UNIVERSIDADE DE ÉVORA

ESCOLA DE CIÊNCIAS E TECNOLOGIA

DEPARTAMENTO DE INFORMÁTICA

Verificação Automática de Metadados em e-books

Felipe Marcellino Cavalcante

Orientação: Prof.ª Dr.ª Irene Pimenta Rodrigues

Mestrado em Engenharia Informática

Dissertação

Évora, 2014

Mestrado em Engenharia Informática

Verificação Automática de Metadados em *e-books*

Felipe Marcellino Cavalcante

Orientador
Prof.ª Dr.ª Irene Pimenta Rodrigues

Sumário

O progresso da Web Semântica - ou Web 3.0 -, nos dias actuais, é um estímulo para o desenvolvimento de aplicações voltadas à partilha e integração de informação entre vários sistemas. Tantas podem ser as suas aplicações, em tão diferentes disciplinas, que a escolhida para este estudo emerge do universo pululante de unidades exploráveis. O presente trabalho tem como meta o desenvolvimento de uma ontologia para uma representação do domínio dos e-books utilizando as linguagens OWL e RDF, além do sistema Protégé para o seu desenvolvimento; a aplicação de técnicas de PLN para extracção de metadados de e-books, com a biblioteca NLTK; o desenvolvimento de uma aplicação para a realização de consultas à ontologia, com a linguagem SPARQL, visando a validação dos metadados.

Keywords: Ontologia, OWL, RDF, PLN, NER, NLTK, e-book

Automatic Verification of Metadata in e-books

Abstract

Nowadays, the progress of Semantic Web - or Web 3.0 - is a stimulus for the development of applications toward sharing and integration of information between various systems. Their applications are as many as we can think, in several subjects, and, due to that, the one that I chose for this study emerges from the teeming universe of exploitable units. The present work has, as its aim, the development of an ontology for the representation of the domain of e-books, using the OWL and RDF languages, and the Protégé system for its development; the application of NLP techniques for the extraction of e-book metadata, using the NLTK library; the development of an application to query the ontology, using the SPARQL language, whose main goal is the validation of metadata.

Agradecimentos

Acima de tudo, agradeço a Deus.

A dissertação que aqui apresentarei é, primariamente, dedicada aos meus pais, à minha avó, à minha querida Ana e à Dona Dina, suportes fundamentais no percurso pela conquista dos meus objectivos. Não só me apoiaram em todos os momentos difíceis e sinuosos, como também souberam valorizar todas as tentativas de que se reveste a procura do saber científico.

Gostaria de mostrar a minha gratidão pessoal para com a Professora Doutora Irene Pimenta Rodrigues, do Departamento de Informática da Universidade de Évora e orientadora desta dissertação.

Aos professores Helga, Raison e Maritel, deixo o meu agradecimento.

Acrónimos

CML Chemical Markup Language

CWA Closed World Assumption

DAML DARPA Agent Markup Language

GRDDL Gleaning Resource Description from Dialects of Languages

HMM Hidden Markov Models

HTML Hypertext Markup Language

HTTP Hypertext Transfer Protocol

JSON JavaScript Object Notation

KIF Knowledge Interchange Format

MQL Metaweb Query Language

NE Named Entity

NER Named Entity Recognition

NLP Natural Language Processing

NLTK Natural Language Toolkit

OCML Operational Conceptual Modelling Language

OIL Ontology Inference Layer

OWA Open World Assumption

OWL Web Ontology Language

OWL DL Web Ontology Language Description Logic

POST Part-of-speech Tagging

QA Question Answering

RDF Resource Description Framework

RDFa Resource Description Framework in Attributes

RDF-DAWG RDF Data Access Working Group

RDF-S Resource Description Framework Schema

RFC Request for Comments

SBD Sentence Boundary Disambiguation

SHOE Simple HTML Ontology Extensions

SPARQL Simple Protocol and RDF Query Language

URI Uniform Resource Identifier

URL Uniform Resource Locator

XHTML Extensible Hypertext Markup Language

XML Extensible Markup Language

XOL Ontology Exchange Language

XSLT Extensible Stylesheet Language for Transformation

W3C World Wide Web Consortium

Conteúdo

Lista de Figuras

Capítulo 1

Introdução

Nos dias actuais e, principalmente, na Web actual, há uma porta cada vez maior para o aproveitamento da informação. Tão imensa é a informação disponível na Internet, que a representação deste conhecimento de forma estruturada, categorizada, com associações entre termos, entidades e valores, é mais que fundamental para trazer benefícios. Benefícios esses que só vêm à tona caso se tenha o domínio das técnicas de organização desse conhecimento e, inclusivamente, meios de extracção, condicionamento e transformação dos dados para um propósito particular.

O surgimento da Web representou uma mudança radical na forma como a maior parte das pessoas usa a Internet. Actualmente, os utilizadores já não necessitam de pensar sobre as aplicações que realizam o tratamento das mensagens que circulam pela rede. A grosso modo, tudo que um utilizador tem de fazer, para ter acesso à Web, é instalar um navegador; depois disso, todas as aplicações ficarão à sua disposição e comando. Antes do aparecimento da World Wide Web, quando um utilizador queria usar uma aplicação na Internet, ele tinha que instalar uma ferramenta capaz de tratar mensagens de tipos específicos de rede na sua máquina. Se um utilizador quisesse localizar os utilizadores de outras redes, teria que instalar uma aplicação capaz de usar o protocolo FINGER. Se um utilizador quisesse trocar um e-mail numa rede, ele teria que instalar uma aplicação capaz de utilizar o protocolo SMTP. Cada uma destas ferramentas entenderia o formato das mensagens, assim como o protocolo específico de determinada tarefa, e saberia, também, qual a melhor forma de exibir a informação ao utilizador. Operando subjacentemente, está um conjunto de mensagens que os desenvolvedores da infraestrutura concordaram em tratar de uma forma padronizada. Por exemplo, está bem definido que quando um servidor HTTP recebe um pedido GET, deve enviar de volta os dados correspondentes à porção do caminho do pedido da mensagem. A semântica destas mensagens tem sido cuidadosamente definida por comités e padronizada e documentada nas recomendações

das RFC e W3C. Esta infraestrutura padronizada permite que os desenvolvedores de aplicações Web operem por detrás de uma camada que os separa dos detalhes inerentes à transmissão dos dados da aplicação entre as máquinas, o que permite um maior foco na forma como essas mesmas aplicações aparecem aos utilizadores. Assim, actualmente, os programadores já não precisam de combinar, para cada projecto, o formato das mensagens ou a forma como as aplicações se devem comportar na presença de determinados dados. [23, pp. 4-5]

Hoje em dia, a maioria das aplicações Web operam em praticamente dois sentidos: o servidor e o navegador. O navegador mostra elementos estáticos e dinâmicos das páginas, enquanto o servidor, na maior parte das aplicações, contém a base de dados e o código gerador das páginas, que, de modo geral, é criado a partir de linguagens de programação *server-side*, como PHP, Ruby e Python. Do ponto de vista semântico, o maior valor está do lado do servidor, que não está aberto à partilha livre de informação. Por isso, o termo pejorativo *stovepipe system* é empregado para designar este tipo de sistema, do qual não se consegue explorar as capacidades de partilha de informação e integração noutras aplicações. [6, p. 5] Para minimizar este revés do ponto de vista da semântica, o enriquecimento da informação contida do lado do navegador foi idealizado para que tal conteúdo pudesse ser explorado semanticamente.

Por ter crescido rapidamente e de forma desvinculada, a informação existente na Internet pode estar presente em uma infinidade de páginas de diferentes sites. Assim como as mesmas unidades de informação podem coexistir no conjunto da Web, definições de objectos que formam a informação também podem estar espalhadas, com acepções distintas. O escopo de trabalho, no qual as técnicas da Web Semântica são úteis, é todo aquele que visa a centralização do conhecimento sobre determinada matéria para fins quaisquer.

Durante muitos anos, a Inteligência Artificial passou por momentos alternantes de desconfiança e esperança, mas com a grande valorização, nos últimos anos, da Web Semântica, o financiamento de projectos voltou a acontecer com mais intensidade. Muitos ramos já estão a tirar proveito disso, como, por exemplo, a Medicina e a Biologia. A comunidade científica talvez consiga, com o auxílio dessa tecnologia, alcançar feitos com mais celeridade, devido à capacidade de descoberta de novos factos e relações de difícil percepção humana. O mesmo se dá para uma corporação, empresa ou grupo de estudo independente que queira servir-se dessas competências.

É comum que os projectos que se aproveitam da Web Semântica utilizem outras subáreas da Inteligência Artificial - ou mesmo de outros ramos - para atingirem o resultado esperado. Ou seja: apesar de não ser requerido este apoio, a depender do projecto que se tem em mãos, dos resultados que se pretende auferir, é perfeitamente possível reunir num sistema vários métodos que se completam. No caso deste trabalho, algumas funções de Processamento de Linguagem Natural são as bases complementares.

Com a popularização dos e-books, cada vez mais pessoas têm acesso a ficheiros com livros inteiros, que podem ser acedidos através de computadores convencionais, portáteis

e dispositivos móveis. Uma aplicação, portanto, que possa ser usada por tantas pessoas e, por haver capacidade para estar online, tende ao sucesso. Para além disso, há casos em que uma pessoa faz *download* de um e-book na Internet e quer, para efeito de confirmação, saber se o nome do escritor e o título da obra contidos no ficheiro equivalem aos verdadeiros metadados daquele livro.

Pois, sendo assim, o meu trabalho nesta tese é demonstrar como podemos integrar diferentes áreas em prol de uma finalidade. A temática do meu trabalho consiste em criar uma ontologia para representar o domínio dos e-books, livros digitais, contendo os seus metadados e as associações entre as diversas entidades que estão relacionadas. Esta primeira tarefa é, resumidamente, a porção que trata da Web Semântica. Fazendo-me valer das linguagens OWL e RDF para iniciar o desenvolvimento dessa ontologia, ela passa a fazer parte do domínio da Web 3.0, ou seja, basta ser publicada para outros interessados terem a oportunidade de fazer interrogações e complementá-la.

A minha intenção também é demonstrar que a análise do conteúdo dos e-books pode ser realizada de forma automática, e, para tal, utilizo a biblioteca NLTK [3]. Embora já haja disponível um recurso como este, um processo de estudo linguístico seria obviamente obrigatório antes de desenvolver a minha própria solução. Isto só mostra que, em Informática, muitas soluções prontas ajudam a completar trabalhos e a poupar tempo.

Por conseguinte, viso a introdução ao mundo da Web Semântica, às suas possibilidades e aos seus alcances por meio do exemplo deste trabalho. Apesar de não resultar num produto final que possa, efectivamente, ser explorado cientificamente ou no mercado, devido ao seu próprio carácter demonstrativo, é um trabalho que descreve um processo completo, concretizável e, caso sirva de interesse para qualquer pessoa, seu alicerce deixa-o pronto para ser expandido.

Capítulo 2

Estado da Arte

2.1 Ontologia

O termo Ontologia pode ter diversos significados, dependendo da área em que é aplicado. Tendo origem filosófica [1, p. 8], no século XVII, o conceito representado pelo termo é ainda mais antigo, remetendo a Aristóteles, como mostrado em [16]. O termo ganhou conotações mais específicas ao longo do tempo e, actualmente, é também de uso comum na Ciência da Computação. Mesmo nesse contexto específico, o termo é difuso quanto à sua definição. Guarino recolheu [9] o que considera ser as oito definições mais empregadas pela comunidade da Representação de Conhecimento. Diante de tantos significados, uma acepção genérica, que presume tanto conceitos concretos quanto abstractos, pode ter em conta que toda ontologia, necessariamente, diz respeito a uma determinada área do conhecimento, relacionando os seus conceitos e atribuindo-lhes propriedades. Assim, é possível derivar axiomas que introduzem factos à ontologia e que lhe dão uma semântica que revela o mundo real. A ontologia, então, tem o fim de "permitir o compartilhamento e a reutilização do conhecimento". [16]

Actualmente, podem ser encontrados projectos que fazem uso de ontologias em variados domínios: "gestão do conhecimento, comércio electrónico, recuperação de informações na Internet, processamento de linguagens naturais, de cunho educacional, entre outros". [1] Algumas dessas aplicações foram ambicionadas para funcionarem segundo os padrões da *Semantic Web*, como é indicado em [22].

As ontologias criadas com um propósito informático podem ser descritas em três tipos de linguagens, que devem ser escolhidas consoante a natureza do problema. Podem ser linguagens de ontologias tradicionais (*Cycl*, *Ontolíngua*, *F-Logic*, *CML*, *OCML*, *Loom* e *KIF*), linguagens do padrão *Web* (*XML* e *RDF*) e linguagens de ontologias *Web-based*

(*OIL*, *DAML+OIL*, *SHOE*, *XOL* e *OWL*) [16]. Essas linguagens têm, normalmente, o poder de expressão da Lógica de Primeira Ordem ou da Lógica Descritiva. Como este trabalho tem a pretensão de ser balizado sob a linguagem *OWL*, que é uma linguagem que proporciona mais vocabulário para descrição de ontologias do que outras linguagens, como *RDF* e *XML*, as aplicações apresentadas a seguir também seguem essa linha. Além disso, a *OWL* é "destinada a ser utilizada quando a informação encapsulada em documentos precisa de ser automaticamente processada por aplicações" [8], como é o caso deste trabalho.

Como a representação de qualquer área do conhecimento é exequível, enquadrando-se nos limites lógicos da *OWL*, é possível encontrar trabalhos feitos em muitos domínios. Nota-se que uma ontologia é, na maior parte das vezes, acompanhada de um sistema planeado para tirar partido do domínio representado, seja para actualizá-lo ou para outras actividades. Alguns exemplos são: a representação do domínio de um jogo de futebol, "de modo a que sistemas de análise ou simulação de jogos possam utilizar as representações conceptuais que a ontologia oferece"; [8] uma ontologia de representação de cenários e domínios de aplicações sensíveis ao contexto, de forma que um motor de inferência possa tomar decisões sobre a adaptação do conteúdo dessas aplicações; [2] o mapeamento de várias ontologias jurídicas para uso em um sistema de recuperação de informação; [25] um sistema de "construção automática de ontologias a partir de um conjunto de documentos e a utilização dessas ontologias em processos de inferência". [22]

2.1.1 Semântica

Pode dizer-se que a semântica é o significado de algo. Uma sequência de símbolos pode ser usada para transmitir o sentido e esta comunicação pode, assim, afectar o comportamento. Nas ontologias, a linguagem, diferentemente da linguagem natural, incide antes no uso da semântica para representar, combinar e partilhar conhecimento entre comunidades de máquinas. Além disso, também se ocupa da formulação escrita dos sistemas que podem agir sobre esse conhecimento. [23, p. 3]

As ontologias são essenciais para aplicações que participam em amplas redes, como a Internet, nas quais o significado das mensagens trocadas precisa de ser muito bem definido e o conhecimento precisa de ser uniformizado para que não haja risco de existirem ambiguidades inesperadas entre os utilizadores da informação.

A Web Semântica utiliza a OWA, que é a assumpção de que tudo que não é afirmado directamente é desconhecido, ao invés de ser considerado falso, como o faz a CWA. Por exemplo, se é afirmado que o autor Dostoiévski nasceu na Rússia, e, em seguida, surge a questão de que ele é português, a resposta não é falsa, mas indefinida. Como a Web é um domínio de informação incompleto, é importante saber que os *reasoners* e as linguagens de consulta, como a SPARQL, interpretam as ontologias desta forma.

2.1.2 RDF

O RDF "é um modelo padronizado pela W3C para o intercâmbio de dados na Web." [26] A capacidade de representação e comunicação de dados do RDF foram decisivos para que este fosse aprovado como recomendação para a tecnologia de Web Semântica, pelo W3C.

"É a linguagem base da Web Semântica. É uma linguagem usada para descrever dados, metadados e até outras linguagens de representação de dados. O RDF usa um formato de dados em grafos, em contraste com os formatos de dados relacionais (como a maioria das bases de dados) e os formatos de dados hierárquicos (como o XML). Qualquer modelo de dados ou linguagem de dados que use o RDF é uma parte da Web Semântica." [18]

Com o RDF, pode-se descrever qualquer área do conhecimento, simplesmente com a descrição de sentenças sobre suas propriedades. Essas sentenças são, em sua essência, compostas por triplos, que são formados por três partes: o sujeito, a relação (a propriedade) e o objecto.

O RDF está assente em protocolos baseados na Web, como URI, HTTP e XML, por exemplo. O propósito do URI (*Universal Resource Identifier*) é identificar, de forma única, um nome no RDF. Pode-se referenciar uma URI assim como faz-se com uma URL, sendo que esta é, de facto, a finalidade desta última. Quando uma URI serve como uma fonte oficial para um certo conjunto de nomes, é chamada de *namespace*. [18]

O plano do grafo RDF requer que todos os itens de dados tenham um URI, pois o cerne de tal plano é que os itens possam se relacionar, podendo as próprias relações terem também URI associados. O grafo refere-se a uma colecção de triplos RDF, mesmo que este grafo seja composto por apenas um triplo. Matematicamente, os grafos RDF são apoiados na Teoria dos Conjuntos. [18]

Cada um dos três elementos pode ser descrito de duas formas: com recurso ou com literal. Quando usa-se um recurso, está-se a usar uma URI, cuja vantagem, em relação ao literal, é que há a possibilidade de extracção de mais informações e relações sobre o ente do triplo, seja um sujeito, uma propriedade ou um objecto. Ao usar um literal, prende-se àquela sentença um dado repetível e estático. Em termos de representação de conhecimento, o uso de recursos em vez de literais é preferível.

Um grafo RDF é chamado de federado, quando, em seu conteúdo, é usada pelo menos uma referência externa. Usam-se *XML Namespaces* para declarar variáveis curtas que representam outros grafos ou vocabulários, cujas partes passam a poder ser descritas.

Para a comunicação de modelos RDF acontecer de modo compatível entre diversas aplicações, o formato escolhido por convenção foi o XML, que é a própria base sintática dos grafos RDF.

A prática de reificação com RDF é desencorajada porque, na prática, a semântica

da representação é confusa e, além disso, torna as consultas SPARQL muito pesadas. [18]

RDF Schema (RDFS)

O *RDF Schema* é usado para a descrição de classes RDF. "O RDFS provê um vocabulário para descrever recursos, propriedades (predicados), classes e subclasses. Pode ser construído com XML serializado, como o próprio RDF." [18] Isto torna possível que inferências (e.g. *subclass reasoning* ou *subsumption reasoning*) sejam feitas sobre o modelo apenas com a representação dada pelo RDFS, o que leva a uma redução da necessidade de expressão do conhecimento, já exprimido anteriormente numa definição.

RDFa, eRDF e GRDDL

O RDFa permite que um conjunto de atributos descritivos sejam implantados em qualquer tipo de página compatível com XML. O formato eRDF tem a mesma função apenas com os formatos HTML e XHTML. Os elementos que contêm alguma semântica importante podem ser denotados com atributos (daí vem o "a" de RDFa), cujos valores levam a recursos externos. Com este aparato, é possível que *softwares* façam uso com mais precisão das informações disponíveis nas páginas Web. Por exemplo, motores de busca da Web podem, de forma inteligente, analisar o conteúdo das páginas dos sites à procura de informação relevante nas *tags*.

O GRDDL é uma recomendação do W3C para a extração do conteúdo RDF presente como atributos nas páginas HTML. A extração é feita por meio de *scripts* XSLT.

Tendo como ferramentas formatos para representação e consulta de conhecimento em páginas Web, consultas mais avançadas ou mesmo inferências podem ser feitas sobre o próprio conteúdo da Internet. Este é o principal foco da Web 3.0.

Especializações sintáticas de RDF

Alguns outros formatos usados para definição de triplos não passam de extensões do RDF com modificações meramente sintáticas. Os formatos N3, Turtle e N-Triples podem ser citados como exemplos.

2.1.3 OWL

A linguagem OWL tem um poder de expressão muito mais amplo em relação ao RDF, que oferece um modelo simples para tratar dados em grafos. Contudo, quando se precisa de uma representação semântica complexa que envolva relações ou modelos de dados avançados, o RDF deixa de ser uma opção e dá lugar a uma linguagem mais sofisticada, a OWL. Assim, podem-se representar "classes, factos sobre essas classes, relações

entre classes ou instâncias e as características dessas relações." [18] A linguagem OWL permite que se crie uma ontologia mais elaborada, com o apoio de regras formuladas com lógica. Há a possibilidade de descrever, na ontologia, com rigor, o que cada entidade é e como se relaciona com as outras entidades.

Sublinguagens OWL

Até à versão 1.0 da OWL, três sublinguagens OWL foram criadas para efeitos de optimização. "Essas sublinguagens podem ser caracterizadas pela sua expressividade em termos das construções válidas e dos axiomas permitidos. A OWL-Lite usa somente alguma da expressão disponível na OWL-DL, que é a intermediária, e que permite um uso pleno do núcleo da linguagem OWL. Por fim, a OWL-Full é a linguagem com poder total de expressão, capacitando os seus utilizadores a definirem que classes podem também ter propriedades e instâncias." [18]

Sintaxe Manchester-OWL

Merece uma especial atenção a Sintaxe Manchester-OWL, considerada mais amigável e de fácil compreensão para pessoas que não têm conhecimentos sobre Lógica Descritiva. Além de a prolixidade do XML tornar difícil a alteração e criação rápida de ontologias, uma sintaxe que é baseada em Lógica Descritiva pode ser um grande obstáculo para *experts* de domínios de conhecimento que não são versados em lógica. [13]

Por esse motivo e por ter se tornado uma das convenções da Web Semântica, a maioria dos editores de ontologia suportam a Sintaxe Manchester-OWL, que é empregada também na descrição de trechos de ontologias em fóruns de discussão e troca de e-mails, por exemplo. [13] O *software* Protégé disponibiliza um separador, para consultas simplificadas com esta sintaxe.

Restrições OWL

Uma restrição descreve uma classe de indivíduos com base nas relações em que os membros desta classe se incluem. Por outras palavras, uma restrição é um tipo de classe, da mesma maneira que uma classe nomeada é um tipo de classe. Na linguagem OWL, podemos descrever todas as classes de indivíduos utilizando restrições. As restrições são compostas por três categorias principais. São elas:

- Restrições Quantificadoras
- Restrições de Cardinalidade
- Restrições do tipo *hasValue*

As restrições quantificadoras podem ser divididas entre restrições existenciais e restrições universais. As restrições existenciais descrevem classes de indivíduos que participam em, pelo menos, uma relação com uma propriedade; os indivíduos pertencem à classe que está descrita na definição da restrição (e.g. uma classe de indivíduos que tenham pelo menos uma relação da propriedade *authorHasAuthorshipOfBook* com a classe *Book* - *authorHasAuthorshipOfBook some Book* é um exemplo da restrição escrita na Sintaxe Manchester-OWL). As restrições universais descrevem classes de indivíduos que, dada uma única propriedade, apenas têm relacionamentos com outros indivíduos que são membros de uma classe específica (e.g. uma classe de indivíduos que tenham todas as suas relações com a propriedade *authorHasAuthorshipOfBook* restritas à classe *Book* - *authorHasAuthorshipOfBook only Book* na Sintaxe Manchester-OWL).

As restrições de cardinalidade funcionam sobre a quantidade de relações que os indivíduos da classe precisam de ter com uma determinada propriedade. Podem ser de três tipos: exacta, mínima e máxima. Por exemplo, com uma restrição de cardinalidade, é possível ordenar que, para um indivíduo pertencer à classe *AuthorOfOnlyOneBook*, precisa de exactamente uma relação da propriedade *authorHasAuthorshipOfBook* a um indivíduo da classe *Book*.

As restrições *hasValue* definem uma classe anónima de indivíduos que estão relacionados com um outro indivíduo específico por meio de uma dada propriedade. A diferença entre uma restrição *hasValue* e uma restrição quantificadora é que o primeiro tipo é moldado com indivíduos e o segundo tipo com classes. Também vale ressaltar que não há impedimento quanto a outras relações com a mesma propriedade. Portanto, semanticamente, neste sentido, há uma equivalência com as restrições quantificadoras existenciais. [12, p. 99]

Uma restrição descreve uma classe anónima *unnamed class* que contém todos os indivíduos que satisfazem a restrição, ou seja, todos os indivíduos que têm as relações necessárias para serem membros da classe. As restrições são utilizadas nas definições de classes OWL para especificar as superclasses anónimas da classe descrita.

O *Reasoner* e as Condições Necessárias e Suficientes

As condições necessárias (*necessary conditions*) são todas as condições que um indivíduo que é membro de uma classe deve cumprir. Ou seja: são todas as definições que precisam de ser satisfeitas para que algo seja considerado de um tipo. Já as condições suficientes dizem respeito à classificação de indivíduos numa espécie de comparação com a definição da classe, para que se possa decidir se é ou não um membro. Esta tarefa é própria do *reasoner*.

Quando uma classe possui pelo menos uma condição necessária e uma suficiente, pode ser chamada de classe definida. Chama-se a classe que só contém condições necessárias de classe primitiva. [12, p. 55]. Com esses tipos de definições, consegue-se usar o *reasoner* para computar a hierarquia da ontologia.

Dá-se o nome de *Subsumption Testing* à tarefa de classificação da ontologia, pois trata-se de verificar se uma classe é subsumida por outra, isto é, se há relação de herança. Outra importante tarefa do *reasoner* é a verificação de consistência da ontologia. Com base na descrição (condições) de uma classe, o *reasoner* pode verificar se é ou não possível que a classe venha a ter alguma instância.

A ontologia pode ser enviada para o reasoner para que este, automaticamente, compute a classificação hierárquica e, também, para que o *reasoner* verifique a consistência lógica da ontologia. No *software* Protégé, distinguem-se a hierarquia manualmente construída e a inferida pelo *reasoner* pelos nomes de hierarquia assertiva e hierarquia inferida, respectivamente.

Classes

As classes OWL são interpretadas como conjuntos que contêm indivíduos. Elas são descritas utilizando-se descrições formais (matemáticas) que atestam, precisamente, os requerimentos para os membros da classe. Na OWL, as classes são construídas a partir das descrições que especificam as condições que devem ser satisfeitas por um indivíduo para que este seja membro da classe.

Propriedades

As propriedades OWL representam relações. São dois os grandes tipos de propriedades - Propriedades de Objectos (*Object Properties*) e Propriedades de Tipo de Dados (*Datatype Properties*). As primeiras são relações entre dois indivíduos, enquanto as outras são propriedades que relacionam indivíduos a valores que correspondem a tipos de dados como múmeros inteiros e texto.

Ainda existe um terceiro tipo de propriedade, chamado Propriedade de Anotação (*Annotation Property*), que pode ser utilizado para adicionar informação a classes, propriedades e indivíduos.

Propriedades Inversas

Cada propriedade de objecto pode ter uma propriedade inversa correspondente. Se alguma propriedade ligar um indivíduo A a um indivíduo B, então a sua propriedade inversa irá ligar o indíviduo B ao indivíduo A. Normalmente, representa uma ligação semântica recíproca. Por exemplo, se um indivíduo está relacionado com outro pela propriedade *comanda*, a propriedade inversa é *éComandadoPor*. Logo, se *A comanda B*, *B éComandadoPor A*.

Características das Propriedades

Em [12, pp. 29-33], pode-se ter uma boa visão de como funcionam as adições semânticas aplicadas às propriedades OWL. O facto de as propriedades terem características como as descritas abaixo é relevante pela vantagem da significação acrescida à ontologia.

- Propriedades Funcionais - representam relações nas quais apenas um único indivíduo pode estar ligado. Por exemplo, as relações *temMãeBiológica* ou *temPaiBiológico* são exemplos de propriedades funcionais, dado que nenhum ser tem mais que uma mãe ou pai biológicos.

- Propriedades Funcionais Inversas - assim como as propriedades funcionais, dizem respeito à unicidade da relação, mas do ponto de vista da propriedade inversa. Se a propriedade *éMãeBiológicaDe* for funcional inversa, isto quer dizer que o indivíduo sujeito da relação só pode ser mãe de um filho. Assim, como mostrado em [12, p. 29], se há mais de uma relação representada na ontologia, o *reasoner* só pode inferir que os indivíduos relacionados são iguais.

- Propriedades Transitivas - representam ligações indirectas entre indivíduos, de modo que o simples facto de haver relações terceiras já é suficiente para relacionar todos os indivíduos envolvidos por essa cadeia.

- Propriedades Simétricas - ligam indivíduos, pela mesma propriedade, de forma recíproca. Sendo assim, se um indivíduo se relaciona com outro por via desta propriedade, logo este último também está relacionado com o primeiro pela mesma relação. Um bom exemplo é a propriedade *éIrmãoDe*. Vale notar que uma propriedade simétrica tem em si a própria relação inversa. [12, p. 30].

- Propriedades Assimétricas - ao contrário das propriedades simétricas, as assimétricas denotam a impossibilidade de conexão inversa pela mesma propriedade. Se um indivíduo está relacionado a outro por uma propriedade assimétrica, este último não pode se relacionar com o primeiro pela mesma propriedade.

- Propriedades Reflexivas - para propriedades que têm a possibilidade de um indivíduo ter um auto-relacionamento, usa-se a característica reflexiva. Por exemplo, a propriedade *conhece*, já que uma pessoa conhece-se. [12, p. 32]

- Propriedades Irreflexivas - contrariamente à característica reflexiva, a irreflexiva representa propriedades que não têm a possibilidade de realizar um auto-relacionamento. Por exemplo, um ser humano não pode ser o seu próprio pai; logo, a propriedade *éPaiDe* é irreflexiva.

2.1.4 Interrogações sobre Ontologias

Desde que o RDF tornou-se um *standard*, em 1999, "dezenas de linguagens de consulta foram desenvolvidas como projectos pessoais, académicos e comerciais. No mesmo ano, a W3C formou o RDF-DAWG." [7, p. 42] Como resultado, no fim de 2004, foi lançado o primeiro *draft* da linguagem SPARQL, que, no ano de 2008, transformou-se numa recomendação oficial do W3C. Com o decorrer do seu uso, novas ideias surgiram para a sua melhoria. Por isso, um ano depois, em 2009, foi criado o *SPARQL Working Group* - que, actualmente, se encontra encerrado - para tratamento exclusivo do desenvolvimento e manutenção da linguagem, que culminou, em 2013, na evolução da versão 1.1 como recomendação da W3C. [7, p. 43]

A recuperação de informação de ontologias é uma área complexa, com diversas técnicas e linguagens. Logo, actualmente, a linguagem SPARQL, por ser um padrão recomendado pela W3C, é amplamente adoptada para este fim. [17, p. 31] A linguagem SPARQL é capaz de retornar dados e relações de uma ontologia assente em RDF. Alguns críticos deste *standard* afirmam que, em vez de inventar uma nova gramática para o SPARQL, a W3C deveria alavancar o trabalho já realizado sobre as linguagens SQL e XQuery. [18, p. 228]

A SPARQL é uma linguagem de consultas para RDF, recomendada pela W3C, que age sobre triplos (sujeito, predicado e objecto). Assim como funciona com a linguagem SQL, a linguagem SPARQL provê uma *interface* declarativa para interagir com uma base de dados RDF. [18, p. 228] A sua semântica é toda baseada na resolução de padrões em grafos, e não lida com as especificidades da linguagem OWL. [17, p. 32]

Os processadores SPARQL aplicam os comandos sobre um conjunto de dados e retornam o resultado. Para designar esses processadores, costumam-se usar os termos *SPARQL processor* e SPARQL engine. [7, p. 5]

Usam-se como convenção as palavras-chave do SPARQL com letras maiúsculas, embora tal uso não seja obrigatório. [7, p. 4] As cláusulas do SPARQL seguem definidas abaixo: [7, p. 104]

- SELECT - a mais popular entre as formas do SPARQL, a cláusula SELECT permite a selecção de atributos identificáveis dentro da colecção a que se interroga. Alguns processadores de SPARQL, como o ARQ, costumam exibir os resultados das consultas em forma tabular, com linhas e colunas, sendo a coluna a representante de cada nome de variável presente após a palavra-chave SELECT;
- ASK - esta cláusula funciona como um questionador de conformidades. Dado o padrão exposto na sentença, ela deve retornar um valor booleano que mostre se a expressão de triplos - que pode ser condicionada - é uma verdade ou não, dentro da ontologia;
- CONSTRUCT - responsável por retornar triplos de uma base sem que estes sejam modificados ou para criar novos triplos. Esta cláusula permite a construção de

relações com triplos, ou seja, a manipulação completa com o fim de construir uma nova parte da ontologia;

- DESCRIBE - apesar de não ser muito usada, devido à inconsistência das suas implementações, esta cláusula devolve triplos que representam a descrição de um dado recurso.

A filtragem dos dados retornados pela consulta, ou das variáveis nela inseridas, estão sob condição do padrão de triplos. Isto acontece porque o sistema utiliza variáveis nas três posições dos triplos, que serão confrontados contra os grafos RDF. Usa-se a cláusula *FILTER* para limpar os triplos retornados, de forma a garantir que o resultado seja o mais aproximadamente possível do que se espera. Também há a capacidade de utilizar operadores lógicos nas expressões do filtro, assim como expressões regulares, para que o conteúdo de uma determinada variável siga exactamente o que define um padrão de caracteres.

Assim como no SQL, o operador *IN* do SPARQL leva a que se utilize uma lista enumerada integrada na consulta. É útil quando se tem um grupo de valores que deve ser usado numa comparação com um outro valor do triplo. Outra facilidade é a capacidade de definir prefixos de grafos com a cláusula *PREFIX*. Com este recurso, pode-se associar a um *alias* um endereço, local ou remoto, de uma base RDF; então, no corpo da consulta, basta que se invoque o nome do prefixo para se alcançar os nomes daquele escopo, sem que seja necessário escrever todo o endereço várias vezes.

A palavra-chave *LIMIT* tem o mesmo significado que a cláusula homónima do SQL, que é limitar o número de resultados da consulta. Isto é especialmente útil quando se está a trabalhar com grafos de porte considerável, e não se deseja perder tempo com uma consulta demorada, ou não se espera mais que um número exacto de resultados. Um conjunto de triplos armazenados não tem uma ordem certa. Já a palavra-chave *OFFSET* dá o poder de omitir, da lista final, alguns resultados que são ignorados a partir de uma determinada posição do índice.

2.2 Processamento de Linguagem Natural

A "linguagem natural é surpreendente. É difícil de imaginar uma melhor API para o conhecimento. É um exemplo de semântica: símbolos referem-se a coisas ou conceitos, e sequências de símbolos carregam um significado." [23, p. 3] A troca de informações entre pessoas na Internet e mesmo fora dela é praticamente quase toda feita em linguagem natural, seja nas redes sociais, nos *e-books*, nos *chats* ou em outros tipos de documentos digitalizados. No entanto, o uso de textos digitais sempre implica uma maior complexidade, ainda mais quando se está a trabalhar com textos sem revisão, com possíveis erros de ortografia ou mesmo problemas semânticos.

Dentro do campo da *PLN*, como foi explicado anteriormente, este trabalho limita-se a explorar as técnicas de *Part-of-speech tagging* e *Named-entity Recognition*. Para fazer

uso da *PLN* nesse sentido, é preciso fundamentar um modelo de linguagem que seja uma aproximação da linguagem natural e descrever formalmente uma gramática a ser usada pelo sistema. Da ambiguidade própria das linguagens naturais, surgiu a necessidade de empregar modelos de distribuição de probabilidade em vez de se considerar apenas um significado para cada palavra ou frase. [21]

2.2.1 Part-of-speech tagging

Como o nome sugere, é uma actividade que tem como principal objectivo a marcação de cada palavra num texto com a sua correcta classe gramatical. Normalmente, é usada em sistemas que executam processamento de linguagem natural. Ao longo dos anos, o aumento do interesse e o sucesso dessa técnica são devidos à criação de *corpora* (conjuntos de documentos) como *Brown Corpus* e *Corpus of Contemporary American English*, que foram desenvolvidos manualmente e aprimorados com o passar dos anos. Esses conjuntos anotados foram utilizados para a criação e aperfeiçoamento de *taggers* (marcadores). Os mais eficientes marcadores chegaram à marca dos 96% de acerto [15], ou seja, dentro dos 96%, cada palavra ou *token*, o que inclui vírgulas e outros caracteres simbólicos, foi categorizado correctamente. Esse resultado, entretanto, não é tão satisfatório, dado que um único texto pode ter milhares de frases, e cada frase tem, em média, cerca de vinte palavras. Isso quer dizer que, de um modo geral, uma palavra em cada frase é anotada erroneamente. [14, 15] Os dois maiores obstáculos apontados por [15] são: a presença maciça de palavras ambíguas na língua inglesa e palavras desconhecidas.

Mesmo com esses dois problemas (o último prejudica, inclusive, a posterior análise sintática, dado que palavras desconhecidas não são marcadas), a técnica de *Part-of-speech tagging* é usada, principalmente, nas áreas de *Information Retrieval*, *Question Answering Systems*, *Partial Parsing*, *Lexical Acquisition*, *Information Extraction* e na mineração de dados textuais. [14, 15]

Em conformidade com [15], quando se trata de *Part-of-speech tagging*, desde o seu início, as abordagens mais vistas e consideradas convencionais são:

- 1) *Rule-based Methods*, que hoje é tida como ineficiente e até impraticável, pois requer apoio teórico de Linguística para escrever as regras da língua a ser ensinada ao *tagger*;
- 2) *Transformation-based Learning*, uma abordagem inspirada na diminuição da carga manual embutida nos *Rule-based Methods*. Para isso, as regras não são introduzidas manualmente, mas aprendidas por meio dos próprios documentos. Ainda assim, num processo iterativo, há uma etapa de comparação dos resultados obtidos com textos marcados correctamente, para que a transformação das regras seja efectuada, até que se consiga alcançar um limiar pré-determinado;
- 3) *Randomized Transformation-based Learning*, com a mesma base da anterior, apenas restringe o número de transformações que corrigem as marcações, com vista a

diminuir o tempo de processamento. As transformações são escolhidas de maneira aleatória;

- 4) *Markov Model Taggers*, abordagem introduzida no início da década de 90, começou a substituir os métodos anteriores. Com a mudança de rumo da *NLP* para os métodos estatísticos, os *Hidden Markov Models* ganharam destaque;

- 5) *Maximum Entropy Methods*, outra abordagem estatística, toma o contexto das palavras no texto para formular a probabilidade condicional usada no cálculo da entropia. A distribuição de probabilidade que maximiza a entropia é, então, assumida como a condição para se ter informação em falta. É com base nesta distribuição que as inferências são feitas. Ainda segundo [15], outras abordagens adicionais têm sido usadas, tais como *Support Vector Machines*, *Neural Networks* e *Decision Trees*.

A Part-of-speech tagging é uma tarefa essencial de pré-processamento para muitas aplicações que usam Processamento de Linguagem Natural. A tarefa de POST é responsável por marcar cada palavra num texto com a sua correcta classe gramatical. Para concretizar essa tarefa, usa-se uma *tagset* - exemplificada na Figura 3.1 -, que é o conjunto de possíveis classes para uma determinada tarefa. [3, p. 179]

Tag	Meaning	Examples
ADJ	adjective	*new, good, high, special, big, local*
ADV	adverb	*really, already, still, early, now*
CNJ	conjunction	*and, or, but, if, while, although*
DET	determiner	*the, a, some, most, every, no*
EX	existential	*there, there's*
FW	foreign word	*dolce, ersatz, esprit, quo, maitre*
MOD	modal verb	*will, can, would, may, must, should*
N	noun	*year, home, costs, time, education*
NP	proper noun	*Alison, Africa, April, Washington*
NUM	number	*twenty-four, fourth, 1991, 14:24*
PRO	pronoun	*he, their, her, its, my, I, us*
P	preposition	*on, of, at, with, by, into, under*
TO	the word *to*	*to*
UH	interjection	*ah, bang, ha, whee, hmpf, oops*
V	verb	*is, has, get, do, make, see, run*
VD	past tense	*said, took, told, made, asked*
VG	present participle	*making, going, playing, working*
VN	past participle	*given, taken, begun, sung*
WH	*wh* determiner	*who, which, when, what, where, how*

Figura 2.1: Exemplo de tagset [3, p. 183]

Alguns problemas

Segundo [15, p. 107], dois são os maiores problemas que impedem que as marcações atinjam a totalidade dos acertos: a presença de palavras ambíguas e palavras desconhecidas. Ainda de acordo [15], os melhores marcadores chegam a 96% de classificações certas. A presença de palavras com mais de um significado gramatical é um dificultador, pois uma palavra que pode ser um adjectivo, um advérbio, um verbo ou um substantivo depende do contexto, ou seja, da frase em que está inserida.

Métodos

As principais abordagens usadas são os métodos baseados em regras, nos quais muitos pesquisadores tiraram vantagem de textos previamente marcados para desenvolver sistemas. O maior problema desta abordagem é a excessiva intervenção manual necessária na marcação dos textos que servem de apoio. [15, p. 108]

Dado o problema relatado acima, os métodos que não precisam de tanta intervenção ganharam logo a atenção dos estudiosos relacionados ao tema. Um desses métodos é o baseado em transformações. Neste método, não há o recurso dos textos já marcados, mas um sistema de aprendizagem a partir dos textos. A tarefa, feita em três passos, começa com a marcação de cada palavra do texto com uma classificação, que pode ser aleatória ou tendo em conta o maior número de marcações do texto de treino ou, simplesmente, atribuindo a classificação de substantivo a cada palavra. [5] Segue-se a fase de aprendizagem, na qual são aplicados *templates* de regras, instanciados com dados dos textos anotados na primeira etapa. O conjunto de regras criado é aplicado a este *corpus*, e o resultado é comparado com um *corpus* anotado manualmente (que é considerado correcto). A partir desta etapa, as transformações são registadas e seleccionadas de acordo com a sua taxa de acerto. "A fase de transformações é dada como encerrada quando não há mais transformações que reduzam o erro abaixo de um determinado limiar." [15, p. 108]

O método dos marcadores do Modelo de Markov começou, junto com outros métodos estatísticos, a ganhar espaço no meio científico, ocupando o lugar dos métodos baseados em regras. Esta mudança de paradigma foi influenciada pela multiplicação e divulgação de *corpus* anotados. [10] As HMM são "a versão do formalismo de Markov mais frequentemente aplicada à tarefa de *POS tagging*". [15, 109] O seu uso, basicamente, abrange a criação de uma tabela de probabilidades das frequências de sequências de palavras; para isto, alguns *corpus* devem ser processados. A tabela é usada no algoritmo para inferir qual a próxima palavra mais provável, dada a palavra que acabou de ser encontrada. Podem ser usadas sequências para além de duas palavras, ou seja, o sistema pode inferir, por exemplo, que, após a ocorrência de dois verbos e um pronome, a maior probabilidade é a de aparecer uma preposição.

2.2.2 Named-entity Recognition (NER)

Named-entities (ou entidades nomeadas) são sintagmas nominais, normalmente compostos por pelo menos um substantivo, que se referem a tipos específicos de indivíduos, tais como organizações, pessoas, datas, entre outros. "Devem ser auto-explicativos, com excepção de alguns tipos de NE, como os de localidade e o tipo GPE (entidades geopolíticas tais como cidades, estados, províncias e países)" [3, p. 281], que, como tal, podem ter elementos que nada tem a ver com o significado do termo. Por exemplo, "Rio de Janeiro"não tem relação imediata com um rio nem com o mês de janeiro.

O principal propósito desta tarefa é reunir todas as referências às entidades nomeadas num texto. Para isso, o sistema pode ser dividido em duas subtarefas. A primeira diz respeito à identificação dos limites das entidades nomeadas, com a consequente identificação do tipo de cada uma. A maior parte dos sistemas de QA, por exemplo, foca-se na limitação do conteúdo à parte que provavelmente mais interessa, para que, com isso, tenha um desempenho mais elevado. [3, p. 281]

Para a identificação das entidades nomeadas, uma das formas é verificar cada palavra numa lista de nomes apropriada. Por exemplo, no caso de localidades, pode-se usar um dicionário geográfico. Contudo, ao fazer isto, existe o risco de se gerarem rapidamente problemas, como se pode ver na figura abaixo:

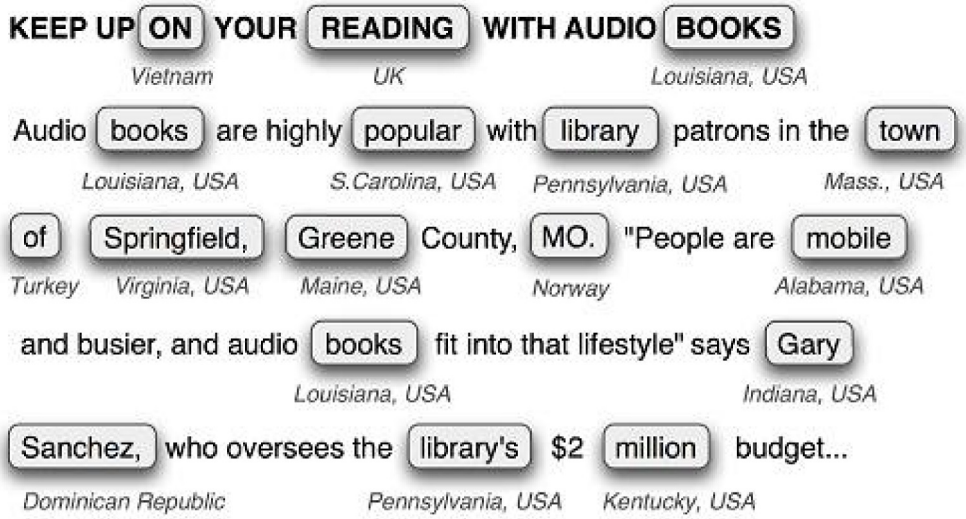

Figura 2.2: Exemplo de detecção de NE de localidade [3, p. 282]

Como se pode ver na imagem, muitos termos foram mal classificados como entidades nomeadas, como é o exemplo de *ON*, que foi categorizado como uma cidade do Vietnam. A ambiguidade de que se revestem estes termos acaba por contaminar os resultados da classificação. Por exemplo, o nome do autor de livros, Peter French, contém um termo que pode se confundir com uma nacionalidade.

Segundo [21], "*Named-entity Recognition* é a tarefa de localizar nomes em um do-

cumento e decidir a que classes eles pertencem". A técnica de *Named-entity Recognition* "foi introduzida nos anos 90 como parte das actividades das *Message Understanding Conferences* (*MUC*). A meta dessas conferências era o reconhecimento de nomes próprios (pessoas, organizações, localidades) e de outras frases contendo datas, intervalos de tempo e medidas, em artigos de jornais escritos em inglês". [24] Sendo assim, pode-se perceber que essa técnica oferece exactamente o que o presente trabalho precisa para buscar alguns dos metadados no conteúdo dos *e-books* e para a posterior confrontação com os recursos na *Web*.

Costumam-se usar modelos de caracteres *n-gram*, baseados em *Markov Chains*, para as tarefas de *Named-entity Recognition*, classificação de géneros literários, reconhecimento de idiomas e correcção ortográfica. Com o uso desses modelos, sistemas computacionais conseguem reconhecer idiomas com uma precisão maior que 99%. [21]

Num trabalho com uma função parecida, mas voltada para textos jurídicos [20], foi proposta a utilização de métodos de *Machine Learning*, tais como *Decision Trees* e *Hidden Markov Models* para atingir a extracção e o reconhecimento de entidades nomeadas. Os seus resultados foram considerados parcialmente bons, como seguem as taxas de acerto: 99,9% para datas, 80-90% para localidades e 35% para nomes de organizações e referências a artigos e legislações. A taxa de acerto mais baixa é explicada pela inadequação do classificador semântico para reconhecer as organizações e pela complexidade elevada da estrutura sintática das referências. Isso demonstra que as ferramentas empregadas devem ser escolhidas e estar reguladas conforme a natureza dos documentos que são acedidos.

POST e NER no NLTK

A biblioteca NLTK dispõe de um classificador treinado para reconhecer entidades nomeadas. A função nltk.ne_chunk() devolve todas as entidades que encontra, e tem a opção, por via de argumento, de devolvê-la tipadas, ou seja, acompanhadas de um marcador relativo à sua característica mais predominante, como localidade, nome de pessoa ou nome de organização. [3, p. 283]

A tarefa de POST na NLTK é feita em conjunto com duas tarefas de pré-processamento, a *tokenização* de orações e, depois, de palavras. O processo de *tokenização* de orações, também conhecido como SBD, é a definição do início e do fim de cada oração de um texto. O maior desafio deste procedimento é a ambiguidade que alguns sinais de pontuação têm: um ponto de exclamação pode estar presente, por exemplo, na expressão *(sic!)*, ou um ponto final pode estar presente numa abreviatura.

O resultado da *tokenização* de uma oração é passado para a segmentação de palavras, ou seja, a partir das orações destacadas, o algoritmo deve determinar quais são as palavras. A função de segmentação de palavras do NLTK utiliza o método de *tokenização* Treebank, que, basicamente, faz alterações em pontos especiais nos textos de língua inglesa, como a separação do apóstrofo da palavra a que está ligado.

A função de *POS tagging* recebe as orações já segmentadas em palavras, categoriza-as e o resultado desta classificação é passado para a função de extracção de entidades nomeadas, que ciente da classe gramatical de cada palavra, decreta no retorno quais são NE.

Capítulo 3

Sistema para verificação de metadados de livros electrónicos

Para efeito ilustrativo, decidi criar uma ontologia para representar, de forma básica, o domínio dos *e-books*. Foram expostas entidades que são comuns e que formam, independentes, metadados. Embora somente os metadados de nome do livro e nome do autor tenham sido utilizados no trabalho prático, quaisquer dos outros poderiam ter sido usados, desde que um método de extracção adequado fosse empregado.

Segue abaixo a ontologia completa, criada com o auxílio do *software* Protégé OWL.

3.1 A Ontologia em OWL

A ontologia é composta pelas seguintes classes detalhadas:

- Author

  É a classe que deve agrupar todos os autores das obras publicadas em formato digital, os e-books. Esta classe tem muitas possíveis subclasses, como EnglishAuthor, ScientificAuthor e NovelistAuthor.

- Book

  A classe Book representa as obras disponíveis em formato de e-book. Os indivíduos desta classe podem ser categorizados em diversas subclasses, como EnglishBook, DramaBook ou HistoryBook.

- Country

Country é uma classe cujo único intuito é a representação de países, que, neste âmbito, podem ser a nacionalidade de um escritor, o país de estabelecimento de uma editora ou o país de alguma língua.

- Language

  Language representa os idiomas existentes, tendo utilidade fundamental na relação com os metadados. Os autores escrevem em um ou mais idiomas; obras são publicadas em um idioma; países falam determinados idiomas; editoras publicam em um ou mais idiomas.

- Publisher

  Publisher é a representação das editoras ou grupos editoriais responsáveis pela publicação dos livros. Suas relações com outras classes da ontologia dão-se pelos idiomas que a editora utiliza, pelos livros que a editora publica, pelo seu ano de fundação e pelos seus países de estabelecimento.

- Year

  A classe Year é a entidade dos anos. É útil para relacionar as entidades da ontologia com os metadados extraídos dos e-books. Como exemplo, um possível metadado é o ano de publicação de um livro. Suas relações incluem o ano de nascimento de um autor, o ano de fundação de uma editora e o ano de publicação de um livro.

As propriedades que compõem as relações da ontologia são:

- authorHasAuthorshipOfBook

  Um autor pode ter muitos livros. Um livro também pode pertencer a mais de um autor.

  - Domínio: Author
  - Contradomínio: Book
  - Inverso de: bookIsAuthorshipOfAuthor
  - Características: assimétrica e irreflexiva

- authorWasBornInCountry

  Um autor nasceu em um país, um único país. Esta propriedade tem carácter funcional.

  - Domínio: Author
  - Contradomínio: Country
  - Inverso de: countryOfBirthOfAuthor
  - Características: funcional, assimétrica e irreflexiva

- authorWasBornInYear

  Um autor nasceu em um determinado ano; o que leva a propriedade a ter um carácter funcional.

  - Domínio: Author
  - Contradomínio: Year
  - Inverso de: yearOfBirthOfAuthor
  - Características: funcional, assimétrica e irreflexiva.

- authorWritesInLanguage

  Um autor pode escrever em mais de um idioma. Um idioma pode ser relacionado, por meio desta propriedade, a muitos autores.

  - Domínio: Author
  - Contradomínio: Language
  - Inverso de: languageIsWrittenByAuthor
  - Características: assimétrica e irreflexiva

- bookIsAuthorshipOfAuthor

  Um livro é da autoria de um ou mais autores. Um autor pode estar relacionado, por meio desta propriedade, a vários livros.

  - Domínio: Book
  - Contradomínio: Author
  - Inverso de: authorHasAuthorshipOfBook
  - Características: assimétrica e irreflexiva

- bookIsPublishedByPublisher

  Um livro é publicado por uma única editora. A propriedade tem carácter funcional. Uma editora pode estar relacionada, por meio desta propriedade, a vários livros.

  - Domínio: Book
  - Contradomínio: Publisher
  - Inverso de: publisherIsPublisherOfBook
  - Características: funcional, assimétrica e irreflexiva

- bookIsPublishedInLanguage

  Um livro é publicado em um único idioma. Apesar de ser possível a existência de trechos escritos noutras línguas, considera-se que o livro foi escrito ou traduzido em

apenas um idioma. A propriedade tem carácter funcional. Um idioma pode estar relacionado, por meio desta propriedade, a muitos livros.

  - Domínio: Book
  - Contradomínio: Language
  - Inverso de: languageOfPublishingOfBook
  - Características: funcional, assimétrica e irreflexiva

- bookWasPublishedInYear

  Um livro foi publicado em um ano específico. A propriedade é funcional. Um ano pode estar relacionado, por meio desta propriedade, a muitos livros.

  - Domínio: Book
  - Contradomínio: Year
  - Inverso de: yearOfPublicationOfBook
  - Características: funcional, assimétrica e irreflexiva

- countryOfBirthOfAuthor

  A nacionalidade de um autor é única, pelo que esta propriedade é funcional inversa. Um país pode ser o berço de muitos autores.

  - Domínio: Country
  - Contradomínio: Author
  - Inverso de: authorWasBornInCountry
  - Características: funcional inversa, assimétrica e irreflexiva

- countryOfEstablishmentOfPublisher

  Um país pode sediar diversas editoras. Uma editora só está relacionada a um país, por meio desta propriedade.

  - Domínio: Country
  - Contradomínio: Publisher
  - Inverso de: publisherIsEstablishedAtCountry
  - Características: funcional inversa, assimétrica e irreflexiva

- countrySpeaksLanguage

  Um país pode ter mais de uma língua. Uma mesma língua pode ser usada por mais de um país.

  - Domínio: Country

  - Contradomínio: Language
  - Inverso de: languageIsSpokenByCountry
  - Características: assimétrica e irreflexiva

- languageIsPublishedByPublisher

  O idioma de publicação de uma editora pode não ser único. Uma mesma editora pode publicar livros em diversas línguas. Um idioma pode ser usado por muitas editoras.

  - Domínio: Language
  - Contradomínio: Publisher
  - Inverso de: publisherPublishesInLanguage
  - Características: assimétrica e irreflexiva

- languageIsSpokenByCountry

  Uma língua pode ser usada por muitos países. Um mesmo país pode usar mais de uma língua.

  - Domínio: Language
  - Contradomínio: Country
  - Inverso de: countrySpeaksLanguage
  - Características: assimétrica e irreflexiva

- languageIsWrittenByAuthor

  Um idioma pode ser utilizado por muitos autores. O facto de um autor utilizar um idioma não implica a inutilização de outros.

  - Domínio: Language
  - Contradomínio: Author
  - Inverso de: authorWritesInLanguage
  - Características: assimétrica e irreflexiva

- languageOfPublishingOfBook

  O idioma de publicação é único para um livro. Por isso, é uma propriedade funcional inversa. O domínio, Language, deve ser único para o contradomínio, Book.

  - Domínio: Language
  - Contradomínio: Book
  - Inverso de: bookIsPublishedInLanguage

  - Características: funcional inversa, assimétrica e irreflexiva

- publisherIsEstablishedAtCountry

  Uma editora está estabelecida em apenas um único país. Caso existam diferentes filiais internacionais de uma editora, devem ser indivíduos diferentes.

  - Domínio: Publisher
  - Contradomínio: Country
  - Inverso de: countryOfEstablishmentOfPublisher
  - Características: funcional, assimétrica e irreflexiva

- publisherIsPublisherOfBook

  Uma editora pode ter publicado muitos livros. Porém, o livro só foi publicado por uma editora, pelo que esta propriedade é funcional inversa. O contradomínio Livro só pode estar ligado a um indivíduo do domínio Editora.

  - Domínio: Publisher
  - Contradomínio: Book
  - Inverso de: bookIsPublishedByPublisher
  - Características: funcional inversa, assimétrica e irreflexiva

- publisherPublishesInLanguage

  Uma editora publica em uma ou mais línguas. Um idioma pode estar relacionado, por meio desta propriedade, a muitas editoras.

  - Domínio: Publisher
  - Contradomínio: Language
  - Inverso de: languageIsPublishedByPublisher
  - Características: assimétrica e irreflexiva

- publisherWasFoundedInYear

  Uma editora foi fundada em um único ano. Esta propriedade tem carácter funcional. Várias editoras podem ter sido fundadas no mesmo ano.

  - Domínio: Publisher
  - Contradomínio: Year
  - Inverso de: yearOfFoundationOfPublisher
  - Características: functional, assimétrica e irreflexiva

- yearOfBirthOfAuthor

  Muitos autores podem ter nascido no mesmo ano. Pelo contrário, um autor nasceu num determinado ano. A propriedade, portanto, é funcional inversa.

  - Domínio: Year
  - Contradomínio: Author
  - Inverso de: authorWasBornInYear
  - Características: funcional inversa, assimétrica e irreflexiva

- yearOfFoundationOfPublisher

  Muitas editoras podem ter sido fundadas no mesmo ano. Por outro lado, uma editora foi fundada num determinado ano. Por este motivo, esta propriedade é funcional inversa.

  - Domínio: Year
  - Contradomínio: Publisher
  - Inverso de: publisherWasFoundedInYear
  - Características: funcional inversa, assimétrica e irreflexiva

- yearOfPublicationOfBook

  O ano de publicação de um livro deve ser único para um livro. Note-se que, se uma obra for reeditada, trata-se de outro indivíduo de livro. Portanto, esta propriedade é funcional inversa.

  - Domínio: Year
  - Contradomínio: Book
  - Inverso de: bookWasPublishedInYear
  - Características: funcional inversa, assimétrica e irreflexiva

Segue abaixo, na Figura 4.1, uma amostra gráfica de como se relacionam as classes da ontologia, com arcos a representar as propriedades acima mencionadas e explicadas.

3.2 Inclusão de factos na ontologia

A Freebase [4], uma base de dados genérica e que oferece acesso aos dados de forma gratuita, serviu como meio para reunir dados suficientes para que este trabalho se concretizasse praticamente. A base oferece suporte a consultas por meio da linguagem MQL, que é inteiramente baseada na sintaxe do JSON. O processo de extracção de autores e livros deu-se pela própria plataforma online da Freebase.

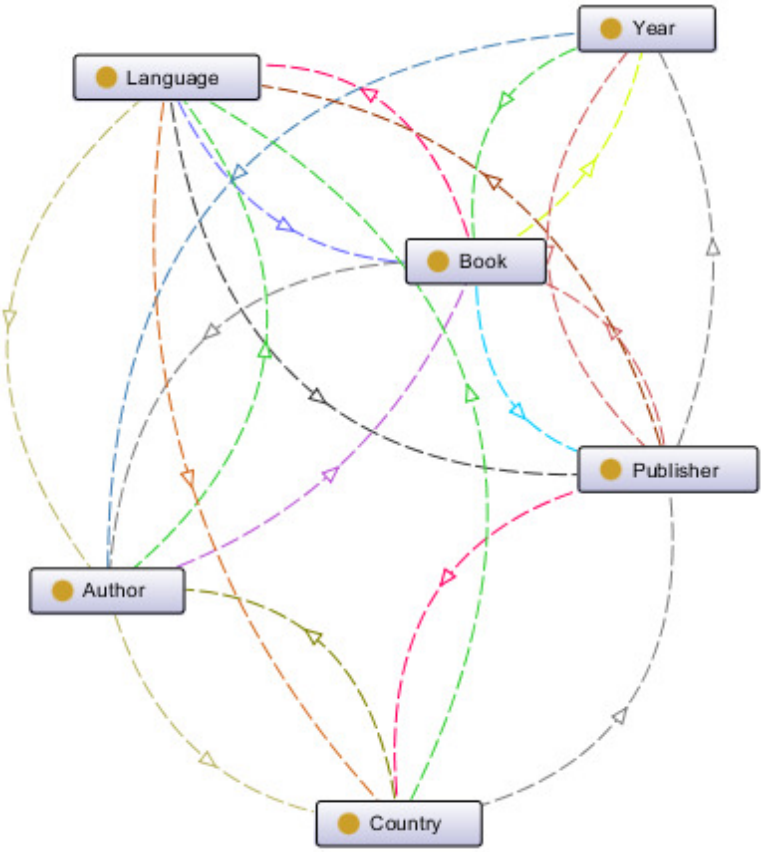

Figura 3.1: Representação gráfica da ontologia

Com o apoio da linguagem PHP, foi codificado um *script* para fazer a transformação dos dados extraídos da Freebase em um dicionário em formato de texto. Em seguida, um outro script foi desenvolvido para, a partir do dicionário gerado, formar triplos que, finalmente, fossem incorporados à ontologia. Esses triplos podem referir-se a autores, livros ou à associação entre ambos. Em formato RDF, os recursos foram introduzidos na ontologia com proveito das propriedades já existentes, como *authorHasAuthorshipOfBook*, *authorHasName*, *bookIsAuthorshipOfAuthor* e *bookHasTitle*, que foram empregadas para efeitos demonstrativos deste trabalho prático.

3.3 Extracção e validação de metadados

O foco da parte prática é mostrado na Figura 3.3. Esta parte inclui o processamento de um *script* implementado em Python para extracção das entidades nomeadas; a execução de consultas criadas com SPARQL para interrogação de cada entidade na ontologia; apresentação dos resultados. Cada um desses procedimentos é descrito com mais pormenor abaixo.

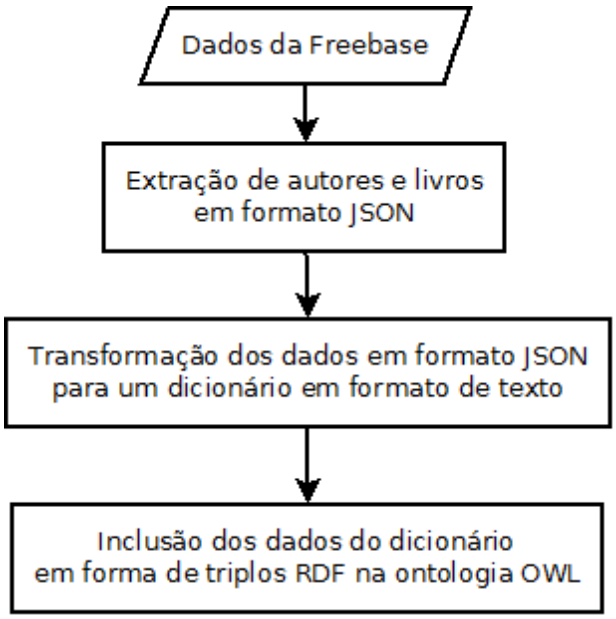

Figura 3.2: Processo detalhado de inclusão de factos

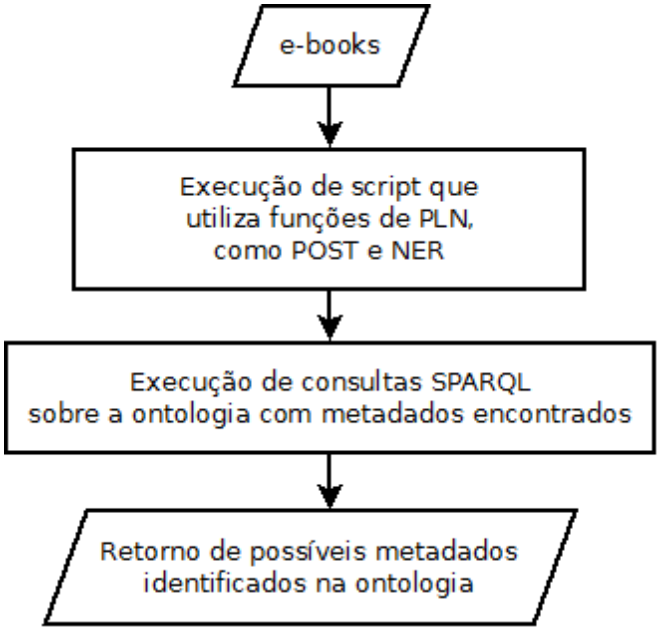

Figura 3.3: Processo detalhado de extracção e validação de metadados

A selecção dos e-books foi feita pelo site Project Gutenbeg [11], por meio do qual foi feito o download de cem *e-books*, de textos clássicos, de trinta e sete diferentes autores.

O *script* Python, que foi produzido para a extracção de entidades nomeadas de *e-books*, teve o fundamental apoio da biblioteca NLTK [3], cujas funções de POST e NER, em conjunto, agiram sobre os primeiros quinhentos caracteres de cada *e-book* processado. Este número foi escolhido com base na observação de que a maioria dos *e-books* contém

os metadados logo no início do corpo do texto. Embora haja casos em que os metadados originais (e.g. nome do autor, título) estão ausentes de todo o corpo ou espalhados aleatoriamente pelo conteúdo, este estudo não se propõe a cobri-los. O resultado da execução do *script*, portanto, é uma lista de entidades retornadas pela respectiva função de NER. Tal lista é então armazenada num ficheiro que será, posteriormente, usado para a comparação dos metadados na ontologia.

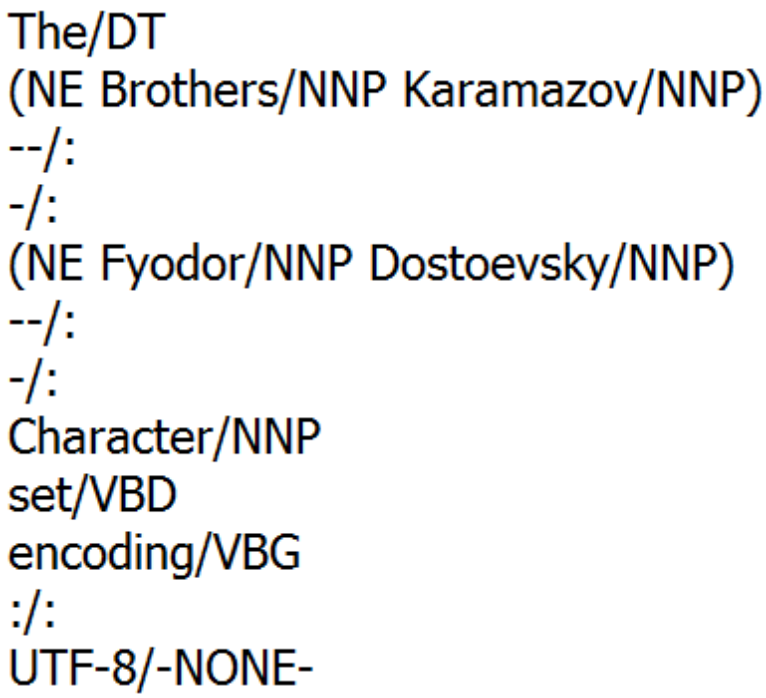

Figura 3.4: Trecho do resultado do script de extracção de entidades nomeadas

A segunda etapa da parte prática dá-se pela verificação das entidades nomeadas. Um script PHP que usa a biblioteca ARC [19] para a execução de consultas SPARQL, primeiramente faz todas as associações possíveis entre as entidades, como se cada combinação tivesse a relação de autor com a obra. Em seguida, empregando esse conjunto de associações, executa, sobre cada par de termos, uma consulta à ontologia, já previamente encorpada com todos os autores e obras relacionados. Como retorno da consulta, há, então, o par correspondente ao autor e ao nome da obra. É possível que seja mostrado mais de um par, caso se encontrem na ontologia relações que satisfaçam os termos buscados.

```
PREFIX ont: <http://www.semanticweb.org/felipe/ontologies/m_ontology#>
SELECT ?authorName ?bookTitle
WHERE {
        ?author ont:authorHasAuthorshipOfBook ?book .
        ?book ont:bookHasTitle ?bookTitle .
        ?author ont:authorHasName ?authorName .
        FILTER regex(?bookTitle, '$book', 'i') .
```

```
            FILTER regex(?authorName, '$author', 'i')
    }
    LIMIT 3
```

A consulta SPARQL, como se pode ver acima ou, melhor enquadrada no seu escopo, no Anexo B, contém uma secção condicional. Nesta, a pesquisa pela conexão entre o autor e o livro está presente pela via do predicado *authorHasAuthorshipOfBook*. Já os metadados são procurados por meio de uma expressão regular com a opção *case-insensitive*, na cláusula *FILTER*, que usa os valores das variáveis que contêm as entidades nomeadas do livro da iteração. O limite de três resultados foi posto por não haver, neste estudo, relevância a mostragem de mais achados, além de ser suficiente tal quantidade, para que haja uma ideia acerca do funcionamento do algoritmo.

```
NER_MODIFIED_fyodor_dostoevsky-the_brothers_karamazov.txt

array (size=8)
  0 => string 'Brothers Karamazov' (length=18)
  1 => string 'Fyodor Dostoevsky' (length=17)
  2 => string 'BrothersS Karamazov' (length=19)
  3 => string 'Russian' (length=7)
  4 => string 'Fyodor Dostoyevsky' (length=18)
  5 => string 'Constance Garnett' (length=17)
  6 => string 'Lowell' (length=6)
  7 => string 'New York' (length=8)

Resultado: Fyodor Dostoevsky --- The Brothers Karamazov

NER_MODIFIED_fyodor_dostoevsky-the_idiot.txt

array (size=11)
  0 => string 'Idiot' (length=5)
  1 => string 'Fyodor Dostoevsky' (length=17)
  2 => string 'Martin Adamson' (length=14)
  3 => string 'David Widger' (length=12)
  4 => string 'Andrew Sly' (length=10)
  5 => string 'IDIOT' (length=5)
  6 => string 'Fyodor Dostoyevsky' (length=18)
  7 => string 'Eva Martin' (length=10)
  8 => string 'PART NN' (length=7)
  9 => string 'Warsaw' (length=6)
  10 => string 'Petersburg' (length=10)

Resultado: Fyodor Dostoevsky --- The Idiot
```

```
NER_MODIFIED_chesterton-orthodoxy.txt

array (size=16)
  0 => string 'Orthodoxy' (length=9)
  1 => string 'Chesterton' (length=10)
  2 => string 'Jonathan Ingram' (length=15)
  3 => string 'Clare Coney' (length=11)
  4 => string 'Online' (length=6)
  5 => string 'Distributed' (length=11)
  6 => string 'ORTHODOXY' (length=9)
  7 => string '_by_' (length=4)
  8 => string 'CHESTERTON' (length=10)
  9 => string 'JOHN' (length=4)
  10 => string 'BODLEY' (length=6)
  11 => string '_First' (length=6)
  12 => string 'in_' (length=3)
  13 => string '_printed_' (length=9)
  14 => string '_Reprinted_' (length=11)
  16 => string '_Reprinte)' (length=10)

Resultado: G K Chesterton --- Orthodoxy
```

O resultado da análise automática pode ser visto acima em alguns exemplos retirados dentre os cem livros analisados. A taxa de acerto, após verificação do processo executado por todos os cem livros, foi de 87%. Os casos de falha devem-se à inexistência de entidades nomeadas na área analisada que pudessem servir de termos para a consulta à ontologia.

Capítulo 4

Conclusões e trabalho futuro

O presente trabalho foi realizado sobre a Web Semântica e o Processamento de Linguagem Natural. A natureza dessas áreas implica sempre um trabalho sobre a Incerteza. Para que se consiga alcançar um desempenho e um resultado apropriado, deve-se restringir o escopo ao objecto de estudo. Tendo o foco em um domínio particular, com todas as suas peculiaridades, há menor chance de cair em abstracções que não levam a resoluções de problemas nem a avanços científicos, mas a um mais valioso e adaptado uso das técnicas.

Por ter um valor académico, o actual trabalho não tem uma aplicabilidade prática e imediata. Deve ser visto pelo seu valor de exploração de algumas das linguagens da *Web 3.0*, que em conjunto com os algoritmos de Processamento de Linguagem Natural, conseguiu bem demonstrar o poder de exploração de uma das possíveis áreas do conhecimento. Apesar de este trabalho ter como tema principal os *e-books*, podia ter como foco qualquer outra extensão prática que pudesse ser representada em forma de ontologia.

O desenvolvimento da ontologia básica foi feito com sucesso. As entidades estão bem representadas e relacionadas. Foi possível provar, sem efeito de comparação com projectos reais, como a Web Semântica tem poder de revolucionar a forma como os dados são tratados na Internet e em grandes bases de dados.

O PLN, como abordagem auxiliadora do projecto, cumpriu bem a sua função. Os metadados são extraídos, combinados em pares de nome de autor e título do livro e, em seguida, verificados por meio de uma consulta SPARQL à ontologia. Esta tarefa foi bem cumprida e mostrou um ciclo inteiro a funcionar, sendo este ciclo passível de modificações para se adequar a qualquer outro fim.

4.1 Limitações

A ontologia criada para este trabalho só contempla a representação de grupos de autores. Está preparada para representar, embora com limitações - dada a noção de que toda ontologia é incompleta em relação à realidade operante do mundo -, todos os autores, autores de uma nacionalidade, autores de um continente, autores de uma editora, por exemplo. Não há, no entanto, a possibilidade, posta a estrutura actual da ontologia, de trabalhar com as peculiaridades de um autor. Caso haja o interesse, tal como o de representar a obra inteira do Fernando Pessoa em uma ontologia, recomendo vivamente a construção de uma ontologia totalmente virada para as entidades da obra que precisam de ser exploradas adequadamente.

O processo de extracção de metadados está completamente limitado para o escopo deste trabalho, pelo que deve ser modificado caso haja interesse em se trabalhar com *e-books* oriundos de qualquer outra fonte. A verificação dos metadados, pelo contrário, só precisa de alguns ajustes, para se amoldar a outros tipos de metadados. Não há outros metadados considerados neste estudo senão o nome do autor e o título do livro. Além de modificar o algoritmo de leitura dos resultados do NLTK, o script de invocação da consulta SPARQL precisa de manutenção para montar a cláusula *WHERE* com as propriedades da ontologia que tenham importância.

4.2 Trabalho Futuro

Dada a forma como as classes e propriedades foram dispostas, é fácil, para um investigador da Web Semântica, aproveitar a estrutura desta ontologia para aprimorá-la às suas necessidades. Também vale ressaltar que, de jeito nenhum, é um requisito que se siga a mesma linha de trabalho feita neste estudo. A ontologia é completamente independente do trabalho de PLN aqui desenvolvido.

Visto que a limitação desta ontologia é a do próprio domínio e a do âmbito deste estudo, toda e qualquer pessoa interessada em explorar a representação de *e-books* genericamente, deve aproveitar somente as partes que interessam.

As classes que não tenham utilidade podem ser removidas e as relações dependentes podem ser desvinculadas. Não há, absolutamente, nenhuma dependência que seja fundamental na representação deste conhecimento. Além disso, pode-se aproveitar a estrutura da ontologia, sem os indivíduos, para trabalhar sobre algum género literário específico, sobre um grupo de autores ou um grupo de editoras de um determinado país, por exemplo.

A exploração da ontologia também pode servir para suprir informação a *softwares* que precisem de usar dados sobre as entidades e relacioná-las de alguma forma, seja com pesquisas ou relatórios. Como, normalmente, as ontologias não são *stovepipe systems*, ou seja, são aproveitadas por outros programas, esta utilidade é usualmente seguida.

A continuação deste projecto com a parte de Processamento de Linguagem Natural

tem um leque amplo de pontos a explorar. Este estudo foi feito com livros provenientes de uma única fonte, o site Project Gutenberg [11]. Tendo em conta que os livros foram processados da mesma forma, sem variantes para formatos distintos - pois os livros têm os metadados localizados em posições não muito diferentes -, um dos trabalhos futuros é criar uma abordagem mais complexa para lidar com livros que tenham os metadados dispersos e, além disso, livros que não tenham os metadados expostos. Para este último caso, deve haver um mecanismo de interacção forte entre a ontologia e os algoritmos de PLN. A ontologia deve fornecer, por exemplo, informações sobre palavras mais utilizadas por determinado autor e o estilo de escrita (e, para isso, a tarefa de *POS tagging* deve ser muito útil).

Bibliografia

[1] M. B. Almeida and M. P. Bax. Uma visão geral sobre ontologias: pesquisa sobre definições, tipos, aplicações, métodos de avaliação e de construção. *Ciência da Informação*, 32:7 – 20, 12 2003.

[2] V. H. Barbosa. Ontologies for context-aware applications. Master's thesis, Faculdade de Engenharia da Universidade do Porto, 2008.

[3] S. Bird, E. Klein, and E. Loper. *Natural Language Processing with Python*. O'Reilly Media, Inc., 1st edition, 2009.

[4] K. Bollacker, P. Tufts, T. Pierce, and R. Cook. A platform for scalable, collaborative, structured information integration. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250, 2008.

[5] E. Brill. Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging. *Comput. Linguist.*, 21(4):543–565, Dec. 1995.

[6] M. C. Daconta, L. J. Obrst, and K. T. Smith. *The Semantic Web: A Guide to the Future of XML, Web Services and Knowledge Management*. Wiley, Indianapolis, IN, 2003.

[7] B. DuCharme. *Learning SPARQL*. St. Laurent, Simon and Perez, Jasmine, 2011.

[8] M. Faria. Definição de uma ontologia aplicada ao futebol. Master's thesis, Faculdade de Engenharia da Universidade do Porto, 2009.

[9] N. Guarino. Understanding, building, and using ontologies. *International Journal of Human Computer Studies*, 1997.

[10] T. Güngör. Part-of-speech tagging. In N. Indurkhya and F. J. Damerau, editors, *Handbook of Natural Language Processing, Second Edition*. CRC Press, Taylor and Francis Group, Boca Raton, FL, 2010. ISBN 978-1420085921.

[11] P. Gutenberg, 2014. Disponível em http://www.gutenberg.org/.

[12] M. Horridge. *Protege OWL Tutorial P4 v1 3 [A Practical Guide To Building OWL Ontologies Using Protege 4 and CO-ODE Tools, Edition 1.3].* The University Of Manchester, 1.3 edition, Mar. 2011.

[13] M. Horridge, N. Drummond, J. Goodwin, A. Rector, R. Stevens, and H. Wang. The manchester owl syntax. *OWLed*, 2006.

[14] C. D. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing.* MIT Press, Cambridge, MA, USA, 1999.

[15] A. R. Martinez. Part-of-speech tagging. *WIREs Comp Stat*, 2012.

[16] E. Morais. Ontologias: conceitos, usos, tipos, metodologias, ferramentas e linguagens. Technical report, Universidade Federal de Goiás, 2007.

[17] F. Parreiras. *Semantic Web and Model-Driven Engineering.* IEEE Press, 2012.

[18] J. T. Pollock. *Semantic Web for dummies.* Hoboken, N.J. Wiley Publishing, 2009. Index.

[19] A. Project, 2014. Disponível em https://github.com/semsol/arc2/wiki.

[20] P. Quaresma and T. Gonçalves. Using linguistic information and machine learning techniques to identify entities from juridical documents. In E. Francesconi, S. Montemagni, W. Peters, and D. Tiscornia, editors, *Semantic Processing of Legal Texts*, volume 6036 of *Lecture Notes in Computer Science*, pages 44–59. Springer Berlin Heidelberg, 2010.

[21] S. J. Russell and P. Norvig. *Artificial Intelligence - A Modern Approach.* Prentice Hall, 3rd edition, 2010.

[22] J. M. G. Saias. Uma metodologia para a construção automática de ontologias e a sua aplicação em sistemas de recuperação de informação. Master's thesis, Universidade de Évora, 2003.

[23] T. Segaran, C. Evans, and J. Taylor. *Programming the Semantic Web - Build Flexible Applications with Graph Data.* O'Reilly Media, 2009.

[24] G. Szarvas, R. Farkas, and R. Ormándi. Improving a state-of-the-art named entity recognition system using the world wide web. In *Industrial Conference on Data Mining*, pages 163–172, 2007.

[25] C. Trojahn, P. Quaresma, and R. Vieira. Ontology mapping for a legal question answering system, 2007.

[26] W3C. W3c resource description framework (rdf), 2014. Disponível em http://www.w3.org/RDF/.

Anexos

Anexo A

As definições de classes e propriedades na linguagem OWL estão todas neste código, que não inclui as instâncias de cada classe.

```
<?xml version="1.0"?>

<!DOCTYPE rdf:RDF [
    <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
    <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
    <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
    <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
    <!ENTITY m_ontology "http://www.semanticweb.org/felipe/ontologies/m_ontology#" >
]>

<rdf:RDF xmlns="http://www.w3.org/2002/07/owl#"
     xml:base="http://www.w3.org/2002/07/owl"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     xmlns:owl="http://www.w3.org/2002/07/owl#"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:m_ontology="http://www.semanticweb.org/felipe/ontologies/m_ontology#">
    <Ontology rdf:about="http://www.semanticweb.org/felipe/ontologies/m_ontology"/>

    <!-- http://www.semanticweb.org/felipe/ontologies/m_ontology#
authorHasAuthorshipOfBook -->


    <ObjectProperty rdf:about="&m_ontology;authorHasAuthorshipOfBook">
        <rdf:type rdf:resource="&owl;AsymmetricProperty"/>
```

```
        <rdf:type rdf:resource="&owl;IrreflexiveProperty"/>
        <rdfs:comment xml:lang="pt">Um autor pode ter muitos livros.
Um livro também pode pertencer a mais de um autor.</rdfs:comment>
        <rdfs:domain rdf:resource="&m_ontology;Author"/>
        <rdfs:range rdf:resource="&m_ontology;Book"/>
        <inverseOf rdf:resource="&m_ontology;bookIsAuthorshipOfAuthor"/>
    </ObjectProperty>

    <!-- http://www.semanticweb.org/felipe/ontologies/m_ontology#
authorWasBornInCountry -->

    <ObjectProperty rdf:about="&m_ontology;authorWasBornInCountry">
        <rdf:type rdf:resource="&owl;AsymmetricProperty"/>
        <rdf:type rdf:resource="&owl;FunctionalProperty"/>
        <rdf:type rdf:resource="&owl;IrreflexiveProperty"/>
        <rdfs:comment xml:lang="pt">Um autor nasceu em um país,
um único país. Esta propriedade tem carácter funcional.
        </rdfs:comment>
        <rdfs:domain rdf:resource="&m_ontology;Author"/>
        <rdfs:range rdf:resource="&m_ontology;Country"/>
        <inverseOf rdf:resource="&m_ontology;countryOfBirthOfAuthor"/>
    </ObjectProperty>

    <!-- http://www.semanticweb.org/felipe/ontologies/m_ontology#
authorWasBornInYear -->

    <ObjectProperty rdf:about="&m_ontology;authorWasBornInYear">
        <rdf:type rdf:resource="&owl;AsymmetricProperty"/>
        <rdf:type rdf:resource="&owl;FunctionalProperty"/>
        <rdf:type rdf:resource="&owl;IrreflexiveProperty"/>
        <rdfs:comment xml:lang="pt">Um autor nasceu em um determinado ano;
o que leva a propriedade a ter um carácter funcional.</rdfs:comment>
        <rdfs:domain rdf:resource="&m_ontology;Author"/>
        <rdfs:range rdf:resource="&m_ontology;Year"/>
    </ObjectProperty>

    <!-- http://www.semanticweb.org/felipe/ontologies/m_ontology#
authorWritesInLanguage -->

    <ObjectProperty rdf:about="&m_ontology;authorWritesInLanguage">
        <rdf:type rdf:resource="&owl;AsymmetricProperty"/>
        <rdf:type rdf:resource="&owl;IrreflexiveProperty"/>
        <rdfs:comment xml:lang="pt">Um autor pode escrever em mais de um
```

```
idioma.</rdfs:comment>
        <rdfs:domain rdf:resource="&m_ontology;Author"/>
        <rdfs:range rdf:resource="&m_ontology;Language"/>
        <inverseOf rdf:resource="&m_ontology;languageIsWrittenByAuthor"/>
    </ObjectProperty>

    <!-- http://www.semanticweb.org/felipe/ontologies/m_ontology#
bookIsAuthorshipOfAuthor -->


    <ObjectProperty rdf:about="&m_ontology;bookIsAuthorshipOfAuthor">
        <rdf:type rdf:resource="&owl;AsymmetricProperty"/>
        <rdf:type rdf:resource="&owl;IrreflexiveProperty"/>
        <rdfs:comment xml:lang="pt">Um livro é da autoria de um ou
mais autores.Um autor pode estar relacionado, por meio desta propriedade,
a vários livros.</rdfs:comment>
        <rdfs:range rdf:resource="&m_ontology;Author"/>
        <rdfs:domain rdf:resource="&m_ontology;Book"/>
    </ObjectProperty>

    <!-- http://www.semanticweb.org/felipe/ontologies/m_ontology#
bookIsPublishedByPublisher -->


    <ObjectProperty rdf:about="&m_ontology;bookIsPublishedByPublisher">
        <rdf:type rdf:resource="&owl;AsymmetricProperty"/>
        <rdf:type rdf:resource="&owl;FunctionalProperty"/>
        <rdf:type rdf:resource="&owl;IrreflexiveProperty"/>
        <rdfs:comment xml:lang="pt">Um livro é publicado por
uma única editora. A propriedade tem carácter funcional.
Uma editora pode estar relacionada, por meio desta propriedade,
a vários livros.</rdfs:comment>
        <rdfs:domain rdf:resource="&m_ontology;Book"/>
        <rdfs:range rdf:resource="&m_ontology;Publisher"/>
    </ObjectProperty>

    <!-- http://www.semanticweb.org/felipe/ontologies/m_ontology#
bookIsPublishedInLanguage -->


    <ObjectProperty rdf:about="&m_ontology;bookIsPublishedInLanguage">
        <rdf:type rdf:resource="&owl;AsymmetricProperty"/>
        <rdf:type rdf:resource="&owl;FunctionalProperty"/>
```

```
        <rdf:type rdf:resource="&owl;IrreflexiveProperty"/>
        <rdfs:comment xml:lang="pt">Um livro é publicado em um único idioma.
Apesar de ser possível a existência de trechos noutras línguas,
considera-se que o livro foi escrito ou traduzido em um idioma.
A propriedade tem carácter funcional. Um idioma pode estar relacionado,
por meio desta propriedade, a muitos livros.</rdfs:comment>
        <rdfs:domain rdf:resource="&m_ontology;Book"/>
        <rdfs:range rdf:resource="&m_ontology;Language"/>
    </ObjectProperty>

    <!-- http://www.semanticweb.org/felipe/ontologies/m_ontology#
bookWasPublishedInYear -->

    <ObjectProperty rdf:about="&m_ontology;bookWasPublishedInYear">
        <rdf:type rdf:resource="&owl;AsymmetricProperty"/>
        <rdf:type rdf:resource="&owl;FunctionalProperty"/>
        <rdf:type rdf:resource="&owl;IrreflexiveProperty"/>
        <rdfs:comment xml:lang="pt">Um livro foi publicado em um
ano específico. A propriedade é &quot;functional&quot;.
Um ano pode estar relacionado, por meio desta propriedade, a muitos
livros.</rdfs:comment>
        <rdfs:domain rdf:resource="&m_ontology;Book"/>
        <rdfs:range rdf:resource="&m_ontology;Year"/>
        <inverseOf rdf:resource="&m_ontology;yearOfPublicationOfBook"/>
    </ObjectProperty>

    <!-- http://www.semanticweb.org/felipe/ontologies/m_ontology#
countryOfBirthOfAuthor -->

    <ObjectProperty rdf:about="&m_ontology;countryOfBirthOfAuthor">
        <rdf:type rdf:resource="&owl;AsymmetricProperty"/>
        <rdf:type rdf:resource="&owl;InverseFunctionalProperty"/>
        <rdf:type rdf:resource="&owl;IrreflexiveProperty"/>
        <rdfs:comment xml:lang="pt">A nacionalidade de um autor é única,
pelo que esta propriedade é &quot;inverse functional&quot;.
Um país pode ser o berço de muitos autores.</rdfs:comment>
        <rdfs:range rdf:resource="&m_ontology;Author"/>
        <rdfs:domain rdf:resource="&m_ontology;Country"/>
    </ObjectProperty>

    <!-- http://www.semanticweb.org/felipe/ontologies/m_ontology#
countryOfEstablishmentOfPublisher -->
```

```
    <ObjectProperty rdf:about="&m_ontology;countryOfEstablishmentOfPublisher">
        <rdf:type rdf:resource="&owl;AsymmetricProperty"/>
        <rdf:type rdf:resource="&owl;InverseFunctionalProperty"/>
        <rdf:type rdf:resource="&owl;IrreflexiveProperty"/>
        <rdfs:comment>Um país pode sediar diversas editoras.
Uma editora só está relacionada a um país, por meio desta
propriedade.</rdfs:comment>
        <rdfs:domain rdf:resource="&m_ontology;Country"/>
        <rdfs:range rdf:resource="&m_ontology;Publisher"/>
    </ObjectProperty>

    <!-- http://www.semanticweb.org/felipe/ontologies/m_ontology#
countrySpeaksLanguage -->


    <ObjectProperty rdf:about="&m_ontology;countrySpeaksLanguage">
        <rdf:type rdf:resource="&owl;AsymmetricProperty"/>
        <rdf:type rdf:resource="&owl;IrreflexiveProperty"/>
        <rdfs:comment>Um país pode ter mais de uma língua.
Uma mesma língua pode ser usada por mais de um país.</rdfs:comment>
        <rdfs:domain rdf:resource="&m_ontology;Country"/>
        <rdfs:range rdf:resource="&m_ontology;Language"/>
        <inverseOf rdf:resource="&m_ontology;languageIsSpokenByCountry"/>
    </ObjectProperty>

    <!-- http://www.semanticweb.org/felipe/ontologies/m_ontology#
languageIsPublishedByPublisher -->


    <ObjectProperty rdf:about="&m_ontology;languageIsPublishedByPublisher">
        <rdf:type rdf:resource="&owl;AsymmetricProperty"/>
        <rdf:type rdf:resource="&owl;IrreflexiveProperty"/>
        <rdfs:comment xml:lang="pt">O idioma de publicação de uma editora
pode não ser único. Uma mesma editora pode publicar livros em
diversas línguas. Um idioma pode ser usado por muitas editoras.
        </rdfs:comment>
        <rdfs:domain rdf:resource="&m_ontology;Language"/>
        <rdfs:range rdf:resource="&m_ontology;Publisher"/>
        <inverseOf rdf:resource="&m_ontology;publisherPublishesInLanguage"/>
    </ObjectProperty>

    <!-- http://www.semanticweb.org/felipe/ontologies/m_ontology#
```

```
languageIsSpokenByCountry -->


    <ObjectProperty rdf:about="&m_ontology;languageIsSpokenByCountry">
        <rdf:type rdf:resource="&owl;AsymmetricProperty"/>
        <rdf:type rdf:resource="&owl;IrreflexiveProperty"/>
        <rdfs:comment>Uma língua pode ser usada por muitos países.
Um mesmo país pode usar mais de uma língua.</rdfs:comment>
        <rdfs:range rdf:resource="&m_ontology;Country"/>
        <rdfs:domain rdf:resource="&m_ontology;Language"/>
    </ObjectProperty>

    <!-- http://www.semanticweb.org/felipe/ontologies/m_ontology#
languageIsWrittenByAuthor -->


    <ObjectProperty rdf:about="&m_ontology;languageIsWrittenByAuthor">
        <rdf:type rdf:resource="&owl;AsymmetricProperty"/>
        <rdf:type rdf:resource="&owl;IrreflexiveProperty"/>
        <rdfs:comment xml:lang="pt">Um idioma pode ser utilizado por muitos
autores. O facto de um autor utilizar um idioma não implica a
inutilização de outros idiomas.</rdfs:comment>
        <rdfs:range rdf:resource="&m_ontology;Author"/>
        <rdfs:domain rdf:resource="&m_ontology;Language"/>
        <rdfs:subPropertyOf rdf:resource="&owl;topObjectProperty"/>
    </ObjectProperty>

    <!-- http://www.semanticweb.org/felipe/ontologies/m_ontology#
languageOfPublishingOfBook -->


    <ObjectProperty rdf:about="&m_ontology;languageOfPublishingOfBook">
        <rdf:type rdf:resource="&owl;AsymmetricProperty"/>
        <rdf:type rdf:resource="&owl;InverseFunctionalProperty"/>
        <rdf:type rdf:resource="&owl;IrreflexiveProperty"/>
        <rdfs:comment xml:lang="pt">O idioma de publicação é único para um
livro. Por isso, é uma propriedade &quot;inverse functional&quot;.
O domínio, language, deve ser único para o contradomínio, book.
        </rdfs:comment>
        <rdfs:range rdf:resource="&m_ontology;Book"/>
        <rdfs:domain rdf:resource="&m_ontology;Language"/>
        <inverseOf rdf:resource="&m_ontology;bookIsPublishedInLanguage"/>
    </ObjectProperty>
```

```
    <!-- http://www.semanticweb.org/felipe/ontologies/m_ontology#
publisherIsEstablishedAtCountry -->


    <ObjectProperty rdf:about="&m_ontology;publisherIsEstablishedAtCountry">
        <rdf:type rdf:resource="&owl;AsymmetricProperty"/>
        <rdf:type rdf:resource="&owl;FunctionalProperty"/>
        <rdf:type rdf:resource="&owl;IrreflexiveProperty"/>
        <rdfs:comment>Uma editora está estabelecida em apenas um único país.
Caso existam diferentes filiais internacionais de uma editora,
devem ser indivíduos diferentes.</rdfs:comment>
        <rdfs:range rdf:resource="&m_ontology;Country"/>
        <rdfs:domain rdf:resource="&m_ontology;Publisher"/>
        <inverseOf rdf:resource="&m_ontology;countryOfEstablishmentOfPublisher"/>
    </ObjectProperty>

    <!-- http://www.semanticweb.org/felipe/ontologies/m_ontology#
publisherIsPublisherOfBook -->

    <ObjectProperty rdf:about="&m_ontology;publisherIsPublisherOfBook">
        <rdf:type rdf:resource="&owl;AsymmetricProperty"/>
        <rdf:type rdf:resource="&owl;InverseFunctionalProperty"/>
        <rdf:type rdf:resource="&owl;IrreflexiveProperty"/>
        <rdfs:comment xml:lang="pt">Uma editora pode ter publicado
muitos livros. Porém, o livro só foi publicado por uma editora,
pelo que esta propriedade é &quot;inverse functional&quot;.
O contradomínio livro só pode estar ligado a um indivíduo do
domínio editora.</rdfs:comment>
        <rdfs:range rdf:resource="&m_ontology;Book"/>
        <rdfs:domain rdf:resource="&m_ontology;Publisher"/>
        <inverseOf rdf:resource="&m_ontology;bookIsPublishedByPublisher"/>
    </ObjectProperty>

    <!-- http://www.semanticweb.org/felipe/ontologies/m_ontology#
publisherPublishesInLanguage -->


    <ObjectProperty rdf:about="&m_ontology;publisherPublishesInLanguage">
        <rdf:type rdf:resource="&owl;AsymmetricProperty"/>
        <rdf:type rdf:resource="&owl;IrreflexiveProperty"/>
        <rdfs:comment xml:lang="pt">Uma editora publica em uma ou mais
línguas. Um idioma pode estar relacionado, por meio desta propriedade,
```

```
a muitas editoras.
        </rdfs:comment>
        <rdfs:range rdf:resource="&m_ontology;Language"/>
        <rdfs:domain rdf:resource="&m_ontology;Publisher"/>
    </ObjectProperty>

    <!-- http://www.semanticweb.org/felipe/ontologies/m_ontology#
publisherWasFoundedInYear -->


    <ObjectProperty rdf:about="&m_ontology;publisherWasFoundedInYear">
        <rdf:type rdf:resource="&owl;AsymmetricProperty"/>
        <rdf:type rdf:resource="&owl;FunctionalProperty"/>
        <rdf:type rdf:resource="&owl;IrreflexiveProperty"/>
        <rdfs:comment xml:lang="pt">Uma editora foi fundada em
um único ano. Esta propriedade tem carácter funcional.
Várias editoras podem ter sido fundadas no mesmo ano.</rdfs:comment>
        <rdfs:domain rdf:resource="&m_ontology;Publisher"/>
        <rdfs:range rdf:resource="&m_ontology;Year"/>
    </ObjectProperty>

    <!-- http://www.semanticweb.org/felipe/ontologies/m_ontology#
yearOfBirthOfAuthor -->

    <ObjectProperty rdf:about="&m_ontology;yearOfBirthOfAuthor">
        <rdf:type rdf:resource="&owl;AsymmetricProperty"/>
        <rdf:type rdf:resource="&owl;InverseFunctionalProperty"/>
        <rdf:type rdf:resource="&owl;IrreflexiveProperty"/>
        <rdfs:comment xml:lang="pt">Muitos autores podem ter nascido
no mesmo ano. Pelo contrário, um autor nasceu num determinado ano.
A propriedade, portanto, é &quot;inverse functional&quot;.</rdfs:comment>
        <rdfs:range rdf:resource="&m_ontology;Author"/>
        <rdfs:domain rdf:resource="&m_ontology;Year"/>
        <inverseOf rdf:resource="&m_ontology;authorWasBornInYear"/>
    </ObjectProperty>

    <!-- http://www.semanticweb.org/felipe/ontologies/m_ontology#
yearOfFoundationOfPublisher -->


    <ObjectProperty rdf:about="&m_ontology;yearOfFoundationOfPublisher">
        <rdf:type rdf:resource="&owl;AsymmetricProperty"/>
        <rdf:type rdf:resource="&owl;InverseFunctionalProperty"/>
```

```
        <rdf:type rdf:resource="&owl;IrreflexiveProperty"/>
        <rdfs:comment xml:lang="pt">Muitas editoras podem ter sido
fundadas no mesmo ano. Por outro lado, uma editora foi fundada num
determinado ano. Por este motivo, esta propriedade é
&quot;inverse functional&quot;.</rdfs:comment>
        <rdfs:range rdf:resource="&m_ontology;Publisher"/>
        <rdfs:domain rdf:resource="&m_ontology;Year"/>
        <inverseOf rdf:resource="&m_ontology;publisherWasFoundedInYear"/>
    </ObjectProperty>

    <!-- http://www.semanticweb.org/felipe/ontologies/m_ontology#
yearOfPublicationOfBook -->


    <ObjectProperty rdf:about="&m_ontology;yearOfPublicationOfBook">
        <rdf:type rdf:resource="&owl;AsymmetricProperty"/>
        <rdf:type rdf:resource="&owl;InverseFunctionalProperty"/>
        <rdf:type rdf:resource="&owl;IrreflexiveProperty"/>
        <rdfs:comment xml:lang="pt">O ano de publicação de um livro
deve ser único para um livro. Note-se que, se uma obra for reeditada,
trata-se de outro indivíduo de livro. Portanto, esta propriedade é
&quot;inverse functional&quot;.</rdfs:comment>
        <rdfs:range rdf:resource="&m_ontology;Book"/>
        <rdfs:domain rdf:resource="&m_ontology;Year"/>
    </ObjectProperty>

    <!--
    //
    // Data properties
    //
     -->

    <!-- http://www.semanticweb.org/felipe/ontologies/m_ontology#
authorHasName -->

    <DatatypeProperty rdf:about="&m_ontology;authorHasName">
        <rdfs:domain rdf:resource="&m_ontology;Author"/>
        <rdfs:range rdf:resource="&xsd;string"/>
    </DatatypeProperty>

    <!-- http://www.semanticweb.org/felipe/ontologies/m_ontology#
bookHasISBN -->
```

```
    <DatatypeProperty rdf:about="&m_ontology;bookHasISBN">
        <rdfs:domain rdf:resource="&m_ontology;Book"/>
        <rdfs:range rdf:resource="&xsd;string"/>
    </DatatypeProperty>

    <!-- http://www.semanticweb.org/felipe/ontologies/m_ontology#
bookHasNumberOfPages -->

    <DatatypeProperty rdf:about="&m_ontology;bookHasNumberOfPages">
        <rdfs:domain rdf:resource="&m_ontology;Book"/>
        <rdfs:range rdf:resource="&xsd;string"/>
    </DatatypeProperty>

    <!-- http://www.semanticweb.org/felipe/ontologies/m_ontology#
bookHasTitle -->

    <DatatypeProperty rdf:about="&m_ontology;bookHasTitle">
        <rdfs:domain rdf:resource="&m_ontology;Book"/>
        <rdfs:range rdf:resource="&xsd;string"/>
    </DatatypeProperty>

    <!--
    //
    // Classes
    //
     -->

    <!-- http://www.semanticweb.org/felipe/ontologies/m_ontology#Author -->

    <Class rdf:about="&m_ontology;Author">
        <rdfs:subClassOf>
            <Restriction>
   <onProperty rdf:resource="&m_ontology;authorWasBornInCountry"/>
   <allValuesFrom rdf:resource="&m_ontology;Country"/>
            </Restriction>
        </rdfs:subClassOf>
        <rdfs:subClassOf>
            <Restriction>
   <onProperty rdf:resource="&m_ontology;authorHasAuthorshipOfBook"/>
   <someValuesFrom rdf:resource="&m_ontology;Book"/>
            </Restriction>
        </rdfs:subClassOf>
        <rdfs:subClassOf>
```

```
         <Restriction>
<onProperty rdf:resource="&m_ontology;authorHasAuthorshipOfBook"/>
<allValuesFrom rdf:resource="&m_ontology;Book"/>
         </Restriction>
     </rdfs:subClassOf>
     <rdfs:subClassOf>
         <Restriction>
<onProperty rdf:resource="&m_ontology;authorWritesInLanguage"/>
<allValuesFrom rdf:resource="&m_ontology;Language"/>
         </Restriction>
     </rdfs:subClassOf>
     <rdfs:subClassOf>
         <Restriction>
<onProperty rdf:resource="&m_ontology;authorWritesInLanguage"/>
<someValuesFrom rdf:resource="&m_ontology;Language"/>
         </Restriction>
     </rdfs:subClassOf>
     <rdfs:subClassOf>
         <Restriction>
<onProperty rdf:resource="&m_ontology;authorWasBornInYear"/>
<allValuesFrom rdf:resource="&m_ontology;Year"/>
         </Restriction>
     </rdfs:subClassOf>
 </Class>

 <!-- http://www.semanticweb.org/felipe/ontologies/m_ontology#Book -->

 <Class rdf:about="&m_ontology;Book">
     <rdfs:subClassOf>
         <Restriction>
<onProperty rdf:resource="&m_ontology;bookWasPublishedInYear"/>
<allValuesFrom rdf:resource="&m_ontology;Year"/>
         </Restriction>
     </rdfs:subClassOf>
     <rdfs:subClassOf>
         <Restriction>
<onProperty rdf:resource="&m_ontology;bookIsPublishedByPublisher"/>
<allValuesFrom rdf:resource="&m_ontology;Publisher"/>
         </Restriction>
     </rdfs:subClassOf>
     <rdfs:subClassOf>
         <Restriction>
<onProperty rdf:resource="&m_ontology;bookIsPublishedInLanguage"/>
```

```
<allValuesFrom rdf:resource="&m_ontology;Language"/>
         </Restriction>
     </rdfs:subClassOf>
     <rdfs:subClassOf>
         <Restriction>
<onProperty rdf:resource="&m_ontology;bookIsPublishedInLanguage"/>
<someValuesFrom rdf:resource="&m_ontology;Language"/>
         </Restriction>
     </rdfs:subClassOf>
     <rdfs:subClassOf>
         <Restriction>
<onProperty rdf:resource="&m_ontology;bookIsAuthorshipOfAuthor"/>
<someValuesFrom rdf:resource="&m_ontology;Author"/>
         </Restriction>
     </rdfs:subClassOf>
     <rdfs:subClassOf>
         <Restriction>
<onProperty rdf:resource="&m_ontology;bookIsPublishedByPublisher"/>
<someValuesFrom rdf:resource="&m_ontology;Publisher"/>
         </Restriction>
     </rdfs:subClassOf>
     <rdfs:subClassOf>
         <Restriction>
<onProperty rdf:resource="&m_ontology;bookIsAuthorshipOfAuthor"/>
<allValuesFrom rdf:resource="&m_ontology;Author"/>
         </Restriction>
     </rdfs:subClassOf>
     <rdfs:subClassOf>
         <Restriction>
   <onProperty rdf:resource="&m_ontology;bookWasPublishedInYear"/>
   <someValuesFrom rdf:resource="&m_ontology;Year"/>
         </Restriction>
     </rdfs:subClassOf>
 </Class>

 <!-- http://www.semanticweb.org/felipe/ontologies/m_ontology#Country -->

 <Class rdf:about="&m_ontology;Country">
     <rdfs:subClassOf>
         <Restriction>
<onProperty rdf:resource="&m_ontology;countryOfBirthOfAuthor"/>
<allValuesFrom rdf:resource="&m_ontology;Author"/>
         </Restriction>
```

```
        </rdfs:subClassOf>
        <rdfs:subClassOf>
            <Restriction>
   <onProperty rdf:resource="&m_ontology;countryOfEstablishmentOfPublisher"/>
   <someValuesFrom rdf:resource="&m_ontology;Publisher"/>
            </Restriction>
        </rdfs:subClassOf>
        <rdfs:subClassOf>
            <Restriction>
      <onProperty rdf:resource="&m_ontology;countryOfBirthOfAuthor"/>
      <someValuesFrom rdf:resource="&m_ontology;Author"/>
            </Restriction>
        </rdfs:subClassOf>
        <rdfs:subClassOf>
            <Restriction>
      <onProperty rdf:resource="&m_ontology;countrySpeaksLanguage"/>
      <someValuesFrom rdf:resource="&m_ontology;Language"/>
            </Restriction>
        </rdfs:subClassOf>
        <rdfs:subClassOf>
            <Restriction>
                <onProperty
rdf:resource="&m_ontology;countryOfEstablishmentOfPublisher"/>
                <allValuesFrom rdf:resource="&m_ontology;Publisher"/>
            </Restriction>
        </rdfs:subClassOf>
        <rdfs:subClassOf>
            <Restriction>
            <onProperty rdf:resource="&m_ontology;countrySpeaksLanguage"/>
            <allValuesFrom rdf:resource="&m_ontology;Language"/>
            </Restriction>
        </rdfs:subClassOf>
    </Class>

    <!-- http://www.semanticweb.org/felipe/ontologies/m_ontology#Language -->

    <Class rdf:about="&m_ontology;Language">
        <rdfs:subClassOf>
            <Restriction>
   <onProperty rdf:resource="&m_ontology;languageOfPublishingOfBook"/>
   <someValuesFrom rdf:resource="&m_ontology;Book"/>
            </Restriction>
        </rdfs:subClassOf>
```

```
     <rdfs:subClassOf>
         <Restriction>
<onProperty rdf:resource="&m_ontology;languageIsWrittenByAuthor"/>
<someValuesFrom rdf:resource="&m_ontology;Author"/>
         </Restriction>
     </rdfs:subClassOf>
     <rdfs:subClassOf>
         <Restriction>
<onProperty rdf:resource="&m_ontology;languageIsSpokenByCountry"/>
<someValuesFrom rdf:resource="&m_ontology;Country"/>
         </Restriction>
     </rdfs:subClassOf>
     <rdfs:subClassOf>
         <Restriction>
<onProperty rdf:resource="&m_ontology;languageIsWrittenByAuthor"/>
<allValuesFrom rdf:resource="&m_ontology;Author"/>
         </Restriction>
     </rdfs:subClassOf>
     <rdfs:subClassOf>
         <Restriction>
<onProperty rdf:resource="&m_ontology;languageIsSpokenByCountry"/>
<allValuesFrom rdf:resource="&m_ontology;Country"/>
         </Restriction>
     </rdfs:subClassOf>
     <rdfs:subClassOf>
         <Restriction>
<onProperty rdf:resource="&m_ontology;languageIsPublishedByPublisher"/>
<allValuesFrom rdf:resource="&m_ontology;Publisher"/>
         </Restriction>
     </rdfs:subClassOf>
     <rdfs:subClassOf>
         <Restriction>
<onProperty rdf:resource="&m_ontology;languageIsPublishedByPublisher"/>
<someValuesFrom rdf:resource="&m_ontology;Publisher"/>
         </Restriction>
     </rdfs:subClassOf>
     <rdfs:subClassOf>
         <Restriction>
<onProperty rdf:resource="&m_ontology;languageOfPublishingOfBook"/>
<allValuesFrom rdf:resource="&m_ontology;Book"/>
         </Restriction>
     </rdfs:subClassOf>
 </Class>
```

```
    <!-- http://www.semanticweb.org/felipe/ontologies/m_ontology#
Publisher -->

    <Class rdf:about="&m_ontology;Publisher">
        <rdfs:subClassOf>
            <Restriction>
   <onProperty rdf:resource="&m_ontology;publisherWasFoundedInYear"/>
   <allValuesFrom rdf:resource="&m_ontology;Year"/>
            </Restriction>
        </rdfs:subClassOf>
        <rdfs:subClassOf>
            <Restriction>
   <onProperty rdf:resource="&m_ontology;publisherIsEstablishedAtCountry"/>
   <someValuesFrom rdf:resource="&m_ontology;Country"/>
            </Restriction>
        </rdfs:subClassOf>
        <rdfs:subClassOf>
            <Restriction>
   <onProperty rdf:resource="&m_ontology;publisherPublishesInLanguage"/>
   <allValuesFrom rdf:resource="&m_ontology;Language"/>
            </Restriction>
        </rdfs:subClassOf>
        <rdfs:subClassOf>
            <Restriction>
                <onProperty rdf:resource="&m_ontology;publisherIsPublisherOfBook"/>
                <someValuesFrom rdf:resource="&m_ontology;Book"/>
            </Restriction>
        </rdfs:subClassOf>
        <rdfs:subClassOf>
            <Restriction>
   <onProperty rdf:resource="&m_ontology;publisherIsEstablishedAtCountry"/>
   <allValuesFrom rdf:resource="&m_ontology;Country"/>
            </Restriction>
        </rdfs:subClassOf>
        <rdfs:subClassOf>
            <Restriction>
                <onProperty rdf:resource="&m_ontology;publisherWasFoundedInYear"/>
                <someValuesFrom rdf:resource="&m_ontology;Year"/>
            </Restriction>
        </rdfs:subClassOf>
        <rdfs:subClassOf>
            <Restriction>
```

```
<onProperty rdf:resource="&m_ontology;publisherPublishesInLanguage"/>
<someValuesFrom rdf:resource="&m_ontology;Language"/>
        </Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
        <Restriction>
            <onProperty rdf:resource="&m_ontology;publisherIsPublisherOfBook"/>
            <allValuesFrom rdf:resource="&m_ontology;Book"/>
        </Restriction>
    </rdfs:subClassOf>
</Class>

<!-- http://www.semanticweb.org/felipe/ontologies/m_ontology#Year -->

<Class rdf:about="&m_ontology;Year">
    <rdfs:subClassOf>
        <Restriction>
    <onProperty rdf:resource="&m_ontology;yearOfPublicationOfBook"/>
    <allValuesFrom rdf:resource="&m_ontology;Book"/>
        </Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
        <Restriction>
        <onProperty rdf:resource="&m_ontology;yearOfBirthOfAuthor"/>
        <allValuesFrom rdf:resource="&m_ontology;Author"/>
        </Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
        <Restriction>
            <onProperty rdf:resource="&m_ontology;yearOfFoundationOfPublisher"/>
            <allValuesFrom rdf:resource="&m_ontology;Publisher"/>
        </Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
        <Restriction>
    <onProperty rdf:resource="&m_ontology;yearOfPublicationOfBook"/>
    <someValuesFrom rdf:resource="&m_ontology;Book"/>
        </Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
        <Restriction>
<onProperty rdf:resource="&m_ontology;yearOfFoundationOfPublisher"/>
<someValuesFrom rdf:resource="&m_ontology;Publisher"/>
```

```
            </Restriction>
        </rdfs:subClassOf>
        <rdfs:subClassOf>
            <Restriction>
            <onProperty rdf:resource="&m_ontology;yearOfBirthOfAuthor"/>
            <someValuesFrom rdf:resource="&m_ontology;Author"/>
            </Restriction>
        </rdfs:subClassOf>
    </Class>

    <!--
    //
    // General axioms
    //
     -->

    <rdf:Description>
        <rdf:type rdf:resource="&owl;AllDisjointClasses"/>
        <members rdf:parseType="Collection">
            <rdf:Description rdf:about="&m_ontology;Author"/>
            <rdf:Description rdf:about="&m_ontology;Book"/>
            <rdf:Description rdf:about="&m_ontology;Country"/>
            <rdf:Description rdf:about="&m_ontology;Language"/>
            <rdf:Description rdf:about="&m_ontology;Publisher"/>
            <rdf:Description rdf:about="&m_ontology;Year"/>
        </members>
    </rdf:Description>
</rdf:RDF>
```

Anexo B

```
import os
import nltk, re, pprint
ebooksFolder = os.path.join('C:\textbackslash\textbackslash',
'Users', 'Felipe', 'Desktop', 'books_txt')
ebooksList = os.listdir(ebooksFolder)
for filename in ebooksList:
    fname, fextension = os.path.splitext(filename)
    if (fextension == '.txt'):
        newName = 'NER_' + filename
        file = open(ebooksFolder + '\\' + filename)
        rawFile = file.read()
        partToCopy = rawFile[:500]
        partToCopy = " ".join(partToCopy.split())
        segmentedSentences = nltk.sent_tokenize(partToCopy)
        tokenizedSentences = [nltk.word_tokenize(sent) for sent in segmentedSentences]
        posTaggedSentences = [nltk.pos_tag(sent) for sent in tokenizedSentences]
        nerResults = [nltk.ne_chunk(pts, binary=True) for pts in posTaggedSentences]
        pathToCopy = 'C:\\Users\\Felipe\\Desktop\\books_txt\\'
        nameToSave = os.path.join(pathToCopy, newName)
        newFile = open(nameToSave, 'w')
        newFile.seek(0)
        for nerResult in nerResults:
            newFile.write(str(nerResult))
        newFile.close()
```

Anexo C

```
<?php
    require_once("easyrdf-0.8.0/lib/EasyRdf.php");
    EasyRdf_Namespace::set('ont',
        'http://www.semanticweb.org/felipe/ontologies/m_ontology#');

    function searchAuthorAndBook($store, $author, $book) {
        $found = false;
        $sparqlQuery = "
            PREFIX ont: <http://www.semanticweb.org/felipe/ontologies/m_ontology#>
            SELECT ?authorName ?bookTitle
            WHERE {
                ?author ont:authorHasAuthorshipOfBook ?book .
                ?book ont:bookHasTitle ?bookTitle .
                ?author ont:authorHasName ?authorName .
                FILTER regex(?bookTitle, '$book', 'i') .
                FILTER regex(?authorName, '$author', 'i')
            }
            LIMIT 3
        ";

        $sparqlResult = '';

        if ($rows = $store->query($sparqlQuery, 'rows')) {
            foreach ($rows as $row) {
                $found = true;
                $sparqlResult .= '<li>' . $row['authorName'] .
                ' --- ' . $row['bookTitle'] . '</li>';
            }
```

```
        }

        echo '<ul>';
        echo $sparqlResult;
        echo '</ul>';
        return $found;
    }
?>
```

Anexo D

```
<?php
    require_once('query_rdf_file.php');
    include_once('arc2-master/ARC2.php');

    function getProcessedNER($possibleMetadatum) {
        $returnedMetadatum = str_replace(
            array(
                '/NNP',
                '/JJ',
            ),
            '',
            $possibleMetadatum
        );
        $returnedMetadatum = trim($returnedMetadatum);
        return $returnedMetadatum;
    }

    $config = array(
        'db_name' => 'my_triples',
        'db_user' => 'root',
        'db_pwd' => '',
        'store_name' => 'arc_tests',
        'max_errors' => 100,
    );

    $store = ARC2::getStore($config);

    if (! $store->isSetUp()) {
```

```
        $store->setUp();
    }

    $store->query('LOAD <http://localhost/tese/ontologia_modificada.rdf>');
    $filePath = '../gutenberg_ner/';
    $directoryIterator = new DirectoryIterator($filePath);

    foreach ($directoryIterator as $fileInfo) {
        $fileFullName = $filePath . $fileInfo->getFilename();

        if (! ($fileInfo->isDot() || is_dir($fileFullName))) {
            echo '<h3>' . $fileInfo->getFilename() . '</h3>';
            $fileContent = file_get_contents($fileFullName);
            preg_match_all('/\(NE(.*)\)/', $fileContent, $foundNERs);
            $possibleMetadata = array();

            foreach ($foundNERs[1] as $foundNER) {
                $possibleMetadata[] = getProcessedNER($foundNER);
            };

            $possibleMetadata = str_replace('/', ' ', $possibleMetadata);
            $possibleMetadata = array_unique($possibleMetadata);
            $possibleMetadata2 = $possibleMetadata;

            foreach ($possibleMetadata as $possibleMetadatum) {
                foreach ($possibleMetadata2 as $possibleMetadatum2) {
                    if ($possibleMetadatum !== $possibleMetadatum2) {
                        $result = searchAuthorAndBook(
$store,
                           $possibleMetadatum,
                           $possibleMetadatum2
                        );

                        if ($result) {
                            break 2;
                        }
                    }
                }
            }
        }
    }
?>
```