UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

LEANDRO ZULIAN GALLINA

Extração e Representação Semântica de Fatos Temporais

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

Prof[a]. Dr. Renata Galante
Orientadora

Porto Alegre, maio de 2012

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Gallina, Leandro Zulian

Extração e Representação Semântica de Fatos Temporais / Leandro Zulian Gallina. – Porto Alegre: PPGC da UFRGS, 2012.

96 f.: il.

Dissertação (mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2012. Orientadora: Renata Galante.

1. Expressões temporais. 2. Recuperação de informação. 3. Gramática formal. 4. Ontologias. I. Galante, Renata. II. Título.

“O sábado é uma ilusão.”
— NELSON RODRIGUES

AGRADECIMENTOS

Agradeço aos professores do Instituto de Informática da Universidade Federal do Rio Grande do Sul pelo trabalho realizado e pela educação gratuita e de qualidade que proporcionaram a mim e a milhares de outros estudantes. Agradeço especialmente a Renata Galante pela orientação fundamental no desenvolvimento deste trabalho. Seus conselhos e revisões foram essenciais para a qualidade desta dissertação.

Agradeço à minha família e à minha namorada pelo apoio e compreensão durante o tempo dispendido para a realização deste trabalho. Agradeço aos meus amigos pela compreensão por todos aqueles eventos não frequentados para que este trabalho pudesse ser concluído. Agradeço aos meus colegas do Instituto de Informática pela troca de ideias.

Agradeço a todos que, de alguma forma, colaboraram com a conclusão deste curso e a realização deste sonho.

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS

OWL	Web Ontology Language
RDF	Resource Description Framework
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
W3C	World Wide Web Consortium
SPARQL	SPARQL Protocol and RDF Query Language
LOD	Linking Open Data
SQL	Structured Query Language
SVM	Support Vector Machines
kNN	k-Nearest-Neighbors
EXTIO	Extraction of Temporal Information Using Ontologies
XML	Extensible Markup Language
N3	Notation3
RDFa	Resource Description Framework in attributes
CSV	Comma-Separated Values

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

Este trabalho descreve EXTIO (*Extraction of Temporal Information Using Ontologies*), uma abordagem que permite a normalização de expressões temporais e a organização em ontologia de fatos temporais extraídos de texto em linguagem natural. Isto permite que motores de busca possam aproveitar melhor a informação temporal de páginas da Web, realizando inferências sobre fatos temporais. EXTIO propõe: a normalização de expressões temporais relativas através de uma gramática formal para a língua inglesa; e a organização de fatos temporais extraídos do texto normalizado em uma ontologia. Expressões temporais relativas são construções textuais de tempo que se referem a uma data absoluta cujo valor é relativo a outra data. Por exemplo, a expressão "three months ago" (três meses atrás) é uma expressão temporal relativa, pois seu surgimento no texto se refere a uma data três meses antes da data de publicação do documento. Experimentos demonstram que a gramática formal proposta para a normalização de expressões temporais relativas supera o *baseline* na eficácia da normalização e no tempo de processamento de documentos em linguagem natural. A principal contribuição deste trabalho é a gramática formal para normalização de expressões temporais relativas de texto na língua inglesa. Também é contribuição deste trabalho o processamento semântico da informação temporal disponível em formato texto em documentos, para que possa ser melhor aproveitada por motores de busca.

Palavras-chave: Expressões temporais, Recuperação de informação, Gramática formal, Ontologias.

EXTIO – Extraction of Temporal Information Using Ontologies

ABSTRACT

This work describes EXTIO, an approach for the normalization of temporal expressions and the semantic organization of temporal facts extracted from natural language text. This approach allows search engines to benefit from temporal information in Web pages, performing inferences on temporal facts. EXTIO proposes: the normalization of relative temporal expressions through a formal grammar for the English language; and the organization of temporal facts extracted from normalized text in an ontology. Relative temporal expressions are textual time structures that refer to an absolute date whose value is relative to another date. For instance, "three months ago" is a relative temporal expression because its appearance in the text refers to a date three months before the document publication date. Experiments show that the proposed formal grammar for the normalization of relative temporal expressions has a better performance than the baseline in effectiveness and processing time. The main contribution of this work is the formal grammar for the normalization of temporal expressions in natural language text in English. Another contribution of this work is the semantic processing of temporal information available in documents, so that search engines may benefit from this information.

Keywords: temporal expressions, information retrieval, formal grammars, ontologies.

1 INTRODUÇÃO

Os motores de busca atualmente são focados na informação sintática disponível nas páginas da Internet. Motores de busca comerciais como Google e Yahoo! têm fornecido resultados cada vez melhores para o usuário final, mas ainda deixam a desejar no que se refere ao processamento de informações temporais. Essas ferramentas utilizam apenas uma pequena parte da informação temporal da Web, especialmente a data de coleta do documento (JIN et al., 2008). No entanto, existe uma grande quantidade de informação semântica disponível na Internet que pode ser inferida a partir de texto em linguagem natural.

Uma fonte de informação temporal importante e pouco explorada em documentos de texto reside nas expressões temporais relativas. A Figura 1.1 exibe um documento extraído da Internet[1] com o título "2005 State of the Union Address". Em meio ao seu conteúdo, a página possui a seguinte frase:

"And in the last year alone, the United States has added 2.3 million new jobs."

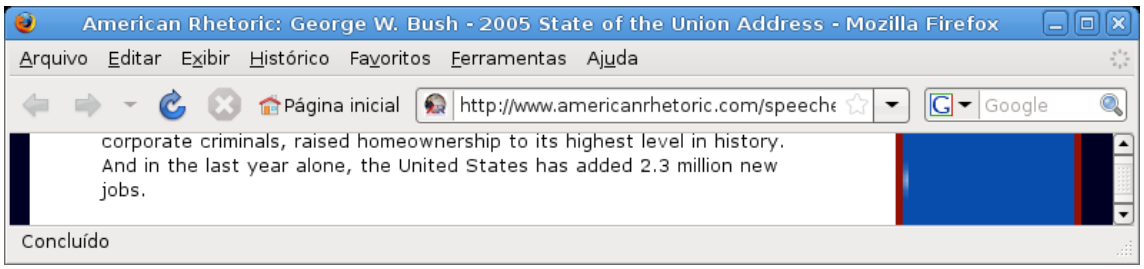


Figura 1.1: Página da Web com expressões temporais relativas

Neste trecho, é encontrada a expressão "last year" (ano passado). Esta expressão refere-se a um momento de tempo que ocorreu no passado. Para determinar o exato momento de tempo a que esta expressão se refere, é necessário saber a data de publicação do documento. Assim, será possível computar a data absoluta correspondente a esta expressão. Por se tratar de uma expressão relativa à data do documento, esta construção é chamada de *expressão temporal relativa*. Em relação à data de publicação do documento, as expressões temporais relativas podem se referir a momentos no passado, no futuro ou no presente, por exemplo "three years ago" (três anos atrás), "next year" (ano que vem) e "today" (hoje), respectivamente. No exemplo acima, dado que o documento foi publicado no ano de 2005, a expressão "last year" é normalizada para a data absoluta de 2004.

[1] http://www.americanrhetoric.com/speeches/stateoftheunion2005.htm

Uma vez realizada esta etapa de normalização, um motor de busca à procura de números de emprego em 2004 nos Estados Unidos poderá encontrar esta página.

Outro tipo de informação temporal disponível e pouco explorado em documentos está nos fatos temporais e períodos de tempo. Para exemplificar, a Figura 1.2 ilustra um documento extraído da Internet[2]. Esta página traz a informação de que Bill Clinton foi presidente dos Estados Unidos de 1993 a 2001. Um ser humano, após ler esta página, será capaz de inferir que, em 1995, o presidente dos Estados Unidos era Bill Clinton. No entanto, se submetermos a pesquisa "United States president 1995" (presidente dos Estados Unidos em 1995) a um motor de busca convencional, ele terá dificuldades em responder corretamente à pesquisa solicitada, por estar centrado apenas na informação sintática disponível nos documentos. O motor de busca poderá, entre outras estratégias, buscar pelo termo "1995" juntamente com os termos "United States president" em um único documento. Caso não exista um documento com esses três termos, ele não será considerado pela pesquisa. Adicionalmente, outros documentos poderão até mesmo confundir o motor de busca atribuindo grande relevância a resultados menos relevantes. Este exemplo mostra a importância de efetivamente interpretar a informação semântica do documento: um mecanismo que extrair o fato de que Bill Clinton foi presidente americano de 1993 a 2001 será capaz também de inferir que, em 1995, o presidente americano era Bill Clinton.

Figura 1.2: Página da Web com períodos de tempo

O objetivo deste trabalho de mestrado é definir uma abordagem que permita a organização semântica de fatos temporais extraídos de texto. Esta abordagem, denominada ***EXTIO*** *(Extraction of Temporal Information Using Ontologies)*, é voltada para o processamento de documentos em linguagem natural na língua inglesa, e está dividida nas seguintes etapas:

1. O texto é submetido a uma etapa de normalização de expressões temporais, em que todas as ocorrências de data são normalizadas para uma representação padronizada. A abordagem EXTIO suporta tanto datas absolutas, como "March 31, 2011" (31 de março de 2011), quanto tempos relativos, como "next Tuesday" (próxima terça-feira). As expressões relativas são normalizadas com base na direção e magnitude em relação a um tempo de referência, usualmente a data de publicação do documento. Para realizar esta tarefa, este trabalho emprega uma gramática formal para identificação e normalização de expressões temporais relativas. Até onde é de conhecimento do autor deste trabalho, esta é a primeira vez que uma gramática formal é proposta para a normalização de expressões temporais em inglês. Esta etapa do trabalho insere esta dissertação no contexto de bancos de dados temporais e recuperação de informações.

[2]http://www.britannica.com/EBchecked/topic/121813/Bill-Clinton

2. Após ser realizada a normalização de expressões temporais, o texto passa por uma etapa de extração de fatos temporais. Fatos temporais relacionam objetos (por exemplo uma pessoa, cidade ou obra musical) a datas através de uma propriedade temporal. Este trabalho define um conjunto de propriedades temporais e utiliza uma ferramenta de extração de informações para buscar no texto novos pares relacionados por estas propriedades temporais. Esta etapa do trabalho insere esta dissertação no contexto de extração de informações.

3. Uma vez extraídos os fatos temporais, eles são utilizados para popular uma ontologia. Desta forma, as tecnologias da Web Semântica podem ser aplicadas aos fatos temporais extraídos, permitindo a realização de consultas sofisticadas, conexão a fontes de dados semânticos e criação de novas aplicações e *mashups*. Para este trabalho foi escolhida uma ontologia existente, o que permite o reuso de uma estrutura à qual são agregadas novas informações. Assim, a base de dados é estendida, e é possível realizar inferências temporais sobre os fatos extraídos, envolvendo também os dados existentes *a priori*. Esta etapa do trabalho insere esta dissertação no contexto de ontologias e Web Semântica.

De forma geral, a contribuição deste trabalho é o processamento adequado da informação temporal disponível em documentos de texto em linguagem natural, para que possa ser melhor aproveitada por motores de busca. Especificamente, a principal contribuição deste trabalho é uma gramática formal para a normalização de expressões temporais relativas, método inédito para esta tarefa. O principal foco de aplicação da abordagem **EXTIO** é em documentos de texto em linguagem natural com qualquer tipo de expressão temporal, como notícias e biografias.

Este trabalho está estruturado da seguinte forma. O Capítulo 2 apresenta conceitos fundamentais para a compreensão do trabalho. No Capítulo 3, é apresentada uma revisão bibliográfica acerca dos trabalhos relacionados a esta dissertação. O Capítulo 4 apresenta o processo de determinação das propriedades temporais utilizadas para os fatos temporais extraídos. O Capítulo 5 apresenta a abordagem proposta neste trabalho (EXTIO). No Capítulo 6, a implementação da abordagem é detalhada. No Capítulo 7, são apresentados os resultados dos experimentos realizados. O Capítulo 8 expõe as conclusões e as atividades futuras. O Apêndice A exibe a listagem completa da gramática formal para normalização de expressões temporais relativas, com o analisador léxico listado no Apêndice B. Finalmente, o Apêndice C lista as propriedades temporais utilizadas neste trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

Para a correta compreensão e embasamento deste trabalho, este capítulo apresenta os principais conceitos sobre informação temporal, gramáticas formais e ontologias. Adicionalmente, também são dadas definições sobre as tecnologias utilizadas para atingir os objetivos deste trabalho.

2.1 Conceitos de Tempo e Bancos de Dados Temporais

Nesta seção são apresentados os principais conceitos sobre tempo e bancos de dados temporais, que auxiliam na compreensão deste trabalho. Estes conceitos foram retirados de (DYRESON et al., 1994) e (EDELWEISS, 1998).

Um modelo de dados temporal pode representar o tempo de duas formas. Uma delas é representar o tempo como no mundo real, de forma *contínua*. Entre dois instantes de tempo, sempre existe no mínimo um outro instante de tempo, estabelecendo um isomorfismo com o conjunto dos números reais. Outra maneira de representar o tempo é de forma *discreta*. Nesse caso, há uma linha do tempo formada por uma sequência de intervalos temporais consecutivos, indivisíveis e de mesma duração, chamados de *chronons*.

A duração de um *chronon* equivale à granularidade de uma aplicação. Assim, em uma aplicação cuja granularidade é diária, o *chronon* pode equivaler a um dia. O *chronon* de uma aplicação é único, mas é possível realizar implementações para que diferentes granularidades sejam adotadas dentro de uma aplicação. Por exemplo, em uma aplicação de granularidade diária, é possível afirmar que Abraham Lincoln nasceu no dia 12 de fevereiro de 1809 e que Aristóteles faleceu no ano de 322 a.C. (sem especificar dia e mês exatos, que são desconhecidos).

Quanto aos elementos primitivos de representação temporal, existem quatro tipos principais: (i) um *instante* consiste de um ponto na linha do tempo, podendo ser um ponto de duração infinitesimal (no caso da representação contínua do tempo) ou de duração igual a um *chronon* (no caso da representação discreta); (ii) um *intervalo temporal* é o tempo entre dois instantes, e, no caso da representação discreta, é constituído por um conjunto de *chronons* contíguos; (iii) um *elemento temporal* é a união finita de diversos intervalos temporais, e, no caso da representação discreta, equivale a um conjunto de *chronons* não necessariamente contíguos; e (iv) a *duração temporal* é uma porção de tempo com comprimento conhecido, mas sem instantes específicos de início e fim, e pode ser fixa (como uma semana, que possui sempre 7 dias) ou variável (como um mês, que possui 28, 29, 30 ou 31 dias).

O *tempo de validade* de um fato compreende o tempo durante o qual o fato é verdadeiro em uma realidade modelada. Um fato pode estar associado a qualquer número de instantes ou intervalos de tempos. Este conceito não deve ser confundido com o *tempo*

de transação, que denota, em um banco de dados temporal, o tempo no qual o fato é registrado no banco de dados.

O tempo pode ser absoluto ou relativo. Tempo *absoluto* indica que um determinado tempo de validade, com determinada granularidade, está associado a um fato. Tal informação não depende do tempo de validade de outro fato ou do instante atual (*now*). Um exemplo é: "Albert Einstein faleceu em 18 de abril de 1955". Já o tempo *relativo* indica que o tempo de validade de um fato está relacionado ou ao instante atual (*now*) ou ao tempo de validade de outro fato. Esta relação de um tempo com outro pode ser qualitativa (antes, depois) ou quantitativa (uma semana antes, 4 anos depois). Exemplos incluem: "João iniciou o Mestrado no ano passado" e "A Copa do Mundo ocorrerá no Brasil 4 anos depois da Copa na África do Sul".

Quanto à representação primitiva do tempo, um sistema pode representar o tempo como um ponto ou como um intervalo. O trabalho de (DOWTY, 1979) atribui a representação primitiva de tempo como um ponto em uma linha temporal. Na representação primitiva, um intervalo pode ser especificado através de seu ponto de início e seu ponto de fim. O trabalho de (ALLEN, 1983) atribui a representação primitiva de tempo a um intervalo. Esse mesmo trabalho especifica 13 relações possíveis entre dois intervalos de tempo. Estas relações são estabelecidas com base na ordenação dos momentos de início e fim destes intervalos.

Uma vez estabelecido um modelo de dados temporal, é necessário submeter consultas temporais a ele. Uma consulta temporal apresenta dois componentes: um componente de seleção e um componente de saída. O *componente de seleção* é o componente responsável por filtrar os dados. Geralmente utiliza condições lógicas, que podem envolver valores temporais ou não. Conforme o componente de seleção, as consultas podem ser classificadas em três tipos: as consultas de seleção sobre dados, que estabelecem condições lógicas somente sobre valores de dados e não valores temporais; as consultas de seleção temporal, que estabelecem condições lógicas somente sobre valores temporais (tempo de validade ou tempo de transação); e as consultas de seleção mista. O *componente de saída* é o componente responsável pela projeção dos dados. Analogamente, as consultas podem ser classificadas em três tipos: as consultas de saída de dados, as consultas de saída temporal e as consultas de saída mista.

2.2 Gramáticas Formais

Esta seção apresenta uma fundamentação teórica a respeito de gramáticas formais. A principal contribuição desta dissertação é uma gramática formal para a normalização de expressões temporais relativas. Assim, ficam aqui apresentados conceitos que serão úteis para a compreensão desta importante parte do trabalho apresentado.

Uma gramática livre de contexto (AHO et al., 2006) é uma notação usada para especificar a sintaxe de uma linguagem. Neste trabalho, o termo gramática livre de contexto é citado nas formas abreviadas de gramática ou gramática formal, com o mesmo significado.

A linguagem gerada por uma gramática formal é chamada de *linguagem livre de contexto*. Este tipo de linguagem dá suporte a elementos típicos de linguagens de programação, como parênteses balanceados, construções estruturadas em blocos, entre outras. Por isso, as gramáticas livres de contexto são usadas para descrever linguagens de programação como Java, C e Pascal (MENEZES, 1998).

O seguinte exemplo, retirado de (AHO et al., 2006), mostra como uma gramática é

usada para descrever elementos de uma linguagem de programação. Seja um bloco *if-else* de Java, que possui a forma:

if (expressão) instrução **else** instrução

Isto significa que o bloco *if-else* é formado pela palavra-chave **if**, um parênteses de início, uma expressão, um parênteses de fim, uma instrução, a palavra-chave **else** e outra instrução. Ao utilizar a variável *expr* para denotar uma expressão e a variável *instr* para denotar uma instrução, esta regra pode ser expressa da seguinte forma:

instr $\rightarrow$ if (*expr*) *instr* else *instr*

Esta regra recebe o nome de *regra de produção*. A seta significa que o elemento à sua esquerda deriva os elementos à sua direita. As regras de produção definem as condições de geração das sequências de caracteres da linguagem. Se uma gramática contém diversas regras de produção com o mesmo componente do lado esquerdo, estas regras podem ser abreviadas para simplificar a sua representação. Assim, a sequência de regras $\alpha \rightarrow \beta_1$, $\alpha \rightarrow \beta_2$, $\ldots \alpha \rightarrow \beta_n$ pode ser abreviada como:

$$\alpha \rightarrow \beta_1 \ \mid \ \beta_2 \ \mid \ \ldots \ \mid \ \beta_n$$

Os elementos léxicos, como a palavra-chave **if** e os parênteses, são chamados *terminais*. As variáveis como *expr* e *instr* representam sequências de terminais e são chamados *não-terminais*.

Formalmente, uma gramática livre de contexto é definida através de quatro componentes:

1. Um conjunto finito de símbolos terminais.
2. Um conjunto finito de símbolos não-terminais.
3. Um conjunto finito de regras de produção. Cada regra de produção consiste de um símbolo não-terminal, uma seta e uma sequência de símbolos terminais e/ou não-terminais.
4. Um elemento do conjunto de símbolos não-terminais escolhido como variável inicial.

Neste texto, uma *palavra* ou *sequência de caracteres* possui o mesmo significado, indicando uma sequência finita de símbolos concatenados.

2.2.1 Derivação

Uma gramática deriva uma palavra iniciando pela variável inicial e repetidamente substituindo um símbolo não-terminal através da aplicação de regras de produção. Em cada aplicação de regra de produção, um símbolo não-terminal que aparece do lado esquerdo da regra de produção é substituído pelos símbolos que aparecem do lado direito daquela regra de produção. Estas regras são aplicadas sucessivamente até que sejam geradas palavras terminais. O conjunto de todas as palavras terminais derivadas a partir da variável inicial forma a linguagem definida pela gramática.

O seguinte exemplo demonstra uma gramática para geração de expressões aritméticas simples. As regras de produção são exibidas na Tabela 2.1.

Tabela 2.1: Gramática para expressões aritméticas

$expr \rightarrow expr + numero$
$expr \rightarrow expr - numero$
$expr \rightarrow numero$
$numero \rightarrow 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$

Considere-se que a variável inicial é *expr*. Neste exemplo, a expressão $5 + 4 - 7$ pode ser gerada através da seguinte sequência de aplicações de regras de produção:

$$
\begin{aligned}
& expr \\
\Rightarrow\ & expr - numero \\
\Rightarrow\ & expr + numero - numero \\
\Rightarrow\ & numero + numero - numero \\
\Rightarrow\ & 5 + 4 - 7
\end{aligned}
$$

Uma regra de produção pode conter, em seu lado direito, uma palavra vazia. A palavra vazia é útil para derivar um conjunto de símbolos que é vazio a partir de um símbolo não-terminal. Palavras vazias podem ser usadas para indicar, por exemplo, atributos opcionais dentro de uma linguagem de programação. O símbolo usado para denotar a palavra vazia é ϵ.

Realizar o *parsing* de uma palavra consiste em tentar derivar esta palavra a partir da variável inicial da gramática. Caso a derivação da palavra não seja possível, a gramática deve indicar que a palavra possui um erro de sintaxe. O problema de *parsing* está entre os mais importantes dentro da área de estudo de compiladores. Um programa escrito em uma linguagem de programação que possua erros de sintaxe não terá sucesso na sua etapa de compilação.

2.2.2 Tradução dirigida por sintaxe

A tradução dirigida por sintaxe é realizada adicionando regras ou fragmentos de programa a produções em uma gramática. Usualmente, esta tradução é realizada de forma incremental, através da execução de fragmentos de programa. Quando esta abordagem é utilizada, é estabelecido um esquema de tradução dirigida por sintaxe.

Os fragmentos de programa listados junto ao lado direito das regras de produção recebem o nome de *ações semânticas*. As ações semânticas aparecem nas regras de produção delimitadas por chaves. Uma ação semântica é executada no momento em que ela aparece no meio da regra de produção.

O exemplo da Tabela 2.1 poderia ser modificado para incluir ações semânticas de forma a imprimir os números envolvidos nas expressões aritméticas. Isto é obtido alterando as regras de produção para a variável *numero* conforme é exibido na Tabela 2.2.

Tabela 2.2: Gramática para expressões aritméticas com ações semânticas

$numero \rightarrow 0$	`{print('0')}`
$numero \rightarrow 1$	`{print('1')}`
$numero \rightarrow 2$	`{print('2')}`
$\vdots$	
$numero \rightarrow 9$	`{print('9')}`

A tradução dirigida por sintaxe é capaz de associar ações às regras de produção, mas

não é suficiente para realizar efetivamente os cálculos entre os números envolvidos. Para isto, é necessário associar valores aos símbolos terminais da gramática, algo que é possível através da análise léxica.

2.2.3 Análise Léxica

Em um compilador, o analisador léxico é responsável por ler caracteres da entrada e agrupá-los em *tokens*. Um *token* possui um símbolo terminal – usado para as decisões das etapas de *parsing* – e informação adicional na forma de atributos. A sequência de caracteres de entrada que compõe um *token* é chamada de lexema. Portanto, o *token* é formado por um lexema e por seus atributos.

O analisador léxico é utilizado para permitir que uma gramática represente números, por exemplo. Na Tabela 2.2, a gramática permite o reconhecimento dos símbolos terminais 0, 1, 2 ... 9 como *numero*, e imprime seus valores ao encontrá-los. No entanto, o valor efetivo do *numero* não foi armazenado. Para a implementação de um compilador, o analisador léxico utilizaria os valores dos símbolos terminais como atributos de *numero*. O exemplo foi modificado conforme consta na Tabela 2.3. O atributo *valor* é utilizado para armazenar a representação em números inteiros equivalente ao número encontrado.

Tabela 2.3: Análise léxica para gramática de operações aritméticas simples

$numero \rightarrow 0$	`{numero.valor = 0}`
$numero \rightarrow 1$	`{numero.valor = 1}`
$numero \rightarrow 2$	`{numero.valor = 2}`
$\vdots$	
$numero \rightarrow 9$	`{numero.valor = 9}`

A combinação da tradução dirigida por sintaxe com a análise léxica permite a implementação de um compilador completo. Ao encontrar, por exemplo, o número 9 no meio das expressões da gramática, o analisador léxico deve ser capaz de atribuir o valor equivalente ao *token* gerado. Assim, este valor pode ser usado em computações posteriores.

2.3 Web Semântica e Ontologias

Um dos objetivos desta dissertação é a organização semântica de fatos temporais extraídos em ontologias. Esta seção apresenta os principais conceitos envolvendo ontologias e Web Semântica, para a correta compreensão deste trabalho.

A World Wide Web (WWW, ou simplesmente Web) é um sistema de documentos de hipertexto interligados através da Internet. Surgiu nos anos 1990 a partir da concepção de Tim Berners-Lee (BERNERS-LEE; CAILLIAU, 1990) como um sistema onde usuários poderiam visualizar conteúdo através de navegadores em uma arquitetura cliente-servidor.

O sucesso da Web foi alcançado em grande parte devido aos padrões propostos por Berners-Lee e outros pioneiros da Internet (BERNERS-LEE; FISCHETTI, 1999), como:

- um sistema globalmente único de identificação de recursos na Web, que deu origem aos atuais URI (*Uniform Resource Identifier*) e URL (*Uniform Resource Locator*);
- uma linguagem para os documentos em hipertexto, o HTML (*HyperText Markup Language*); e

- um protocolo para transferência de hipertexto, o HTTP (*Hypertext Transfer Protocol*).

Os esforços de padronização da Web deram origem ao W3C (*World Wide Web Consortium*)[1], fundado para desenvolver e manter esses padrões.

Para visualizar um documento (ou página) na Web, o usuário precisa digitar a sua URL no navegador ou clicar um *link* de alguma outra página para aquele documento. Este foi um dos primeiros problemas que surgiu na Web: por ser orientada a documentos em vez de orientada a conteúdo, ela não permitia ao usuário encontrar facilmente páginas de seu interesse. Para resolver esta demanda, surgiram os motores de busca, que permitem ao usuário encontrar o conteúdo desejado mesmo sem saber a URL da página procurada. Esta demanda deu origem a motores de busca de grande sucesso comercial, como Google, Yahoo! e Altavista.

A Web atual possui determinadas limitações (BERNERS-LEE; HENDLER; LASSILA, 2001). Ela se baseia em sua maior parte em documentos escritos em HTML, linguagem que dá maior ênfase a texto, formulários e conteúdo multimídia, como imagens e vídeos. A Web popularizou-se ao exibir este tipo de conteúdo de forma amigável ao usuário final. No entanto, este mesmo conteúdo, em grande parte, não está estruturado de forma a ser interpretado por máquinas. O HTML fornece a maneira de representar o conteúdo visualmente, através de marcações que representam texto em negrito, itálico, diferentes fontes, diferentes tamanhos de texto etc. Entretanto, o HTML não se especializa na descrição do conteúdo das marcações, limitando a habilidade que um programa de computador teria para reconhecer em seu conteúdo a descrição de um livro, de uma pessoa, de uma cidade, de um evento etc. Sem marcações semânticas, a interpretação do que é apresentado fica totalmente a cargo do usuário que visualiza a página.

Motores de busca como os mencionados anteriormente realizam buscas sintáticas sobre o conteúdo da Web. Isto é suficiente para a realização de vários tipos de consulta, como o exemplo de um usuário em busca de informações sobre a cidade de Brasília. Se ele digitar "Brasília" no motor de busca, muito provavelmente encontrará conteúdo de interesse sobre essa cidade. Isto porque uma busca sintática é baseada no texto e nas palavras escritas, realizando buscas pelos termos digitados (ou termos semelhantes). No entanto, nem todas as buscas podem ser realizadas de forma sintática. É possível, por exemplo, que o usuário esteja buscando por "cidades sul americanas com mais de 2 milhões de habitantes". Digitar esta frase em um motor de busca somente daria resultado se existisse uma página específica com estas exatas palavras. Se os dados da Web estivessem dispostos de alguma maneira estruturada e padronizada, seria possível realizar esta busca e retornar uma lista específica de cidades correspondentes à consulta. Para endereçar as dificuldades encontradas pela Web, surgiu a *Web Semântica*, com o objetivo de estender a atual rede de documentos dando significado ao seu conteúdo. O termo Web Semântica foi definido por Berners-Lee como "uma rede de dados que pode ser processada direta e indiretamente por máquinas" (BERNERS-LEE; HENDLER; LASSILA, 2001).

Dado o enorme sucesso alcançado pela Web tradicional, a Web Semântica foi proposta por Berners-Lee como sua extensão, de forma a aproveitar a sua infraestrutura e alguns dos serviços já oferecidos. A Web Semântica emprega, por exemplo, as tecnologias HTTP e URI da mesma forma que a Web já utiliza. Adicionalmente, novos padrões foram introduzidos para apoiá-la. Berners-Lee definiu a arquitetura da Web Semântica em três camadas (BERNERS-LEE, 1998):

[1] http://www.w3.org/Consortium/

1. **Camada de metadados:** é responsável pela descrição de conceitos. O padrão RDF (*Resource Description Framework*), recomendado pela W3C, fornece uma maneira flexível de descrever conceitos como pessoas, locais ou abstrações (HEATH; BIZER, 2011). Além disso, é possível estabelecer *links* em RDF que representam as relações entre conceitos. Estas relações são tipificadas. Um modelo RDF pode ser serializado para diversos formatos, como RDF/XML, N3, Turtle e RDFa.

2. **Camada de esquema:** define a descrição hierárquica de conceitos – permitindo relações do tipo "é-um" entre diferentes classes – e propriedades, através da introdução de *ontologias*. Uma ontologia é definida como a especificação formal e explícita de uma conceitualização compartilhada (GRUBER, 1993). Como resultado, é obtido um vocabulário compartilhado, útil para a padronização da representação de conhecimento. *RDF Schema* (BRICKLEY; GUHA, 2004) é o padrão recomendado pela W3C para esta camada.

3. **Camada lógica:** define linguagens de ontologias mais poderosas, que permitem a realização de inferências. Uma inferência é realizada quando um conjunto de axiomas permite a descoberta de informação adicional baseada nos dados fornecidos explicitamente. Esta camada é baseada em linguagens formais de descrição do conhecimento chamadas *Description Logics* (HORROCKS; SATTLER; TOBIES, 1999). A recomendação da W3C para esta camada é a linguagem OWL (*Web Ontology Language*), criada para utilização na Web (MCGUINNESS; HARMELEN, 2004). Todos os elementos desta linguagem (classes, propriedades e indivíduos) são definidos como recursos RDF e identificados por URIs. Esta linguagem inclui o padrão *RDF Schema*. Assim, as ontologias também podem integrar a rede global de dados, provendo um padrão para o compartilhamento de vocabulários entre aplicações.

Um exemplo de informação representada na camada de metadados seria um recurso em RDF para a cidade de Porto Alegre, um recurso em RDF para o Estado do Rio Grande do Sul e um recurso em RDF para o país Brasil. Um exemplo de informação representada pela camada de esquema seria uma ligação entre os dois recursos, registrando que a cidade de Porto Alegre "está localizada no" Rio Grande do Sul. Representada como parte de uma ontologia, esta propriedade se torna padronizada, de forma que outros pares de recursos poderão ser representados da mesma maneira. Assim, esta propriedade também pode representar que o Rio Grande do Sul "está localizado no" Brasil. Já as regras de inferência da camada lógica podem estabelecer a seguinte relação: se Porto Alegre "está localizada no" Rio Grande do Sul e o Rio Grande do Sul "está localizado no" Brasil, então deduz-se que Porto Alegre "está localizada no" Brasil através do mecanismo de inferências.

O exemplo acima evidencia uma das grandes vantagens da Web Semântica sobre a Web tradicional: o processamento automatizado de marcações através do compartilhamento de um vocabulário, mesmo que apenas de forma parcial. No exemplo, mesmo que nem todos os recursos geográficos sejam representados através da mesma ontologia, é possível realizar inferências desde que a propriedade em questão ("está localizada no") seja a mesma em diferentes cenários.

A interação entre aplicações e provedores de dados semânticos na Web pode se dar através de navegadores, ou através de aplicações que enfocam a Web como um banco de dados ao qual pode-se realizar consultas. Com esse intuito, foi criada a SPARQL – *SPARQL Protocol and RDF Query Language* (SEGARAN et al., 2009) –, uma linguagem semelhante ao SQL, com a diferença de visar consultas sobre fontes de dados em

RDF na Web. Através de consultas SPARQL é possível expressar buscas mais complexas tais como “cidades sul americanas com mais de 2 milhões de habitantes”. SPARQL é atualmente a recomendação da W3C para recuperação e manipulação de dados em RDF. A linguagem SPARQL permite a realização de uma *consulta federada*. Neste cenário, uma mesma consulta é submetida a diversos *endpoints* – serviços que recebem consultas SPARQL e em troca fornecem resultados – onde a consulta é computada e os resultados são consolidados.

2.3.1 Linked Data (Dados Ligados)

Tecnologias como RDF e SPARQL permitem a publicação e busca de grandes quantidades de dados distribuídos pela Web. No entanto, para que estas tecnologias agreguem valor, é necessário que efetivamente volumes significativos de dados sejam publicados de forma compatível a elas. Com isto em mente, o trabalho de (BERNERS-LEE, 2006) propôs o termo *Linked Data* (Dados Ligados) como um conjunto de melhores práticas para publicar e interligar dados estruturados na Web. Estas melhores práticas, também conhecidas como Princípios de *Linked Data*, são as seguintes:

1. Usar URIs para dar nome a conceitos.
2. Usar URIs baseadas em HTTP, para que estes nomes possam ser consultados pela Internet.
3. Responder a uma consulta por uma URI com informações úteis, usando os padrões recomendados pela W3C.
4. Incluir *links* para outras URIs, permitindo a descoberta de itens relacionados.

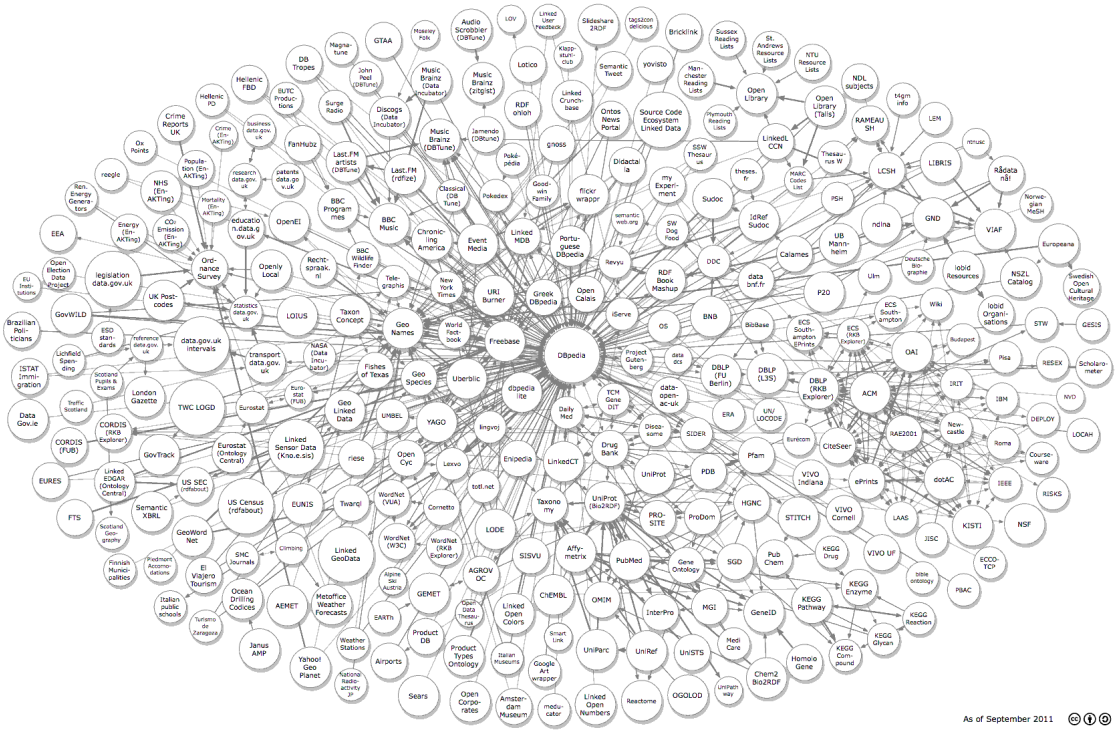


Figura 2.1: Situação da *LOD cloud* em setembro de 2011

A aderência a estes princípios pode ser parcial, o que significa que um determinado conjunto de dados pode seguir alguns princípios e não outros. Isto gerou um sistema de classificação de cinco estrelas, onde o conjunto de dados recebe uma nota de acordo com os seguintes critérios:

- **1 estrela:** dados disponibilizados na Web em qualquer formato, com uma licença aberta;
- **2 estrelas:** dados disponíveis em formato estruturado compreensível por máquinas (por exemplo, um arquivo no formato Microsoft Excel em vez de uma imagem de uma tabela);
- **3 estrelas:** dados disponíveis em um formato não proprietário (por exemplo, em CSV em vez de Excel);
- **4 estrelas:** dados disponíveis como no caso de 3 estrelas, mais o uso de padrões da W3C (RDF e SPARQL) para identificar conceitos, de maneira que as pessoas possam criar *links* para elas;
- **5 estrelas:** dados disponíveis como no caso de 4 estrelas, mais o uso de *links* para os dados de outras fontes para providenciar contexto.

É importante notar que cada classificação pode ser obtida de forma gradual, representando uma transição suave para um estado em que os dados publicados por uma organização podem ser considerados *Linked Data*. A publicação crescente de fontes de dados compatíveis com *Linked Data*, ligadas umas às outras, deu origem à *LOD cloud* (nuvem de Dados Ligados). Esta nuvem é ilustrada pela Figura 2.1. Cada círculo representa uma fonte de dados, cujo tamanho na figura é proporcional ao tamanho da base de dados. As setas de um círculo para outro indicam que os dados da fonte representada pelo primeiro círculo possuem *links* para os dados da fonte representada pelo segundo círculo.

A imagem da Figura 2.1 foi atualizada pela última vez em setembro de 2011. Novas versões desta figura são publicadas regularmente na Internet[2]. Qualquer organização pode publicar dados e ligá-los à *LOD cloud*. O endereço do projeto traz instruções sobre como proceder para que os dados publicados sejam exibidos em versões futuras desta figura.

[2]O endereço do projeto na Internet é http://lod-cloud.net/

3 TRABALHOS RELACIONADOS

Este capítulo apresenta os trabalhos relacionados a esta dissertação. Para cada área de pesquisa relacionada – a saber, expressões temporais, extração de informações e ontologias – os trabalhos mais significativos são apresentados de forma comparativa. Quando aplicável, os trabalhos também são descritos em termos de sua utilidade para a abordagem proposta nesta dissertação.

Para clareza do texto, neste capítulo uma ferramenta ou software é denominada disponível livremente quando é possível ter acesso de forma gratuita ao seu código. Casos em que a ferramenta pode ser utilizada gratuitamente sem disponibilização de código são explicitamente indicados.

Este capítulo está estruturado da seguinte forma. A seção 3.1 apresenta ferramentas na área de reconhecimento de expressões temporais, que realizam a anotação ou a normalização de expressões. Na seção 3.2, são apresentadas as abordagens de extração de informações a partir de texto em linguagem natural. A seção 3.3 apresenta iniciativas de pesquisa com o objetivo de criar ontologias para representar conhecimento extraído de bases existentes. A seção 3.4 encerra o capítulo com as considerações finais.

3.1 Reconhecimento de Expressões Temporais

Esta seção apresenta abordagens existentes para reconhecimento de expressões temporais. Este estudo é pertinente, pois a abordagem proposta nesta dissertação inclui o tratamento de expressões temporais. Algumas destas abordagens realizam a anotação de expressões temporais, ou seja, indicam, através de um determinado formato, a localização delas em meio ao texto. Outras realizam a normalização de expressões temporais, substituindo sua presença no texto pelo valor da data absoluta a que se referem. Existem propostas que realizam tanto a anotação quanto a normalização. Ao final desta seção, é apresentado um quadro comparativo entre as abordagens analisadas.

TempEx (MANI; WILSON, 2000) é uma ferramenta para anotação e normalização de expressões temporais em documentos na língua inglesa. Suporta expressões temporais nas granularidades de dia, semana, mês, ano, década e século, além de hora, minuto e segundo. A normalização é feita em relação a um tempo de referência. As expressões temporais normalizadas por TempEx são representadas no formato TIMEX2 (FERRO et al., 2001). Este formato é baseado no padrão ISO-8601 (ISO, 2000) e no formato predecessor TIMEX (SETZER, 2001). Informações temporais são representadas através de pontos em uma linha do tempo. TIMEX2 segue o padrão ISO-8601 ao adotar uma formatação baseada em calendário, para representar as granularidades variando de século a segundo. TempEx também normaliza expressões temporais não suportadas pelo padrão ISO-8601, por exemplo estações do ano e períodos do dia como manhã, tarde e noite.

GUTime (VERHAGEN et al., 2005) é o sucessor de TempEx. Neste projeto, o formato TIMEX2, empregado por TempEx, é substituído pelo padrão TimeML TIMEX3 (PUSTEJOVSKY et al., 2004). TimeML é um padrão para representação de informação temporal baseado em XML. A *tag* do tipo TIMEX3 é utilizada para marcação de expressões temporais. Esta *tag* é baseada no TIMEX2 com a inclusão de algumas melhorias, como suporte a formatos europeus de data. Como o TimeML é baseado em XML, o resultado final do processamento de GUTime pode ser consultado por linguagens de manipulação de XML como XPath e XQuery (HUNTER et al., 2007). GUTime é parte do pacote TARSQI (VERHAGEN; PUSTEJOVSKY, 2008). Este pacote fornece implementação para o processamento das *tags* do padrão TimeML. Além da *tag* TIMEX3, que é de responsabilidade de GUTime, existem *tags* de TimeML que permitem marcação específica de eventos, sinais, conectores temporais e outros. O pacote TARSQI está disponível no endereço do projeto TARSQI[1].

O trabalho de (SAQUETE; MARTINEZ-BARCO, 2000) realiza a identificação e a normalização de expressões temporais em textos em espanhol. Para isto é utilizada uma gramática formal para reconhecimento das expressões temporais na língua espanhola. A normalização é realizada em relação à data de publicação do documento. Esse trabalho também normaliza correferências, isto é, expressões temporais relativas a datas nomeadas em meio ao texto. Exemplos de correferências tratadas por esse trabalho são as expressões "dos dias antes" (dois dias antes) e "la semana anterior" (a semana anterior).

O projeto GATE (CUNNINGHAM et al., 2011) provê uma arquitetura, um *framework* e um ambiente de desenvolvimento de módulos de processamento de texto em linguagem natural. Dentro deste ambiente de desenvolvimento, existe um conjunto de módulos disponibilizado livremente através do pacote Annie (CUNNINGHAM et al., 2002). Annie inclui ferramentas para separação de texto em frases, separação de texto em palavras, *POS tagger* (para determinação de funções morfossintáticas dentro do texto) e outros componentes. Este conjunto de módulos é capaz de extrair diferentes tipos de informação do texto, incluindo expressões temporais. Adicionalmente, também são extraídas informações a respeito de localizações, pessoas, organizações e outros tipos diversos. A extração é feita através do reconhecimento de entidades nomeadas. O resultado do processamento de texto pode ser armazenado em um banco de dados relacional, um objeto serializado na linguagem Java ou um XML em um formato próprio de Annie. Nenhum destes formatos segue um padrão específico, como o padrão ISO-8601 utilizado por GUTime. Annie realiza apenas a anotação de expressões temporais em inglês, sem normalizá-las. Adaptações para outros idiomas são feitas individualmente (MAYNARD; CUNNINGHAM, 2003).

PorTexTO (CRAVEIRO; MACEDO; MADEIRA, 2009) é um sistema de reconhecimento de entidades nomeadas temporais em texto na língua portuguesa. As expressões temporais são identificadas através de padrões de expressões. PorTexTO foi projetado visando baixo tempo de processamento, mesmo que não identifique todas as expressões temporais. Para isto, são utilizados estudos estatísticos para definir as expressões temporais com o maior número de ocorrências. Os padrões de expressões normalizados por este sistema são criados a partir de co-ocorrências existentes em referências temporais. PorTexTO realiza apenas a anotação de expressões em português, sem a normalização. Como resultado, marcações em XML, designando as expressões temporais, são inseridas no meio do texto.

A Tabela 3.1 exibe um resumo comparativo entre as ferramentas de reconhecimento de expressões temporais discutidas nesta seção. As ferramentas são comparadas através

[1] http://timeml.org/site/tarsqi/index.html

dos seguintes aspectos: “Anotação” indica se a ferramenta realiza a anotação das expressões temporais; “Normalização” denota se as expressões temporais encontradas pela ferramenta são normalizadas; “Saída” indica o formato gerado pela ferramenta; “Idioma” é a língua que os documentos de entrada devem possuir; e “Disponível” indica se a ferramenta está disponível livremente para utilização.

Tabela 3.1: Comparativo entre ferramentas de reconhecimento de expressões temporais

	GUTime	**(SAQUETE)**	**Annie**	**PorTexTO**
Anotação	Sim	Não	Sim	Sim
Normalização	Sim	Sim	Não	Não
Saída	TimeML	Textual	BD, Java ou XML	Textual e XML
Idioma	Inglês	Espanhol	Inglês	Português
Disponível	Sim	Não	Sim	Não

Esta seção apresentou trabalhos relacionados a esta dissertação, que realizam a tarefa de extração e normalização de expressões temporais. Conforme descrito nesta seção, não foi encontrado nenhum trabalho que realize a normalização de expressões temporais para a língua inglesa utilizando uma gramática formal. Isto torna inédito o método proposto neste trabalho, que especifica uma gramática para a normalização de documentos em inglês.

3.2 Extração de Informações

Esta seção apresenta algumas ferramentas existentes para extração de informações. Extração de informações é a tarefa de identificar instâncias de determinada classe de eventos ou relacionamentos em texto em linguagem natural, para em seguida extrair os argumentos relevantes do evento ou relacionamento em questão (GRISHMAN, 1997). Desta forma, a extração de informações também envolve a manutenção de uma representação estruturada, como um banco de dados, de informação obtida a partir do texto.

É importante ressaltar que esta seção não é exaustiva. Existe um grande número de iniciativas de extração de informações. Por premissas de espaço, apenas algumas são destacadas aqui. Estes projetos são voltados à extração de informações a partir de texto em linguagem natural na língua inglesa, que é o foco deste trabalho.

3.2.1 Leila

Esta subseção apresenta a ferramenta de extração de informações Leila (SUCHANEK; IFRIM; WEIKUM, 2006a), que extrai instâncias de relações binárias a partir de texto em linguagem natural. Esta ferramenta é apresentada de forma isolada nesta subseção por ser escolhida para a implementação do protótipo da abordagem proposta neste trabalho. Leila se baseia na análise de estruturas linguísticas de sentenças. Estas estruturas são submetidas a métodos de aprendizagem de máquina para a definição de modelos. A repetição de modelos dá origem a fatos extraídos do texto.

Para obter a estrutura linguística de uma sentença, é utilizada uma gramática de dependências (*Link Grammar Parser*) (SLEATOR; TEMPERLEY, 1993). Para cada sentença, a gramática de dependências gera uma estrutura linguística chamada ligação em cadeia (*linkage*). A ligação em cadeia é um grafo planar, conexo, não-direcionado, que conecta todas as palavras da frase original. As arestas do grafo são rotuladas com conectores, que

expressam a relação entre as palavras. A ligação em cadeia deve obedecer a certas restrições linguísticas, dadas pela gramática de dependências. Esta gramática especifica que palavras podem ser ligadas antes e depois de uma palavra através dos conectores. A gramática de dependências também especifica a classe gramatical de uma palavra, fazendo o papel de um *part-of-speech tagger*.

A Figura 3.1, retirada de (SUCHANEK; IFRIM; WEIKUM, 2006a), ilustra uma ligação em cadeia de uma frase. O conector *subj* rotula a aresta que liga o sujeito ao verbo da sentença. A identificação da classe gramatical (*part-of-speech*) identifica que "was" é um verbo (sufixo ".v") e "composers" e "time" são substantivos (sufixo ".n").

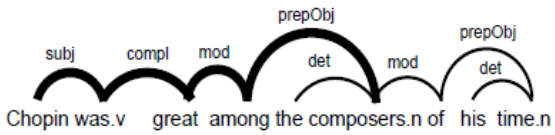


Figura 3.1: Exemplo de ligação em cadeia de Leila, retirado de (SUCHANEK; IFRIM; WEIKUM, 2006a)

Um *modelo* é uma ligação em cadeia onde duas palavras são substituídas por espaços reservados. Na Figura 3.2, retirada de (SUCHANEK; IFRIM; WEIKUM, 2006a), é exibido um modelo com os espaços reservados X e Y. O caminho mais curto de um espaço reservado a outro, marcado em negrito, é denominado *ponte*.

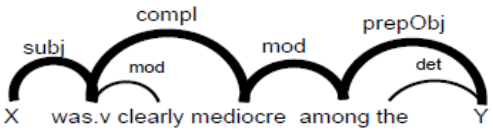


Figura 3.2: Exemplo de padrão de Leila, retirado de (SUCHANEK; IFRIM; WEIKUM, 2006a)

Diz-se que um padrão *combina* com uma ligação em cadeia se a ponte do padrão aparece na ligação em cadeia. A combinação ocorre mesmo se substantivos e adjetivos forem diferentes. O padrão da Figura 3.2 combina com a ligação em cadeia da Figura 3.1, pois a ponte do padrão aparece na ligação em cadeia, apesar da diferença dos adjetivos.

Para descobrir pares para uma relação, o algoritmo de Leila descrito a seguir exige a implementação de uma função para decidir em qual das seguintes categorias um par de palavras deve ser classificado: (i) um exemplo para a relação; (ii) um contra-exemplo da relação; (iii) um candidato da relação; ou (iv) nenhuma das categorias anteriores. Para especificar os exemplos, a implementação pode ser feita através de uma lista, recuperada de um arquivo de exemplos.

O algoritmo funciona em três fases:

1. Na fase de descoberta, a ferramenta procura por ligações nas quais aparecem os pares de exemplo, gerando modelos positivos. Em seguida, o algoritmo repassa todas as sentenças e encontra todas as ligações que correspondem a modelos positivos, mas produzem um contra-exemplo. Estes modelos são armazenados como modelos negativos.

2. Na fase de treinamento, técnicas de aprendizado de máquina, como SVM (CHERKASSKY; MA, 2004) e *k-Nearest-Neighbor* (TAN; STEINBACH; KUMAR, 2005),

são aplicadas para descobrir tendências em modelos positivos. O resultado é um classificador para modelos.

3. Na fase de teste, são usadas todas as sentenças do corpus. Para cada ligação, o sistema gera todos os modelos possíveis ao substituir um par de palavras por espaços reservados. Se duas palavras formam um par candidato e o modelo é classificado como positivo, então Leila propõe que o par faça parte da relação.

Para a fase de treinamento e a fase de testes, são fornecidos *corpora* diferentes. Como resultado, são gerados novos pares para a relação proposta.

Leila foi escolhida para o protótipo apresentado neste trabalho devido ao fato de que realiza o tratamento adequado de expressões temporais absolutas, assim eliminando a necessidade de resolver ambiguidades na representação de datas. Adicionalmente, a ferramenta realiza a extração de fatos temporais binários de forma adequada. Além disso, o protótipo desta dissertação emprega arquivos de exemplo, utilizando para isto informação estruturada do projeto DBpedia[2].

3.2.2 Outras Ferramentas de Extração de Informações

KnowItAll (ETZIONI et al., 2004) é um sistema de extração de fatos, conceitos e relacionamentos de páginas da Web. Inicialmente, um pequeno conjunto de regras genéricas e independentes de domínio é construído de forma manual. Em seguida, KnowItAll expande o conjunto de regras através de técnicas de aprendizado de máquina. A aplicação das regras obtidas sobre o texto em linguagem natural dá origem a um conjunto de fatos extraídos. Para validar estes fatos, KnowItAll realiza consultas em motores de busca sobre a Web. Para cada fato extraído, o número de resultados obtidos na busca é utilizado como métrica para medir a veracidade do fato. Estes resultados são submetidos a um classificador Naive Bayes (DOMINGOS; PAZZANI, 1997), atribuindo a um fato a qualidade de verdadeiro ou falso.

TextRunner (BANKO et al., 2007) é uma ferramenta de extração de informações a partir de texto em linguagem natural que visa escalabilidade e baixo tempo de processamento. TextRunner consiste de três módulos. O primeiro é o módulo de aprendizado, que recebe como entrada um *corpus* de exemplo e gera como saída um classificador que rotula candidatos à extração como prováveis ou não. O segundo módulo é o extrator, que realiza uma única passagem sobre todos os documentos para extrair informações de todas as relações possíveis. A informação extraída é submetida ao classificador gerado pelo módulo de aprendizado, eliminando os candidatos não aptos à extração. O terceiro módulo é o avaliador, que atribui probabilidades às informações extraídas pelo extrator através de um modelo probabilístico de redundância em texto. A principal virtude de TextRunner consiste na passagem única do módulo extrator, de forma que são extraídas de uma única vez informações para as relações estudadas. Em um experimento comparativo com KnowItAll apresentado em (BANKO et al., 2007), TextRunner teve um tempo de execução muito inferior e apresentou menos erros nos fatos extraídos. No entanto, neste experimento, TextRunner obteve um número menor de extrações do que KnowItAll.

[2]http://dbpedia.org/

3.3 Ontologias de Propósito Geral

Existem iniciativas de pesquisa com o objetivo de popular ontologias para representar conhecimento extraído de forma automática ou semiautomática de bases estabelecidas. Duas dessas iniciativas visam a população de ontologias a partir da organização semântica de informações extraídas da Internet: DBpedia (BIZER et al., 2009) e YAGO (SUCHANEK; KASNECI; WEIKUM, 2007). Estas iniciativas são apresentadas nesta seção.

3.3.1 DBpedia

DBpedia (BIZER et al., 2009) é um projeto que busca extrair informações estruturadas da enciclopédia aberta Wikipedia[3] e disponibilizá-las livremente na Internet. A Wikipedia é um dos portais mais visitados da Internet e está sob constante revisão, dando origem a uma rica fonte de dados de diferentes domínios (AUER et al., 2007). A Wikipedia está limitada pela possibilidade de busca baseada apenas em texto, o que limita o acesso a sua base de conhecimento. O projeto DBpedia possui o objetivo de converter o conteúdo da Wikipedia em conhecimento estruturado, de forma que técnicas da Web Semântica possam ser aplicadas no resultado. Desta forma, é possível realizar consultas sofisticadas, conectar os dados da DBpedia a outras fontes dados da Web e criar novas aplicações e *mashups*.

Periodicamente, novos dados são coletados da Wikipedia, gerando uma base atualizada da DBpedia que está disponível online[4]. A versão mais recente de DBpedia (versão 3.7 de setembro de 2011) possui 3,64 milhões de entidades, das quais 1,83 milhões estão classificadas de forma consistente em ontologia. Para cada entidade é definido um identificador globalmente único, que pode ser traduzido para uma descrição em RDF da mesma, incluindo: definições em 30 idiomas, relacionamentos com outros recursos, classificações em quatro hierarquias conceituais e links para outras fontes da Web descrevendo a entidade.

É possível realizar consultas sofisticadas sobre a base de dados da DBpedia através da linguagem SPARQL (SEGARAN et al., 2009) ou de interfaces gráficas como a *Faceted Wikipedia Search* (HAHN et al., 2010). Todo o conteúdo da DBpedia está disponível gratuitamente para *download* sob licenças livres.

A DBpedia é considerada uma das partes mais importantes da *LOD cloud* por Tim Berners-Lee (TALIS, 2008). Pela Figura 2.1 (página 22), é possível visualizar a DBpedia como uma das maiores bases de *Linked Data* atualmente. Isto ocorre pelo fato desta fonte de dados ter se tornado ponto de referência e base para muitas outras iniciativas de *Linked Data*.

Cada descrição em RDF de uma entidade da DBpedia possui uma URI (*Universal Resource Identifiers*). Para determinada entidade, a URI utilizada na DBpedia é baseada no endereço provido pela Wikipedia. Por exemplo, a página na Wikipedia sobre a cidade de Porto Alegre está disponível no endereço `http://en.wikipedia.org/wiki/Porto_Alegre`. Como resultado, o recurso que representa esta mesma cidade na DBpedia encontra-se na URI `http://dbpedia.org/page/Porto_Alegre`.

A DBpedia estabelece uma ontologia para representar semanticamente o conteúdo extraído. Esta ontologia consiste de 170 classes e mais de 700 propriedades com definições de domínio e imagem. A ontologia foi criada manualmente a partir dos tipos de *infobox*

[3] http://www.wikipedia.org
[4] http://dbpedia.org/

mais comumente usados na versão em inglês da Wikipedia. Na Wikipedia, uma *infobox* é uma tabela empregada no topo dos artigos para sumarizar informações comuns a diversos textos e melhorar a navegação para outros artigos relacionados.

O projeto DBpedia deu origem a uma ferramenta de anotação semântica. Esta ferramenta, denominada DBpedia Spotlight (MENDES et al., 2011), anota páginas da Web com recursos semânticos da DBpedia. Anotações semânticas podem ser definidas como descritores de conteúdo inseridos em HTML de forma a reduzir a ambiguidade de interpretação no texto. Por exemplo, a palavra "Cruzeiro" pode ser interpretada como um clube de futebol mineiro ou uma moeda utilizada no Brasil, dentre outros significados. Nesse contexto, anotações associam URIs com elementos HTML de forma a prover uma interpretação não ambígua.

DBpedia Spotlight é voltada para a anotação com recursos semânticos da DBpedia. Dado um determinado texto, a ferramenta DBpedia Spotlight analisa suas palavras e expressões e retorna recursos correspondentes na DBpedia, caso existam. A anotação semântica de uma página em HTML traz como benefícios: o enriquecimento de dados não estruturados ou semi-estruturados com um contexto, descobrindo novas relações dentro de um domínio; e a possibilidade de uma busca encontrar resultados que não estão explicitamente relacionados à consulta original.

A página do projeto DBpedia Spotlight[5] disponibiliza o uso interativo da ferramenta via Internet, e seu código está disponível livremente. A ferramenta permite que sejam configurados os valores de confiança e suporte. A confiança deve ser um valor entre 0 e 1 indicando o quão confiável é a anotação. Valores maiores indicam que mais rigor deve ser usado na anotação, resultando em anotações em menor número e mais confiáveis. Valores menores costumam resultar em mais anotações e podem aumentar a chance de erros. O suporte pode variar de 0 até infinito, indicando um valor mínimo de referências que determinada entidade deve receber dentro da Wikipedia, ou seja, quantas páginas da Wikipedia possuem *links* para a entidade. Valores maiores de suporte permitem a anotação apenas de entidades consideradas populares.

O projeto *Faceted Wikipedia Search* (HAHN et al., 2010) possui o objetivo de fornecer uma interface de busca facetada à DBpedia. Esta interface permite a um usuário leigo, sem conhecimento técnico de buscas sobre a Web Semântica, como SPARQL, consultar toda a base de dados da DBpedia de forma gráfica. É possível realizar consultas complexas que obtêm resultados superiores à busca textual usual, como a fornecida pela Wikipedia e por motores de busca comerciais. Consultas permitem descobrir: "Que jogadores de futebol nascidos no Brasil entre 1940 e 1950 atuaram na Espanha?", "Que rios desaguam no Rio Amazonas e possuem mais de 50 km?" e "Que espécies de plantas foram catalogadas entre 1990 e 1995?" A busca fornecida por esta ferramenta é de propósito geral, englobando informações temporais. O resultado do projeto pode ser acessado gratuitamente pela Internet[6]. No entanto, o código-fonte não está disponível, sendo possível realizar consultas apenas sobre a base de dados da DBpedia existente na Internet.

3.3.2 YAGO

Esta seção apresenta o trabalho YAGO (SUCHANEK; KASNECI; WEIKUM, 2007). Este projeto utiliza WordNet (FELLBAUM, 1998), um banco de dados léxicos para a língua inglesa. WordNet foi designado para servir como dicionário e tesauro eletrônico, além de auxiliar na análise de texto automática e aplicações de inteligência artificial. Os

[5] http://spotlight.dbpedia.org/
[6] http://dbpedia.neofonie.de/

conceitos de WordNet podem estabelecer relações léxicas entre si, como: sinônimos, antônimos, hiperônimos, hipônimos, holônimos, merônimos e outros. Os conceitos também estão classificadas em classes morfossintáticas, como substantivo, verbo ou adjetivo. Quando um par de conceitos está ligado por uma relação do tipo hiperônimo/hipônimo, é possível estabelecer uma hierarquia do tipo "é-um" entre eles. Por exemplo, "animal" é hiperônimo de "mamífero" (e o segundo é hipônimo do primeiro), de forma que todo mamífero é um animal.

YAGO (SUCHANEK; KASNECI; WEIKUM, 2007) é uma ontologia composta de conhecimento extraído da Wikipedia. Diferentemente da DBpedia, que visa a extração de informações de *infoboxes* da enciclopédia, YAGO se baseia nas páginas de categorias da Wikipedia. As páginas de categorias são listas de artigos que pertencem a uma categoria específica. As categorias da Wikipedia estão organizadas de forma hierárquica. No entanto, esta hierarquia não possui muita utilidade para criação de uma ontologia. Para isto, o projeto YAGO aplica a hierarquia de conceitos de WordNet, unificando os termos providos por este com os fatos extraídos da Wikipedia. Esta abordagem permite que o conhecimento da Wikipedia seja explorado através da taxonomia de conceitos de WordNet. YAGO é baseado em entidades e relações binárias. Para aumentar a expressividade da ontologia, é possível estabelecer relacionamentos entre relações binárias e definir propriedades para as relações (como transitividade). O projeto YAGO define seu próprio modelo de dados, denominado *modelo YAGO*, para definir estes relacionamentos entre relações. Desta forma, a ontologia de YAGO acaba por não seguir os padrões recomendados pela W3C, como OWL e RDF.

3.3.3 TOB

TOB (*Timely Ontologies for Business Relations*) (ZHANG et al., 2008) possui o objetivo de representar relações temporais entre empresas, produtos e consumidores através de uma ontologia. Estes dados são extraídos a partir de texto em linguagem natural e inseridos em uma ontologia que segue um modelo de dados baseado no modelo YAGO. Um fato é associado a um tempo de validade. Por exemplo, é possível representar que determinada pessoa ocupava o cargo de CEO de determinada organização em um certo intervalo de tempo. Para representação temporal, TOB utiliza um modelo de quatro componentes temporais, que permite que o tempo seja representado mesmo quando não há certeza absoluta sobre a validade dos fatos temporais. Os dois primeiros componentes temporais estabelecem o intervalo de início, ou seja, o período de tempo em que a validade do fato tem início. Os dois últimos componentes temporais estabelecem o intervalo de fim, ou seja, o período de tempo em que a validade do fato se encerra. Se não houver dúvida a respeito do momento de início da validade do fato, o intervalo de início terá seus componentes de início e fim iguais, tornando o intervalo igual a um ponto. O mesmo é observado para o intervalo de fim.

3.3.4 T-YAGO

T-YAGO (WANG et al., 2010) estende a base de dados de YAGO com aspectos temporais. Neste projeto, os fatos de YAGO são associados a um tempo de validade. Um fato pode ser válido em um ponto de tempo ou dentro de um intervalo temporal. Para representar um fato temporal em um modelo de relações binárias, T-YAGO decompõe o fato n-ário em um fato primário com diversos fatos secundários associados. O fato primário recebe um identificador e os fatos secundários são representados através de uma relação entre o identificador e os demais argumentos. Quando houver dúvida a respeito

do exato momento de início ou de fim de um período de validade, é possível representar a validade de um fato através do modelo de representação temporal de TOB. Desta forma, o momento de início do período de validade possui um instante de início e um instante de fim (que podem ser idênticos, no caso de não haver dúvida sobre o início). O mesmo pode ser dito do momento de fim do período de validade.

3.4 Considerações Finais

Neste capítulo foram apresentados trabalhos relacionados a esta dissertação. Para esta dissertação, são realizados experimentos comparativos entre a gramática formal para normalização de expressões temporais relativas de EXTIO e um *baseline*. Para *baseline*, foi escolhida a normalização de expressões temporais de GUTime. Esta escolha foi feita porque GUTime é o estado da arte na normalização de expressões temporais. Adicionalmente, GUTime apresenta as seguintes semelhanças com EXTIO:

- são voltados para o processamento de documentos em linguagem natural da língua inglesa;
- para representar o tempo de forma primitiva, utilizam pontos, em vez de intervalos. Em ambas, um intervalo de tempo é representado através de seus pontos de início e fim;
- fazem a normalização em relação à data de publicação do documento, que pode ser configurada de forma independente para cada documento analisado;
- suportam expressões temporais nas granularidades de dia, semana, mês, ano, década e século;
- buscam normalizar expressões temporais da maneira que um humano faria ao ler os documentos;
- normalizam as expressões temporais em formatos sem ambiguidades, baseados em calendário, apesar dos formatos serem distintos para as duas abordagens.

O trabalho desta dissertação realiza a população de uma ontologia com fatos temporais extraídos de texto em linguagem natural. Para isto, é necessário selecionar uma ontologia para esta tarefa. As opções incluem as ontologias apresentadas neste capítulo, além da possibilidade de criar uma ontologia especificamente para ser populada com os fatos temporais.

As melhores práticas da Web Semântica aconselham o reuso de ontologias existentes sempre que possível (SIMPERL, 2009). O reuso do conhecimento ontológico é apontado como um dos fatores principais para que a Web Semântica alcance sucesso. Neste cenário, o uso de ontologias foi concebido como um meio para permitir e aumentar a interoperabilidade entre aplicações da computação. A extensibilidade de ontologias também deve ser levada em conta, pois permite que uma ontologia seja acrescida de novos conceitos e propriedades quando necessário expandir sua estrutura. Adicionalmente, existem problemas de custo e qualidade na criação de uma ontologia do princípio. Desta forma, o trabalho desta dissertação utiliza uma ontologia existente para a população com fatos temporais.

Para este trabalho foi selecionada a ontologia da DBpedia para a população com fatos temporais. Diversos fatores motivaram esta escolha. Um deles é o apontamento da

DBpedia como uma das partes mais importantes da *LOD cloud* por Berners-Lee (TALIS, 2008). A DBpedia é uma das maiores bases de *Linked Data* disponível. O trabalho desta dissertação busca usufruir disto, tornando-se mais uma iniciativa que gera dados com *links* para a DBpedia. Outro fator importante é que a DBpedia segue os padrões recomendados pela W3C para tecnologias de Web Semântica. Também é importante ressaltar que a DBpedia possui *links* para dados de diversas outras ontologias, como EuroStat, CIA World Factbook, Freebase, OpenCyc e a própria base de YAGO (BIZER et al., 2009). Por estes motivos, a ontologia da DBpedia foi selecionada para este trabalho.

4 ANÁLISE DE PROPRIEDADES TEMPORAIS DE ONTOLOGIA

Um dos objetivos da abordagem ***EXTIO*** *(Extraction of Temporal Information Using Ontologies)* é extrair fatos temporais de texto em linguagem natural e inseri-los em uma ontologia. A tarefa de inserir fatos em ontologia também é chamada de *população de ontologia* (MAYNARD; LI; PETERS, 2008). Uma vez populada, a ontologia fica disponível para a realização de consultas semânticas sobre os fatos extraídos do texto.

O objetivo deste capítulo é identificar propriedades temporais de ontologia para populá-los com os fatos temporais extraídos pela proposta EXTIO. Dentro da arquitetura de três camadas da Web Semântica (BERNERS-LEE, 1998), uma ontologia permite a relação hierárquica entre conceitos e a descrição de propriedades. Para esta tarefa, foi selecionada a ontologia da DBpedia (BIZER et al., 2009).

Este capítulo está organizado da seguinte forma. A seção 4.1 apresenta um estudo sobre as propriedades temporais da ontologia da DBpedia. A seção 4.2 apresenta um estudo realizado sobre as ocorrências destas propriedades temporais. A seção 4.3 encerra o capítulo com as considerações finais.

4.1 Descoberta das Propriedades Temporais

Esta seção apresenta um estudo sobre as propriedades temporais da ontologia da DBpedia. Esta ontologia possui propriedades que descrevem as mais diversas características, como temperatura, área, distância, lucro etc. Para armazenar informações temporais nesta ontologia, o trabalho apresentado nesta dissertação utiliza as propriedades que representam informação temporal. Para este estudo, a versão da DBpedia utilizada é a 3.6, disponível na época em que o estudo foi realizado.

A ontologia da DBpedia está disponível livremente na Internet[1], onde é possível obter a estrutura da ontologia na linguagem OWL (*Web Ontology Language*), a linguagem recomendada pelo W3C (*World Wide Web Consortium*) para definição de ontologias. Esta linguagem permite que, para cada propriedade da ontologia, seja definida uma imagem (*range*), que indica os valores que a propriedade pode assumir.

O estudo da ontologia pode ser conduzido de duas maneiras. Uma delas é através da visualização em um programa gráfico de edição de ontologias, como Protégé[2]. Outra é pela análise do código-fonte OWL. O código-fonte OWL é um arquivo no formato XML, que segue uma definição fornecida pelo W3C.

Para classificar cada propriedade da ontologia da DBpedia como temporal ou não,

[1] http://wiki.dbpedia.org/Downloads
[2] http://protege.stanford.edu/

este estudo definiu os seguintes critérios, que englobam todas as relações temporais da DBpedia:

1. São classificadas como propriedades temporais todas as propriedades cuja imagem é uma data. Estas propriedades possuem como imagem o tipo *date*[3] de XML Schema.

   A propriedade que representa a data de nascimento de uma pessoa é um exemplo de propriedade temporal do tipo *date*. A listagem da Figura 4.1 exibe esta propriedade, descrita na linguagem OWL. A linha 1 inicia a descrição da propriedade com a *tag* `owl:DatatypeProperty`. A linha 2 apresenta o atributo `rdf:about` para atribuir a URI `http://dbpedia.org/ontology/birthDate` para esta propriedade. As linhas 3 e 4 atribuem rótulos (`rdfs:label`) nos idiomas inglês e alemão, respectivamente, a esta propriedade. As linhas 5 a 7 trazem o domínio (`rdfs:domain`) da propriedade. Este domínio é a classe que descreve uma Pessoa (`http://dbpedia.org/ontology/Person`), o que significa que, toda a vez que um recurso possuir uma data de nascimento, este recurso é necessariamente do tipo Pessoa de DBpedia. As linhas 8 a 10 denotam a imagem (`rdfs:range`) da propriedade. Finalmente, a linha 11 encerra a descrição da propriedade com a *tag* `/owl:DatatypeProperty`.

```
1  <owl:DatatypeProperty
2  rdf:about="http://dbpedia.org/ontology/birthDate">
3  <rdfs:label xml:lang="en">birth date</rdfs:label>
4  <rdfs:label xml:lang="de">Geburtsdatum</rdfs:label>
5  <rdfs:domain
6  rdf:resource="http://dbpedia.org/ontology/Person">
7  </rdfs:domain>
8  <rdfs:range
9  rdf:resource="http://www.w3.org/2001/XMLSchema#date">
10 </rdfs:range>
11 </owl:DatatypeProperty>
```

Figura 4.1: Propriedade que representa a data de nascimento de uma pessoa

2. São classificadas como propriedades temporais todas as propriedades cuja imagem representa um ano do calendário. Estas propriedades possuem como imagem o tipo *gYear*[4] de XML Schema.

   A propriedade que representa o ano de formação de uma organização é um exemplo de propriedade temporal do tipo *gYear*. A listagem da Figura 4.2 exibe esta propriedade. A linha 2 utiliza o atributo `rdf:about` para atribuir a URI `http://dbpedia.org/ontology/formationYear` a esta propriedade. A linha 3 atribui um rótulo no idioma inglês a esta propriedade. As linhas 4 a 6 indicam que o domínio (`rdfs:domain`) da propriedade é a classe de DBpedia que descreve uma Organização

[3] http://www.w3.org/2001/XMLSchema#date
[4] http://www.w3.org/2001/XMLSchema#gYear

(`http://dbpedia.org/ontology/Organisation`). Finalmente, as linhas 7 a 9 denotam a imagem da propriedade.

```
1  <owl:DatatypeProperty
2  rdf:about="http://dbpedia.org/ontology/formationYear">
3  <rdfs:label xml:lang="en">formation year</rdfs:label>
4  <rdfs:domain
5  rdf:resource="http://dbpedia.org/ontology/Organisation">
6  </rdfs:domain>
7  <rdfs:range
8  rdf:resource="http://www.w3.org/2001/XMLSchema#gYear">
9  </rdfs:range>
10 </owl:DatatypeProperty>
```

Figura 4.2: Propriedade que representa o ano de formação de uma organização

3. São classificadas como propriedades temporais todas as propriedades cuja imagem representa um período de tempo medido em dias. Estas propriedades possuem como imagem o tipo *day*[5] de DBpedia.

   A propriedade que representa o período orbital, em dias, de um planeta é um exemplo de propriedade temporal do tipo *day*. A listagem da Figura 4.3 exibe esta propriedade. A linha 2 utiliza o atributo `rdf:about` para atribuir a URI `http://dbpedia.org/ontology/Planet/orbitalPeriod` a esta propriedade. A linha 3 atribui um rótulo no idioma inglês a esta propriedade. As linhas 4 a 6 indicam que o domínio (`rdfs:domain`) da propriedade é a classe de DBpedia que descreve um planeta (`http://dbpedia.org/ontology/Planet`). Finalmente, as linhas 7 a 9 denotam a imagem da propriedade.

```
1  <owl:DatatypeProperty
2  rdf:about="http://dbpedia.org/ontology/Planet/orbitalPeriod">
3  <rdfs:label xml:lang="en">orbital period (d)</rdfs:label>
4  <rdfs:domain
5  rdf:resource="http://dbpedia.org/ontology/Planet">
6  </rdfs:domain>
7  <rdfs:range
8  rdf:resource="http://dbpedia.org/datatype/day">
9  </rdfs:range>
10 </owl:DatatypeProperty>
```

Figura 4.3: Propriedade que representa o período orbital de um planeta

4. São classificadas como propriedades temporais todas as propriedades cuja imagem representa um período de tempo medido em horas. Estas propriedades possuem como imagem o tipo *hour*[6] de DBpedia.

[5] http://dbpedia.org/datatype/day
[6] http://dbpedia.org/datatype/hour

A propriedade que representa o período de órbita de uma missão espacial ao redor da lua, em horas, é um exemplo de propriedade temporal do tipo *hour*. A listagem da Figura 4.4 exibe esta propriedade. As linhas 2 e 3 utilizam o atributo `rdf:about` para atribuir a URI `http://dbpedia.org/ontology/SpaceMission/lunarOrbitTime` a esta propriedade. A linha 4 atribui um rótulo no idioma inglês a esta propriedade. As linhas 5 a 7 indicam que o domínio (`rdfs:domain`) da propriedade é a classe de DBpedia que descreve uma missão espacial. Finalmente, as linhas 8 a 10 denotam a imagem da propriedade.

```
1  <owl:DatatypeProperty
2  rdf:about=
3      "http://dbpedia.org/ontology/SpaceMission/lunarOrbitTime">
4  <rdfs:label xml:lang="en">lunar orbit time (h)</rdfs:label>
5  <rdfs:domain
6  rdf:resource="http://dbpedia.org/ontology/SpaceMission">
7  </rdfs:domain>
8  <rdfs:range
9  rdf:resource="http://dbpedia.org/datatype/hour">
10 </rdfs:range>
11 </owl:DatatypeProperty>
```

Figura 4.4: Propriedade que representa período de órbita de uma missão espacial ao redor da lua

5. São classificadas como propriedades temporais todas as propriedades cuja imagem representa um período de tempo medido em minutos. Estas propriedades possuem como imagem o tipo *minute*[7] de DBpedia.

   A propriedade que representa o tempo de duração de uma obra, em minutos, é um exemplo de propriedade temporal do tipo *minute*. A listagem da Figura 4.5 exibe esta propriedade. A linha 2 utiliza o atributo `rdf:about` para atribuir a URI `http://dbpedia.org/ontology/Work/runtime` a esta propriedade. A linha 3 atribui um rótulo no idioma inglês a esta propriedade. As linhas 4 a 6 indicam que o domínio (`rdfs:domain`) da propriedade é a classe de DBpedia que descreve uma obra, como uma música ou um filme (`http://dbpedia.org/ontology/Work`). Finalmente, as linhas 7 a 9 denotam a imagem da propriedade.

6. São classificadas como propriedades temporais todas as propriedades cuja imagem representa um período de tempo medido em segundos. Estas propriedades possuem como imagem o tipo *second*[8] de DBpedia.

   A propriedade que representa o tempo de aceleração, em segundos, de um automóvel é um exemplo de propriedade temporal do tipo *second*. A listagem da Figura 4.6 exibe esta propriedade. As linhas 2 e 3 utilizam o atributo `rdf:about` para atribuir a URI `http://dbpedia.org/ontology/AutomobileEngine/acceleration` a esta propriedade. A linha 4 atribui um rótulo no idioma inglês

[7] http://dbpedia.org/datatype/minute
[8] http://dbpedia.org/datatype/second

```
1  <owl:DatatypeProperty
2  rdf:about="http://dbpedia.org/ontology/Work/runtime">
3  <rdfs:label xml:lang="en">runtime (m)</rdfs:label>
4  <rdfs:domain
5  rdf:resource="http://dbpedia.org/ontology/Work">
6  </rdfs:domain>
7  <rdfs:range
8  rdf:resource="http://dbpedia.org/datatype/minute">
9  </rdfs:range>
10 </owl:DatatypeProperty>
```

Figura 4.5: Propriedade que representa o tempo de duração de uma obra

a esta propriedade. As linhas 5 a 7 indicam que o domínio (`rdfs:domain`) da propriedade é a classe de DBpedia que descreve um motor de automóvel. Finalmente, as linhas 8 a 10 denotam a imagem da propriedade.

```
1  <owl:DatatypeProperty
2  rdf:about=
3   "http://dbpedia.org/ontology/AutomobileEngine/acceleration">
4  <rdfs:label xml:lang="en">acceleration (s)</rdfs:label>
5  <rdfs:domain
6  rdf:resource="http://dbpedia.org/ontology/AutomobileEngine">
7  </rdfs:domain>
8  <rdfs:range
9  rdf:resource="http://dbpedia.org/datatype/second">
10 </rdfs:range>
11 </owl:DatatypeProperty>
```

Figura 4.6: Propriedade que representa o tempo de aceleração de um automóvel

7. As demais propriedades foram analisadas conforme sua imagem para determinar quais delas são temporais. Neste processo foi verificado que algumas propriedades com imagem do tipo *double*[9] (número de precisão dupla) de XML Schema são temporais.

   A propriedade que representa o tempo de permanência de um objeto ou pessoa no espaço é um exemplo de propriedade temporal do tipo *double*. A listagem da Figura 4.7 exibe esta propriedade. A linha 2 utiliza o atributo `rdf:about` para atribuir a URI `http://dbpedia.org/ontology/timeInSpace` a esta propriedade. A linha 3 atribui um rótulo no idioma inglês a esta propriedade. As linhas 4 a 6 denotam a imagem da propriedade.

Estes critérios foram aplicados à DBpedia em sua versão 3.6. Nesta versão, a ontologia possui um total de 706 propriedades. A aplicação destes critérios resulta em um total

[9] http://www.w3.org/2001/XMLSchema#double

```
1 <owl:DatatypeProperty
2 rdf:about="http://dbpedia.org/ontology/timeInSpace">
3 <rdfs:label xml:lang="en">time in space (s)</rdfs:label>
4 <rdfs:range
5 rdf:resource="http://www.w3.org/2001/XMLSchema#double">
6 </rdfs:range>
7 </owl:DatatypeProperty>
```

Figura 4.7: Propriedade que representa o tempo de permanência de um objeto ou pessoa no espaço

de 167 propriedades temporais de DBpedia, o que equivale a 23,7% das propriedades totais existentes. Estas propriedades são utilizadas para receber os fatos temporais extraídos pela abordagem EXTIO.

A listagem completa das 167 propriedades temporais descritas nesta seção pode ser consultada no Apêndice C. Por simplicidade, não é listado o código OWL completo, apenas a URI de cada propriedade temporal.

4.2 Frequência das Propriedades Temporais

A ontologia da DBpedia é criada através de um esforço conjunto de uma comunidade aberta de pesquisadores (AUER et al., 2007) (BIZER et al., 2009). Uma propriedade (temporal ou não) desta ontologia pode ser modelada, mas não é garantido que ela seja efetivamente preenchida com dados dentro do escopo do projeto. Assim, o objetivo geral desta seção é verificar como as propriedades temporais descobertas neste capítulo são utilizadas para o preenchimento de instâncias dentro da ontologia.

Especificamente, para cada propriedade temporal descoberta na ontologia, são contados quantos registros (instâncias) existem na base de dados da DBpedia. Esta base de dados é preenchida com informações extraídas da Wikipedia (BIZER et al., 2009). É possível que uma propriedade temporal tenha sido modelada, mas não preenchida com dados extraídos da Wikipedia. Adicionalmente, é interessante detectar quais propriedades temporais possuem o maior número de registros dentro da base de dados da DBpedia, pois estes registros são usados pela abordagem EXTIO como insumo de entrada para a extração de informações a partir de texto.

Para detectar o número de registros de uma propriedade temporal, este trabalho definiu uma consulta na linguagem SPARQL. A linguagem SPARQL (SEGARAN et al., 2009) permite recuperar e manipular dados de uma base em RDF, como a DBpedia. Esta consulta é realizada através de um *endpoint* SPARQL. Um *endpoint* é um serviço que recebe uma consulta SPARQL e retorna o resultado da mesma sobre uma fonte de dados organizados em RDF, como é o caso da DBpedia. O *endpoint* utilizado está disponível na Internet, na página do projeto DBpedia[10].

Para esta contagem, são considerados apenas os registros que referenciam períodos temporais de 1980 a 2011. Como o principal foco de aplicação da abordagem EXTIO é em páginas da Web, as ocorrências temporais anteriores ao surgimento da rede foram descartadas. O ano de 1980 foi selecionado como limite para permitir a resolução de expressões temporais relativas com referência ao passado.

[10]http://dbpedia.org/sparql

A listagem da Figura 4.8 exibe a consulta SPARQL utilizada. A linguagem SPARQL possui uma sintaxe similar a SQL, com a diferença de ter sido projetada para permitir a realização de consultas sobre fontes de dados em RDF. O campo «URI DA PROPRIEDADE» indica a URI de uma das 167 propriedades temporais da DBpedia. Por exemplo, para a data de nascimento de uma pessoa, este campo seria substituído por `dbpedia-owl:birthDate`.

```
1 PREFIX dbpedia-owl: <http://dbpedia.org/ontology/>
2 SELECT COUNT(*) WHERE {
3   ?item <<URI DA PROPRIEDADE>> ?data
4   FILTER(
5     (?data) >= "1980-01-01"^^xsd:dateTime &&
6     (?data) <= "2011-12-31"^^xsd:dateTime) .
7 }
```

Figura 4.8: Consulta em SPARQL para detectar o número de registros de uma propriedade temporal

A linha 1 estabelece um prefixo para a consulta SPARQL. Isto define um *namespace*, de forma que um trecho de URI pode ser substituído por um prefixo. No exemplo da propriedade de data de nascimento, a URI original `http://dbpedia.org/ontology/birthDate` é transformada em `dbpedia-owl:birthDate` pela substituição por um prefixo. A linha 2 estabelece que a consulta deve retornar a contagem de registros obtidos. A linha 3 estabelece a relação entre um objeto `?item` e um objeto `?data`, através da propriedade temporal desejada. As linhas 4 a 6 estabelecem um filtro no qual apenas os registros cuja informação temporal esteja entre 01 de janeiro de 1980 e 31 de dezembro de 2011 são considerados.

Esta consulta foi executada de forma iterativa para as 167 propriedades temporais da ontologia da DBpedia. Para cada propriedade temporal, foi obtida uma contagem de registros. Os 15 resultados com mais ocorrências estão listados na Tabela 4.1. Os resultados estão em ordem decrescente.

A tabela evidencia quais propriedades temporais de DBpedia possuem o maior número de registros com datas entre 1980 e 2011. Por premissas de espaço, apenas as 15 propriedades com mais ocorrências estão listadas nesta tabela. O Apêndice C apresenta a listagem completa da contagem de ocorrências das 167 propriedades temporais.

4.3 Considerações Finais

Este capítulo apresentou um estudo sobre as propriedades da ontologia da DBpedia, utilizada para representação semântica dos fatos extraídos pela abordagem EXTIO. Cada propriedade foi analisada para verificar se se tratava de uma propriedade temporal ou não. Finalmente, foi realizada uma contagem das ocorrências das propriedades temporais sobre a base de dados existente. Esta análise serve de base para a extração de informações realizada pela abordagem EXTIO, com o objetivo de inserir novos registros na ontologia da DBpedia.

Tabela 4.1: As 15 propriedades temporais com mais ocorrências entre 1980 e 2011

Nome Propriedade	**Ocorrências**
release date	129325
birth date	85901
active years start year	60664
death date	56907
active years start date	32459
active years end date	22368
active years end year	22234
formation year	22166
added	21299
population as of	16874
death year	12527
draft year	9165
discovered	8335
founding year	8280

5 EXTIO – EXTRACTION OF TEMPORAL INFORMATION USING ONTOLOGIES

Este capítulo apresenta **EXTIO** – *Extraction of Temporal Information Using Ontologies* –, uma abordagem proposta para normalizar expressões temporais e extrair fatos temporais de texto em linguagem natural na língua inglesa. São apresentadas a visão geral da abordagem proposta e a especificação de seus componentes, que incluem a etapa de normalização de expressões temporais e a etapa de extração de fatos com informação temporal.

O capítulo está organizado da seguinte forma. A seção 5.1 apresenta uma visão geral da abordagem EXTIO. A seção 5.2 explica como é obtida a data de publicação do documento. A seção 5.3 apresenta em detalhes a gramática formal para normalização de expressões temporais relativas. Na seção 5.4, é apresentada a etapa de extração de fatos temporais. A seção 5.5 explica a organização dos fatos temporais extraídos em uma ontologia. Por fim, a seção 5.6 encerra o capítulo com as considerações finais.

A proposta inicial do trabalho foi publicada em (GALLINA; GALANTE, 2011). A gramática formal para normalização de expressões temporais relativas foi publicada em (GALLINA; GALANTE; DORNELES, 2011).

5.1 Visão Geral

A abordagem EXTIO é baseada em três etapas. A primeira etapa consiste em identificar e normalizar expressões temporais para datas absolutas. Isto inclui a substituição de expressões temporais relativas à data de publicação do documento por datas absolutas. A segunda etapa consiste em realizar a extração de fatos temporais do texto. A terceira e última etapa consiste em armazenar os fatos extraídos em uma ontologia, em um processo chamado *população de ontologia*. Ontologia é uma especificação formal e explícita de uma conceitualização compartilhada (GRUBER, 1993). Como resultado, é obtida uma ontologia preenchida com os fatos temporais que foram extraídos do texto. Esta ontologia fica disponível para que, posteriormente, seja possível realizar consultas sobre os fatos extraídos.

A Figura 5.1 apresenta a arquitetura da abordagem EXTIO. O passo 1 calcula a data do documento, que serve como tempo de referência para a normalização de expressões temporais relativas. No passo 2, todas as ocorrências de datas são normalizadas, incluindo as expressões temporais relativas. Esta normalização é baseada em um método inédito proposto neste trabalho, a normalização de expressões temporais relativas através de uma gramática formal. O *parser* da gramática faz a normalização do texto com base em um conjunto de regras. Uma vez que o texto possua todas as ocorrências de data represen-

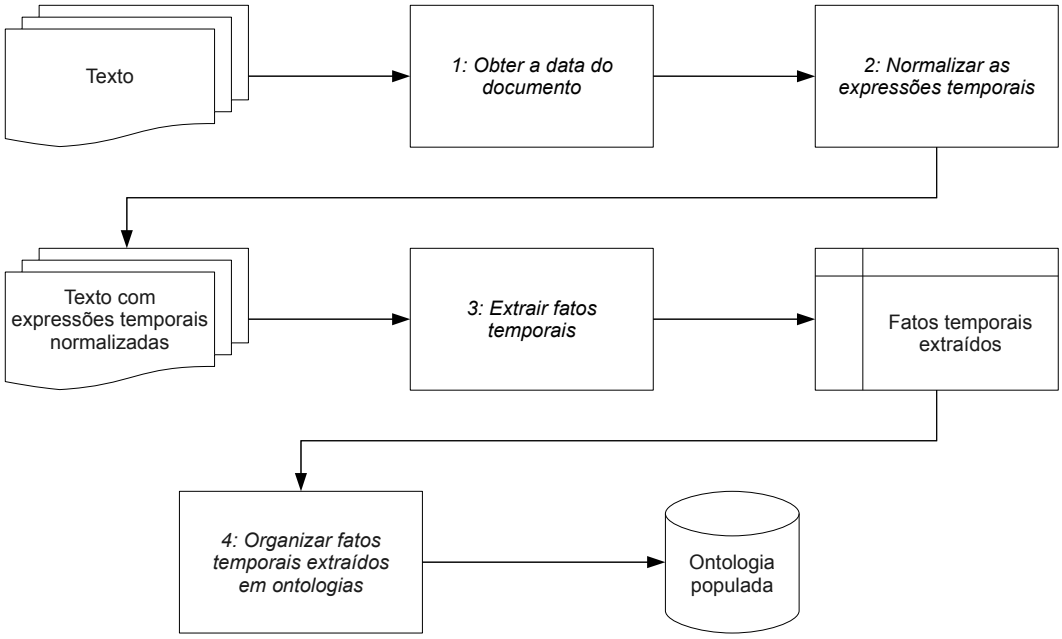


Figura 5.1: Arquitetura da abordagem EXTIO

tadas como datas absolutas, é possível realizar a extração de fatos temporais no passo 3. Finalmente, no passo 4, os fatos temporais extraídos são armazenados em uma ontologia. Estes dados ficam disponíveis para consultas através de ferramentas usuais de consulta a ontologias.

A abordagem EXTIO recebe como entrada um documento (ou conjunto de documentos) em linguagem natural. A abordagem EXTIO é voltada para documentos na língua inglesa. Conforme estudo feito em (LINGUA DTIL, 2007), no ano de 2007, 45% das páginas da Web estavam escritas em inglês. As página da Web escritas em português somam 1,39%. Por este motivo, a língua inglesa foi escolhida para a abordagem EXTIO.

No restante deste capítulo, cada um dos passos da abordagem EXTIO é explicado em detalhes. Também são apresentados exemplos para mostrar o funcionamento da abordagem proposta.

5.2 Obtenção da Data de Documentos

Esta seção apresenta o método de identificação da data de publicação de documentos. Esta data é utilizada como tempo de referência para a normalização de expressões temporais relativas. Por exemplo, se uma notícia possui a frase "Google acquired YouTube last Monday" e é possível detectar que a data de publicação do documento é 02 de abril de 2008, então é possível inferir que a aquisição ocorreu no dia 31 de março de 2008, que é a segunda-feira anterior à data do documento[1].

Segundo (ZHANG et al., 2008) e (KOEN; BENDER, 2000), a data de publicação do documento pode ser obtida das seguintes fontes:

1. a URL da página na Internet. Por exemplo, na página:

[1]Exemplo extraído de (ZHANG et al., 2008).

www.guardian.co.uk/football/2011/apr/13/chelsea-carlo-ancelotti

é possível inferir que a data de publicação é 13 de abril de 2011;

2. os metadados da página da Internet, em meio ao HTML. Para isto, a procura pela data é realizada na *tag* meta; ou

3. no conteúdo do documento. A data de publicação pode ser encontrada próxima de palavras-chave como "*published*" (publicado), "*posted*" (postado) ou "*publication date*" (data de publicação). No caso de uma página que permite comentários de usuários, é importante evitar confundir a data da página com as datas dos comentários.

A abordagem EXTIO apresentada neste trabalho utiliza a data de publicação para normalização das expressões temporais relativas. Devido a este assunto ter sido tratado nestes trabalhos anteriores, a implementação realizada para esta dissertação não inclui um novo método de obtenção da data do documento.

5.3 Normalização de Expressões Temporais Relativas

Esta seção apresenta a etapa de normalização de expressões temporais a partir de texto em linguagem natural na língua inglesa, que representa uma das principais contribuições deste trabalho. Neste passo, todas as expressões temporais do documento são padronizadas em um formato único de representação de data. As expressões temporais relativas do documento são normalizadas para datas absolutas. O resultado final desta etapa é o documento de texto original, com as expressões temporais normalizadas.

A normalização é importante para que a extração de fatos temporais – passo seguinte a esta etapa na abordagem EXTIO – receba como entrada um documento com informação temporal padronizada, de forma que a extração seja capaz de obter mais fatos a partir do texto. Por exemplo, a ocorrência no texto da expressão temporal relativa yesterday (ontem) é substituída pela data absoluta a que se refere, isto é, o dia anterior à data de publicação do documento. A extração de fatos temporais usufrui desta normalização ao associar fatos à data absoluta, em vez de usar o texto yesterday.

A gramática deste trabalho atribui diferentes granularidades temporais à data normalizada, de acordo com a expressão temporal relativa encontrada no texto. A granularidade temporal indica a duração do período de tempo tratado pela regra de produção. As granularidades adotadas pelas regras de produção da gramática são: dia, semana, mês, ano, década e século. A granularidade mais fina adotada por este trabalho é a diária, devido às expressões temporais encontradas nos documentos de texto. A escolha por esta granularidade é validada pela realização de experimentos, pois são incomuns as ocorrências de expressões temporais envolvendo granularidades mais finas, como a granularidade horária.

Esta seção apresenta a gramática, suas regras de produção e as etapas da análise léxica. Para cada regra, um exemplo retirado de documentos reais, disponíveis na Web, é apresentado para ilustrar a aplicabilidade da regra. Por premissas de espaço, algumas regras de produção são suprimidas devido à similaridade com outras regras apresentadas previamente. A gramática formal completa pode ser encontrada no Apêndice A. O analisador léxico completo é listado no Apêndice B.

A Tabela 5.1 traz exemplos de expressões que podem ocorrer em um documento e a correspondente substituição proposta pela gramática. As datas absolutas desta tabela foram calculadas considerando seu valor caso fossem encontradas em um documento datado de 01 de abril de 2011.

Tabela 5.1: Exemplos de resolução de expressões temporais relativas

Expressão temporal relativa	**Expressão temporal normalizada**
tomorrow (amanhã)	02 de abril de 2011
last Tuesday (última terça-feira)	29 de março de 2011
next month (mês que vem)	maio de 2011
three years ago (três anos atrás)	2008

O restante desta seção mostra as regras de produção da gramática para a normalização de expressões temporais. Quando pertinente, também são mostradas etapas do analisador léxico, que é responsável por ler caracteres da entrada e agrupá-los em *tokens*. Em todas as regras, a variável `documentDate` armazena a data de publicação do documento.

A listagem da Figura 5.2 especifica a regra de produção principal da gramática, que gera todas as expressões temporais bem formadas. Todos os *tokens* que não se enquadram nesta regra de produção são transcritos para a saída de forma idêntica à entrada, através de regras de produção auxiliares.

A resolução da expressão yesterday (ontem) é feita através da regra de produção listada a seguir. A expressão é substituída pelo resultado de adicionar -1 dia à data de publicação do documento. Este cálculo é realizado pela função `addDays`. A resolução de today (hoje) e tomorrow (amanhã) é análoga, somando zero e um dia, respectivamente. A granularidade destas regras de produção é diária.

$yesterday \rightarrow$ "$yesterday$"
```
 {addDays(documentDate, -1);}
```

$today \rightarrow$ "$today$"
```
 {addDays(documentDate, 0);}
```

$tomorrow \rightarrow$ "$tomorrow$"
```
 {addDays(documentDate, 1);}
```

O seguinte exemplo ilustra a aplicação da regra de produção `yesterday`. A seguinte frase foi extraída de um documento cuja data de publicação é 22 de julho de 2011[2]:

"There was an agreement, some additional revenues, until yesterday when the president demanded $400 billion more."

No trecho "There was an agreement, some additional revenues, until", nenhum *token* é reconhecido pela gramática formal como pertencente a uma expressão temporal relativa. Desta forma, o trecho é transcrito para a saída de forma idêntica. Em seguida, o *token* yesterday é reconhecido pela gramática formal como uma expressão temporal relativa, gerada pela regra de produção `yesterday`. A ação semântica associada a esta

[2]O site http://transcripts.cnn.com/TRANSCRIPTS/1107/22/jkusa.01.html traz a transcrição de um programa de televisão transmitido no dia 22 de julho de 2011.

$temporal\_expression \rightarrow today$
$| \; tomorrow$
$| \; yesterday$
$| \; tonight$
$| \; months\_ago$
$| \; years\_ago$
$| \; weeks\_ago$
$| \; decades\_ago$
$| \; centuries\_ago$
$| \; this\_year$
$| \; last\_year$
$| \; next\_year$
$| \; this\_month$
$| \; last\_month$
$| \; next\_month$
$| \; this\_week$
$| \; last\_week$
$| \; next\_week$
$| \; this\_decade$
$| \; last\_decade$
$| \; next\_decade$
$| \; this\_century$
$| \; last\_century$
$| \; next\_century$
$| \; last\_occurrence\_of\_month$
$| \; last\_weekday$
$| \; next\_weekday$
$| \; weekday$
$| \; date\_without\_year$
$| \; day\_month$
$| \; month\_without\_year$

Figura 5.2: Regra de produção principal da gramática

regra de produção faz uma chamada à função `addDays`, utilizando como parâmetro a data de publicação do documento (`documentDate` que, no caso, armazena a data de 22 de julho de 2011) e o valor -1. Como resultado, a função `addDays` retorna a data de 21 de julho de 2011, que substitui o *token* yesterday no texto original. Quanto aos demais *tokens* da frase, nenhum é reconhecido como parte de uma expressão temporal relativa, e todos são transcritos para a saída de forma idêntica. Como resultado, a frase passa a ter a seguinte redação:

"There was an agreement, some additional revenues, until 2011/07/21 when the president demanded $400 billion more."

A expressão tonight (esta noite) é resolvida de forma análoga a today, somando zero dia à data de publicação do documento. Isto é feito através da função `addDays`. Adicionalmente, é impressa também a expressão "at night" (à noite), para dar ao texto o contexto

de que a expressão está no período noturno. Para esta regra de produção, é aplicada a granularidade diária. Nenhuma semântica adicional é dada ao período da noite.

$tonight \rightarrow$ "*tonight*"
`{addDays(documentDate, 0); print("at night");}`

A aplicação da regra de produção `tonight` é ilustrada pelo seguinte exemplo. Em um documento com data de publicação de 22 de fevereiro de 2012[3], a seguinte frase foi encontrada:

"Five things to watch for in tonight's debate"

O *token* tonight é reconhecido pela gramática formal como uma expressão temporal relativa. A ação semântica associada à regra de produção utilizada faz chamadas a duas funções: (i) `addDays`, com a data de publicação do documento de 22 de fevereiro de 2012 e o valor 0, o que obtém como resultado a data de 22 de fevereiro de 2012; e (ii) `print`, que imprime o texto "at night". Os demais *tokens* não são reconhecidos por nenhuma regra de produção da gramática formal e são transcritos para a saída de forma idêntica à entrada. Como resultado, a frase passa a ter a seguinte forma:

"Five things to watch for in 2012/02/22 at night's debate"

Neste exemplo, verifica-se que a sentença resultante não é gramaticalmente correta do ponto de vista da língua inglesa. No entanto, isto não é um problema para a abordagem EXTIO, pois estas frases não são destinadas à leitura por um humano. Elas são construídas de forma intermediária, para que, mais tarde, seja realizada a extração de informações sobre elas.

Para a resolução de expressões temporais que envolvem numerais, é utilizado o analisador léxico. O analisador léxico possui a responsabilidade de atribuir a `number` o valor de um número no texto, incluindo números por extenso. A expressão regular `[0-9]+` captura qualquer sequência de dígitos. Assim, qualquer número em sua representação inteira é compreendido pela gramática. O analisador léxico deste trabalho também compreende números por extenso de 1 (one) até 10 (ten). Desta forma, quando o analisador léxico encontra, por exemplo, a expressão three months ago (três meses atrás), ele é capaz de converter o texto three para o valor inteiro 3, armazenando o valor no campo `value`. Estas regras do analisador léxico estão na listagem a seguir.

$number \rightarrow [0-9]+$
`{number.value = valor da sequência de dígitos;}`
$number \rightarrow$ "*one*"
`{number.value = 1;}`
$number \rightarrow$ "*two*"
`{number.value = 2;}`
$\vdots$
$number \rightarrow$ "*ten*"
`{number.value = 10;}`

[3]http://articles.cnn.com/2012-02-22/politics/politics_preston-five-things_1_presidential-debate-republican-presidential-candidates-newt-gingrich?_s=PM:POLITICS

A expressão five months ago (cinco meses atrás) é resolvida através da regra de produção `months_ago`. Esta regra resolve todas as expressões compostas de um número seguido pelas palavras months ago. A função `addMonths` é definida para adicionar ou subtrair um número de meses da data atual. Nesta regra de produção, esta função soma o valor negativo do número de meses da expressão. O analisador léxico registra o valor de um número no campo `value` do *token* `number`. Por exemplo, na expressão five months ago, este valor é igual a 5, pois o analisador léxico converte números por extenso, de 1 até 10, para sua representação inteira. A regra de produção, cuja granularidade é mensal, está na listagem a seguir.

$token\_month \rightarrow$ "$month$" | "$months$"
$token\_ago \rightarrow$ "ago"
$months\_ago \rightarrow number\ token\_month\ token\_ago$
`{addMonths(documentDate, -number.value);}`

O seguinte exemplo ilustra a aplicação da regra de produção `months_ago`. Uma notícia[4], com data de publicação de 27 de fevereiro de 2011, possui a seguinte frase:

"UK arms companies visited Tripoli three months ago."

A gramática identifica a expressão temporal three months ago, que é decomposta em um *token* `number`, com o valor inteiro 3 no campo `value`, seguido dos *tokens* `token_month` e `token_ago`. O *parsing* desta expressão é realizado pela regra de produção `months_ago`. A ação semântica associada a esta regra faz uma chamada à função `addMonths`, passando como argumentos `documentDate` (27 de fevereiro de 2011) e `-number.value` (ou seja, -3). A função subtrai três meses da data de publicação do documento, obtendo a data de novembro de 2010. Não é possível saber exatamente a que dia do mês de novembro de 2010 a expressão temporal relativa se refere, portanto a informação de dia é descartada pela função `addMonths`. Como resultado, a frase passa a apresentar a seguinte redação:

"UK arms companies visited Tripoli 2010/11."

Quatro outras regras de produção funcionam de forma análoga a `months_ago`:

- A regra de produção `years_ago`, para expressões como six years ago (seis anos atrás). Para subtrair o número de anos da data de publicação do documento, é utilizada a função `addYears`. Esta regra de produção possui granularidade de ano:

  $token\_year \rightarrow$ "$year$" | "$years$"

  $years\_ago \rightarrow number\ token\_year\ token\_ago$

  `{addYears(documentDate, -number.value);}`

- A regra de produção `weeks_ago`, para expressões como one week ago (uma semana atrás). Para subtrair o número de semanas da data de publicação do documento, é utilizada a função `addWeeks`. Esta regra de produção possui granularidade de semana:

[4]http://www.guardian.co.uk/world/2011/feb/27/libyan-arms-fair-attended-by-uk-firms

$token\_week \rightarrow$ "$week$" | "$weeks$"

$weeks\_ago \rightarrow number\ token\_week\ token\_ago$

```
{addWeeks(documentDate, -number.value);}
```

- A regra de produção `decades_ago`, para expressões como two decades ago (duas décadas atrás). Para subtrair o número de décadas da data de publicação do documento, é utilizada a função `addDecades`. Esta regra de produção possui granularidade de década:

  $token\_decade \rightarrow$ "$decade$" | "$decades$"

  $decades\_ago \rightarrow number\ token\_decade\ token\_ago$

  ```
  {addDecades(documentDate, -number.value);}
  ```

- A regra de produção `centuries_ago`, para expressões como four centuries ago (quatro séculos atrás). Para subtrair o número de séculos da data de publicação do documento, é utilizada a função `addCenturies`. Esta regra de produção possui granularidade de século:

  $token\_century \rightarrow$ "$century$" | "$centuries$"

  $centuries\_ago \rightarrow number\ token\_century\ token\_ago$

  ```
  {addCenturies(documentDate, -number.value);}
  ```

As expressões this year (este ano), last year (ano passado) e next year (ano que vem) são normalizadas pelas regras de produção `this_year`, `last_year` e `next_year`, respectivamente. A ocorrência da expressão no texto é substituída pelo resultado de somar, respectivamente, 0, -1 e 1 ano à data de publicação do documento. Esta operação de soma é realizada pela função `addYears`. As três regras de produção destas expressões são listadas a seguir.

$token\_this \rightarrow$ "$this$"
$token\_last \rightarrow$ "$last$"
$token\_next \rightarrow$ "$next$"
$this\_year \rightarrow token\_this\ token\_year$
```
{addYears(documentDate, 0);}
```
$last\_year \rightarrow token\_last\ token\_year$
```
{addYears(documentDate, -1);}
```
$next\_year \rightarrow token\_next\ token\_year$
```
{addYears(documentDate, 1);}
```

Para ilustrar o uso da regra de produção `last_year`, é utilizado o seguinte exemplo. Um documento[5] datado de 11 de janeiro de 2012 possui a seguinte frase em seu conteúdo:

"Loss to Steelers in last year's playoffs still stings"

O *parsing* da expressão temporal last year é feito pela regra de produção `last_year`. A ação semântica associada a esta regra faz uma chamada à função `addYears`, com os argumentos `documentDate` e -1. A função subtrai um ano da data de publicação do

[5] http://articles.baltimoresun.com/2012-01-11/sports/bal-loss-to-steelers-in-last-years-playoffs-still-stings-20120110_1_experience-fuels-stings-ravens

documento, obtendo como resultado o ano de 2011. Não é possível saber exatamente a que dia e mês de 2011 a expressão temporal relativa se refere, portanto estas informações são descartadas. Desta forma, a granularidade resultante é de ano. A frase produzida possui o seguinte texto:

"Loss to Steelers in 2011's playoffs still stings"

As regras de produção a seguir funcionam de forma análoga. Para abreviação, as regras não são listadas neste capítulo, mas podem ser consultadas no apêndice A, que contém a gramática formal completa.

- As regras de produção `this_month`, `last_month` e `next_month` normalizam as expressões this month (este mês), last month (mês passado) e next month (mês que vem), respectivamente. Estas regras de produção possuem a granularidade de mês.

- As regras de produção `this_week`, `last_week` e `next_week` normalizam as expressões this week (esta semana), last week (semana passada) e next week (semana que vem), respectivamente. Estas regras de produção possuem a granularidade de semana.

- As regras de produção `this_decade`, `last_decade` e `next_decade` normalizam as expressões this decade (esta década), last decade (década passada) e next decade (década que vem), respectivamente. Estas regras de produção possuem a granularidade de década.

- As regras de produção `this_century`, `last_century` e `next_century` normalizam as expressões this century (este século), last century (século passado) e next century (século que vem), respectivamente. Estas regras de produção possuem a granularidade de século.

Uma expressão como last August (último agosto) é normalizada pela regra de produção `last_occurrence_of_month`. A expressão temporal é substituída pela última ocorrência do mês da expressão temporal (no exemplo, agosto) anterior à data de publicação do documento. O cálculo desta ocorrência é realizado pela função `getLastDateByMonth`. Esta função recebe como entrada a data de publicação do documento e a identificação do mês desejado (janeiro, fevereiro, ..., dezembro) e retorna como saída a data absoluta que contém o mês e o ano, de forma normalizada. A data absoluta resultante possui a granularidade de mês, pois o dia exato não é conhecido.

A regra de produção está na listagem a seguir. Para o *token* `month`, o analisador léxico armazena no campo `value` um valor de 1 a 12, onde 1 identifica o mês de janeiro (January) e 12 identifica o mês de dezembro (December). Este mesmo valor[6] é recebido pela função `getLastDateByMonth` para identificar o mês contido pela expressão temporal.

$month \rightarrow$ "$January$" `{month.value = 1;}`
$month \rightarrow$ "$February$" `{month.value = 2;}`
⋮

[6]De acordo com as boas práticas de programação, a implementação deste trabalho utilizou constantes identificando os meses, no lugar dos números de 1 a 12.

$month \rightarrow$ "$December$" `{month.value = 12;}`

$last\_occurrence\_of\_month \rightarrow token\_last\ month$
`{getLastDateByMonth(documentDate, month.value);}`

O seguinte exemplo ilustra a aplicação da regra de produção `last_occurrence_of_month`. A seguinte frase foi extraída de um documento[7] cuja data de publicação é 5 de dezembro de 2011:

"The most recent extension, passed last December, kept 7 million people on the rolls."

A gramática identifica que a expressão temporal last December é produzida pela regra `last_occurrence_of_month`, pois são encontrados os *tokens* `token_last` e `month`. O analisador léxico atribui ao *token* `month` o valor 12, armazenado no campo `value`, pois trata-se do mês de dezembro (December). A ação semântica associada à regra de produção faz uma chamada à função `getLastDateByMonth`, com os argumentos `documentDate` (igual a 5 de dezembro de 2011) e `month.value` (igual a 12). Como resultado, a função obtém a última ocorrência do mês de dezembro anterior à data de 5 de dezembro de 2011. Trata-se, neste caso, de dezembro de 2010, pois a última ocorrência do mês de dezembro não pode ser o próprio mês de dezembro em que o documento foi criado. O dia exato não é conhecido, sendo adotada a granularidade de mês. A frase passa a possuir a seguinte redação:

"The most recent extension, passed 2010/12, kept 7 million people on the rolls."

Uma expressão como next Thursday (próxima quinta-feira) é normalizada pela regra de produção `next_weekday`. A expressão é substituída pela primeira ocorrência daquele dia da semana após a data de publicação do documento. O cálculo desta ocorrência é realizado pela função `getNextWeekday`. Esta função recebe como entrada a data de publicação do documento e a identificação do dia da semana desejado (domingo, segunda-feira, terça-feira, ..., sábado) e retorna como saída a data absoluta equivalente à primeira ocorrência do dia da semana desejado após a data do documento.

A regra de produção está na listagem a seguir. Para o *token* `token_weekday`, o analisador léxico armazena no campo `value` um valor de 0 a 6, onde 0 identifica o domingo (Sunday) e 6 identifica o sábado (Saturday). Este mesmo valor[8] é recebido pela função `getNextWeekday` para identificar o dia da semana contido pela expressão temporal.

$token\_weekday \rightarrow$ "$Sunday$" `{token_weekday.value = 0;}`
$token\_weekday \rightarrow$ "$Monday$" `{token_weekday.value = 1;}`
$token\_weekday \rightarrow$ "$Tuesday$" `{token_weekday.value = 2;}`
$token\_weekday \rightarrow$ "$Wednesday$" `{token_weekday.value = 3;}`
$token\_weekday \rightarrow$ "$Thursday$" `{token_weekday.value = 4;}`
$token\_weekday \rightarrow$ "$Friday$" `{token_weekday.value = 5;}`

[7] http://money.cnn.com/2011/12/05/news/economy/unemployment_benefits_extension/index.htm

[8] De acordo com as boas práticas de programação, a implementação deste trabalho utilizou constantes identificando os dias da semana, no lugar dos números de 0 a 6.

token_weekday → "*Saturday*" `{token_weekday.value = 6;}`

next_weekday → *token_next token_weekday*
`{getNextWeekday(documentDate, token_weekday.value);}`

A regra de produção `next_weekday` é ilustrada pelo seguinte exemplo. Um documento[9] com data de publicação de 30 de setembro de 2011 possui a seguinte frase em seu conteúdo:

"Next Saturday, she will talk at the BD & Comics Passion festival"

A sequência de *tokens* `token_next` e `token_weekday` indica para a gramática que a expressão Next Saturday é produzida pela regra de produção `next_weekday`. O analisador léxico atribui a `token_weekday` o valor de 6, armazenado na variável `value`, pois o dia da semana na expressão é o sábado (Saturday). A ação semântica associada à regra de produção faz uma chamada à função `getNextWeekday`, com os argumentos `documentDate` (no caso, 30 de setembro de 2011) e `token_weekday.value` (no caso, 6). A função calcula qual é o sábado seguinte à data de publicação do documento e o resultado é 01 de outubro de 2011. A frase passa a ter a seguinte redação:

"2011/10/01, she will talk at the BD & Comics Passion festival"

Uma expressão como last Monday (a última segunda-feira) é normalizada pela regra de produção `last_weekday`. Esta regra de produção realiza a operação contrária à regra `next_weekday`. A expressão é substituída pela ocorrência mais recente daquele dia da semana antes da data de publicação do documento. O cálculo desta ocorrência é realizado pela função `getPreviousWeekday`. Esta função recebe como entrada a data de publicação do documento e a identificação do dia da semana desejado e retorna como saída a data absoluta equivalente à última ocorrência do dia da semana desejado anterior à data do documento. A regra de produção está na listagem a seguir.

last_weekday → *token_last token_weekday*
```
{getPreviousWeekday(documentDate,
    token_weekday.value);}
```

Uma expressão como Tuesday (terça-feira), sem nenhuma indicação de passado (last) ou futuro (next), é normalizada pela regra de produção `weekday`. Para a normalização de uma expressão temporal, neste caso, EXTIO define a seguinte heurística: buscar qual a ocorrência daquele dia da semana (no caso da expressão Tuesday, terça-feira) está mais próxima da data de publicação do documento. O cálculo deste dia da semana é realizado pela função `getNearestWeekday`. Esta função recebe como entrada a data de publicação do documento e a identificação do dia da semana desejado e retorna como saída a data absoluta equivalente à ocorrência do dia da semana desejado mais próxima à data do documento. A regra de produção é apresentada na listagem a seguir.

weekday → *token_weekday*
`{getNearestWeekday(documentDate, token_weekday.value);}`

[9] http://www.guardian.co.uk/lifeandstyle/2011/sep/30/audrey-niffenegger-interview-books

O pseudocódigo da função `getNearestWeekday` é apresentado no Algoritmo 1. Este algoritmo recebe como entrada a data de publicação do documento (*data_do_documento*) e o dia da semana da expressão a ser normalizada (*dia_da_semana*). A saída é a expressão temporal normalizada para uma data absoluta (*data_absoluta*). Na linha 5, a função `getPreviousWeekday` é usada para calcular a ocorrência anterior de *dia_da_semana* em relação a *data_do_documento*. A linha 6 calcula a diferença, em dias, entre a data obtida pela função `getPreviousWeekday` e a *data_do_documento*. A linha 7 obtém a ocorrência posterior de *dia_da_semana* em relação a *data_do_documento*. Na linha 8 é calculada a diferença entre a data obtida pela função `getNextWeekday` e a *data_do_documento*. O bloco que contém as linhas de 9 a 13 realiza a comparação entre a ocorrência anterior e a ocorrência posterior de *dia_da_semana*. Se a ocorrência anterior é a que apresenta a menor diferença em relação à data de publicação do documento, então a função retorna a ocorrência anterior; caso contrário, a ocorrência posterior é retornada. Não é necessário testar pela igualdade entre as diferenças, pois uma semana tem 7 dias, o que significa que, no pior caso, as diferenças são de 4 dias para uma ocorrência e 3 dias para a outra ocorrência, sem que jamais haja empate.

Algoritmo 1 Algoritmo da função getNearestWeekday

```
1:  INPUT: data_do_documento
2:  INPUT: dia_da_semana
3:  OUTPUT: data_absoluta
4:  begin
5:  ocorr_anterior ← getPreviousWeekday(data_do_documento, dia_da_semana)
6:  dif_anterior ← data_do_documento − ocorr_anterior)
7:  ocorr_posterior ← getNextWeekday(data_do_documento, dia_da_semana)
8:  dif_posterior ← ocorr_posterior − data_do_documento)
9:  if dif_anterior < dif_posterior then
10:    return ocorr_anterior
11: else
12:    return ocorr_posterior
13: end if
14: end
```

O seguinte exemplo ilustra a aplicação da regra de produção `weekday`. De um documento[10], com data de publicação de 10 de fevereiro de 2012, é extraída a seguinte frase:

"On Thursday the National Portrait Gallery opened its exhibition of Lucian Freud's portraits"

A gramática identifica que Thursday é gerada pela regra de produção `weekday`. O analisador léxico atribui ao *token* `token_weekday` o valor de 4, armazenado na variável `value`, pois o dia da semana na expressão é a quinta-feira (Thursday). A ação semântica associada à regra de produção faz uma chamada à função `getNearestWeekday`, com os argumentos `documentDate` (no caso, 10 de fevereiro de 2012) e `token_weekday.value` (no caso, 4). A função segue os passos do Algoritmo 1. A ocorrência de quinta-feira à data de 10 de fevereiro de 2012 é o dia 9 daquele mês, que apresenta um dia de diferença. A ocorrência de quinta-feira posterior

[10]http://www.guardian.co.uk/artanddesign/2012/feb/10/yoko-ono-leonardo-cohen-artists

a 10 de fevereiro de 2012 é o dia 16 daquele mês, que apresenta seis dias de diferença. Portanto, a ocorrência mais próxima é o dia 9 de fevereiro de 2012, que é a data retornada pela função. A frase passa a ter a seguinte redação:

"On 2012/02/09 the National Portrait Gallery opened its exhibition of Lucian Freud's portraits"

Uma expressão como January 15th (15 de janeiro) é uma expressão temporal relativa que possui informação de mês e dia. Este tipo de expressão traz o mês e o dia da data referenciada, mas não informa o ano. Para normalizar esta expressão, é necessário descobrir qual o valor do ano, para complementar o mês e o dia que já são conhecidos. Quando esta expressão temporal relativa aparece com o mês primeiro, seguido pelo dia (como January 15th), a normalização é realizada pela regra de produção `date_without_year`. Quando a expressão temporal relativa aparece com o dia primeiro, seguido pelo mês (como 15 January), a normalização é realizada pela regra de produção `day_month`.

Para a normalização de `date_without_year` e `day_month`, EXTIO define a seguinte heurística: buscar a ocorrência daquele dia do ano mais próxima da data de publicação do documento. O cálculo desta ocorrência é realizado pela função `getDateWithoutYear`. Esta função recebe como entrada a data de publicação do documento (no formato ano, mês e dia) e o dia do ano desejado (no formato mês e dia) e retorna como saída a data absoluta equivalente à ocorrência daquele dia do ano mais próxima à data do documento. As regras de produção são apresentadas na listagem a seguir.

$date\_without\_year \rightarrow month\ number$
```
{getDateWithoutYear(documentDate,
    month.value, number.value);}
```

$day\_month \rightarrow number\ month$
```
{getDateWithoutYear(documentDate,
    month.value, number.value);}
```

O Algoritmo 2 apresenta o pseudocódigo da função `getDateWithoutYear`. Este algoritmo recebe como entrada a data de publicação do documento (ano, mês e dia) e os dados da expressão temporal relativa sem ano (apenas mês e dia). Como saída, retorna a data absoluta à qual a expressão temporal relativa se refere. Como o mês e o dia são conhecidos, falta apenas calcular o ano. A função `obterAno` recebe como entrada uma data completa e retorna o ano da data. Na linha 6, esta função é utilizada para obter o ano de publicação do documento. Em seguida são analisadas as três datas absolutas candidatas a retorno pelo algoritmo: (i) a data composta pelo dia e mês conhecidos, com o mesmo ano em que o documento foi publicado (linha 7); (ii) a data composta pelo dia e mês conhecidos, com o ano anterior ao ano de publicação do documento (linha 12); e (iii) a data composta pelo dia e mês conhecidos, com o ano posterior ao ano de publicação do documento (linha 17). Dentre estas três candidatas, é retornada aquela que estiver mais próxima da data de publicação do documento. Não é necessário calcular todas as três diferenças, pois a data mais próxima da data de publicação do documento é a candidata que estiver a menos de meio ano de diferença em relação à data do documento. Nas linhas de 8 a 10, verifica-se se a candidata (i) está menos de meio ano de diferença,

retornando-a em caso positivo. Caso não esteja, as linhas 13 a 15 verificam se a candidata (ii) apresenta esta diferença, retornando-a em caso positivo. Em caso negativo, não é necessário calcular a diferença entre a candidata (iii) e a data de publicação do documento, pois por eliminação ela é a data procurada.

Algoritmo 2 Algoritmo da função getDateWithoutYear

```
1: INPUT: data_do_documento
2: INPUT: mes_do_ano
3: INPUT: dia_do_ano
4: OUTPUT: data_absoluta
5: begin
6: ano_do_documento ← obterAno(data_do_documento)
7: data_mesmo_ano ← construirData(ano_do_documento, mes_do_ano, dia_do_ano)
8: dif_data_mesmo_ano ← modulo(data_do_documento − data_mesmo_ano)
9: if dif_data_mesmo_ano < 1/2 Ano then
10:   return data_mesmo_ano
11: else
12:   data_ano_anterior ← construirData(ano_do_documento − 1, mes_do_ano, dia_do_ano)
13:   dif_data_ano_anterior ← modulo(data_do_documento − data_ano_anterior)
14:   if dif_data_ano_anterior < 1/2 Ano then
15:     return data_ano_anterior
16:   else
17:     data_ano_posterior ← construirData(ano_do_documento + 1, mes_do_ano, dia_do_ano)
18:     return data_ano_posterior
19:   end if
20: end if
21: end
```

A regra de produção `date_without_year` é ilustrada pelo seguinte exemplo. Um documento[11] com data de publicação de 28 de fevereiro de 2012 possui a seguinte frase em seu conteúdo:

"I have applied for my H1-B 3 year extension after my 6th year on November 30th"

A gramática identifica que November 30th é uma sequência de *tokens* `month` (com o valor 11 armazenado no campo `value`, devido ao mês de novembro), e `number` (com o valor 30 armazenado no campo `value`). Esta sequência de *tokens* é produzida pela regra `date_without_year`. A ação semântica associada à regra de produção faz uma chamada à função `getDateWithoutYear`, com os argumentos `documentDate` – igual a 28 de fevereiro de 2012 –, `month.value` – igual a 11, para indicar novembro – e 30 – para indicar o dia do mês. A função segue os passos do Algoritmo 2, que se dedica a procurar a ocorrência de 30 de novembro mais próxima da data de publicação

[11] http://www.trackitt.com/usa-discussion-forums/h1b/916661877/applied-for-h1-b-extention-on-november-30-2012

do documento. Primeiramente, a função constrói a data de 30 de novembro no mesmo ano de publicação, isto é, 30 de novembro de 2012. A diferença desta data para a data de publicação do documento é maior do que meio ano, portanto esta candidata é descartada. Em seguida a função verifica se a data procurada está no ano anterior ao ano de publicação do documento e constrói a data de 30 de novembro de 2011. Ao calcular a diferença entre esta data e a data de publicação do documento, verifica-se que, neste caso, a diferença é menor do que meio ano. Portanto, a data de 30 de novembro de 2011 é a data procurada e é retornada pela função. A frase passa a ter a seguinte redação:

"I have applied for my H1-B 3 year extension after my 6th year on 2011/11/30"

Finalmente, uma expressão como March (março), sem indicação de passado ou futuro, é normalizada pela regra de produção `month_without_year`. A heurística utilizada é buscar a ocorrência daquele mês mais próxima à data de publicação do documento. O cálculo deste mês é realizado pela função `getDateWithoutYear`, cujo pseudocódigo está no Algoritmo 2. O primeiro parâmetro de entrada desta função é a data de publicação do documento (no formato ano, mês e dia). Como segundo parâmetro de entrada, é necessário passar o dia do ano desejado, no formato mês e dia. Como o dia exato é desconhecido, é passado como dia 1 do mês. A saída é a data absoluta daquele dia do ano mais próxima à data do documento. A granularidade para esta regra de produção é mensal, o que significa que a informação sobre o dia calculado pela função `getDateWithoutYear` é descartada. Na listagem a seguir, deve-se interpretar que a função `discardDay` transforma a granularidade da data retornada pela função `getDateWithoutYear` de diária para mensal, descartando a informação do dia do mês.

$month\_without\_year \rightarrow month$
```
{discardDay(getDateWithoutYear(
    documentDate, month.value, 1));}
```

Para ilustrar a aplicação da regra de produção `month_without_year`, é utilizado o seguinte exemplo. Um documento[12] com a data de publicação de 22 de dezembro de 2011 possui a seguinte frase:

"We will spend in January if we need to."

A gramática identifica que January é gerado pela regra de produção `month_without_year`. O analisador léxico atribui ao *token* `month` o valor de 1, armazenado na variável `value`, pois o mês da expressão é janeiro (January). A ação semântica associada à regra de produção faz uma chamada à função `getDateWithoutYear`, com os argumentos `documentDate` – igual a 22 de dezembro de 2011 –, `month.value` – igual a 1, para indicar janeiro – e 1 – para indicar o primeiro dia do mês, visto que a expressão possui granularidade mensal e não indica um dia exato. A função segue os passos do Algoritmo 2, que se dedica a procurar a ocorrência de 01 de janeiro mais próxima da data de publicação do documento. Primeiramente, a função constrói a data de 01 de janeiro no mesmo ano de publicação, isto é, 01 de janeiro de 2011. A diferença desta data para a data de publicação do documento

[12] http://www.examiner.co.uk/huddersfield-town-fc/huddersfield-town-news/2011/12/22/huddersfield-town-chairman-dean-hoyle-we-will-spend-in-january-if-we-need-to-86081-29992803/

é maior do que meio ano, portanto esta candidata é descartada. Em seguida a função analisa se a data procurada está no ano anterior à data de publicação do documento e a data de 01 de janeiro de 2010 é construída; no entanto, a diferença desta data para a data de publicação do documento também é maior do que meio ano, descartando esta candidata. Por eliminação, a data procurada está no ano seguinte ao ano de publicação do documento, ou seja, 2012, resultando em 01 de janeiro de 2012. A aplicação da função `discardDay` descarta a informação de dia da data calculada e o resultado final é o mês de janeiro de 2012. A frase passa a ter a seguinte redação:

"We will spend in 2012/01 if we need to."

5.4 Extração de Fatos Temporais

Esta seção apresenta a etapa de extração de fatos temporais a partir de texto em linguagem natural na língua inglesa. O texto é analisado em busca de fatos temporais, que expressam relações entre um objeto e uma data. O resultado final desta etapa é um conjunto de fatos temporais, extraídos do texto fornecido como entrada.

O texto utilizado como entrada para esta etapa deve ter sido submetido à normalização de expressões temporais. Esta seção evidencia que a normalização prévia permite a descoberta de fatos temporais em maior quantidade e qualidade. A extração de fatos temporais funciona corretamente mesmo que o texto não tenha sido submetido à normalização de expressões temporais. No entanto, a informação temporal é perdida, neste caso.

Uma *entidade* representa um objeto, com uma descrição idealmente independente de idioma (SUCHANEK; KASNECI; WEIKUM, 2007). Uma entidade representa, por exemplo, cidades, estados e pessoas. Um *fato* é definido por um conjunto formado por entidades ligadas através de uma relação. Um fato pode ser *binário*, por conectar um sujeito a um predicado através de uma relação. Fatos podem assumir aridades maiores, relacionando três entidades (fatos ternários), quatro entidades (fatos quaternários) ou qualquer outra quantidade (fatos n-ários).

A listagem a seguir exibe um exemplo de fato. A entidade Brasília se relaciona com a entidade Brasil através da relação é capital de. A entidade à esquerda da relação é o *sujeito* e a entidade à direita é o *predicado*.

"Brasília é capital de Brasil"

Uma entidade também pode representar objetos como números, datas e outros literais. Desta forma, é possível construir um fato que descreve a população de uma cidade, por exemplo. A listagem a seguir exibe um fato no qual a entidade Brasília possui uma população de 2.500.000 habitantes.

"Brasília tem população de 2.500.000"

Um *fato temporal* é um fato cujo predicado é uma data. Um exemplo de fato temporal é apresentado na listagem a seguir. Este fato afirma que a entidade Brasília se relaciona com a data de 21 de abril de 1960 através da relação foi fundada em.

"Brasília foi fundada em 21/04/1960"

Para a representação de um fato com aridade maior do que binária, são utilizados

identificadores nos fatos. Um fato ternário, por exemplo, liga três entidades através de uma relação. Para representar este fato, dois dos argumentos são relacionados através de um fato intermediário, que recebe um identificador; o fato intermediário, por sua vez, relaciona-se com o terceiro argumento, complementando a representação da informação. Este método adotado pela abordagem EXTIO foi proposto por (SUCHANEK; KASNECI; WEIKUM, 2007).

Um exemplo de fato ternário seria: Ronaldo foi contratado pelo Real Madrid em 31/08/2002. Neste fato, existem três entidades relacionadas, e o fato apenas faz sentido quando relacionado com todas as suas entidades. Para representar este fato, é gerado um fato temporário, relacionando as entidades Ronaldo e Real Madrid. Este fato temporário recebe o identificador #1. Em seguida, este fato temporário relaciona-se com a entidade restante, que é a data de 31/08/2002.

#1: "Ronaldo foi contratado pelo Real Madrid"
#2: "#1 em 31/08/2002"

Para a representação de um fato quaternário, o mesmo método é adotado pela abordagem EXTIO. Um exemplo de fato quaternário seria expresso pela informação de que John Kennedy foi presidente dos Estados Unidos de 20/01/1961 a 22/11/1963. Neste caso, serão necessários dois identificadores adicionais para representar esta informação.

#1: "John Kennedy foi presidente dos Estados Unidos"
#2: "#1 a partir de 20/01/1961"
#3: "#2 até 22/11/1963"

Através deste método, um fato com qualquer aridade pode ser representado. Um fato n-ário é desmembrado em $n-1$ fatos intermediários com identificadores. A informação é representada sem perdas por este conjunto de fatos.

5.5 Organização de Fatos Temporais em Ontologias

Esta seção apresenta a etapa de organização dos fatos temporais extraídos em uma ontologia. O objetivo desta etapa é converter os fatos obtidos na etapa de Extração de Fatos Temporais em triplas RDF. No padrão RDF (*Resource Description Framework*) (HEATH; BIZER, 2011), uma tripla é formada por sujeito, propriedade e predicado. A tripla representa que determinado objeto (o sujeito) possui como propriedade determinado valor (o predicado). Desta forma, a tripla é utilizada para representar um fato binário. A representação de fatos ternários, quarternários e outras aridades maiores é feita através da criação de fatos intermediários com identificadores.

Nos fatos binários temporais, as propriedades são temporais e o predicado representa uma data. A data pode estar nas granularidades de dia, semana, mês, ano, década e século, em congruência com as datas normalizadas pela gramática formal para expressões temporais relativas de EXTIO.

A tripla RDF possui três componentes: o sujeito, o predicado e a propriedade que os relaciona. Estes componentes são gerados a partir do fato temporal extraído da seguinte maneira:

1. **Sujeito:** a etapa de extração de fatos temporais gera um par de sujeito e predicado em formato textual. O sujeito extraído é um texto sem significado semântico. Para

atribuir significado semântico ao sujeito, é necessário utilizar uma ferramenta de anotação semântica.

2. **Propriedade:** é o identificador da propriedade em questão.

3. **Predicado:** é a data à qual o sujeito se relaciona.

5.6 Considerações Finais

Este capítulo apresentou EXTIO, uma proposta de padronização de expressões temporais e extração de informações temporais a partir de texto em linguagem natural. Foram apresentadas a visão geral de EXTIO, seu módulo de normalização de expressões temporais relativas e seu módulo de extração e representação semântica de fatos temporais.

Neste capítulo foi apresentada a gramática formal para normalização de expressões temporais relativas a partir de texto em linguagem natural, uma das principais contribuições deste trabalho. É importante ressaltar que, até onde é conhecido, esta gramática formal é inédita, no sentido de que não foi apresentada até o momento uma gramática para normalização de expressões temporais relativas na língua inglesa.

O texto resultante da normalização de expressões temporais também pode ser utilizado por um motor de busca. Nesta situação, os documentos teriam suas expressões normalizadas antes da indexação. Como resultado, as consultas submetidas ao motor de busca seriam realizadas sobre o conteúdo dos documentos sobre as expressões temporais normalizadas. As próprias consultas também poderiam ser normalizadas, em relação ao momento de uso do motor de busca. Este possível uso da gramática formal para normalização de expressões temporais não é abordado nesta dissertação.

Este trabalho visa extrair fatos temporais a partir de texto em linguagem natural. A abordagem proposta neste capítulo poderia ser generalizada a fatos de qualquer propósito. No entanto, trabalhos como (HERMAN, 2007) indicam que a organização semântica de fatos encontra maior sucesso em iniciativas menos gerais, e mais específicas a uma certa área do conhecimento. Desta forma, este trabalho está focado apenas na extração de fatos especificamente de natureza temporal.

6 PROTÓTIPO DA ABORDAGEM EXTIO

Este capítulo apresenta detalhes de implementação aplicados com o objetivo de construir um protótipo para a abordagem EXTIO. Os componentes da abordagem são especificados, detalhando o ferramental utilizado. Quando aplicável, também são denotados módulos de outros projetos já existentes que foram empregados para a construção do protótipo.

O capítulo está organizado da seguinte forma. A seção 6.1 apresenta uma visão geral do protótipo. A seção 6.2 descreve como foi realizada a implementação da etapa de normalização de expressões temporais, envolvendo a gramática formal para normalização de expressões relativas e uma ferramenta para padronização de expressões temporais absolutas. Na seção 6.3, são explicados os passos realizados para a extração de fatos temporais. A seção 6.4 detalha como os fatos temporais extraídos são inseridos na ontologia da DBpedia, escolhida para este trabalho. Por fim, a seção 6.5 encerra o capítulo com as considerações finais.

6.1 Arquitetura

Esta seção apresenta a arquitetura do protótipo implementado para validação da abordagem EXTIO. Inicialmente, o texto é submetido à normalização de expressões temporais. Em seguida, os fatos temporais são extraídos. Finalmente, os fatos extraídos são organizados em ontologias. É importante observar que, para a validação deste trabalho, foi escolhida a extração e organização apenas de fatos temporais binários. Assim, esta implementação trabalha com fatos temporais que relacionam um objeto (o sujeito) a uma data (o predicado). Os fatos temporais com outras aridades podem ser implementados através da atribuição de identificadores a fatos intermediários. Por simplicidade, esta implementação ateve-se aos fatos temporais binários para validar a proposta.

A Figura 6.1 apresenta o funcionamento do protótipo. No passo 1, o texto é submetido à normalização de expressões temporais. Este passo é realizado para todos os documentos de texto, e é realizado uma única vez. O passo 2 consiste na extração dos fatos temporais binários a partir do texto com expressões temporais normalizadas. Este passo é realizado de forma iterativa para cada uma das 167 propriedades temporais da DBpedia apresentadas no Capítulo 4. No passo 3, os fatos temporais binários extraídos são organizados para que preencham as propriedades temporais. Este passo também é realizado de forma iterativa para as 167 propriedades temporais em estudo.

No passo de normalização de expressões temporais, são normalizadas as expressões relativas e as expressões absolutas. A extração de fatos temporais é realizada através de uma ferramenta de extração de informações a partir de texto em linguagem natural. A organização de fatos temporais é realizada através da anotação semântica e criação de

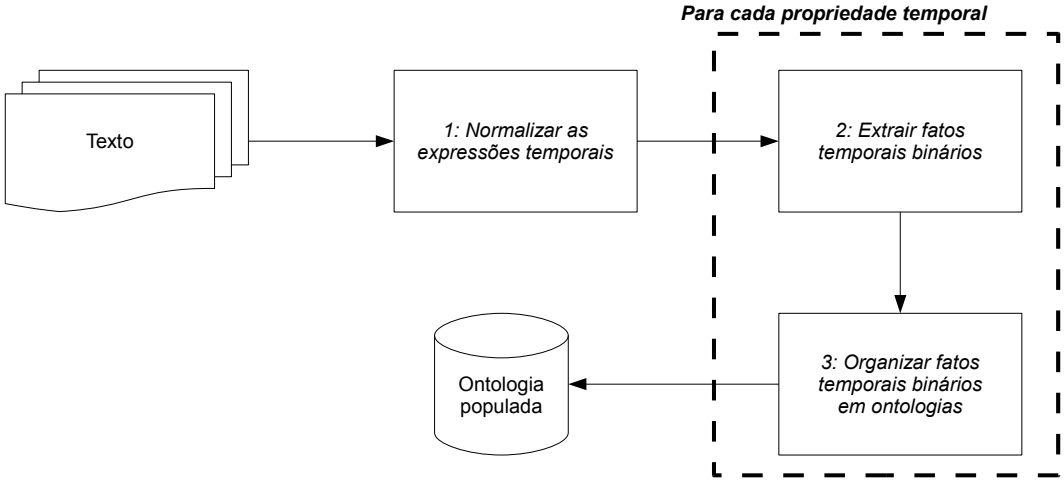


Figura 6.1: Funcionamento do protótipo de EXTIO

novas triplas RDF.

6.2 Normalização de Expressões Temporais

Esta seção descreve as ferramentas empregadas para implementação do módulo de normalização de expressões temporais de EXTIO. As expressões temporais relativas são normalizadas através da gramática formal de EXTIO. As expressões absolutas são normalizadas através de Leila (SUCHANEK; IFRIM; WEIKUM, 2006a), uma ferramenta existente de padronização de datas.

6.2.1 Gramática Formal

A gramática formal para normalização de expressões temporais de EXTIO foi implementada através das ferramentas de geração de *parsers* Lex e Yacc (LEVINE; MASON; BROWN, 1992). Para analisador léxico, foi utilizado o Flex versão 2.5.35. Para processamento da gramática, foi utilizado o Bison versão 2.4.1.

É importante ressaltar que a gramática formal proposta neste trabalho possui um objetivo diferente da utilização comum de compiladores. Ferramentas de geração de *parsers* são quase sempre empregadas para verificar se um determinado programa segue uma sintaxe definida, antes de realizar uma compilação, que consiste em aplicar regras de produção para gerar um código em linguagem compreensível por máquina (AHO et al., 2006). Quando um erro de sintaxe é encontrado, a compilação é interrompida. Na gramática formal apresentada neste trabalho, o objetivo é aplicar as regras de produção para gerar um texto com expressões temporais normalizadas, sem se preocupar com a correção da sintaxe dos documentos. Desta forma, os erros de sintaxe são ignorados pela implementação desta gramática.

Na gramática, frequentemente a resolução de expressões temporais relativas é realizada através de cálculos envolvendo datas. Por exemplo, para a expressão next Thursday (próxima quinta-feira), é necessário obter a próxima ocorrência de quinta-feira, após a data de publicação do documento, com a regra de produção `next_weekday`. Isto é feito através de funções da biblioteca Boost (KARLSSON, 2005).

A biblioteca Boost fornece implementações para diversas funções utilizadas para a resolução das expressões temporais relativas. Para a regra de produção `next_weekday`, por exemplo, é chamada a função `getNextWeekday` de Boost[1]. Para a regra de produção `last_weekday`, a função `getPreviousWeekday` também tem sua implementação dada pela biblioteca Boost[2].

6.2.2 Padronização de Datas

Dentro de um documento, uma data pode ser representada de diversas maneiras. Por exemplo, uma mesma data pode ser representada como "February 1st, 2011", "02/01/2011", "Feb 01, 2011" e de muitas outras maneiras (MANICA; DORNELES; GALANTE, 2010). É necessário, portanto, realizar uma normalização das datas absolutas dos documentos.

Diversas abordagens para padronização de datas foram propostas. Dentre elas, o trabalho desta dissertação optou por adotar a ferramenta Leila (SUCHANEK; IFRIM; WEIKUM, 2006a) para realizar a desambiguação de expressões temporais. Esta escolha foi feita devido ao formato padronizado para o qual Leila converte as datas encontradas no texto. Desta forma, a gramática formal proposta neste trabalho está focada na normalização de expressões temporais relativas, partindo do pressuposto que os diversos formatos de representação de data encontrados em um documento são padronizados pelo módulo de padronização de datas da ferramenta Leila. Este módulo é executado previamente à execução da gramática formal de normalização de expressões temporais relativas. Isto significa que o texto em linguagem natural é submetido à padronização de datas, e o texto resultante desta padronização serve como entrada para a execução da gramática formal.

O módulo de padronização de datas da ferramenta Leila converte as ocorrências temporais para o formato *@YYYY_MM_DD@* e suas variações. Isto ocorre da seguinte maneira:

- Uma data onde o dia, o mês e o ano são conhecidos é representada no formato *@YYYY_MM_DD@*. Por exemplo, a data de 31 de março de 2011 é traduzida para *@2011_03_31@*.
- Se apenas o mês e o ano são conhecidos, mas não o dia, a data é normalizada no formato *@YYYY_MM_#@*. O mês de dezembro de 2010, por exemplo, é representado como *@2010_12_#@*.
- Um ano sem nenhuma informação adicional é representado como *@YYYY_#_#@*. Por exemplo, o ano de 2009 se torna *@2009_#_#@*.
- Analogamente, décadas são representadas no formato *@YYY#_#_#@*. Por exemplo, a década de 1980 é representada como *@198#_#_#@*.
- Se apenas o dia e o mês são conhecidos, mas não o ano, a data é representada como *@#_MM_DD@*. Por exemplo, a expressão 11 de fevereiro é traduzida por *@#_2_11@*.
- Se apenas o mês é conhecido, mas não o dia nem o ano, o mês é substituído por *MONTH01* para o mês de janeiro, *MONTH02* para o mês de fevereiro e assim sucessivamente.

[1] A função da biblioteca Boost equivalente possui o nome `next_weekday`.
[2] A função da biblioteca Boost equivalente possui o nome `previous_weekday`.

O seguinte exemplo ilustra a padronização de datas absolutas de Leila. A frase a seguir possui uma expressão temporal absoluta, referente à data de 29 de junho de 1992.

"The three-month bill rate was the highest since the rate of 3.59% on June 29, 1992."

A ferramenta Leila reconhece o trecho "June 29, 1992" como uma data e a converte para a representação padronizada no formato *@YYYY_MM_DD@*. Como resultado, a frase é substituída da seguinte forma:

"The three-month bill rate was the highest since the rate of 3.59% on @1992_6_29@."

O próximo exemplo ilustra uma situação em que uma expressão temporal é normalizada através da combinação de Leila com a gramática formal proposta por este trabalho. Seja a seguinte frase, encontrada em um documento cuja data de publicação é 01 de março de 1994:

"As of Dec. 31, the bank had a Tier 1 capital ratio of 6.24% and a leverage ratio of 5.49%."

A padronização de datas da ferramenta Leila reconhece a expressão "Dec. 31" como uma data. A expressão é padronizada, de forma que a frase é substituída pela seguinte:

"As of @#_12_31@, the bank had a Tier 1 capital ratio of 6.24% and a leverage ratio of 5.49%."

Esta frase é utilizada como entrada para a gramática formal. A gramática reconhece a expressão através da regra de produção `date_without_year`. Esta regra determina que a expressão temporal relativa deve ser substituída pela ocorrência do dia 31 de dezembro mais próxima da data de publicação do documento. O resultado é a data de 31 de dezembro de 1993. Para manter a padronização proposta por Leila, a gramática formal foi implementada de forma que o resultado seja gerado no mesmo formato de data de Leila. Assim, a seguinte frase é gerada:

"As of @1993_12_31@, the bank had a Tier 1 capital ratio of 6.24% and a leverage ratio of 5.49%."

A combinação entre a ferramenta existente Leila e a gramática formal proposta neste trabalho permite que todas as expressões temporais encontradas no texto sejam normalizadas. Isto ocorre tanto com expressões temporais relativas como absolutas. Por utilizar uma ferramenta existente, este trabalho não precisou propor um método para padronização de datas, que ocorrem no texto em formatos variados e, por vezes, complexos.

6.3 Extração de Fatos Temporais

Para validação da proposta apresentada, esta seção apresenta a implementação do módulo de extração de fatos temporais de EXTIO. Por simplicidade, o protótipo realiza a extração de fatos temporais binários. Isto significa que são descobertas relações entre dois objetos, onde um destes objetos é uma data.

As relações temporais utilizadas na implementação deste trabalho são as propriedades temporais da ontologia da DBpedia (BIZER et al., 2009), listadas no Apêndice C. Cada propriedade temporal descreve uma relação entre um objeto – o sujeito da relação – e uma data – o predicado da relação. Exemplos de propriedades temporais são:

- A data de nascimento de uma pessoa. Neste caso, o sujeito da relação é a pessoa e o predicado da relação é a sua data de nascimento.
- A data de lançamento de uma obra. O sujeito da relação é a obra e o predicado da relação é a sua data de lançamento.
- A data de fundação de uma organização. O sujeito da relação é a organização e a data de fundação é o predicado.

Existem diversas abordagens para extração de fatos a partir de texto em linguagem natural em inglês. O trabalho desta dissertação optou pela ferramenta Leila (SUCHANEK; IFRIM; WEIKUM, 2006a). A implementação foi obtida na página do projeto na Internet[3].

A ferramenta Leila descobre novas ocorrências de uma relação – temporal ou não – a partir de dois insumos: um arquivo de exemplos e um *corpus* de treinamento. Um *corpus* de treinamento é um conjunto de documentos utilizado para calibrar um conjunto de regras de determinado método linguístico (JACQUEMIN, 2001). Um *corpus* de teste é usado para avaliação das regras para um determinado conjunto de termos e um determinado idioma (JACQUEMIN, 2001).

Para uma relação binária, o arquivo de exemplos contém uma lista onde cada linha contém o sujeito e o predicado de uma ocorrência daquela relação binária. A etapa de aprendizagem de Leila consiste em analisar o *corpus* de treinamento em busca de ocorrências do sujeito e do predicado da relação binária. As ocorrências das palavras dos componentes da relação em uma mesma frase são aprendidas como modelos. Finalmente, estes modelos são executados sobre o *corpus* de teste. Quando os modelos são encontrados no *corpus* de teste, novas ocorrências da relação binária são produzidas por Leila.

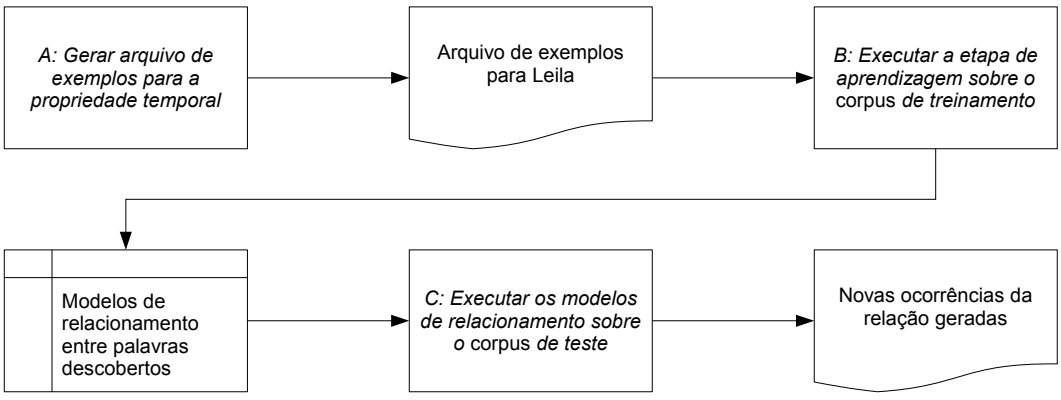


Figura 6.2: Passos da extração de fatos temporais

[3]http://www.mpi-inf.mpg.de/yago-naga/leila/

O trabalho desta dissertação utiliza as propriedades temporais como relações binárias, cabendo à ferramenta Leila a extração de novos pares destas relações. Para cada propriedade temporal, são executados os passos descritos na Figura 6.2. No Passo A, para cada propriedade temporal desejada, é criado um arquivo de exemplos. No Passo B, é executada a etapa de aprendizagem sobre o *corpus* de treinamento, que é o mesmo para todas as propriedades temporais. Como resultado, são descobertos novos modelos de relacionamento entre palavras de texto. Finalmente, o Passo C consiste em executar os modelos descobertos sobre o *corpus* de teste, que é o mesmo para todas as propriedades. Como resultado, novos pares de sujeito e predicado da propriedade temporal são descobertos. No restante desta seção, cada um dos passos da Figura 6.2 é explicado em detalhe.

6.3.1 Criação de Arquivos de Exemplo

Esta etapa consiste em gerar, para cada uma das 167 propriedades temporais em estudo – listadas no Apêndice C –, um arquivo de exemplos. Este arquivo é constituído de uma tabela de duas colunas, uma para o sujeito da relação e outra para o predicado. Como este trabalho está focado em relações temporais, o predicado é uma informação temporal.

O arquivo de exemplos é um pré-requisito da ferramenta de extração de informações Leila. Cada relação, seja ela temporal ou não, possui um arquivo de exemplos diferente. A função do arquivo de exemplos é relatar ocorrências da relação que podem ocorrer no *corpus* de treinamento. Quando o sujeito e o predicado de uma mesma linha do arquivo de exemplos aparecem em uma sentença do *corpus* de treinamento, esta sentença é analisada por sua estrutura sintática, através da gramática de dependências (SLEATOR; TEMPERLEY, 1993). Se esta estrutura sintática se repetir no *corpus* de teste, Leila identifica a descoberta de um novo par que pertence à relação em estudo.

A criação manual dos arquivos de exemplo é uma tarefa humana demorada e suscetível a erros. Seria necessário escrever manualmente arquivos de exemplo com pares de sujeito e predicado que podem ser encontrados no *corpus* de treinamento. Adicionalmente, isto teria que ser feito para todas as propriedades temporais, uma de cada vez. Para evitar esta tarefa laboriosa, o trabalho aqui apresentado optou pela geração automática de arquivos de exemplo a partir de dados existentes na base da DBpedia.

Para geração dos arquivos de exemplos, os dados da DBpedia são recuperados através de uma consulta SPARQL definida por este trabalho. A linguagem SPARQL (SEGARAN et al., 2009) permite a recuperação e a manipulação de dados de uma base em RDF, caso da DBpedia. A página do projeto da DBpedia na Internet[4] disponibiliza um *endpoint* SPARQL onde são executadas as consultas apresentadas nesta seção. A geração automática de arquivos de exemplo a partir de dados da DBpedia, através de uma consulta SPARQL, é uma contribuição deste trabalho. A recomendação da ferramenta de extração de informações (SUCHANEK; IFRIM; WEIKUM, 2006a) é criar arquivos de exemplo manualmente.

A listagem da Figura 6.3 exibe a consulta SPARQL utilizada. O campo «URI DA PROPRIEDADE» indica a URI de uma propriedade temporal da DBpedia. Por exemplo, para a data de nascimento de uma pessoa, este campo deve ser substituído por `dbpedia-owl:birthDate`. A linha 1 instrui a consulta a retornar a informação temporal (predicado) e o rótulo do objeto relacionado a ela (sujeito). A linha 2 estabelece a relação entre o objeto e a informação temporal através da propriedade em questão. A linha 3 indica que deve ser buscado o rótulo do objeto, que se relaciona com o objeto

[4] http://dbpedia.org/sparql

através da propriedade `rdfs:label`. O filtro das linhas 5 e 6 estabelece que apenas a informação temporal do período de 1980 a 2011 deve ser considerada[5]. Por fim, a linha 7 indica que devem ser retornados apenas os rótulos na língua inglesa (`en`), pois a proposta apresentada neste trabalho é voltada para o processamento de informação temporal neste idioma.

```
1 SELECT ?data ?label WHERE {
2   ?item <<URI DA PROPRIEDADE>> ?data .
3   ?item rdfs:label ?label .
4   FILTER(
5     (?data) >= "1980-01-01"^^xsd:dateTime &&
6     (?data) <= "2011-12-31"^^xsd:dateTime &&
7     langMatches(lang(?label), "en")) .
8 }
```

Figura 6.3: Consulta SPARQL para geração de arquivos de exemplos

Esta consulta é executada para todas as propriedades temporais desejadas. No caso, trata-se das 167 propriedades temporais da DBpedia detalhadas no Apêndice C. Para ilustrar, a Tabela 6.1 exibe um trecho do arquivo de exemplos gerado para a propriedade `releaseDate` (data de lançamento). Esta propriedade possui como sujeito uma obra, como um filme ou uma música, e como predicado a data de lançamento desta obra. A data está no formato *YYYY-MM-DD*, com quatro dígitos para o ano (*YYYY*), dois dígitos para o mês (*MM*) e dois dígitos para o dia (*DD*).

Tabela 6.1: Trecho do arquivo de exemplos da propriedade `releaseDate`

1993-01-01	Lake Consequence (film)
1993-01-01	Jam (album)
1993-01-01	Dr. Quinn, Medicine Woman
1993-01-01	The Wonder Years (Michael W. Smith box set)
1993-01-01	Background (album)
1993-01-01	Home Alone 2: Lost in New York (video game)
1993-01-01	Blinky Bill (TV Series)
1993-01-01	The Terminator: Rampage
1993-01-01	Ken's Labyrinth
1993-01-01	Aitebar
1993-01-02	Simon the Sorcerer
1993-01-03	Star Trek: Deep Space Nine
1993-09-03	Kalifornia

É importante ressaltar que a Tabela 6.1 exibe somente um trecho do arquivo de exemplos gerado para a propriedade temporal em estudo. O tamanho do arquivo de exemplos, em linhas, é igual à contagem de ocorrências da propriedade temporal no período de tempo contemplado. No caso da propriedade `releaseDate`, o arquivo de exemplos possui 129.325 linhas. O Apêndice C exibe a listagem completa do número de linhas do arquivo de exemplos de cada propriedade temporal (que é igual ao número de ocorrências da propriedade).

[5]Em congruência com a Seção 4.2.

6.3.2 Etapa de Aprendizagem

Este passo consiste em executar a etapa de aprendizagem da ferramenta de extração de informações. Esta etapa é executada para cada uma das propriedades temporais em estudo. Os insumos de entrada são o arquivo de exemplos da propriedade temporal em questão e o *corpus* de treinamento. Como saída, são produzidos os modelos de relacionamento entre palavras para esta propriedade temporal.

A ferramenta Leila analisa os documentos do *corpus* de treinamento pela sua estrutura sintática. Cada documento é separado em frases. Em seguida, a ferramenta submete cada frase à gramática de dependências (SLEATOR; TEMPERLEY, 1993). Como resultado, é produzida, para cada frase, uma estrutura linguística chamada *ligação em cadeia*. Uma ligação em cadeia é um grafo planar conexo não-direcionado, onde as palavras são os nós do grafo e as arestas são rotuladas com o relacionamento entre as palavras.

Para a ferramenta Leila, uma determinada relação r é *expressa* por uma ligação em cadeia quando duas entidades da frase produtora da ligação em cadeia estão em uma linha do arquivo de exemplos da relação r. Quando isto acontece, a ferramenta Leila gera um *modelo* para r. Um modelo é uma ligação em cadeia na qual as entidades pertencentes à relação r são substituídas por espaços reservados. A etapa de aprendizagem tem o objetivo de encontrar todos os modelos para a relação. Uma relação pode ter diversos modelos. É possível também que a etapa de aprendizagem não encontre, para uma determinada relação, nenhum modelo.

O seguinte exemplo ilustra a aplicação da ferramenta de extração de informações Leila às propriedades temporais. Neste exemplo, a seguinte frase foi extraída do *corpus* de treinamento:

Kalifornia was released on @1993_09_03@.

A data absoluta de 03 de setembro de 1993 está normalizada na representação padronizada. O primeiro passo da ferramenta Leila é gerar as ligações em cadeia. A seguinte listagem exibe a ligação em cadeia produzida para esta frase.

```
    +--------------------------Xp--------------------------+
    +----Wd----+---Ss--+---Pv---+--MVp-+----Jp---+         |
    |          |       |        |      |         |         |
LEFT-WALL Kalifornia was.v released.v on @1993_09_03@[?].n .
```

O passo seguinte é analisar todas as ligações em cadeia para todas as relações desejadas – no caso, todas as propriedades temporais em estudo. A Tabela 6.1 apresenta um trecho do arquivo de exemplos da propriedade `releaseDate` (data de lançamento). Nesta tabela verifica-se que a relação possui um par no qual o sujeito é o objeto `Kalifornia` e o predicado é a data `@1993_09_03@`. Com esta informação, a ferramenta Leila conclui que a relação `releaseDate` é expressa por esta ligação em cadeia. Assim, é gerado um modelo, apresentado na listagem a seguir. Neste modelo, os espaços reservados são representados por `X` (para o sujeito) e `Y` (para o predicado).

```
    +--------------------------Xp--------------------------+
    +----Wd----+---Ss--+---Pv---+--MVp-+----Jp---+         |
    |          |       |        |      |         |         |
LEFT-WALL      X     was.v released.v on         Y    [?].n .
```

Para otimização do processo, a construção das ligações em cadeia de cada frase é realizada apenas uma vez. Não é necessário reconstruir as ligações em cadeia para cada propriedade temporal. Isto pode ser feito *a priori*, ou apenas no processamento da primeira propriedade temporal.

6.3.3 Descoberta de Novos Pares da Relação Temporal

Este passo consiste em executar a ferramenta de extração de informações sobre o *corpus* de teste. Esta etapa é executada para cada propriedade temporal em estudo. Os insumos de entrada são os modelos da propriedade temporal em questão e o *corpus* de teste. Como saída, são gerados novos pares da propriedade temporal.

Analogamente ao passo anterior com o *corpus* de treinamento, a ferramenta Leila analisa os documentos do *corpus* de teste pela sua estrutura sintática, gerando as ligações em cadeia para todas as frases. O processo é otimizado através da geração das ligações em cadeia apenas uma vez. Não há necessidade de reconstruir as ligações em cadeia para cada propriedade temporal.

Os modelos de cada propriedade temporal (gerados no passo anterior) são comparados às ligações em cadeia do *corpus* de teste. A ferramenta Leila considera que um modelo *combina* com uma ligação em cadeia quando existe similaridade entre o modelo e a ligação em cadeia. O método utilizado pela ferramenta Leila permite que um modelo combine com uma ligação em cadeia mesmo quando não há correspondência exata de substantivos, adjetivos e outras estruturas linguísticas. Este método não faz parte do escopo deste trabalho, e é explicado em artigos sobre a ferramenta Leila (SUCHANEK; IFRIM; WEIKUM, 2006a) (SUCHANEK; IFRIM; WEIKUM, 2006b).

Se um modelo da relação r combina com uma ligação em cadeia, a ferramenta Leila produz para r o par de palavras que a ligação em cadeia possui no lugar dos espaços reservados. Assim, conclui-se que o par de palavras está relacionado pela propriedade temporal em questão.

O seguinte exemplo ilustra esta etapa. É dada continuidade ao exemplo iniciado na subseção 6.3.2. A seguinte frase foi extraída do *corpus* de teste:

Robocop was released on @1987_07_17@.

A data absoluta de 17 de julho de 1987 está normalizada na representação padronizada. O primeiro passo da ferramenta Leila é gerar as ligações em cadeia. A seguinte listagem exibe a ligação em cadeia produzida para esta frase.

```
    +-----------------------Xp--------------------------+
    +---Wd---+--Ss--+---Pv---+--MVp-+----Jp---+         |
    |        |       |        |      |          |         |
LEFT-WALL Robocop was.v released.v on @1987_07_17@[?].n .
```

A tarefa seguinte da ferramenta Leila é comparar esta ligação em cadeia com todos os modelos produzidos para todas as relações temporais. Neste momento, a ferramenta descobre que esta ligação em cadeia combina com o modelo produzido para a relação `releaseDate` (listado na subseção 6.3.2). A entidade Robocop corresponde ao espaço reservado `X`, enquanto a data @1987_07_17@ corresponde ao espaço reservado `Y`. Assim, a ferramenta Leila produz um novo par para a relação `releaseDate`, onde Robocop é o sujeito (espaço reservado `X`) e @1987_07_17@ é o predicado (espaço reservado `Y`).

Este processo é utilizado para dar origem a novos pares da relação temporal. O processo é repetido para cada uma das propriedades temporais estudadas neste trabalho. Esta etapa não organiza os pares da propriedade temporal em uma estrutura de dados, apenas produz os pares em formato textual para cada propriedade temporal em estudo.

6.4 Organização de Fatos Temporais em Ontologias

Esta seção apresenta a etapa de organização dos fatos temporais extraídos em uma ontologia. O objetivo desta etapa é converter os fatos obtidos na etapa de Extração de Fatos Temporais em triplas RDF. Para a implementação apresentada neste capítulo, a ontologia da DBpedia é utilizada como estrutura de dados a ser populada. A implementação é feita desta maneira para que as triplas RDF possam ser incorporadas à base de dados da DBpedia. Assim, a realização de consultas sobre a base de dados da DBpedia passa a retornar, além da informação existente na Wikipedia, os fatos temporais extraídos de qualquer conjunto de documentos em linguagem natural. Esta etapa é realizada de forma iterativa para cada uma das propriedades temporais da DBpedia.

O sujeito do fato temporal extraído é um texto sem significado semântico. Como é utilizada a estrutura de ontologias da DBpedia para organização dos fatos temporais, é necessário converter o sujeito extraído em uma entidade da base de dados da DBpedia. Para isto, a abordagem EXTIO utiliza a ferramenta DBpedia Spotlight (MENDES et al., 2011). Esta ferramenta recebe como entrada uma palavra ou expressão e retorna a URI de um recurso da DBpedia equivalente à expressão. O trabalho aqui proposto utiliza a ferramenta DBpedia Spotlight para fornecer significado semântico ao sujeito em formato textual. Desta forma, a partir de uma palavra ou expressão extraída do texto, o DBpedia Spotlight é utilizado para retornar o objeto da DBpedia equivalente.

A propriedade temporal é identificada através da URI que a propriedade possui na ontologia da DBpedia.

O predicado do fato temporal é uma data no formato *@YYYY_MM_DD@*. Para ser inserida na estrutura de ontologias da DBpedia, ela precisa ser convertida para um formato compatível. Conforme descrito na seção 4.1, o domínio de uma propriedade temporal da DBpedia pode ter diversos tipos diferentes, como o tipo *date* de XML Schema[6]. Para criação do predicado da tripla RDF, a data é convertida do formato *@YYYY_MM_DD@* para o tipo da propriedade temporal através de expressões regulares.

O seguinte exemplo ilustra esta etapa. É dada continuidade ao exemplo iniciado na seção 6.3. No exemplo, para a propriedade temporal `releaseDate` (data de lançamento), foi extraído do texto um fato temporal que possui o sujeito Robocop e o predicado @1987_07_17@. Isto significa que a obra Robocop foi lançada no dia 17 de julho de 1987.

Para organização deste fato temporal, é gerada uma tripla RDF cuja listagem está na Figura 6.4. Para obtenção do sujeito, o texto Robocop é submetido à ferramenta DBpedia Spotlight. Como resultado, é obtida a URI da linha 1 da listagem. A propriedade `releaseDate` possui a URI `http://dbpedia.org/ontology/releaseDate`. Esta URI está na linha 2 da listagem. Para obtenção do predicado, a data de @1987_07_17@ deve ser convertida para o tipo *date* de XML Schema, que é o domínio da propriedade `releaseDate`. O resultado está na linha 3 da listagem. A listagem está no formato de serialização *N-Triples* (HEATH; BIZER, 2011).

[6] http://www.w3.org/2001/XMLSchema#date

```
1 <http://dbpedia.org/resource/RoboCop>
2 <http://dbpedia.org/ontology/releaseDate>
3 "1987-07-17"^^<http://www.w3.org/2001/XMLSchema#date> .
```

Figura 6.4: Tripla RDF que representa um fato extraído de texto

Esta tripla RDF é incorporada à base de dados da DBpedia. Para obter uma base de dados que integrasse os dados da DBpedia provenientes da Wikipedia com os dados provenientes da extração de fatos temporais, a implementação deste trabalho realizou a configuração de um servidor de RDF. Um servidor de RDF permite que uma base de dados em RDF seja disponibilizada, usualmente através da Internet ou de uma intranet, para que sejam realizadas consultas sobre os dados. Para a implementação deste trabalho, foram testados os servidores de RDF Virtuoso (ERLING; MIKHAILOV, 2007) e Joseki[7].

6.5 Considerações Finais

Este capítulo descreveu como foi implementada a abordagem EXTIO proposta neste trabalho. Para obter o resultado final desejado por EXTIO, foi utilizada uma combinação entre ferramentas existentes, explicadas neste capítulo, e contribuições apresentadas neste trabalho. Como resultado, são normalizadas as ocorrências temporais, para em seguida ser realizada a extração de fatos temporais de texto em linguagem natural e a organização destes fatos na ontologia da DBpedia.

[7] http://www.joseki.org/

7 EXPERIMENTOS

Este capítulo apresenta os experimentos realizados com o objetivo de determinar a qualidade dos resultados de EXTIO para a normalização de expressões temporais. São realizados experimentos para mensurar a qualidade da abordagem proposta neste trabalho nos sentidos de eficácia e de tempo de processamento.

O capítulo está organizado da seguinte forma. A seção 7.1 explica como foram configurados os experimentos. A seção 7.2 descreve os resultados obtidos a partir dos experimentos. Os resultados dos experimentos são analisados na seção 7.3.

7.1 Configurações dos Experimentos

Esta seção descreve o projeto da validação experimental. A seção 7.1.1 descreve as métricas utilizadas. A seção 7.1.2 apresenta as configurações de hardware e software empregadas. A metodologia utilizada é descrita na seção 7.1.3. A seção 7.1.4 explica as coleções de teste utilizadas.

7.1.1 Métricas de Avaliação

Os experimentos realizados utilizaram as seguintes métricas para avaliação de resultados: precisão, revocação, *F-measure*, tempo de processamento e Teste-T.

Em um contexto de Recuperação de Informações, a precisão é medida pelo quociente de documentos relevantes recuperados dividido pelo total de documentos recuperados (BAEZA-YATES; RIBEIRO-NETO, 1999). Quando aplicada à proposta de normalização de expressões, como no caso dos experimentos realizados, a precisão é medida pelo quociente de expressões temporais normalizadas corretamente dividido pelo total de expressões temporais normalizadas (correta ou incorretamente). A precisão pode ser calculada pela fórmula:

$$P = \frac{|Corretas \cap Normalizadas|}{|Normalizadas|}$$

onde $Corretas$ denota o conjunto de ocorrências normalizadas corretamente, $Normalizadas$ denota o conjunto de ocorrências normalizadas e $|C|$ denota o número de elementos em um conjunto C.

A revocação é medida pelo quociente de documentos relevantes recuperados dividido pelo total de documentos relevantes existentes na coleção (BAEZA-YATES; RIBEIRO-NETO, 1999). Quando aplicada à proposta de normalização de expressões, a revocação é medida pelo quociente de expressões temporais normalizadas corretamente dividido pelo total de expressões temporais existentes. A revocação pode ser calculada pela fórmula:

$$R = \frac{|Corretas \cap Normalizadas|}{|Corretas|}$$

A *F-measure* (RIJSBERGEN, 1979) é uma média ponderada harmônica entre precisão e revocação, que permite atribuir pesos diferentes para as duas métricas. A *F-measure* é calculada pela fórmula:

$$F = \frac{(1 + \alpha) \times P \times R}{\alpha \times P + R}$$

onde P é a precisão e R é a revocação. Para estes experimentos, foi atribuído o mesmo peso para precisão e revocação, através do valor de $\alpha = 1$. A *F-measure*, quando emprega pesos idênticos para precisão e revocação, também é chamada de *F1-measure*. A *F1-measure* é interpretada como uma média ponderada da precisão e da revocação, onde ambas possuem o mesmo peso. Neste trabalho, o termo *F-measure* refere-se à métrica *F1-measure*, ou seja, com o mesmo peso atribuído à precisão e à revocação.

O tempo de processamento é medido através do tempo (em segundos ou milissegundos) de execução de determinada tarefa. Quando aplicada a uma proposta de normalização de expressões, o tempo de processamento mede o tempo necessário para normalizar todas as expressões temporais de um documento ou conjunto de documentos.

Para comparar diferentes métodos de Recuperação de Informações, um teste estatístico garante a promoção de um método verdadeiramente melhor que outro. O teste estatístico ajuda a evitar conclusões obtidas ao acaso (SMUCKER; ALLAN; CARTERETTE, 2007). Um teste poderoso permite a detecção de melhorias significativas mesmo quando estas melhorias forem pequenas. Um teste preciso relata significância apenas quando realmente existe.

O teste estatístico utilizado neste trabalho é o Teste-T de Student (FISHER, 1937). Este teste parte de uma hipótese nula que pressupõe que as duas médias não possuem diferenças estatísticas significativas. Os experimentos deste trabalho utilizaram o Teste-T bicaudal com limiar de significância estatística de $\alpha = 0,05$. Para o cálculo do Teste-T, foram considerados os valores de precisão, revocação e *F-measure* individuais para cada um dos documentos utilizados no experimento. Se o valor do Teste-T for menor que α, então a hipótese nula é refutada, ou seja, é possível afirmar, com 95% de certeza (95% = $1 - \alpha$), que existe diferença significativa entre os desempenhos dos métodos de normalização de expressões temporais.

7.1.2 Configurações de Ambiente

A gramática formal para normalização de expressões temporais de EXTIO foi implementada na linguagem C, utilizando Lex/Yacc. Para cálculos envolvendo datas (necessários para a resolução de expressões temporais relativas), foi criado um módulo em C++ para chamar funções da biblioteca Boost (KARLSSON, 2005).

Os experimentos foram executados em um ambiente Linux (Ubuntu 10.04) virtualizado através de VMWare sobre um ambiente com Windows (Vista) como hospedeiro. A máquina utilizada possui processador Intel Pentium Dual-Core T4200 de 2 GHz, e 3 GB de memória RAM foram alocados para a máquina virtual.

7.1.3 Metodologia

Os experimentos foram executados usando duas implementações: (i) GUTime (VERHAGEN et al., 2005), utilizado como *baseline*, conforme apresentado na seção 3.1;

e (ii) a gramática formal para normalização de expressões temporais relativas de EXTIO, conforme explicado na seção 5.3.

7.1.4 Descrição das Coleções

Os experimentos realizados utilizaram notícias extraídas da coleção da LATimes[1] de 1994. De acordo com um identificador existente nos próprios documentos, os experimentos utilizaram notícias das seguintes categorias: *Business*, *Metro*, *Sports* e *World Report*.

Para obter a eficácia da normalização de expressões temporais relativas de EXTIO e do *baseline*, é necessário, *a priori*, determinar todas as ocorrências de expressões temporais relativas na coleção de documentos. Isto consiste em uma etapa manual, em que todos os documentos são lidos, e todas as expressões temporais relativas encontradas são anotadas, gerando um gabarito. Cada um dos métodos de normalização é executado sobre a coleção de documentos, e os resultados produzidos são comparados com o gabarito.

Existe uma dificuldade em gerar o gabarito utilizado, pois é necessário identificar manualmente todas as expressões temporais relativas existentes na coleção. Por este motivo, os experimentos para cálculo da eficácia foram realizados sobre uma amostra desta coleção. Para elaboração desta amostra, buscou-se respeitar a proporção de cada tipo de notícia encontrado na coleção completa.

A Tabela 7.1 especifica a quantidade de notícias selecionadas da coleção da LATimes 1994 para a realização destes experimentos. Nesta tabela são exibidas as categorias de notícias selecionadas. Ao todo, 128 notícias foram selecionadas para os experimentos.

Tabela 7.1: Notícias selecionadas da coleção LATimes para realização dos experimentos

Categoria	**Analisadas**
Business	30
Metro	39
Sports	30
World Report	29
Total	128

Para a medição do tempo de processamento, não é necessária a geração prévia de um gabarito. É suficiente medir o tempo necessário para a execução de EXTIO e do *baseline*. Portanto, foi possível usar um conjunto maior de documentos para a medição do tempo de processamento. Trata-se de um superconjunto do conjunto de documentos utilizados para obter a eficácia da normalização de expressões temporais relativas.

7.2 Experimentos Realizados

Esta seção descreve os experimentos realizados e os resultados obtidos. A subseção 7.2.1 descreve os experimentos realizados para avaliar a eficácia de EXTIO e do *baseline*, em termos de precisão, revocação e *F-measure*. A subseção 7.2.2 descreve os experimentos para avaliação do tempo de processamento.

7.2.1 Eficácia da Normalização de Expressões Temporais

Este experimento analisou a eficácia da aplicação de EXTIO e do *baseline* às notícias escolhidas a partir da coleção LATimes 1994. O objetivo deste experimento é verificar se

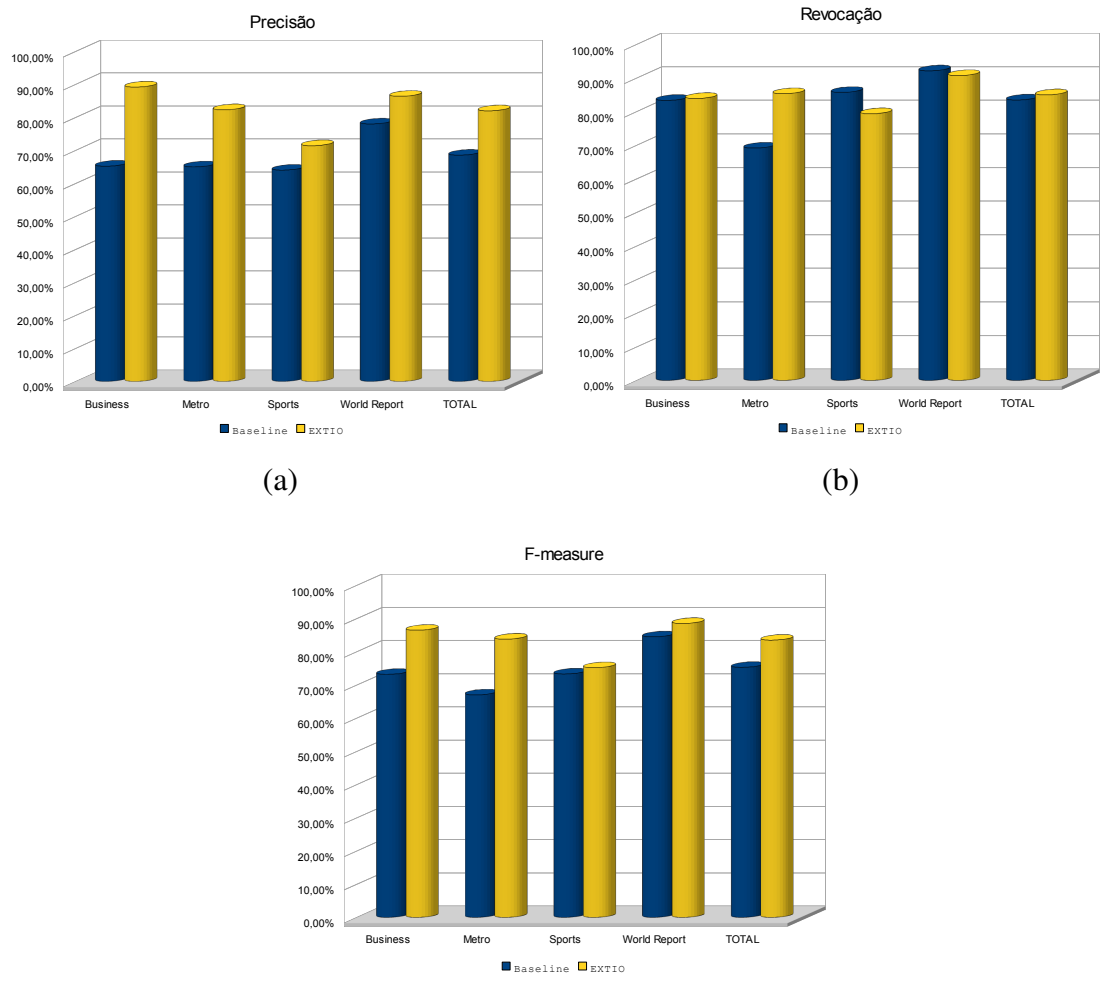
[1] http://www.latimes.com

EXTIO apresenta uma melhoria significativa ou não em relação ao *baseline*, ao realizar a tarefa de normalização de expressões temporais.

O procedimento empregado é descrito da seguinte forma. EXTIO e o *baseline* são aplicados sobre os documentos da coleção de teste para que seja realizada a normalização de expressões temporais. Em seguida, as expressões normalizadas são comparadas com o gabarito. A partir da contagem de expressões normalizadas correta e incorretamente e das expressões não normalizadas por cada uma das abordagens, são calculados os valores de precisão, revocação e *F-measure* para EXTIO e o *baseline*.

A primeira etapa é a geração do gabarito. Cada notícia foi manualmente verificada em busca de expressões temporais relativas. Isto consiste em ler as notícias por completo, identificar cada trecho de texto que corresponde a uma expressão temporal relativa e calcular a data para a qual a expressão deve ser normalizada. Como resultado desta etapa manual, foi constatado que as 128 notícias selecionadas possuíam um total de 585 expressões temporais relativas.

Em seguida, EXTIO e o *baseline* foram executados sobre as notícias selecionadas. Os resultados foram avaliados através das métricas de precisão, revocação e *F-measure*. A Figura 7.1 apresenta gráficos com os resultados dos experimentos realizados. Os números são exibidos na Tabela 7.2.



(a) (b)

(c)

Figura 7.1: Precisão, revocação e *F-measure* de EXTIO e do *baseline*

A avaliação indica que EXTIO supera o *baseline* nas métricas de precisão, revocação e *F-measure*. EXTIO obteve 82,00% de precisão e 83,47% de *F-measure*. O desempenho do *baseline* foi de 68,54% em precisão e de 75,27% em *F-measure*. Na comparação através de Teste-T entre os resultados, $p = 0,00001$ para a precisão e $p = 0,00019$ para a *F-measure*. A revocação foi de 84,99% para o EXTIO e de 83,45% para o *baseline*. Na comparação através do Teste-T entre os resultados de revocação, $p = 0,47$.

Tabela 7.2: Precisão, revocação e *F-measure* do EXTIO e do *baseline*

Categoria		**Business**	**Metro**	**Sports**	**World Report**	**Total**
Ocorrências		130	129	166	160	585
Corretos	*Baseline*	75	65	96	117	353
	EXTIO	**99**	**93**	**100**	**127**	**419**
Incorretos	*Baseline*	40	35	54	33	162
	EXTIO	**12**	**20**	**40**	**20**	**92**
Não Tratados	*Baseline*	15	29	16	10	70
	EXTIO	19	**16**	26	13	74
Precisão	*Baseline*	65,22%	65,00%	64,00%	78,00%	68,54%
	EXTIO	**89,19%**	**82,30%**	**71,43%**	**86,39%**	**82,00%**
Revocação	*Baseline*	83,33%	69,15%	85,71%	92,13%	83,45%
	EXTIO	**83,90%**	**85,32%**	79,37%	90,71%	**84,99%**
F-measure	*Baseline*	73,17%	67,01%	73,28%	84,48%	75,27%
	EXTIO	**86,46%**	**83,78%**	**75,19%**	**88,50%**	**83,47%**

O Teste-T indica que esta é uma melhoria significativa de um trabalho em relação ao outro, pois $p < 0,05$ para as métricas, exceto para a revocação. Estes números mostram que o método inédito proposto neste trabalho, de utilização de uma gramática formal, é adequado e produz bons resultados para a normalização de expressões temporais relativas, superando o *baseline* empregado para comparação. A seguir é analisado o desempenho do EXTIO e do *baseline* nestes experimentos, em cada uma das categorias de notícias utilizadas.

Em duas das quatro categorias analisadas, o desempenho do EXTIO foi superior ao do *baseline* em precisão e *F-measure*, com aprovação no Teste-T. Na categoria *Business* (notícias sobre economia), o EXTIO obteve 89,19% de precisão, contra 65,22% do *baseline*. O Teste-T apresentou valor de $p = 0,00018$. Nesta mesma categoria, a *F-measure* de EXTIO foi de 86,46%, enquanto a do *baseline* foi de 73,17%. O Teste-T apresentou valor de $p = 0,01824$. Estes números indicam que EXTIO apresentou melhoria significativa sobre o *baseline* na categoria *Business*, pois $p < 0,05$ tanto para a precisão quanto para a *F-measure*.

Na categoria *Metro* (notícias locais), EXTIO apresentou precisão de 82,30%, enquanto o *baseline* obteve 65,00%, com o Teste-T apontando o valor de $p = 0,00170$. Na *F-measure*, EXTIO obteve 83,78% nesta categoria contra 67,01% do *baseline*, e o Teste-T apresentou valor de $p = 0,00164$. Estes números indicam que EXTIO apresentou melhoria significativa sobre o *baseline* na categoria *Metro*, pois $p < 0,05$ tanto para a precisão quanto para a *F-measure*.

Nestas duas categorias, EXTIO demonstrou excelentes resultados, mostrando-se como melhoria significativa em relação ao *baseline*. Algumas diferenças entre EXTIO e o *baseline* explicam este sucesso. A primeira delas está na resolução de algumas expressões

temporais que são exclusividade de EXTIO, como as expressões "years ago". Por exemplo, na seguinte frase:

"There is not nearly as much asbestos in the Valley as there was five years ago."

a expressão "five years ago" é normalizada apenas pelo EXTIO, e não pelo *baseline*. Outra diferença significativa entre EXTIO e o *baseline* está no tratamento dado a expressões temporais relativas sem indicação de futuro ou passado. Isto ocorre, por exemplo, na seguinte frase:

"Lawrason weeks ago announced his plans to seek another term as Moorpark mayor, and said Monday that he will keep those plans."

Nesta frase, a expressão temporal relativa "Monday" (segunda-feira) não possui indicação de passado ou futuro. Nesta situação, EXTIO busca a ocorrência temporal da expressão relativa mais próxima à data de publicação do documento. O *baseline*, por sua vez, busca resolver o tempo verbal para decidir se a ocorrência está no passado ou no futuro. Neste exemplo, a notícia é datada de 01 de março de 1994, uma terça-feira. A estratégia escolhida por EXTIO para a resolução desta expressão é encontrar a segunda-feira mais próxima da data de publicação do documento. Neste caso, trata-se do dia anterior, 28 de fevereiro de 1994. Já o *baseline* busca resolver esta expressão através do tempo verbal da frase. No entanto, devido à expressão "he will keep", o *baseline* conclui que a frase está no futuro, e calcula que "Monday" refere-se à próxima ocorrência de uma segunda-feira depois da data de publicação do documento. Como resultado, o *baseline* erroneamente normaliza a expressão para a data 07 de março de 1994. Este exemplo ocorreu repetidas vezes nas notícias destas categorias, resultando em vantagem para o EXTIO sobre o *baseline*.

Nas categorias *Business* e *Metro*, também a revocação foi superior para o EXTIO em relação ao *baseline*. Na categoria *Business*, esta métrica foi de 83,90% para o EXTIO e de 83,33% para o *baseline*. Na categoria *Metro*, o EXTIO obteve 85,32% e o *baseline* obteve 69,15%. No entanto, em ambas as categorias, o Teste-T apontou valor de $p > 0,05$ na comparação entre EXTIO e o *baseline*. Isto significa que não é possível afirmar que EXTIO apresentou uma melhoria significativa de desempenho em relação ao *baseline* na revocação.

Na categoria *World Report* (notícias do mundo), tanto EXTIO quanto o *baseline* apresentaram bom desempenho. Nesta categoria, EXTIO obteve 86,39% de precisão, contra 78,00% do *baseline*, com o Teste-T obtendo o valor de $p = 0,29$. A revocação ficou em 90,71% para o EXTIO e em 92,13% para o *baseline*, e o Teste-T obteve $p = 0,73$. A *F-measure* do EXTIO foi de 88,50%, e a do *baseline*, 84,48%. Os números de EXTIO apresentaram-se positivos, com alta precisão, revocação e *F-measure*. No entanto, o fato do Teste-T apontar o valor de $p > 0,05$ para as três métricas que não é possível afirmar que existe diferença significativa entre as duas abordagens para os documentos desta categoria.

O pior desempenho de EXTIO entre todas as categorias foi na *Sports* (notícias sobre esporte), com 71,43% de precisão, 79,37% de revocação e 75,19% de *F-measure*. O Teste-T para a revocação calculou $p > 0,05$ em todas as métricas, o que significa que não houve diferença significativa de desempenho na comparação com o *baseline*. Esta baixa revocação ocorre devido ao surgimento de expressões temporais relativas especí-

ficas de notícias esportivas, como *"last season"* (na temporada anterior). O significado desta expressão temporal é variável de acordo com o contexto, podendo representar diferentes períodos temporais de acordo com o esporte. Para o basquete, por exemplo, uma temporada costuma decorrer de outubro de um ano a junho do ano seguinte, enquanto que para a Fórmula 1 a temporada vai de março a novembro de um mesmo ano. O correto tratamento desta expressão temporal – e de outras do assunto do esporte – apresenta regras complexas que não foram abordadas neste trabalho.

7.2.2 Tempo de Processamento

Nesta etapa dos experimentos, EXTIO e o *baseline* foram executados em modo de linha de comando, para evitar a influência da execução de interfaces gráficas sobre o tempo de processamento. O tempo foi medido através do comando de Linux `time`.

O tempo medido englobou todas as etapas necessárias para a execução dos dois métodos de normalização de expressões temporais relativas, desde a entrada do arquivo em formato HTML ou XML até a geração do arquivo de saída com as expressões temporais absolutas. Os experimentos foram executados para os 285 documentos selecionados. Para evitar que processos de sistema operacional pudessem impactar no tempo de execução, cada experimento foi executado três vezes com o EXTIO e três vezes com o *baseline*.

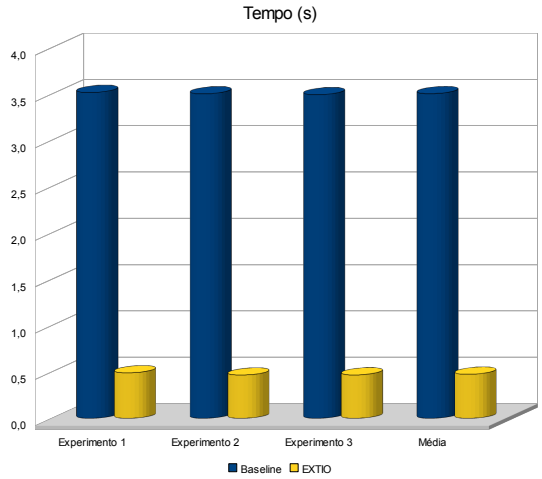


Figura 7.2: Tempo de processamento médio (em segundos) de EXTIO e do *baseline*

O gráfico da Figura 7.2 apresenta os resultados de cada um dos três experimentos realizados e a média de tempo destes experimentos. Os números são exibidos na Tabela 7.3. É visível que, no quesito tempo de processamento, o desempenho do trabalho apresentado nesta dissertação foi superior ao *baseline*, pois o EXTIO obteve tempos de execução menores. O tempo de execução médio do EXTIO para um documento foi de 0,47579 segundos, enquanto o tempo médio do *baseline* foi de 3,50433 segundos. O Teste-T para comparação dos resultados de tempo de execução resultou em $p \sim 10^{-311}$. Estes números evidenciam que o tempo de execução foi significativamente melhor – ou seja, menor – no trabalho apresentado nesta dissertação.

7.3 Análise Geral dos Experimentos

Este capítulo apresentou experimentos para medir, de forma comparativa, o desempenho de EXTIO e um *baseline* para normalizar expressões temporais relativas. Os experi-

Tabela 7.3: Tempo de processamento médio (em segundos) de EXTIO e do *baseline*

	Exp. 1	**Exp. 2**	**Exp. 3**	**Média**
EXTIO	0,49140	0,46695	0,46902	**0,47579**
Baseline	3,51744	3,50263	3,49291	**3,50433**

mentos foram realizados para comparar o desempenho em dois sentidos: eficácia e tempo de processamento.

Como resultado, verificou-se que, para um mesmo conjunto de documentos, EXTIO obteve números maiores de precisão e *F-measure* do que o *baseline*. Ao mesmo tempo, EXTIO obteve números menores de tempo de processamento do que o *baseline*. Desta forma, o trabalho apresentado nesta dissertação superou o *baseline* tanto no quesito eficácia quanto no quesito tempo de processamento.

É necessário frisar que o *baseline*, além de normalizar expressões temporais relativas, também extrai algumas outras categorias de normalização temporal não incluídas neste trabalho. Por exemplo, na seguinte frase:

"The automobile is a century old and showing no signs of putting on the brakes."

a expressão *"a century"* (um século) é identificada pelo *baseline* e extraída como um período de 100 anos. Este tipo de informação sobre períodos temporais não faz parte do escopo de EXTIO e, portanto, não foi incluída nas medições dos experimentos realizados. Devido ao processamento destas expressões, o *baseline* apresenta, em trabalhos como (VERHAGEN et al., 2005), números de precisão e revocação melhores do que os obtidos nos experimentos deste capítulo.

No trabalho realizado em (GALLINA; GALANTE; DORNELES, 2011), os autores concluíram que, para melhorar a precisão do módulo de normalização de expressões temporais relativas de EXTIO, seria interessante analisar os tempos verbais envolvidos nas frases. No entanto, conforme visto nos experimentos, o EXTIO mostrou números superiores ao *baseline* nas categorias *Business* e *Metro*. Em parte, estes números foram obtidos justamente pelo fato de EXTIO não fazer o tratamento considerando o tempo verbal, ao contrário do *baseline*. Assim, após realizar estes experimentos, o trabalho aqui presente concluiu que a política de resolução de expressões temporais aplicando o tempo verbal, como faz o *baseline*, não necessariamente é a melhor estratégia. Portanto, para futuras versões de EXTIO, a análise de tempos verbais não será realizada, e a implementação será mantida desta maneira.

Em alguns documentos, verificou-se que ainda existem expressões temporais tratadas pelo *baseline* que não são tratadas pelo EXTIO. A revocação, nas categorias *Sports* e *World Report*, foi inferior no EXTIO em relação ao *baseline*, apesar do Teste-T não indicar diferença significativa. Um exemplo é a expressão *"this morning"* (esta manhã), não constituinte da gramática formal apresentada neste trabalho. Isto indica que há espaço para tratar mais expressões temporais na gramática formal de EXTIO.

8 CONCLUSÃO

Este trabalho descreveu EXTIO, uma abordagem para extração e organização semântica de fatos temporais extraídos de texto em linguagem natural em inglês. Previamente à extração de informações, esta abordagem realiza a normalização de expressões temporais dos documentos, tornando explícita a informação temporal que encontra-se implícita no texto na forma de expressões temporais relativas. A principal contribuição desta abordagem é o processamento adequado da informação temporal disponível em texto, incluindo um processo de normalização de expressões temporais relativas.

A abordagem foi avaliada através de experimentos de qualidade dos resultados e tempo de processamento de EXTIO para a normalização de expressões temporais. Os experimentos foram realizados sobre um *corpus* de documentos reais, comparando com a ferramenta de normalização de expressões temporais relativas GUTime. Os resultados indicam que EXTIO é significativamente melhor na eficácia da normalização (precisão e *F-measure*) e possui tempo de processamento menor do que GUTime para os mesmos documentos.

Esta dissertação inclui a definição de uma gramática formal para normalização de expressões temporais relativas em documentos na língua inglesa. Até onde é de conhecimento do autor deste trabalho, esta é a primeira vez que uma gramática formal é utilizada para este objetivo, tornando este um trabalho inédito. Este trabalho também inclui os passos detalhados de extração de fatos temporais a partir de texto, com o uso de uma ferramenta existente de extração de informações. Em seguida, os fatos temporais são utilizados para popular a ontologia da DBpedia, empregando as propriedades temporais desta. O reuso de uma ontologia existente segue as melhores práticas da Web Semântica. A população de ontologia com os fatos extraídos permite a realização de consultas sofisticadas, conexão a fontes de dados da Web Semântica e criação de novas aplicações e *mashups*.

Como produção científica, os seguintes artigos foram publicados:

1. (GALLINA; GALANTE, 2011) **Extração e Representação Semântica de Fatos Temporais**. In: WORKSHOP DE TESES E DISSERTAÇÕES EM BANCOS DE DADOS, WTDBD, 10., SIMPÓSIO BRASILEIRO DE BANCO DE DADOS, SBBD, 26. Anais... Belo Horizonte: SBC, 2011. p.25–30. Este artigo descreve a proposta inicial da dissertação.

2. (GALLINA; GALANTE; DORNELES, 2011) **Formal Grammar For The Normalization Of Relative Temporal Expressions**. In: IADIS INTERNATIONAL CONFERENCE WWW/INTERNET 2011, Lisboa, Portugal. Proceedings... International Association for Development of the Information Society, 2011. p.373–380.

Este artigo descreve a gramática formal para normalização de expressões temporais relativas.

Os seguintes trabalhos foram desenvolvidos em paralelo com esta dissertação, nos qua o autor é coorientador:

- (BATTASSINI, 2011) **Uma ferramenta para busca temporal na DBpedia a partir de uma ontologia**. 2011. Monografia (Graduação em Ciência da Computação) – Instituto de Informática, UFRGS, Porto Alegre. Esta monografia de conclusão de curso foi desenvolvida em paralelo com este trabalho. Uma vez populada a ontologia da DBpedia com as novas triplas RDF geradas pela abordagem EXTIO, este trabalho fornece uma interface gráfica de consulta dos dados extraídos.

- (MARCANT, 2012) **Plug In para um motor de busca utilizando a abordagem EXTIO**. 2012. Monografia (Graduação em Ciência da Computação) – Instituto de Informática, UFRGS, Porto Alegre. Esta monografia de conclusão de curso foi desenvolvida em paralelo com este trabalho. O objetivo deste trabalho é implementar um motor de busca que utilize a gramática para normalização de expressões temporais de EXTIO para obter informação temporal em consultas a documentos.

A abordagem EXTIO é aplicável a texto em linguagem natural em inglês. A escolha por esta língua deu-se devido à grande porcentagem de páginas da Web neste idioma. Um possível trabalho futuro seria aplicar esta abordagem para a língua portuguesa. Para atingir este objetivo, seria necessário traduzir a gramática formal para a língua portuguesa, fazendo os devidos mapeamentos. Adicionalmente, seria necessário escolher uma implementação de extração de fatos temporais para português, visto que a ferramenta escolhida para implementação deste trabalho não possui versão em outros idiomas que não o inglês.

Ao longo deste trabalho, alguns aspectos da gramática formal para normalização de expressões temporais relativas foram identificados como passíveis de melhorias ou extensões:

- permitir a aplicação de uma granularidade mais fina para as expressões temporais relativas. Pela gramática apresentada neste trabalho, a expressão tonight (esta noite) é normalizada para o dia que ocorreu, sem ser dado significado temporal ao período da noite. Expressões relacionadas a outros períodos do dia, como manhã e tarde, poderiam se beneficiar desta granularidade fina. A aplicação da granularidade de hora, minuto e segundo também é sugerida como um trabalho futuro.

- realizar um estudo específico para a normalização de expressões temporais de notícias esportivas. Os experimentos realizados neste trabalho mostraram que existe espaço para melhoria do módulo de normalização de expressões temporais relativas de EXTIO nesta categoria de documentos.

- estabelecer condições nas quais determinadas expressões temporais não devem ser normalizadas. Eventualmente, o texto apresenta expressões temporais com a função de substantivo, cuja normalização ocorre em desacordo com o sentido do texto. Por exemplo, a expressão September 11 muitas vezes se refere, em notícias, aos eventos ocorridos especificamente no dia 11 de setembro de 2001. A gramática apresentada neste trabalho ignora esta possibilidade, normalizando a expressão para a ocorrência de 11 de setembro mais próxima à data do documento. É sugerido como trabalho futuro implementar algum mecanismo para identificar as situações em que isto ocorre e não realizar a normalização.

- normalizar expressões temporais que representam feriados e outras datas especiais, como Christmas Day (dia de Natal) e Thanksgiving (dia de Ação de Graças), além de períodos como estações do ano.

- realizar o tratamento de *correferências temporais*. Uma correferência temporal é uma expressão temporal relativa a uma data no meio do texto. Isto difere das expressões temporais relativas tratadas pelo trabalho aqui apresentado, que são normalizadas sempre em relação à data de publicação do documento. Correferências englobam expressões relativas a outras datas, que surgem no conteúdo do texto. A seguinte frase é um exemplo de correferência:

  "On Dec. 2nd, 2008, the President (. . .) Then, the next day, (. . .)"

  Neste exemplo, a expressão the next day é uma correferência. O valor da expressão é relativo a uma data que está no meio do texto – no caso, 02 de dezembro de 2008. Portanto, a expressão deveria ser normalizada para 03 de dezembro de 2008. Esta expressão não é tratada por este trabalho e permanece inalterada.

- implementar um motor de busca integrado com o módulo de normalização de expressões temporais apresentado neste trabalho. Para otimizar o desempenho na realização de consultas, um motor de busca usualmente realiza a indexação dos documentos de texto que podem ser pesquisados. Esta indexação tem como objetivo construir uma estrutura de dados para realizar consultas sobre o texto de forma mais rápida que a simples busca sequencial dentro de todos os documentos (FRAKES; BAEZA-YATES, 1992). Como trabalho futuro, sugere-se a implementação de um motor de busca no qual as ocorrências de expressões temporais de todos os documentos são normalizadas antes dos documentos serem indexados. Desta forma, as expressões temporais – normalizadas pela gramática formal proposta neste trabalho – seriam indexadas pelo motor de busca, no lugar da representação textual relativa, que usualmente não é de interesse para o usuário final. Uma página da Internet[1] com data de publicação de outubro de 2011 ilustra este exemplo com a frase "Consumer prices rose 7.31% in September". Ao submeter esta página à normalização de expressões temporais, a frase seria alterada para "Consumer prices rose 7.31% in September, 2011". Desta forma, uma pessoa que estiver procurando o valor da inflação no mês de setembro de 2011 encontrará esta página como resultado da sua consulta.

- aplicar o módulo de normalização de expressões temporais apresentado neste trabalho na geração de metadados temporais para páginas da Web. Como trabalho futuro, sugere-se que as expressões temporais normalizadas sejam representadas de forma oculta nas páginas da Web, através de metadados. Assim, um motor de busca poderia realizar consultas sobre a informação temporal oculta contida nos metadados.

Ao longo deste trabalho, alguns aspectos da extração e organização de fatos temporais também foram identificados como passíveis de melhorias ou extensões:

[1] http://www.bbc.co.uk/news/business-15218174

- realizar experimentos para verificar o benefício que a extração de informações e organização em ontologia de fatos temporais de EXTIO traz à base de dados da DBpedia. Para estes experimentos, devem ser elaboradas consultas (na linguagem SPARQL) que são submetidas à base de dados da DBpedia. Estas consultas são executadas antes e depois da adição de triplas RDF, permitindo mensurar o ganho de informação trazido pela abordagem EXTIO.

- realizar experimentos para medir precisão, revocação, *F-measure* e tempo de processamento da extração de informações de EXTIO. Estes experimentos devem ser realizados de forma comparativa utilizando como *baseline* um dos trabalhos relacionados citados nesta dissertação, como TOB (*Timely Ontologies for Business Relations*) (ZHANG et al., 2008).

- implementar um protótipo de extração de fatos temporais para contemplar fatos de outras aridades, como ternários e quarternários. No protótipo criado para validação deste trabalho, apenas fatos binários são extraídos do texto e armazenados, pois as propriedades temporais da ontologia relacionam um objeto a uma informação temporal. Um fato quarternário poderia, por exemplo, relacionar um empregado, uma empresa e a data de início e data de fim da relação empregatícia.

- na implementação deste trabalho, foi utilizada a ferramenta Leila (SUCHANEK; IFRIM; WEIKUM, 2006a) para extração de fatos temporais. Para cada par extraído para determinada relação, a ferramenta gera um valor de confiança, que não foi utilizado nesta dissertação. Um trabalho futuro poderia aplicar a confiança para disponibilizar a geração de triplas RDF apenas acima de um determinado limiar de confiança, dentre outras possíveis utilidades.

REFERÊNCIAS

AHO, A. V. et al. **Compilers**: principles, techniques, and tools (2nd edition). Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2006.

ALLEN, J. F. Maintaining knowledge about temporal intervals. **Commun. ACM**, New York, NY, USA, v.26, p.832–843, November 1983.

AUER, S. et al. DBpedia: a nucleus for a web of open data. In: THE SEMANTIC WEB AND 2ND ASIAN CONFERENCE ON ASIAN SEMANTIC WEB CONFERENCE, 6., Berlin, Heidelberg. **Proceedings...** Springer-Verlag, 2007. p.722–735. (ISWC'07/ASWC'07).

BAEZA-YATES, R. A.; RIBEIRO-NETO, B. **Modern Information Retrieval**. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1999.

BANKO, M. et al. Open information extraction from the web. In: ARTIFICAL INTELLIGENCE, 20., San Francisco, CA, USA. **Proceedings...** Morgan Kaufmann Publishers Inc., 2007. p.2670–2676. (IJCAI'07).

BATTASSINI, R. **Um mecanismo de consulta temporal por palavras-chave em páginas Web**. 2011. Monografia (Graduação em Ciência da Computação) — Instituto de Informática, UFRGS, Porto Alegre.

BERNERS-LEE, T. **Semantic Web Roadmap**. Disponível em: <http://www.w3.org/DesignIssues/Semantic.html>. Acesso em: 30 nov. 2011.

BERNERS-LEE, T. **Linked Data - Design Issues**. Disponível em: <http://www.w3.org/DesignIssues/LinkedData.html>. Acesso em: 30 nov. 2011.

BERNERS-LEE, T.; CAILLIAU, R. **WorldWideWeb**: proposal for a hypertext project. [S.l.]: CERN, 1990. Proposal.

BERNERS-LEE, T.; FISCHETTI, M. **Weaving the Web**: the original design and ultimate destiny of the world wide web by its inventor. 1st.ed. [S.l.]: Harper San Francisco, 1999.

BERNERS-LEE, T.; HENDLER, J.; LASSILA, O. The Semantic Web. **Scientific American**, [S.l.], v.284, n.5, p.34–43, May 2001.

BIZER, C. et al. DBpedia - A crystallization point for the Web of Data. **Web Semant.**, Amsterdam, The Netherlands, The Netherlands, v.7, p.154–165, September 2009.

BRICKLEY, D.; GUHA, R. **RDF Vocabulary Description Language 1.0**: rdf schema. Disponível em: <http://www.w3.org/TR/rdf-schema/>. Acesso em: 30 nov. 2011.

CHERKASSKY, V.; MA, Y. Practical selection of SVM parameters and noise estimation for SVM regression. **Neural Netw.**, Oxford, UK, UK, v.17, p.113–126, January 2004.

CRAVEIRO, O.; MACEDO, J.; MADEIRA, H. Use of Co-occurrences for Temporal Expressions Annotation. In: INTERNATIONAL SYMPOSIUM ON STRING PROCESSING AND INFORMATION RETRIEVAL, 16., Berlin, Heidelberg. **Proceedings...** Springer-Verlag, 2009. p.156–164. (SPIRE '09).

CUNNINGHAM, H. et al. GATE: a framework and graphical development environment for robust NLP tools and applications. In: ANNUAL MEETING OF THE ASSOCIATION FOR COMPUTATIONAL LINGUISTICS, PHILADELPHIA, PA, USA, 40. **Proceedings...** [S.l.: s.n.], 2002.

CUNNINGHAM, H. et al. **Text Processing with GATE (Version 6)**. [S.l.: s.n.], 2011.

DOMINGOS, P.; PAZZANI, M. On the Optimality of the Simple Bayesian Classifier under Zero-One Loss. **Mach. Learn.**, Hingham, MA, USA, v.29, p.103–130, November 1997.

DOWTY, D. **Word meaning and Montague grammar** : the semantics of verbs and times in generative semantics and in montague's ptq. Dordrecht Boston: D. Reidel Pub. Co, 1979.

DYRESON, C. et al. A consensus glossary of temporal database concepts. **SIGMOD Rec.**, New York, NY, USA, v.23, p.52–64, March 1994.

EDELWEISS, N. Bancos de Dados Temporais: teoria e prática. In: XVII JORNADA DE ATUALIZAÇÃO EM INFORMÁTICA, XVIII CONGRESSO NACIONAL DA SOCIEDADE BRASILEIRA DE COMPUTAÇÃO, Belo Horizonte, Brasil. **Anais...** Sociedade Brasileira de Computação, 1998. v.2, p.225–282.

ERLING, O.; MIKHAILOV, I. RDF Support in the Virtuoso DBMS. In: CONFERENCE ON SOCIAL SEMANTIC WEB. **Anais...** GI, 2007. p.59–68. (LNI, v.113).

ETZIONI, O. et al. Web-scale information extraction in knowitall: (preliminary results). In: WORLD WIDE WEB, 13., New York, NY, USA. **Proceedings...** ACM, 2004. p.100–110. (WWW '04).

FELLBAUM, C. (Ed.). **WordNet: an electronic lexical database**. [S.l.]: MIT Press, 1998.

FERRO, L. et al. **TIDES Temporal Annotation Guidelines - Version 1.0.2**. MITRE Technical Report MTR 01W0000041. McLean, Virginia: The MITRE Corporation. June 2001.

FISHER, R. A. **The design of experiments**. London: Oliver & Boyd, 1937.

FRAKES, W. B.; BAEZA-YATES, R. (Ed.). **Information retrieval**: data structures and algorithms. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1992.

GALLINA, L.; GALANTE, R. Extração e Representação Semântica de Fatos Temporais. In: WORKSHOP DE TESES E DISSERTAÇÕES EM BANCOS DE DADOS, WTDBD, 10., SIMPÓSIO BRASILEIRO DE BANCO DE DADOS, SBBD, 26. **Anais...** Belo Horizonte: SBC, 2011. p.25–30.

GALLINA, L.; GALANTE, R.; DORNELES, C. Formal Grammar For The Normalization Of Relative Temporal Expressions. In: IADIS INTERNATIONAL CONFERENCE WWW/INTERNET 2011, Lisboa, Portugal. **Proceedings...** International Association for Development of the Information Society, 2011. p.373–380.

GRISHMAN, R. Information Extraction: techniques and challenges. In: INTERNATIONAL SUMMER SCHOOL ON INFORMATION EXTRACTION: A MULTIDISCIPLINARY APPROACH TO AN EMERGING INFORMATION TECHNOLOGY, London, UK, UK. **Anais...** Springer-Verlag, 1997. p.10–27. (SCIE '97).

GRUBER, T. R. A translation approach to portable ontology specifications. **Knowl. Acquis.**, London, UK, UK, v.5, p.199–220, June 1993.

HAHN, R. et al. Faceted Wikipedia Search. In: BUSINESS INFORMATION SYSTEMS, 13TH INTERNATIONAL CONFERENCE, BIS 2010, BERLIN, GERMANY, MAY 3-5, 2010. PROCEEDINGS. **Anais...** Springer, 2010. p.1–11. (Lecture Notes in Business Information Processing, v.47).

HEATH, T.; BIZER, C. **Linked Data**: evolving the web into a global data space. 1st.ed. [S.l.]: Morgan & Claypool, 2011.

HERMAN, I. **State of the Semantic Web**. Disponível em: <http://www.w3.org/2007/Talks/0424-Stavanger-IH/Slides.pdf>. Acesso em: 30 nov. 2011.

HORROCKS, I.; SATTLER, U.; TOBIES, S. Practical Reasoning for Expressive Description Logics. In: INTERNATIONAL CONFERENCE ON LOGIC PROGRAMMING AND AUTOMATED REASONING, 6., London, UK. **Proceedings...** Springer-Verlag, 1999. p.161–180. (LPAR '99).

HUNTER, D. et al. **Beginning XML, 4th Edition**. Birmingham, UK, UK: Wrox Press Ltd., 2007.

ISO. **ISO 8601:2000. data elements and interchange formats — information interchange — representation of dates and times**. 2000. 29p.

JACQUEMIN, C. **Spotting and discovering terms through natural language processing**. Cambridge, MA, USA: MIT Press, 2001.

JIN, P. et al. TISE: a temporal search engine for web contents. In: SECOND INTERNATIONAL SYMPOSIUM ON INTELLIGENT INFORMATION TECHNOLOGY APPLICATION - VOLUME 03, 2008., Washington, DC, USA. **Proceedings...** IEEE Computer Society, 2008. p.220–224.

KARLSSON, B. **Beyond the C++ Standard Library**. [S.l.]: Addison-Wesley Professional, 2005.

KOEN, D. B.; BENDER, W. Time frames: temporal augmentation of the news. **IBM Syst. J.**, Riverton, NJ, USA, v.39, p.597–616, July 2000.

LEVINE, J.; MASON, T.; BROWN, D. **Lex & yacc, 2nd edition**. 2.ed. [S.l.]: O'Reilly, 1992.

LINGUA DTIL, U. L. D. T. e Indústrias da. **Línguas e Culturas na Web - Estudo 2007**. Disponível em: http://dtil.unilat.org/LI/2007/pt/resultados_pt.htm. Acesso em: 30 nov. 2011.

MANI, I.; WILSON, G. Robust temporal processing of news. In: ANNUAL MEETING ON ASSOCIATION FOR COMPUTATIONAL LINGUISTICS, 38., Stroudsburg, PA, USA. **Proceedings...** Association for Computational Linguistics, 2000. p.69–76. (ACL '00).

MANICA, E.; DORNELES, C. F.; GALANTE, R. Supporting Temporal Queries on XML Keyword Search Engines. In: JOURNAL OF INFORMATION AND DATA MANAGEMENT. **Anais...** [S.l.: s.n.], 2010. v.1, p.471–486.

MARCANT, E. **Plug In para um motor de busca utilizando a abordagem EXTIO**. 2012. Monografia (Graduação em Ciência da Computação) — Instituto de Informática, UFRGS, Porto Alegre.

MAYNARD, D.; CUNNINGHAM, H. Multilingual adaptations of ANNIE, a reusable information extraction tool. In: EUROPEAN CHAPTER OF THE ASSOCIATION FOR COMPUTATIONAL LINGUISTICS - VOLUME 2, Stroudsburg, PA, USA. **Proceedings...** Association for Computational Linguistics, 2003. p.219–222. (EACL '03).

MAYNARD, D.; LI, Y.; PETERS, W. NLP Techniques for Term Extraction and Ontology Population. In: ONTOLOGY LEARNING AND POPULATION: BRIDGING THE GAP BETWEEN TEXT AND KNOWLEDGE, 2008., Amsterdam, The Netherlands, The Netherlands. **Proceedings...** IOS Press, 2008. p.107–127.

MCGUINNESS, D.; HARMELEN, F. van. **OWL Web Ontology Language Overview**. Disponível em: <http://www.w3.org/TR/2004/REC-owl-features-20040210/>. Acesso em: abr. 2010.

MENDES, P. N. et al. DBpedia Spotlight: shedding light on the web of documents. In: INTERNATIONAL CONFERENCE ON SEMANTIC SYSTEMS (I-SEMANTICS), 7. **Proceedings...** [S.l.: s.n.], 2011.

MENEZES, P. B. **Linguagens Formais e Autômatos**. [S.l.]: Editora Sagra-Luzzatto, 1998.

PUSTEJOVSKY, J. et al. The Specification Language TimeML. In: MANI, I.; PUSTEJOVSKY, J.; GAIZAUSKAS, R. (Ed.). **The Language of Time**: a reader. [S.l.]: Oxford University Press, 2004.

RIJSBERGEN, C. J. V. **Information Retrieval**. 2nd.ed. Newton, MA, USA: Butterworth-Heinemann, 1979.

SAQUETE, E.; MARTINEZ-BARCO, P. **Grammar Specification for the Recognition of Temporal Expressions**. 2000.

SEGARAN, T. et al. **Programming the Semantic Web**. 1st.ed. [S.l.]: O'Reilly Media, Inc., 2009.

SETZER, A. **Temporal information in newswire articles**: an annotation scheme and corpus study. 2001. PhD thesis — University of Sheffield, Sheffield, Inglaterra.

SIMPERL, E. Reusing ontologies on the Semantic Web: a feasibility study. **Data Knowl. Eng.**, Amsterdam, The Netherlands, The Netherlands, v.68, p.905–925, October 2009.

SLEATOR, D.; TEMPERLEY, D. **Parsing English with a Link Grammar**. 1993.

SMUCKER, M. D.; ALLAN, J.; CARTERETTE, B. A comparison of statistical significance tests for information retrieval evaluation. In: ACM CONFERENCE ON CONFERENCE ON INFORMATION AND KNOWLEDGE MANAGEMENT, New York, NY, USA. **Proceedings...** ACM, 2007. p.623–632. (CIKM '07).

SUCHANEK, F. M.; IFRIM, G.; WEIKUM, G. Combining linguistic and statistical analysis to extract relations from web documents. In: ACM SIGKDD INTERNATIONAL CONFERENCE ON KNOWLEDGE DISCOVERY AND DATA MINING, 12., New York, NY, USA. **Proceedings...** ACM, 2006. p.712–717. (KDD '06).

SUCHANEK, F. M.; IFRIM, G.; WEIKUM, G. LEILA: learning to extract information by linguistic analysis. In: WORKSHOP ON ONTOLOGY LEARNING AND POPULATION: BRIDGING THE GAP BETWEEN TEXT AND KNOWLEDGE, 2., Sydney, Australia. **Proceedings...** Association for Computational Linguistics, 2006. p.18–25.

SUCHANEK, F. M.; KASNECI, G.; WEIKUM, G. Yago: a core of semantic knowledge. In: WORLD WIDE WEB, 16., New York, NY, USA. **Proceedings...** ACM, 2007. p.697–706. (WWW '07).

TALIS. **Sir Tim Berners-Lee Talks with Talis about the Semantic Web**. Disponível em: <http://talis-podcasts.s3.amazonaws.com/twt20080207_TimBL.html>. Acesso em: 30 nov. 2011.

TAN, P.-N.; STEINBACH, M.; KUMAR, V. **Introduction to Data Mining, (First Edition)**. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2005.

VERHAGEN, M. et al. Automating temporal annotation with TARSQI. In: ACL 2005 ON INTERACTIVE POSTER AND DEMONSTRATION SESSIONS, Stroudsburg, PA, USA. **Proceedings...** Association for Computational Linguistics, 2005. p.81–84. (ACL-demo '05).

VERHAGEN, M.; PUSTEJOVSKY, J. Temporal processing with the TARSQI toolkit. In: INTERNATIONAL CONFERENCE ON ON COMPUTATIONAL LINGUISTICS: DEMONSTRATION PAPERS, 22., Stroudsburg, PA, USA. **Anais...** Association for Computational Linguistics, 2008. p.189–192. (COLING '08).

WANG, Y. et al. Timely YAGO: harvesting, querying, and visualizing temporal knowledge from wikipedia. In: INTERNATIONAL CONFERENCE ON EXTENDING DATABASE TECHNOLOGY, 13., New York, NY, USA. **Proceedings...** ACM, 2010. p.697–700. (EDBT '10).

ZHANG, Q. et al. TOB: timely ontologies for business relations. In: WEBDB. **Anais...** [S.l.: s.n.], 2008.

APÊNDICE A GRAMÁTICA FORMAL PARA NORMALIZAÇÃO DE EXPRESSÕES TEMPORAIS RELATIVAS

A variável `documentDate` armazena a data de publicação do documento. Esta data é obtida em uma etapa prévia à aplicação da gramática.

$temporal\_expression \rightarrow$
$today$
$| \; tomorrow$
$| \; yesterday$
$| \; tonight$
$| \; years\_ago$
$| \; months\_ago$
$| \; weeks\_ago$
$| \; decades\_ago$
$| \; centuries\_ago$
$| \; this\_year$
$| \; last\_year$
$| \; next\_year$
$| \; this\_month$
$| \; last\_month$
$| \; next\_month$
$| \; this\_week$
$| \; last\_week$
$| \; next\_week$
$| \; this\_decade$
$| \; last\_decade$
$| \; next\_decade$
$| \; this\_century$
$| \; last\_century$
$| \; next\_century$
$| \; last\_occurrence\_of\_month$
$| \; last\_weekday$
$| \; next\_weekday$
$| \; weekday$
$| \; date\_without\_year$
$| \; day\_month$
$| \; month\_without\_year$

today → “*today*”
```
 {addDays(documentDate, 0);}
```

tomorrow → “*tomorrow*”
```
 {addDays(documentDate, 1);}
```

yesterday → “*yesterday*”
```
 {addDays(documentDate, -1);}
```

tonight → “*tonight*”
```
 {addDays(documentDate, 0); print("at night");}
```

years_ago → *number token_year token_ago*
```
 {addYears(documentDate, -number.value);}
```

months_ago → *number token_month token_ago*
```
 {addMonths(documentDate, -number.value);}
```

weeks_ago → *number token_week token_ago*
```
 {addWeeks(documentDate, -number.value);}
```

decades_ago → *number token_decade token_ago*
```
 {addDecades(documentDate, -number.value);}
```

centuries_ago → *number token_century token_ago*
```
 {addCenturies(documentDate, -number.value);}
```

this_year → *token_this token_year*
```
 {addYears(documentDate, 0);}
```

last_year → *token_last token_year*
```
 {addYears(documentDate, -1);}
```

next_year → *token_next token_year*
```
 {addYears(documentDate, 1);}
```

this_month → *token_this token_month*
```
 {addMonths(documentDate, 0);}
```

last_month → *token_last token_month*
```
 {addMonths(documentDate, -1);}
```

next_month → *token_next token_month*
```
 {addMonths(documentDate, 1);}
```

this_week → *token_this token_week*
```
 {addWeeks(documentDate, 0);}
```

last_week → *token_last token_week*
```
 {addWeeks(documentDate, -1);}
```

next_week → *token_next token_week*
```
 {addWeeks(documentDate, 1);}
```

this_decade → *token_this token_decade*
```
 {addDecades(documentDate, 0);}
```

last_decade → *token_last token_decade*
```
 {addDecades(documentDate, -1);}
```

next_decade → *token_next token_decade*
```
 {addDecades(documentDate, 1);}
```

this_century → *token_this token_century*
```
 {addCenturies(documentDate, 0);}
```

last_century → *token_last token_century*
```
 {addCenturies(documentDate, -1);}
```

next_century → *token_next token_century*
```
 {addCenturies(documentDate, 1);}
```

last_occurrence_of_month → *token_last month*
```
 {getLastDateByMonth(documentDate, month.value);}
```

last_weekday → *token_last token_weekday*
```
{getPreviousWeekday(documentDate,
    token_weekday.value);}
```

next_weekday → *token_next token_weekday*
```
 {getNextWeekday(documentDate, token_weekday.value);}
```

weekday → *token_weekday*
```
 {getNearestWeekday(documentDate, token_weekday.value);}
```

date_without_year → *month number*
```
{getDateWithoutYear(documentDate,
    month.value, number.value);}
```

day_month → *number month*
```
{getDateWithoutYear(documentDate,
    month.value, number.value);}
```

month_without_year → *month*
```
{discardDay(getDateWithoutYear(
    documentDate, month.value, 1));}
```

APÊNDICE B ANALISADOR LÉXICO PARA NORMALIZAÇÃO DE EXPRESSÕES TEMPORAIS RELATIVAS

$number \rightarrow [0-9]+$
 {number.value = *valor da sequência de dígitos*;}
$number \rightarrow$ "*one*"
 {number.value = 1;}
$number \rightarrow$ "*two*"
 {number.value = 2;}
$number \rightarrow$ "*three*"
 {number.value = 3;}
$number \rightarrow$ "*four*"
 {number.value = 4;}
$number \rightarrow$ "*five*"
 {number.value = 5;}
$number \rightarrow$ "*six*"
 {number.value = 6;}
$number \rightarrow$ "*seven*"
 {number.value = 7;}
$number \rightarrow$ "*eight*"
 {number.value = 8;}
$number \rightarrow$ "*nine*"
 {number.value = 9;}
$number \rightarrow$ "*ten*"
 {number.value = 10;}

$token\_ago \rightarrow$ "*ago*"
$token\_year \rightarrow$ "*year*" | "*years*"
$token\_month \rightarrow$ "*month*" | "*months*"
$token\_week \rightarrow$ "*week*" | "*weeks*"
$token\_decade \rightarrow$ "*decade*" | "*decades*"
$token\_century \rightarrow$ "*century*" | "*centuries*"
$token\_this \rightarrow$ "*this*"
$token\_last \rightarrow$ "*last*"
$token\_next \rightarrow$ "*next*"

$month \rightarrow$ "*January*"
 {month.value = 1;}
$month \rightarrow$ "*February*"

```
 {month.value = 2;}
```
month → “*March*”
```
 {month.value = 3;}
```
month → “*April*”
```
 {month.value = 4;}
```
month → “*May*”
```
 {month.value = 5;}
```
month → “*June*”
```
 {month.value = 6;}
```
month → “*July*”
```
 {month.value = 7;}
```
month → “*August*”
```
 {month.value = 8;}
```
month → “*September*”
```
 {month.value = 9;}
```
month → “*October*”
```
 {month.value = 10;}
```
month → “*November*”
```
 {month.value = 11;}
```
month → “*December*”
```
 {month.value = 12;}
```

token_weekday → “*Sunday*”
```
 {token_weekday.value = 0;}
```
token_weekday → “*Monday*”
```
 {token_weekday.value = 1;}
```
token_weekday → “*Tuesday*”
```
 {token_weekday.value = 2;}
```
token_weekday → “*Wednesday*”
```
 {token_weekday.value = 3;}
```
token_weekday → “*Thursday*”
```
 {token_weekday.value = 4;}
```
token_weekday → “*Friday*”
```
 {token_weekday.value = 5;}
```
token_weekday → “*Saturday*”
```
 {token_weekday.value = 6;}
```

APÊNDICE C LISTA DE PROPRIEDADES TEMPORAIS DA ONTOLOGIA SELECIONADA

O prefixo `http://dbpedia.org/ontology/` foi suprimido da coluna *URI da propriedade*.

Na coluna *Imagem*, foram suprimidos os prefixos `http://www.w3.org/2001/` e `http://dbpedia.org/datatype/`.

A coluna *Ocorr.* possui a contagem de ocorrências de 1980 a 2011, conforme processo descrito no Capítulo 4. Os registros estão ordenados por esta coluna.

URI da Propriedade	Imagem	Ocorr.
releaseDate	XMLSchema#date	129325
birthDate	XMLSchema#date	85901
activeYearsStartYear	XMLSchema#gYear	60664
deathDate	XMLSchema#date	56907
activeYearsStartDate	XMLSchema#date	32459
activeYearsEndDate	XMLSchema#date	22368
activeYearsEndYear	XMLSchema#gYear	22234
formationYear	XMLSchema#gYear	22166
added	XMLSchema#date	21299
populationAsOf	XMLSchema#date	16874
deathYear	XMLSchema#gYear	12527
draftYear	XMLSchema#gYear	9165
discovered	XMLSchema#date	8335
foundingYear	XMLSchema#gYear	8280
birthYear	XMLSchema#gYear	7290
completionDate	XMLSchema#date	5767
recordDate	XMLSchema#date	4664
openingDate	XMLSchema#date	4106
formationDate	XMLSchema#date	4035
censusYear	XMLSchema#date	3890
productionStartYear	XMLSchema#gYear	3619
date	XMLSchema#date	3504
productionEndYear	XMLSchema#gYear	3007
startDate	XMLSchema#date	2445
foundingDate	XMLSchema#date	2341
extinctionYear	XMLSchema#gYear	2150
undraftedYear	XMLSchema#gYear	2116
publicationDate	XMLSchema#date	1883
serviceEndYear	XMLSchema#gYear	1724
latestReleaseDate	XMLSchema#date	1488

statisticYear	XMLSchema#date	928
shareDate	XMLSchema#date	592
serviceStartYear	XMLSchema#gYear	510
debut	XMLSchema#date	499
buildingStartDate	XMLSchema#date	477
modelStartYear	XMLSchema#gYear	356
functionStartYear	XMLSchema#gYear	312
extinctionDate	XMLSchema#date	311
closingDate	XMLSchema#date	310
modelEndYear	XMLSchema#gYear	276
landingDate	XMLSchema#date	235
eruptionYear	XMLSchema#gYear	222
beatifiedDate	XMLSchema#date	176
canonizedDate	XMLSchema#date	144
dissolutionYear	XMLSchema#gYear	119
maidenFlight	XMLSchema#date	118
dateOfAbandonment	XMLSchema#date	115
rebuildingDate	XMLSchema#date	114
electionDate	XMLSchema#date	111
buildingEndDate	XMLSchema#date	106
firstAscentYear	XMLSchema#gYear	79
dissolutionDate	XMLSchema#date	75
launch	XMLSchema#date	69
port1DockingDate	XMLSchema#date	69
port1UndockingDate	XMLSchema#date	68
decay	XMLSchema#date	63
finalFlight	XMLSchema#date	62
demolitionDate	XMLSchema#date	56
launchDate	XMLSchema#date	53
dissolved	XMLSchema#date	42
latestPreviewDate	XMLSchema#date	41
endDate	XMLSchema#date	38
ethnicGroupsInYear	XMLSchema#gYear	38
productionStartDate	XMLSchema#date	35
lastLaunchDate	XMLSchema#date	30
anniversary	XMLSchema#date	28
retired	XMLSchema#date	28
disbanded	XMLSchema#date	19
supplementalDraftYear	XMLSchema#gYear	19
certificationDate	XMLSchema#date	14
introductionDate	XMLSchema#date	13
closed	XMLSchema#date	13
productionEndDate	XMLSchema#date	10
introduced	XMLSchema#date	8
firstFlightEndDate	XMLSchema#date	7
firstFlightStartDate	XMLSchema#date	7
firstLaunchDate	XMLSchema#date	7
deliveryDate	XMLSchema#date	7
destructionDate	XMLSchema#date	6
restoreDate	XMLSchema#date	6
lastFlightEndDate	XMLSchema#date	6

lastFlightStartDate	XMLSchema#date	6
dateOfBurial	XMLSchema#date	3
port2UndockingDate	XMLSchema#date	3
season	XMLSchema#date	3
port2DockingDate	XMLSchema#date	3
feastDay	XMLSchema#date	2
spacewalkBegin	XMLSchema#date	1
contractAward	XMLSchema#date	1
spacewalkEnd	XMLSchema#date	1
Spacecraft/port2DockedTime	day	0
Spacecraft/freeFlightTime	day	0
Spacecraft/port1DockedTime	day	0
SpaceShuttle/timeInSpace	day	0
Spacecraft/dockedTime	day	0
SpaceMission/missionDuration	day	0
Planet/orbitalPeriod	day	0
SpaceMission/lunarOrbitTime	hour	0
SpaceMission/lunarSurfaceTime	hour	0
SpaceMission/lunarEvaTime	hour	0
SpaceMission/stationVisitDuration	hour	0
SpaceMission/cmpEvaDuration	hour	0
SpaceMission/stationEvaDuration	hour	0
Astronaut/timeInSpace	minute	0
Work/runtime	minute	0
AutomobileEngine/acceleration	second	0
dateCompleted	XMLSchema#date	0
years	XMLSchema#date	0
suppreddedDate	XMLSchema#date	0
modelStartDate	XMLSchema#date	0
discontinued	XMLSchema#date	0
dateUse	XMLSchema#date	0
dateExtended	XMLSchema#date	0
lastLaunch	XMLSchema#date	0
productionYears	XMLSchema#date	0
functionStartDate	XMLSchema#date	0
managerYears	XMLSchema#date	0
dateClosed	XMLSchema#date	0
modelEndDate	XMLSchema#date	0
nationalYears	XMLSchema#date	0
functionEndDate	XMLSchema#date	0
whaDraftYear	XMLSchema#date	0
day	XMLSchema#date	0
youthYears	XMLSchema#date	0
firstLaunch	XMLSchema#date	0
argueDate	XMLSchema#date	0
wineYear	XMLSchema#date	0
airDate	XMLSchema#date	0
rebuildDate	XMLSchema#date	0
highestBuildingInYear	XMLSchema#date	0
dateConstruction	XMLSchema#date	0
floodingDate	XMLSchema#date	0

decideDate	XMLSchema#date	0
serviceEndDate	XMLSchema#date	0
reopened	XMLSchema#date	0
notifyDate	XMLSchema#date	0
serviceStartDate	XMLSchema#date	0
dateAct	XMLSchema#date	0
ordination	XMLSchema#date	0
lastSeason	XMLSchema#date	0
accessDate	XMLSchema#gYear	0
functionEndYear	XMLSchema#gYear	0
endYearOfSales	XMLSchema#gYear	0
year	XMLSchema#gYear	0
endYearOfInsertion	XMLSchema#gYear	0
ascentDate	XMLSchema#gYear	0
creationYear	XMLSchema#gYear	0
startYearOfInsertion	XMLSchema#gYear	0
startYearOfSales	XMLSchema#gYear	0
orbitalPeriod	XMLSchema#double	0
cmpEvaDuration	XMLSchema#double	0
freeFlightTime	XMLSchema#double	0
port1DockedTime	XMLSchema#double	0
stationEvaDuration	XMLSchema#double	0
timeInSpace	XMLSchema#double	0
acceleration	XMLSchema#double	0
runtime	XMLSchema#double	0
setupTime	XMLSchema#double	0
missionDuration	XMLSchema#double	0
port2DockedTime	XMLSchema#double	0
lunarOrbitTime	XMLSchema#double	0
lunarSurfaceTime	XMLSchema#double	0
playingTime	XMLSchema#double	0
lunarEvaTime	XMLSchema#double	0
dockedTime	XMLSchema#double	0
stationVisitDuration	XMLSchema#double	0
flyingHours	XMLSchema#double	0