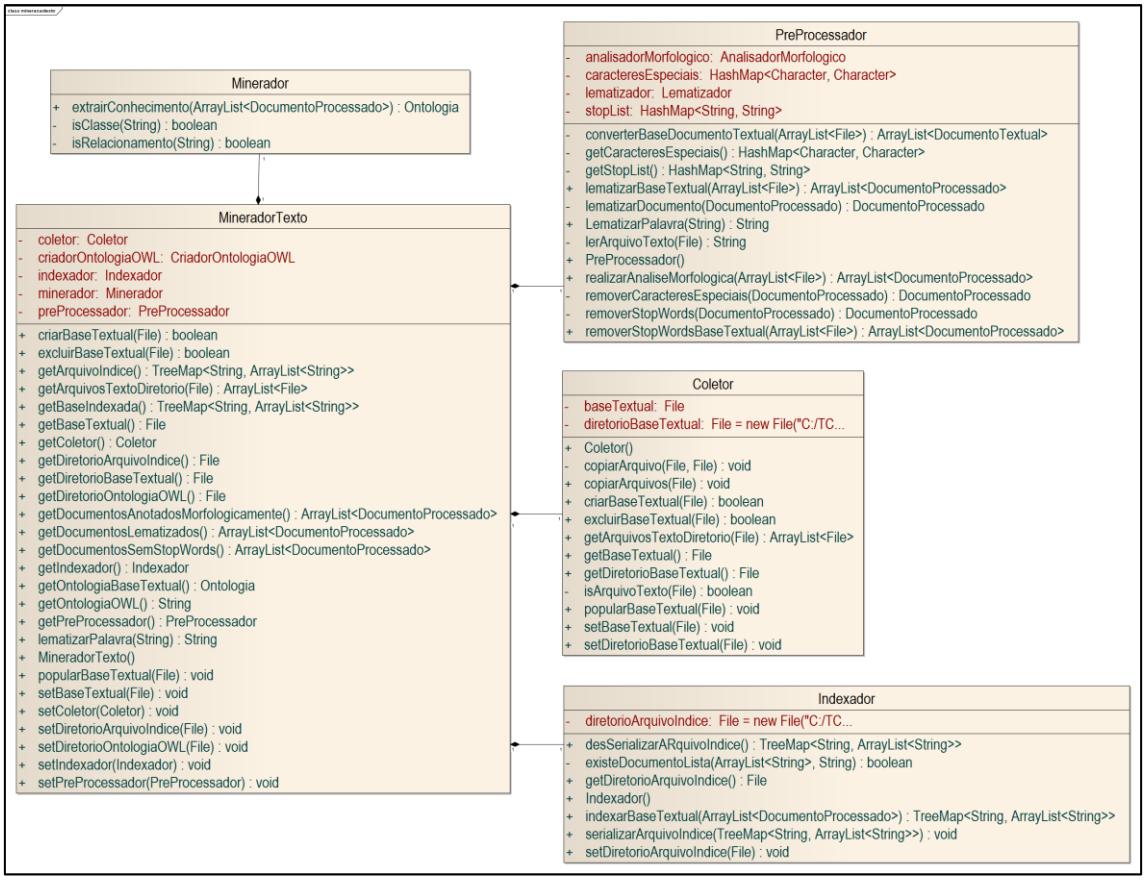
3. Desenvolvimento

Nesta seção é apresentada parte da especificação e da implementação da ferramenta desenvolvida. Inicialmente, na seção 3.1, é apresentado o principal diagrama de classes da ferramenta, juntamente com seu detalhamento. A seguir, na seção 3.2, é descrito um fluxo geral do funcionamento da ferramenta, ao nível de sua implementação.

3.1 Especificação

A especificação da ferramenta foi realizada com o uso dos conceitos do paradigma de análise e modelagem de sistemas orientados a objeto, por meio de diagramas da *Unified Modeling Language* (UML). Na Figura 1 é descrito o diagrama de classes do principal pacote da ferramenta, denominado `controle.mineracaotexto`. Ele foi modelado utilizando o padrão de projeto *Facade*. De acordo com Pfleeger (2004, p. 120), esse padrão objetiva prover uma interface única de acesso a várias funcionalidades de uma API ou subsistemas. A classe `MineradorTexto` é a interface única de acesso a todas as funcionalidades da ferramenta. Ela integra as chamadas de métodos das classes `Coletor`, `PreProcessador`, `Indexador` e `Minerador`.

Figura 1. Diagrama de classes do pacote controle.mineracaotexto



A classe `Coletor` implementa as funcionalidades da etapa de coleta do processo de MT. Quando se navega por um diretório buscando por documentos para compor a base textual de trabalho atual, é necessário filtrar os arquivos encontrados. Isso foi implementado através do método `isArquivoTexto`, onde é analisada a extensão do arquivo. Os arquivos que forem da extensão `txt` serão copiados para a base textual de trabalho atual através do método `copiarArquivos`. Após a cópia, entra em execução a etapa de pré-processamento, modelada através da classe `PreProcessador`. Ela implementa sucessivas operações sobre os documentos textuais, resultando como saída desse processo, um objeto da classe `DocumentoProcessado` com seu conteúdo no formato ideal para as etapas de indexação e mineração.

A classe `Indexador` prove as funcionalidades do processo de indexação da base textual de trabalho atual. Seu método principal é o `indexarBaseTextual`, que recebe como entrada uma lista de arquivos pré-processados e gera como saída um arquivo de

índice invertido. Por fim, tem-se a etapa de mineração do conhecimento presente na base textual atual de trabalho. Essa função é desempenhada através da classe `Minerador` e do método `extrairConhecimento`, que se utiliza de análise morfológica e reconhecimento de padrões para descobrir o conhecimento.

3.2 Implementação

As técnicas e ferramentas utilizadas para o desenvolvimento da ferramenta proposta foram as seguintes: 1) a linguagem de programação Java, na versão 7.0, para codificar a ferramenta; 2) a IDE Eclipse, na versão Juno, como ambiente de desenvolvimento; 3) a biblioteca PTStemmer versão na 2.0, e; 4) a biblioteca Cogroo na versão 4.0.

A biblioteca PTStemmer implementa três algoritmos de lematização de palavras da língua portuguesa, sendo eles: Orengo, Porter e Savoy. É uma biblioteca de código livre tendo implementações em Java, Python e C#. Cogroo é uma biblioteca de código livre escrita em Java para realizar a análise morfológica da base textual de trabalho. Trata-se de uma ferramenta sob licensa GNU *Lesser General Public License* (LGPL), sendo seus mantenedores oficiais o Centro de Competência em Software Livre (CCSL) da Universidade de São Paulo (USP).

Para que seja possível realizar a descoberta do conhecimento a partir dos documentos contidos na base de trabalho atual, primeiro é feita a coleta dos referidos documentos. Em seguida pode-se iniciar o pré-processamento dos documentos. Após transformar os arquivos da base textual atual de trabalho em objetos presentes na memória, são realizados sucessivos processamentos. O primeiro deles é a realização da análise morfológica. Isso dar-se-á através do método `processarDocumento` da classe `AnalisadorMorfologico`. O método `processarDocumento` utiliza a biblioteca Cogroo para realizar as análises.

A última etapa que o objeto da classe `DocumentoProcessado` é submetido é a lematização. Esta foi implementada através do método `lematizarDocumento`, o qual tem como início de seu processamento iterar sobre todas as sentenças do documento. Para cada uma, é iterado sobre todas as suas palavras realizando a lematização das mesmas. Para isso, é utilizado o método `lematizar` da classe `Lematizador`, que faz uso da biblioteca PTStemmer.

Com a etapa de pré-processamento concluída, o arquivo de índice invertido é criado através do método `indexarBaseTextual`. O método `indexarBaseTextual` inicia seu processamento criando o objeto que representa o arquivo de índice em memória. Em seguida o arquivo de índice é populado com todas as palavras distintas encontradas nos textos e os documentos nos quais elas ocorrem.

Com a conclusão da indexação inicia-se a etapa de descoberta de conhecimento, implementada através do método `extrairConhecimento`. O primeiro passo é criar a ontologia para armazenar o conhecimento descoberto. Em seguida, é iterado sobre a lista de `DocumentoProcessado` passada como argumento. Para cada objeto contido na lista, é realizada uma iteração sobre todas as suas sentenças. Em cada sentença é iterado sobre todas as suas palavras. É testado se a palavra é uma classe, através do método `isClasse`, analisando a sua etiqueta morfológica. Caso o resultado do teste for verdadeiro, é verificado se foi achado algum relacionamento na frase. O fato do resultado do teste anterior ser falso, significa que foi identificada a primeira classe do relacionamento. Antes da classe ser inserida na ontologia, é verificado se ela existe na

mesma. Caso a mesma não contenha a classe identificada, um novo objeto de `Classe` é criado e inserido na ontologia.

A ontologia criada no processo de descoberta de conhecimento é transformada em um arquivo OWL, gravado em disco, caracterizando assim a construção da base de conhecimento. Esse processo consiste em transformar os componentes que modelam o conhecimento (classes e relacionamentos), contidos na ontologia, em *tags* OWL (seguindo o padrão de *tags* utilizado pelo Protégé).

4. Estudo de Caso e Resultados

Para avaliar a aplicabilidade da ferramenta desenvolvida, foi realizado um estudo de caso. Como entrada para o processo de MT foi utilizada uma base textual, formada por cinco arquivos (três mil seiscentos e noventa e sete palavras), que descreve a vida e obra de diversos cientistas da computação. No Quadro 2 é apresentado um trecho de código de um dos arquivos utilizados no estudo de caso, denominado `Alan Mathison Turing`.

Quadro 2. Trecho do arquivo Alan Mathison Turing

```
Alan Mathison Turing foi um matemático, lógico, criptoanalista e cientista da computação britânico.
```

O referido trecho do texto foi então submetido à ferramenta desenvolvida, resultando na ontologia OWL apresentada no Quadro 3. Analisando-a pode-se notar que foram identificadas as classes: `Alan_Mathison_Turing` (linha 3), `matemático` (linha 6), `lógico` (linha 9), `criptoanalista` (linha 12) `e cientista_da_computação` (linha 15). Além das classes, foi identificada a propriedade `foi` (linha 18). Para esta, por sua vez, foi identificado o domínio `Alan_Mathison_Turing` (linhas 20 a 23) e os escopos: `matemático` (linhas 24 a 27), `lógico` (linhas 28 a 31), `criptoanalista` (linhas 32 a 35) `e cientista_da_computação` (linhas 36 a 39).

Quadro 3. OntologiaOWL gerada

```
1  <Ontology>
2      <Declaration>
3            <Class IRI="#Alan_Mathison_Turing"/>
4      </Declaration>
5      <Declaration>
6            <Class IRI="# matemático"/>
7      </Declaration>
8      <Declaration>
9            <Class IRI="# lógico "/>
10     </Declaration>
11     <Declaration>
12           <Class IRI="# criptoanalista"/>
13     </Declaration>
14     <Declaration>
15           <Class IRI="# cientista_da_computação"/>
16     </Declaration>
17     <Declaration>
18           <ObjectProperty IRI="#foi"/>
19     </Declaration>
20           <ObjectPropertyDomain>
21                 <ObjectProperty IRI="#foi"/>
22                 <Class IRI="#Alan_Mathison_Turing"/>
23           </ObjectPropertyDomain>
24           <ObjectPropertyRange>
25                 <ObjectProperty IRI="#foi"/>
26                 <Class IRI="#matemático"/>
27           </ObjectPropertyRange>
28           <ObjectPropertyRange>
```

```
29                  <ObjectProperty IRI="#foi"/>
30                  <Class IRI="#lógico"/>
31            </ObjectPropertyRange>
32            <ObjectPropertyRange>
33                  <ObjectProperty IRI="#foi"/>
34                  <Class IRI="#criptoanalista"/>
35            </ObjectPropertyRange>
36            <ObjectPropertyRange>
37                  <ObjectProperty IRI="#foi"/>
38                  <Class IRI="#cientista_da_computação"/>
39            </ObjectPropertyRange>
40  </Ontology>
```

5. Conclusões

Para o desenvolvimento da solução foram utilizadas técnicas e ferramental que se mostraram adequados para que o projeto alcançasse o objetivo proposto. A escolha pela MT mostrou-se adequada para lidar com informações armazenadas em formato textual. O PLN, fazendo uso do conhecimento linguístico, permitiu extrair ao máximo as características do texto, sendo utilizada durante duas etapas da MT: pré-processamento e mineração. Na primeira etapa ela foi utilizada na remoção das *stopwords*, lematização e análise morfológica das palavras presentes na base textual de trabalho. Na segunda foi utilizada para buscar pelos padrões de descoberta de conhecimento adotados para a implementação da ferramenta. O analisador morfológico Cogroo foi imprescindível para o desenvolvimento da ferramenta. O uso da linguagem de programação Java facilitou o uso do Cogroo. Como ele é implementado nessa linguagem, bastou baixar os fontes e integrá-los ao projeto. Também foi utilizada a biblioteca PTStemmer para realizar a lematização das palavras da base textual. A mesma também possui implementação em Java, seguindo a mesma ideia do Cogroo de baixar os fontes e inserí-los no projeto. Assim, o objetivo principal do presente trabalho, que era de disponibilizar uma ferramenta para a criação de bases de conhecimento na forma de uma ontologia OWL a partir de textos não estruturados, foi alcançado.

Porém, duas escolhas feitas durante a revisão bibliográfica merecem ser destacadas. A primeira delas foi a adoção de um pré-processamento simples, baseado no modelo de representação de palavras. A partir dela, foi concluído que quanto mais elaborado o pré-processamento, melhor são os resultados obtidos. A outra foi a opção pelo padrão para descoberta do conhecimento. Ele se mostrou efetivo para a descoberta das classes da ontologia, porém simplista demais para os relacionamentos. Tem-se como alternativa utilizar estruturas sintáticas para melhorar a qualidade do conhecimento descoberto.

Referências

ARANHA, Christian N. et al. Um modelo de desambiguização de palavras e contextos. In: TIL: WORKSHOP DE TECNOLOGIA DA INFORMAÇÃO E DA LINGUAGEM HUMANA, 2., 2004, São Carlos. **Anais...** São Carlos: Sociedade Brasileira de Computação, 2004. p. 101-110.

ARANHA, Christian N. **Uma abordagem de pré-processamento automático para mineração de textos em português:** sob o enfoque da inteligência computacional. 2007. 144 f. Tese (Doutorado em Engenharia Elétrica) – Programa de Pós-Graduação em Engenharia Elétrica, Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro.

BASTOS, Vitor M. **Ambiente de descoberta de conhecimento na web para a língua portuguesa**. 2006. 144 f. Tese (Doutorado em Engenharia Civil) – Programa de Pós-

Graduação em Engenharia Civil, Universidade Federal do Rio de Janeiro, Rio de Janeiro.

CEGALLA, Domingos P. **Novíssima gramatica da língua portuguesa**. São Paulo: Nacional, 2005.

CHANDRASEKARAN, Bharath; JOSEPHSON, John R. What are ontologies, and why do we need them? **Intelligent Systems and their Applications,** Amsterdam, v. 14, n. 1, p. 20-26, jan/fev 1999.

HEINZLE, Roberto**. Um modelo de engenharia do conhecimento para sistemas de apoio a decisão com recursos para raciocínio abdutivo**. 2011. 251 f. Tese (Doutorado em Engenharia e Gestão do Conhecimento) - Programa de Pós-Graduação em Engenharia e Gestão do Conhecimento, Universidade Federal de Santa Catarina, Florianópolis.

KONCHADY, Manu. **Text mining application programming**. Newton: Charles River Media, 2006.

KUECHLER, William L. Business applications of unstructured text. **Communications of ACM**, New York, v. 50, n. 10, p. 86–93, out. 2007.

MANNING, Christopher D.; RAGHAVAN, Prabhakar; SCHUTZE, Hinrich. **Introduction to information retrival.** Cambridge: University Press, 2007.

MARCACINI, Ricardo M.; MOURA, Maria F.; REZENDE, Solange O. O uso da mineração de textos para extração e organização não supervisionada de conhecimento. **Revista de Sistemas de Informação da FSMA**, Macaé, v. 1, n.7, p. 7-21, nov. 2011.

MÜLLER, Daniel N. **Processamento de linguagem natural**. Porto Alegre, 2003. Disponível em: <http://www.inf.ufrgs.br/~danielnm/docs/pln.pdf>. Acesso em: 14 out. 2014.

PFLEEGER, Shari L. **Engenharia de software:** teoria e prática. São Paulo: Person, 2004.

PROTÉGÉ. W**hat is protégé-owl?** [Califórnia], [2013]. Disponível em: <http://protege.stanford.edu/overview/protege-owl.html>. Acesso em: 6 out. 2014.

SOARES, Fabio de A. **Categorização automática de textos baseada em mineração de texto**. 2013. 158 f. Tese (Doutorado em Engenharia Elétrica) – Programa de Pós-Graduação em Engenharia Elétrica, Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro.

W3C. **OWL web ontology language overview** - W3C recommendation 10 February 2004. Massachusetts, 2004. Disponível em: <http://www.w3.org/TR/owl-features>. Acesso em: 4 out. 2014.

WIVES, Leandro. **Recursos de text mining**. Porto Alegre, 2000. Disponível em: <http://www.inf.ufrgs.br/~wives/portugues/textmining.html>. Acesso em: 19 out. 2014.

ZHU, Xingquan; DAVIDSON, Ian. **Knowledge discovery and data mining**: challenges and realities. New York: Hershey, 2007.