Extração Automática de Conteúdos Documentais

Filipe Alexandre Ginja Carapinha

Dissertação para obtenção do Grau de Mestre em
Engenharia Informática e de Computadores

Júri

Presidente:	Professor Doutor Ernesto José Marques Morgado
Orientador:	Professor Doutor Nuno João Neves Mamede
Co-Orientador:	Professor Doutor Jorge Manuel Evangelista Baptista
Vogal:	Doutora Paula Cristina Quaresma da Fonseca Carvalho

Junho 2013

Agradecimentos

Gostaria de agradecer a todas as pessoas que me apoiaram e ajudaram a realização deste trabalho. Em primeiro lugar agradeço ao meu orientador, Professor Nuno Mamede, por me ter guiado na realização deste trabalho mostrando-se sempre disponível para me ajudar e ao meu co-orientador, Professor Jorge Baptista, pela pronta e preciosa revisão dos meus documentos.

Aos meus colegas e amigos por me terem apoiado nos momentos mais difíceis e por terem ouvido inúmeras vezes sem contestar a reposta "Não posso, estou a trabalhar na tese".

Por sempre me ter encorajado a continuar e dar o meu melhor agradeço à minha namorada Filipa Silva. Por fim um obrigado muito especial aos meus pais, que, no final de contas, foram aqueles que me deram a oportunidade de fazer este curso.

Lisboa, 7 de Junho de 2013
Filipe Carapinha

Resumo

Atualmente, devido à grande quantidade de informação disponível em formato de texto, existe uma necessidade crescente de conseguir aceder, tratar e analisar esta informação de forma automática. STRING, a cadeia de processamento de língua natural do L2F, possui alguns dos recursos necessários para processar textos a fim de extrair de forma estruturada a informação neles presente.

Esta dissertação descreve algumas das tarefas-chave necessárias para a extração de informação estruturada a partir de textos e apresenta um conjunto de trabalhos que permitem enquadrar melhor este problema.

Em seguida, apresentam-se as soluções adotadas no desenvolvimento de um novo módulo da STRING que, utilizando o processamento da cadeia, permite extrair e agregar informação proveniente de textos de forma estruturada.

A saída da cadeia é processada e a informação relativa a entidades mencionadas e às relações entre essas entidades, e é então guardada sob a forma de *slots*. Utilizando um mecanismo de resolução de correferência, estes *slots* são agregados pela entidade a que dizem respeito e o resultado é devolvido em formato XML.

Abstract

Nowadays, due to the large amount of information available in text format, there is an increasing need to access, process and analyse this information automatically. STRING, the natural language processing chain of L^2F, has some of the mechanisms required to process texts in order to structure the information they contain.

This paper describes some of the key tasks, necessary for the extraction of structured information from texts and presents the system developed to solve this problem using the resources that STRING already features.

The chain output is processed and the information about named entities and entity relations is stored in the form of slots. Using a correference analysis mechanism, this slots are then aggregated by entity and the output is returned in XML format.

Palavras Chave
Keywords

Palavras Chave

Processamento de Língua Natural
Preenchimento de *Slots*
Extração de Informação
Reconhecimento de Entidades Mencionadas

Keywords

Natural Language Processing
Slot-Filling
Information Extraction
Named Entity Recognition

Índice

1 Introdução **1**

1.1 Extração de Informação . . . 1

1.2 STRING . . . 2

1.3 Objetivos do Trabalho . . . 3

1.4 Estrutura do Documento . . . 4

2 Trabalho Relacionado **5**

2.1 OpenCalais . . . 5

2.2 Desambiguação de Entidades Mencionadas com Recurso a Bases de Conhecimento . . . 7

2.3 Preenchimento de *Slots* . . . 10

2.3.1 New York University . . . 11

2.3.2 UCD IIRG . . . 15

3 Arquitetura da Solução **17**

3.1 Eventos da STRING . . . 18

3.1.1 Eventos **LIFETIME** . . . 19

3.1.1.1 Eventos **BIRHT** e **DEATH** . . . 20

3.1.1.2 Evento de **AGE** . . . 20

3.1.1.3 Eventos **FAMILY** . . . 21

3.1.1.4 Evento **SPOUSE** . . . 21

3.1.1.5 Evento **RESIDENCE** . . . 22

3.1.1.6 Evento **EDUCATION** . . . 23

3.1.2 Eventos **BUSINESS** . . . 23

3.1.2.1 Evento **WORK** . . . 23

3.1.2.2 Evento **FOUNDATION** . . . 24

3.1.2.3 Evento **OWNERSHIP** . . . 25

3.1.2.4 Evento **CLIENT** . . . 25

3.1.2.5 Evento **AFFILIATION** . . . 25

3.1.3 **BUSINESS_PROFESSION** . . . 26

3.2 Módulos AfterXIP . . . 26

3.2.1 Normalização de Expressões Temporais . . . 27

3.2.2 Resolução de Expressões Anafóricas . . . 28

3.2.3 Preenchimento de *Slots* . . . 30

3.2.3.1 Propriedades das Relações . . . 32

3.2.3.2 *Slots* . . . 34

3.2.3.3 Agregação de Entidades . . . 39

3.2.3.4 Saída do módulo de preenchimento de *slots* . . . 41

4 Avaliação **43**
4.1 Metodologia de Avaliação e Resultados 43
4.1.1 Avaliação da Qualidade 43
4.1.2 Avaliação do Desempenho 45

5 Conclusões e Trabalho Futuro **47**

Lista de Figuras

1.1 Cadeia de Processamento de Língua Natural STRING 2

2.1 Fluxo de dados do OpenCalais 5
2.2 Procedimento de geração de padrões 12

3.1 Módulos de processamento Pós-XIP 17
3.2 Árvore gerada 28
3.3 Excerto do ficheiro XML com uma dependência que contem uma anáfora. 29
3.4 UML AfterXIP 31

Lista de Tabelas

2.1 Atributos da entidade *Person* . . . 6
2.2 Atributos da entidade *Company* . . . 6
2.3 Atributos da entidade *Organization* . . . 6
2.4 Atributos da relação *EmploymentRelation* . . . 7
2.5 *Slots* de valor simples (*sigle-valued slots*) . . . 11
2.6 *Slots* de valor em lista (*list-valued slots*) . . . 11

3.1 Eventos . . . 20
3.2 Eventos **BIRTH** e **DEATH** . . . 21
3.3 Evento **AGE** . . . 21
3.4 Eventos **FAMILY** . . . 22
3.5 Relação **SPOUSE** . . . 22
3.6 Evento **RESIDENCE** . . . 23
3.7 Evento **EDUCATION** . . . 23
3.8 Evento **WORK** . . . 24
3.9 Evento **FOUNDATION** . . . 24
3.10 Evento **OWNERSHIP** . . . 25
3.11 Evento **CLIENT** . . . 26
3.12 Evento **AFFILIATION** . . . 26
3.13 Exemplo de uma relação simétrica. . . . 32
3.14 Relações Simétricas . . . 32
3.15 Exemplo de simetria de relações em eventos UNCLE e NEPHEW . . . 32
3.16 Exemplo de simetria de relações em evento EMPLOYEE . . . 33
3.17 Subtipos de eventos que apresentam simetria de relações organização-pessoa . . . 33
3.18 Subtipos de eventos que apresentam simetria de relações pessoa-pessoa . . . 34
3.19 Mapeamentos entre os *slots* simples de uma entidade do tipo pessoa e os eventos da STRING 37
3.20 Mapeamentos entre os *slots* compostos de uma entidade do tipo pessoa e os eventos da STRING . . . 38
3.21 Mapeamentos entre os *slots* simples de uma entidade do tipo organização e os eventos da STRING . . . 39
3.22 Mapeamentos entre os *slots* compostos de uma entidade do tipo organização e os eventos da STRING . . . 39

4.1 Resultados das métricas de qualidade. . . . 45
4.2 Tempos de processamento (em segundos) dos diversos módulos da STRING. . . . 45

Capítulo 1

Introdução

Atualmente, existe uma grande quantidade de informação disponível em formato de texto dispersa pela Web. Para uma máquina, esta informação não é por si só diretamente acessível, tornando-se por isso necessário efetuar algum processamento que permita transformar um texto em língua natural num formato que possa ser automaticamente acedido, tratado e analisado, ou seja, é necessário estruturar estes conteúdos. Esta transformação é um dos objetivos do Processamento de Língua Natural e denomina-se Extração de Informação. Este trabalho apresenta uma abordagem para a resolução deste problema e os detalhes da solução implementada.

1.1 Extração de Informação

A Extração de Informação decompõe-se em várias tarefas e este estudo foca-se sobretudo numa delas, o Preenchimento de *Slots*. Apesar disto, é necessário descrever também algumas das outras tarefas por forma a melhor se entender o problema. Para o fazer, seguem-se dois exemplos:

(1.1) *Chomsky* nasceu a *7 de dezembro de 1928*, na cidade de *Filadélfia*.

(1.2) *Chomsky* é um linguista norte-americano. *Ele* é conhecido por ter criado a gramática generativa.

A primeira das tarefas de Extração de Informação a ter em conta é o Reconhecimento de Entidades Mencionadas (*Named Entity Recognition*), que consiste em delimitar e classificar as entidades presentes num texto. Um sistema capaz de realizar Reconhecimento de Entidades Mencionadas deve, no exemplo (1.1), conseguir identificar e classificar adequadamente três entidades: (a) *Chomsky* como uma PESSOA, (b) *7 de dezembro de 1928* como uma DATA e (c) *Filadélfia* como uma CIDADE. O Reconhecimento de Entidades Mencionadas pode ainda englobar um passo adicional que corresponde à identificação de relações entre entidades. No exemplo (1.1) poderia identificar-se uma relação de *nascimento* entre a entidade PESSOA (Chomsky) e a DATA (7 de dezembro de 1928).

O problema de Desambiguação de Entidades Mencionadas (*Named Entity Disambiguation*) consiste em mapear uma entidade mencionada num texto (referência), previamente identificada pela tarefa de Reconhecimento de Entidades Mencionadas, num único identificador do conceito ao qual a entidade se refere. Usando o exemplo (1.1), a referência *Chomsky* é relativa a Noam Chomsky, linguista norte-americano, e não a nenhum outro indivíduo que possa ter o mesmo nome. Ainda relacionado com esta tarefa, também se pretende estabelecer relações de equivalência entre diversas formas de designar a mesma entidade num texto. Esta tarefa designa-se de resolução de correferências. Um caso particular de resolução de correferências é a resolução de anáfora.

Uma anáfora é uma relação de correferência entre duas expressões num texto: uma delas requer, para poder ser interpretada corretamente, que se estabeleça essa relação com a outra expressão - o antecedente - que ocorre num momento anterior do discurso (excecionalmente, o antecedente pode vir depois da expressão anafórica, nesse caso usa-se o termo *catáfora* para a relação). No exemplo (1.2), *Ele* é uma expressão anafórica que se refere a *Chomsky*. À tarefa de identificar relações anafóricas chama-se Resolução de Anáforas e o exemplo (1.2) mostra um dos vários tipos de anáfora existentes, denominado *anáfora pronominal*. Finalmente, o Preenchimento de *Slots* (*Slot Filling*) consiste em recolher e agregar informação acerca de entidades mencionadas. A informação pode ser recolhida de um ou vários textos e é guardada em atributos associados à entidade à qual a informação se refere.

O conjunto de atributos de uma dada entidade é a *frame* dessa mesma entidade. Estes atributos representam uma relação entre a entidade em questão e determinado facto ou evento, ou um conjunto de factos ou eventos. As *frames* são portanto estruturas de dados para agrupar informação sobre determinada entidade. No exemplo (1.1), pode ser usado o valor da relação DATA_NASCIMENTO(Chomsky, 7/12/192), para preencher um atributo PESSOA_DATA_NASCIMENTO(7/12/1928) de uma hipotética *frame* da entidade PESSOA(*Chomsky*).

Para a realização destas tarefas é necessário recorrer a diversas ferramentas de PLN. Nesta dissertação utilizar-se-á a cadeia de PLN STRING[1] (*STatistical and Rule-based Natural lanGuage processing chain*) e os recursos linguísticos para ela desenvolvidos.

1.2 STRING

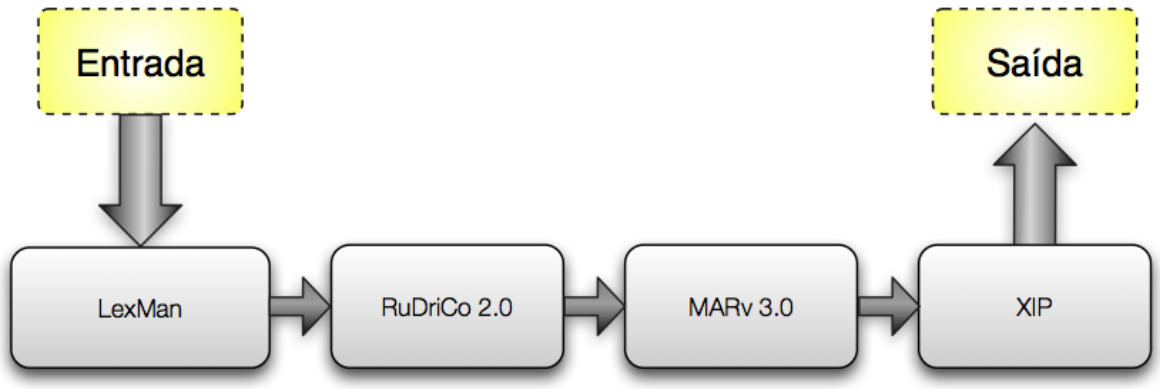

Figura 1.1: Cadeia de Processamento de Língua Natural STRING

STRING é uma cadeia de Processamento de Língua Natural (PLN) desenvolvida no Laboratório de Sistemas de Língua Falada (L^2F), constituída por quatro módulos, como mostra a figura 1.1. O código das diferentes aplicações está escrito nas linguagens C++, Java, Perl e XSLT.

O *LexMan* (Lexical Morphological Analiser [Vicente, 2013]) é responsável por receber um texto de entrada e proceder à divisão do mesmo em segmentos, onde cada segmento corresponde a, por exemplo, uma palavra, um sinal de pontuação ou um endereço e-mail. É ainda responsável pela etiquetagem morfossintática (*POS tagging*) da cadeia, ou seja, classifica um segmento como sendo um símbolo, um número, um verbo, etc. No caso das categorias com flexão, indica ainda os respetivos valores flexionais (tempo, modo, pessoa-número, género, número, grau, etc.). Uma palavra com mais do que uma etiqueta

[1] https://string.l2f.inesc-id.pt/w/index.php/Main_Page (data de acesso: 25/04/2013)

é uma palavra ambígua, de um ponto de vista morfossintático. Por fim, tem como tarefa agrupar os segmentos em frases de acordo com os sinais de pontuação, tendo em conta exceções como as abreviaturas.

O segundo módulo da cadeia é o *desambiguador morfossintático* RuDriCo 2.0 [Diniz, 2010], tendo este como principal objetivo a resolução das ambiguidades geradas anteriormente pelo processo de etiquetação. Adicionalmente, este módulo é também capaz de mudar a segmentação inicial, corrigindo-a para que esta melhor se adapte às necessidades dos módulos seguintes. Por exemplo, junta num só *token* palavras compostas (*'à esquerda de'*), e desfaz as contrações ('*nas*': '*em*' + '*as*'). Este módulo usa um conjunto de regras de transformação para realizar a sua função.

O MARv 3.0 [Ribeiro, 2003] é uma ferramenta de *desambiguação probabilística*, baseada em Modelos de Markov [Rabiner and Juang, 2003] e que usa o algoritmo de Viterbi [Forney, 1973] para selecionar a etiqueta mais adequada para uma palavra dado o contexto em que se insere (esta ferramenta não elimina as etiquetas preteridas). O modelo de língua do MARv assenta em modelos de segunda ordem (*trigramas*), que codificam informação contextual relativa às etiquetas; e *unigramas*, que codificam informação lexical.

O módulo final da cadeia de processamento é o *analisador sintático* XIP (Xerox Incremental Parser [Xerox, 2003]). Este módulo, é responsável por várias tarefas, nomeadamente, por fazer a segmentação das frases em constituintes sintáticos elementares (*chunks*) e calcular dependências entre eles, ou seja, determinar as relações sintáticas (sujeito, complemento direto, etc.). O XIP efetua o processamento de textos frase a frase. É no XIP que se processam as tarefas relacionadas com Reconhecimento de Entidades Mencionadas e de Extração de Relações entre elas; e é neste módulo que se centra também o trabalho desta dissertação. De entre os diversos tipos de relações que o XIP deteta, existe um grupo que é mais importante para o trabalho desenvolvido nesta dissertação os eventos. Na secção 3.1 é apresentada uma descrição detalhada dos tipos de eventos que estão a ser utilizados e das respetivas variações. Quanto à deteção de relações, é implementada no XIP por meio de regras feitas manualmente pelos investigadores do L^{2}F [Baptista et al., 2011]. As regras do XIP são constituídas por 3 partes, sendo possível omitir qualquer delas. As regras têm o seguinte formato:

|***padrão***| **if <*condição*> <termos de dependência>**

Para melhor compreensão, tome-se como exemplo a frase: *"O João nasceu em Janeiro de 1990 em Lisboa."*. Nesta frase, através a regra na listagem 1.1 extrai a relação EVENT[lex=+] entre *nascer* e *nascimento*. O *output* da cadeia quando processa a frase acima indicada é **EVENT_LEX(nasceu,nascimento)**, esta representa a relação entre a palavra normalizada e o elemento predicativo (nasceu) que a gerou. O *padrão* (omitido na regra representada na listagem 1.1), permite estabelecer restrições sobre a estrutura dos nós (*chunks*) e *features* (eg. género, número, etc.), identificadas em determinada frase. A <*condição*> permite verificar a presença de determinadas dependências na frase, por exemplo, a dependência SUBJ estabelece uma relação entre o verbo e o sujeito da frase. No exemplo anterior, tem o valor SUBJ_PRE(*nasceu,João*). Por fim, o segmento <termos de dependência>, é o retorno da regra.

Listagem 1.1: Exemplo de uma regra para deteção de relação de nascimento no XIP 3.16

```
1 if((SUBJ[ELIPS](#2[lemma:nascer],#1)  || SUBJ(#2[lemma:nascer],#1[human])
2           || SUBJ(#2[lemma:nascer],#1[lemma:eu])  )
3
4 EVENT[lex=+](#2,##noun#6[surface:nascimento])
```

1.3 Objetivos do Trabalho

Este trabalho tem como ponto de partida a cadeia de processamento de língua natural do L^{2}F - STRING. A STRING possui alguns dos mecanismos necessários para extração de informação estruturada acerca de entidades presentes num documento único. Nesta dissertação descreve-se a abordagem

tomada para agregar e reestruturar a informação acerca de entidades proveniente da STRING. Para agregar a informação acerca de entidades mencionadas foi necessário implementar um método de resolução de correferências num documento. É ainda descrito o trabalho de integração realizado por forma a ser possível utilizar dois módulos (Resolução de Anáfora e Normalização de Expressões Temporais) previamente desenvolvidos para a STRING. Estes módulos foram desenvolvidos no contexto de duas dissertações anteriores [Nobre, 2011] [Maurício, 2011] mas não estavam ainda integrados na cadeia de processamento.

1.4 Estrutura do Documento

Este documento encontra-se estruturado do seguinte modo: o capítulo 2 descreve alguns trabalhos relacionados e está dividida em 3 secções. A secção 2.1 descreve o sistema *OpenCalais*, que é um exemplo de um sistema comercial que realiza o tipo de tarefas que se pretende para o sistema a desenvolver neste trabalho; a secção 2.2 descreve algumas técnicas e trabalhos relacionados com desambiguação de entidades mencionadas recorrendo a bases de conhecimento; e, por fim, a secção 2.3 descreve a tarefa de preenchimento de *slots* definida para a *Text Analysis Conference 2011* (TAC11) e ainda alguns dos sistemas que nela participaram. No capítulo seguinte (3), descreve-se o *output* da STRING. São também descritas as alterações efetuadas nos módulos já existentes e, por fim, é descrito o módulo desenvolvido no decorrer deste trabalho. No capítulo 4, são descritos os métodos utilizados para avaliar o sistema desenvolvido bem como os resultados desta mesma avaliação. Por fim, no último capítulo apresentam-se as conclusões do trabalho e algumas sugestões de trabalho a realizar por forma a melhorar este sistema.

Capítulo 2

Trabalho Relacionado

Este capítulo descreve alguns trabalhos que se relacionam com este estudo. Em alguns dos trabalhos descritos, devido à natureza comercial dos mesmos, não foi possível obter informações mais precisas sobre o modo como o sistema realiza as diversas tarefas. Na secção 2.1 descreve-se o sistema OpenCalais[1] quanto às suas funcionalidades consideradas mais relevantes para este trabalho. Na secção seguinte (2.2), são descritas algumas técnicas de desambiguação de entidades mencionadas que fazem uso de bases de conhecimento. Na secção 2.3, descrevem-se alguns estudos realizados para a *Text Analysis Conference 2011* (TAC11), com especial ênfase na tarefa de Preenchimento de *Slots*.

2.1 OpenCalais

O OpenCalais é um projeto desenvolvido pela Thomson Reuters[2] com a intenção de construir um sistema capaz de eliminar a necessidade de anotação (*tagging*) manual por parte dos produtores de conteúdos na Internet.

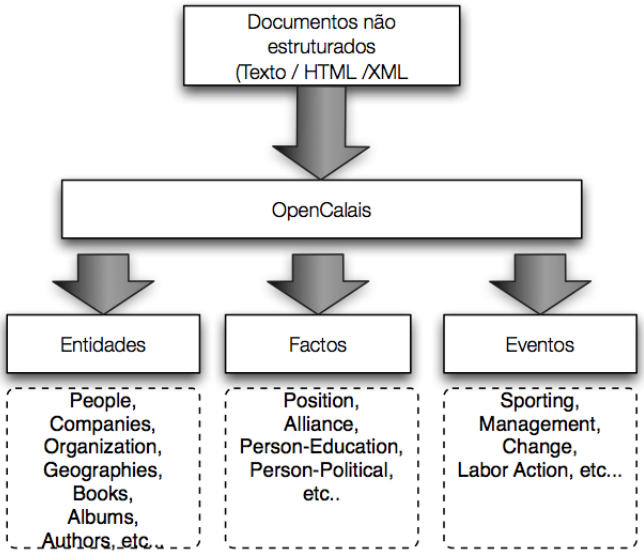

Figura 2.1: Fluxo de dados do OpenCalais

Trata-se de um sistema capaz de processar textos e associar aos mesmos metadados de natureza semântica. O sistema analisa os conteúdos submetidos usando métodos de PLN e de *Aprendizagem*

[1] http://www.opencalais.com (data de acesso: 4/1/2012)
[2] http://www.thomsonreuters.com (data de acesso: 4/1/2012)

Person	**Person**: Nome da pessoa.
	PersonType: Classificação da área de atividade principal da pessoa, de acordo com os seguintes tópicos: desporto, entretenimento, política, economia, militar. Se não for possível classificar é atribuído o valor "N/A".
	Nationality: Nacionalidade da pessoa se for mencionada no documento, caso contrário é atribuído o valor "N/A".

Tabela 2.1: Atributos da entidade *Person*

Company	**Company**: Nome da empresa.
	Nationality: Nacionalidade da empresa se for mencionada no documento, caso contrário é atribuído o valor "N/A".

Tabela 2.2: Atributos da entidade *Company*

Automática e identifica entidades (Pessoa, Empresa, etc.), eventos (pessoa P foi contratada pela empresa E) e factos (pessoa P trabalha na empresa E). Estes últimos representam relações entre entidades. Na realidade, não se verificam diferenças entre os dois conceitos (eventos e factos), pelo que podem ambos ser tratados apenas como relações. A figura 2.1 mostra o fluxo de dados no sistema e alguns exemplos de entidades, factos e eventos identificados pelo sistema.

Os eventos e factos predefinidos no sistema são, normalmente, relativos a domínios específicos: Fusões e Aquisições (*Mergings and Acquisitions*) para o domínio de Negócios (*Business*); Estreia de Filmes (*MovieRelease*) para Notícias de Entretenimento, etc. No entanto, o sistema também possibilita a identificação de relações genéricas, denominadas de *Generic Relations*, as quais não têm associado qualquer domínio específico. As *Generic Relations* são uma tentativa de identificar todas as relações do tipo Sujeito-Predicado-Objecto, em que, ou o sujeito, ou o objeto, têm de ser uma entidade mencionada já reconhecida pelo sistema (*Person, Company, TVShow*, etc).

O OpenCalais, à data atual, identifica cerca de quarenta tipos diferentes de entidades e cerca de uma centena de relações (factos e eventos). As entidades mais relevantes para este trabalho são as entidades do tipo Pessoa (*Person*), Empresa (*Company*) e Organização (*Organization*). As tabelas 2.1, 2.2 e 2.3 mostram os atributos associados a entidades dos tipos *Person*, *Company* e *Organization*, respectivamente. A tabela 2.4 mostra os atributos associados à relação *EmploymentRelation* no OpenCalais.

Quando identificada uma entidade, esta recebe uma pontuação de relevância (*Entity Relevance Score*[3]). Esta pontuação representa a importância atribuída pelo sistema a essa entidade e é usada durante o processo de desambiguação de entidades. O valor de pontuação atribuído varia entre 0 e 1 (em que "1"significa mais relevante). A pontuação de determinada entidade em diferentes textos pode ser usada para determinar a relevância da mesma não só num documento mas numa coleção de documentos.

[3]Não foi possível encontrar informação relativa ao método de cálculo desta pontuação.

Organization	**Organization**: Nome da organização.
	OrganizationType: Classificação da área de actividade principal da organização, de acordo com os seguintes tópicos: desporto, partido político, governamental militar, governamental civil. Se não for possível classificar é atribuído o valor "N/A".
	Nationality: Nacionalidade da organização se for mencionada no documento, caso contrário é atribuído o valor "N/A".

Tabela 2.3: Atributos da entidade *Organization*

Relação de Emprego (*EmploymentRelation*)	**Person_Employer**: Nome do empregador.
	Person_Employee: Nome do empregado.
	Position: Posto que ocupa ou vai ocupar.
	Date: Expressão temporal que descreve a data de contratação.
	DateString: Data de contratação normalizada.

Tabela 2.4: Atributos da relação *EmploymentRelation*

Durante o processo de identificação de entidades podem surgir diferentes tipos de ambiguidade. Por exemplo, a mesma entidade pode ser referida de diferentes formas (*Apple*, *Apple Inc.*) ou entidades distintas podem ser referidas pela mesma expressão (*George Bush* e *George W. Bush*); podem igualmente ocorrer erros de ortografia (*Goerge* em vez de *George*). Para realizar a tarefa de desambiguação de entidades, o OpenCalais usa um conjunto finito de identificadores de entidades, por exemplo, uma lista de nomes de empresas, quando se trata de desambiguar nomes de empresas.

Uma entidade é considerada desambiguada quando se consegue mapear a entidade num único elemento do conjunto de identificadores de entidades. No entanto, existem situações onde este mapeamento não é possível pelo que é escolhido o identificador de entidade que é tido como o mais provável (neste ponto é usada a *Entity Relevance Score*). Para realizar esta tarefa, o sistema recorre à expressão que designa a entidade e, se necessário, à informação contextual presente no texto.

Atualmente, o OpenCalais suporta três tipos distintos de desambiguação, a desambiguação geográfica (cidades, províncias ou estados e países), a desambiguação de produtos eletrónicos (resolve diminutivos de nomes de produtos e também variações dos mesmos recorrendo ao catálogo da Shopping.com como lista de identificadores) e a desambiguação de empresas. Quando o sistema consegue desambiguar com sucesso uma entidade, devolve um conjunto de informações de resolução. Usando como exemplo o caso do nome de uma empresa, o sistema devolve o nome formal da empresa, um URI (*Uniform Resource Identifier*) que é único em todos os documentos e, para os casos de uma empresa pública, o símbolo que identifica a empresa em bolsa. Caso a desambiguação não seja possível, o sistema não devolve qualquer informação de resolução.

Quanto ao *output* do sistema, os metadados gerados são devolvidos sob a forma de construções RDF (Resource Description Framework[4]), que são um formato de metadados-padrão para representar informação na Internet.

2.2 Desambiguação de Entidades Mencionadas com Recurso a Bases de Conhecimento

As técnicas de Desambiguação de Entidades Mencionadas com recurso a bases de conhecimento, tais como o YAGO[5], o Freebase[6], o DBpedia[7] ou o GeoNames[8], que são bases de conhecimento que extraem relações da Wikipédia[9], consistem no mapeamento entre determinada entidade mencionada (referência) num texto e uma entidade canónica (conceito) presente numa base de conhecimento. Se este mapeamento

[4]http://www.w3.org/RDF (data de acesso: 16/12/2011)
[5]http://www.mpi-inf.mpg.de/yago-naga/yago/ (data de acesso: 4/1/2012)
[6]http://www.freebase.com (data de acesso: 4/1/2012)
[7]http://www.dbpedia.org/About (data de acesso: 4/1/2012)
[8]http://www.geonames.org (data de acesso: 4/1/2012)
[9]http://www.wikipedia.org/ (data de acesso: 4/1/2012)

for único, então a referência pode considerar-se desambiguada. Para melhor demonstrar o problema que se pretende resolver através destas técnicas, tome-se como exemplo a seguinte frase:

*"**Fonda** protagonizou Easy Rider em 1969."*

Fonda pode ser uma referência para *Henry Fonda* ou *Peter Fonda* ou até *Jane Fonda*. Logo, é necessário selecionar apenas uma destas pessoas.

Existem várias estratégias para a escolha do conceito correspondente a uma dada referência. As estratégias mais simples consistem na escolha do conceito baseada num critério de relevância ou *popularidade*, como, por exemplo, o conceito com maior artigo na Wikipédia ou, para o caso de uma cidade, a que tem maior número de habitantes. Isto permite evitar os conceitos menos frequentes. No entanto, pode dar-se o caso de a referência a classificar ser na realidade um desses conceitos menos frequentes, pelo que nesses casos a desambiguação falha. Este tipo de estratégias pode ser visto como uma *baseline* e não tem em conta o contexto em que a referência surge, pelo que uma possível melhoria da performance passa pela utilização do contexto envolvente da entidade mencionada.

O contexto de uma dada referência pode ser comparado com o contexto dos potenciais conceitos para que assim seja possível escolher apenas um mapeamento. O contexto do conceito pode ser, por exemplo, o artigo da Wikipédia em que esse conceito é descrito. Uma das técnicas usadas consiste em utilizar os contextos tanto da referência como dos potenciais conceitos, representando cada um dos contextos como um *bag-of-words*[10] e utilizando algum tipo de critério de comparação, como a *similaridade de cosseno*[11] ou a *distância de Jaccard*[12], para obter uma medida de similaridade entre os contextos, sendo então escolhido o conceito com maior valor de similaridade.

No trabalho de Bunescu e Pasca [Bunescu and Pasca, 2006] encontra-se um exemplo deste tipo de estratégias. Neste trabalho, o problema de desambiguação de entidades é abordado como um problema de ordenação de candidatos (*Rank*). Esta abordagem consiste em utilizar uma função de avaliação para ordenar os candidatos, sendo escolhido o que apresentar um valor mais elevado segundo esta função. A função de avaliação utilizada neste trabalho, baseia-se na *similaridade de cosseno* para comparar os contextos das referências e dos conceitos, que, neste caso, são extraídos da Wikipédia. Por forma a tentar melhorar os resultados obtidos com esta abordagem, os autores, refinaram este método através de aprendizagem. Os autores desenvolveram para isso uma nova função de avaliação para utilizar em conjunto com a anterior. Esta nova função, baseia-se na correlação existente entre determinadas palavras e as categorias da Wikipédia (todos os artigos do Wikipédia pertencem pelo menos a uma categoria; eg. John Towner Williams pertence a: *American film score composers, 21st-century classical composers,* etc.), e o seu treino é feito recorrendo ao SVM^{light} [Joachims, 1999]. Quanto aos resultados obtidos, utilizando apenas a *similaridade de cosseno*, os autores reportam valores de precisão entre os 55,4% e os 82,3%. Com a segunda abordagem, que combina as duas funções de avaliação, os valores situam-se entre os 68% e os 84,8%.

Em [Milne and Witten, 2008] as referências não ambíguas são usadas como contexto por forma a desambiguar as referências ambíguas. O classificador desenvolvido utiliza a popularidade e a similaridade de contextos como parâmetros, e é treinado através de métodos de Aprendizagem Automática por forma a balancear o peso de cada um dos parâmetros. Este treino do classificador permite que os parâmetros possam ter pesos diferentes dependendo do documento a classificar. Existe ainda um ajustamento que é efetuado a nível dos contextos, aos quais são atribuídos pesos que representam o seu grau de relevância para a desambiguação. Para este efeito, é utilizada uma função (*relatedness*) que mede o grau de relacionamento entre dois contextos: quanto mais relacionados forem, maior é o valor deste parâmetro. Este

[10] http://en.wikipedia.org/wiki/Bag_of_words_model (data de acesso: 5/1/2012)
[11] http://en.wikipedia.org/wiki/Cosine_similarity (data de acesso: 30/12/2011)
[12] http://en.wikipedia.org/wiki/Jaccard_index (data de acesso: 30/12/2011)

parâmetro é útil, pois reduz a computação necessária ao eliminar contextos que não são relevantes para a desambiguação. Os autores reportam que, no pior caso, conseguem obter uma precisão de 88%.

Ambas as estratégias descritas anteriormente tratam as várias referências presentes no mesmo documento independentemente umas das outras, não considerando qualquer tipo de relação entre elas. Quando se considera apenas um documento, é possível que exista *coerência semântica* relativa aos assuntos presentes (i.e. o documento é relativo a apenas um assunto ou um pequeno número de assuntos dentro do mesmo domínio). Este facto pode ser tido em conta para a tarefa de desambiguação, pois, implica que exista algum tipo de relação entre as entidades presentes no mesmo texto.

No trabalho de Kulkarni et al. [Kulkarni et al., 2009] é apresentado um método que faz uso das três estratégias apresentadas (*popularidade, contexto* e *coerência semântica*) e formula o problema como um problema de otimização, em que as equações combinam a popularidade com similaridade de contextos e a coerência semântica. A formulação do problema contempla ainda um parâmetro ajustável para decidir não etiquetar determinada referência por forma a aumentar a precisão (evitando erros de desambiguação) em detrimento da cobertura (*recall*). Para resolver o problema usam-se duas abordagens algorítmicas, uma através de programação linear (*Linear Programing*[13]) e outra através de um algoritmo de procura local, o *Hill-Climbing*[14].

O sistema AIDA [Hoffart et al., 2011] é outro caso de utilização das três estratégias acima descritas. No entanto, no AIDA é utilizado um algoritmo de coerência baseado em grafos. É construído um grafo, pesado e não direcionado, de referência-entidade no qual os nós são referências e entidades da base de conhecimento, e os arcos são de 2 tipos: arcos referência-entidade, com pesos calculados recorrendo à medida de similaridade ou uma combinação da medida de similaridade e de popularidade; arcos entidade-entidade, para os quais o peso é calculado recorrendo às sobreposições de *links* de redirecionamento na Wikipédia (capturam as relações de coerência entre entidades). O objetivo do algoritmo de coerência é obter um grafo no qual só exista um arco referência-entidade a ligar cada referência a uma única entidade da base de conhecimento, ficando desta forma desambiguada a referência em questão. Este trabalho foi testado e comparado com outros sistemas entre os quais [Kulkarni et al., 2009]. Este sistema conseguiu obter valores de precisão na ordem dos 80% ao passo que [Kulkarni et al., 2009] conseguiu sensivelmente 75%.

No trabalho de Anastácio et al. [Anastácio et al., 2011] o problema da desambiguação de entidades é abordado como um problema de *ranks* normalmente conhecido como *Learning to Rank* [Liu, 2009]. Este trabalho foi um dos participantes na *Text Analysis Conference 2011* (TAC11), na tarefa de Desambiguação de Entidades Mencionadas para a língua inglesa e conseguiu valores de precisão na ordem dos 80%. Os autores utilizam uma arquitetura, comum em sistemas de desambiguação de entidades, constituída por 5 módulos, os quais são descritos sucintamente de seguida:

Expansão de *query* (*Query expansion*): Módulo responsável por identificar os vários nomes pelos quais uma entidade pode ser referida num texto (*General Electric, (GE)*; *SMART Comunications, SMART*, etc.). Foram utilizados métodos baseados em padrões (séries de palavras começadas com letra maiúscula seguidas de uma expressão entre parêntesis, etc.).

Geração de Candidatos (*Candidate generation*): Módulo responsável por filtrar os conceitos da base de conhecimento (neste caso a Wikipédia) que podem corresponder à referência. Esta filtragem é feita através de critérios de similaridade de *strings*, tendo sido utilizada uma versão modificada da *similaridade de cosseno* como métrica de comparação. A modificação efetuada consiste em utilizar *n-gramas*, em que n varia entre 1 e 4, em vez de palavras para fazer a comparação.

[13] http://en.wikipedia.org/wiki/Linear_programming (data de acesso: 30/12/2011)
[14] http://en.wikipedia.org/wiki/Hill_climbing (data de acesso: 30/12/2011)

Ordenação de candidatos (*Candidate ranking*): Módulo responsável por ordenar os candidatos, selecionados anteriormente, de acordo com a probabilidade de serem a correspondência correta para a referência que se pretende desambiguar. Neste trabalho este módulo foi baseado numa abordagem *Supervised Learning to Rank*. Foram testados 5 métodos já desenvolvidos por outros autores: *Ranking SVM* [Joachims, 2002], *Ranking Perceptron* [Collins and Duffy, 2002], *Pairwise Learning to Rank Methods* e a *ListNet* [Cao et al., 2007], *Coordinate Ascent* [Metzler and Croft, 2007] e *AdaRank* [Xu and Li, 2007]. Por fim os candidatos melhor classificados são filtrados através de um classificador supervisionado que deteta as referências sem correspondência (*nil query*).

Validação de candidatos (*Candidate validation*): Este módulo é responsável por decidir se o candidato melhor classificado pelo módulo anterior é um erro, resultante de a referência não ter correspondência na base de conhecimento (*nil query*). As abordagens tipicamente usadas para este módulo consistem: a) na utilização de um limite inferior para o valor da pontuação atribuída pelo módulo de ordenação de candidatos, limite abaixo do qual a correspondência é considerada errada; b) utilização de um modelo de classificador treinado para este efeito; c) aplicação de um esquema de votação. Neste trabalho, a tarefa executada normalmente por este módulo foi delegada no módulo anterior.

Resolução de entidades sem correspondência (*Nil entity resolution*): Módulo responsável por agrupar as referências sem correspondência e devolver um resultado para as mesmas. Neste trabalho é construído um grafo constituído pelas referências sem correspondência como nós e os arcos iniciais estimados através de um classificador treinado para encontrar duplicados nas referências. Por fim é aplicado o fecho transitivo ao grafo para agrupar as referências (se A é B e B é C então A é C).

Todos os sistemas descritos anteriormente foram concebidos para desambiguação de entidades mencionadas em textos escritos em Inglês.

2.3 Preenchimento de *Slots*

O desafio proposto na *Text Analysis Conference 2011* (TAC11) consistiu em descobrir informação acerca de entidades mencionadas num texto (escrito em Inglês) e incorporar essa informação numa base de conhecimento. Estas bases de conhecimento (*frames*) são compostas por atributos (*slots*) relativos à entidade em questão. Cada tipo de entidade tem, portanto, uma *frame* que lhe está associada, e, por sua vez, cada *frame* é constituída por um conjunto de *slots*. A descrição anterior refere-se apenas à tarefa de Preenchimento de *Slots*, no entanto, na TAC11 foram definidas várias outras tarefas entre as quais: Desambiguação de Entidades Mencionadas (*Entity Linking*), Preenchimento de *Slots* Regulares e Preenchimento de *Slots* Temporais (consiste em incluir informação temporal, como data de início e data de fim, em alguns dos *slots*). Nesta dissertação aborda-se apenas o Preenchimento de *Slots* Regulares doravante designada por PS.

Para a tarefa de PS na TAC11 foram considerados dois tipos de entidades, Pessoa (*Person*) e Organização (*Organization*). Quanto aos *slots* das *frames* destas entidades, foram definidos dois tipos distintos: os de valores simples (*single-valued*) e os de valores em lista (*list-valued*). Os *slots* do tipo *single-valued* (e.g. idade) são atributos de valor único, enquanto os *slots list-valued* (e.g. membros da família) podem apresentar múltiplos valores, nomeadamente uma lista de valores. A tabela 2.5 apresenta todos os atributos do tipo *single-valued*, que foram definidos para a TAC11, enquanto a tabela 2.6 apresenta todos os atributos do tipo *list-valued*.

Quanto aos recursos fornecidos para o PS na TAC11, foi disponibilizado um *corpus* constituído por um conjunto de documentos a partir dos quais os sistemas devem extrair informação, uma base de

Person	**Organization**
per:date_of_birth	org:number_of_employees/members
per:age	org:founded
per:country_of_birth	org:dissolved
per:stateorprovince_of_birth	org:country_of_headquarters
per:city_of_birth	org:stateorprovince_of_headquarters
per:date_of_death	org:city_of_headquarters
per:country_of_death	org:website
per:stateorprovince_of_death	-
per:city_of_death	-
per:cause_of_death	-
per:religion	-

Tabela 2.5: *Slots* de valor simples (*sigle-valued slots*)

Person	**Organization**
per:alternate_names	org:alternate_names
per:origin	org:political/religious_affiliation
per:countries_of_residence	org:top_members/employees
per:stateorprovince_of_residence	org:members
per:cities_of_residence	org:member_of
per:schools_attended	org:subsidiaries
per:title	org:parents
per:member_of	org:founded_by
per:employee_of	org:shareholders
per:spouse	-
per:children	-
per:parents	-
per:siblings	-
per:other_family	-
per:charges	-

Tabela 2.6: *Slots* de valor em lista (*list-valued slots*)

conhecimento de referência (inicial) derivada das *infoboxes* da Wikipedia e várias *queries* que representam a entidade sobre a qual deve ser preenchida a informação. As *queries* fornecidas eram compostas pelos seguintes elementos: nome e tipo da entidade; um documento do *corpus* no qual a entidade aparece (para desambiguar no caso de existirem múltiplas entidades com o mesmo nome); o ID do nó (se a entidade aparece na base de conhecimento); e os atributos que não precisam ser preenchidos. A listagem 2.1 é um exemplo de uma das *queries* referidas anteriormente.

Listagem 2.1: Exemplo de uma *Query* da TAC11

```
<query id=SF114>
        <name>Msi Oka</name>
        <docid>eng-WL-11-174592-12943233</docid>
        <enttype>E0300113</enttype>
        <ignore>per:date_of_birth per:country_of_birth</ignore>
</query>
```

Segue-se a descrição dos dois trabalhos, que apresentaram melhores resultados (apenas se apresentam os aspetos relativos à tarefa de Preenchimento de *Slots* Regulares).

2.3.1 New York University

O sistema que conseguiu melhores resultados a nível de cobertura e medida F na TAC11 foi o apresentado pela *New York University* [Sun et al., 2011b]. Este sistema foi construído refinando e acrescentando algumas inovações àquele que participou na TAC10[15] [Grishman and Min, 2010]. O sistema desenvolvido possui 3 componentes básicos: componente de recolha de documentos; componente de extração de respostas; componente de agregação.

O componente de recolha de documentos utiliza a ferramenta *Lucene*[16] para recolher documentos do *corpus*. A ferramenta *Lucene* utiliza o nome da entidade, presente na *query*, e ainda algumas variações do mesmo (omissão de iniciais a meio de nome, omissão de sufixos de empresa, etc.) para recolher documentos onde este surja.

Antes da passagem o próximo componente do sistema, é feito algum pré-processamento, sendo para isso utilizado o *Java Extraction Toolkit*[17] - JET. Esta ferramenta efetua uma série de tarefas de análise de texto (etiquetação morfossintática, análise sintática, etiquetação de nomes, etiquetação de expressões temporais, e análise de correferências). Em paralelo é também utilizada outra ferramenta, o *Stanford parser*[18], que gera árvores de dependências denominadas *Stanford dependency trees*. Estas são utilizadas mais adiante, para emparelhamento de padrões.

O componente de extração de respostas utiliza os resultados da análise de correferência para preencher o *slot alternate_names*. Os outros *slots* são preenchidos recorrendo a padrões especificados manualmente e ainda a alguns padrões criados semiautomaticamente (através de um método de geração).

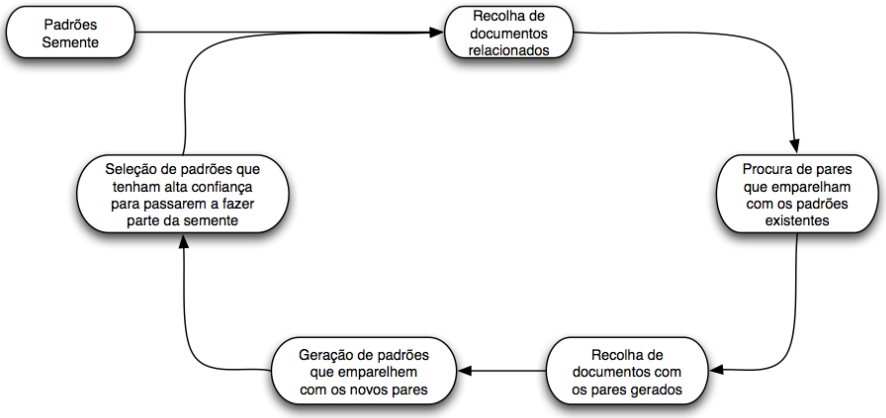

Figura 2.2: Procedimento de geração de padrões

Os padrões manuais são construídos para serem aplicados a pares de nomes ou pares de grupos de nomes, ligados através de uma preposição. Algumas das relações que são usadas para a construção manual de padrões são: títulos que surgem antes dos nomes (*"General Smith"*); certos substantivos (*"linguist Fred Smith"*); nomes (*"Ford president Smith"*); pronomes possessivos (*"Ford's president Smith"*) e estruturas do tipo *x of y* (*"president of Ford"*). Os padrões manuais construídos recorrendo a estas relações são usados diretamente para o preenchimento dos seguintes *slots* de entidades do tipo pessoa: *title*; *employee_of*; *parent*; *spouse*; *children*; *other_family* e *origin*. Quanto aos *slots* do tipo organização, os padrões manuais são usados para o preenchimento de: *top_members*; *employees*; *parent* e *subsidiary*.

A figura 2.2 resume o processo de construção de padrões através de um método semelhante ao *Snowball* [Agichtein and Gravano, 2000], com a diferença de que são dados padrões e não pares como sementes.

[15] *Text Analysis Conference 2010*
[16] http://lucene.apache.org/java/docs/index.html (data de acesso:19/12/2012)
[17] http://cs.nyu.edu/grishman/jet/license.html (data de acesso: 30/12/2011)
[18] http://nlp.stanford.edu/software/lex-parser.shtml (data de acesso: 30/12/2011)

Os padrões-semente são um pequeno grupo de padrões de alta precisão para cada *slot* (exemplo: *", also known as"* para o *slot alternative_names* de uma entidade do tipo Pessoa). Os padrões e pares são classificados com uma medida aproximada, que permite ao editor (pessoa responsável por validar os padrões gerados) ter algum indício da eficácia de determinado padrão, para que desta forma possa selecionar os padrões que apresentem maior valor de confiança. Os padrões gerados e aceites, são utilizados nas iterações seguintes do ciclo de geração de padrões. A avaliação de confiança dos padrões gerados é definida da seguinte forma:

$$Conf(P) = \frac{P.positivo}{P.positivo + P.negativo}$$

$$P.positivo = \sum_{i=1}^{m} Conf(T_i)$$

$$P.negativo = n.\beta$$

m - Número de vezes que o padrão P emparelha um par que faz parte do semente + número de vezes que emparelha um par no conjunto de pares extraídos nas iterações anteriores.

$Conf(T_i)$ - Representa a confiança de um par.

n - Representa o número de vezes que o padrão P emparelha com um par que não consta no grupo de pares corretos.

β - É um fator de penalização por ter sido emparelhado um par errado. Os autores indicam que por uma questão de conveniência ajustaram este parâmetro para o valor 1.0.

Os padrões são ordenados pela seguinte métrica:

$$Conf_{RlogF}(P) = Conf(P) \times log_2(P.positivo)$$

A confiança dos novos pares de entidade-valor é calculada recorrendo à seguinte fórmula:

$$Conf(T) = 1 - \prod_{i=0}^{|P|}(1 - Conf(P_i))$$

A confiança inicialmente atribuída a todos os padrões semente é 1.0. A precisão do valor produzido por *Conf(P)* depende da completude do conjunto de pares corretos atribuídos a cada *slot*. Isto significa que, dado um padrão para o qual não existe o número suficiente de pares corretos, o valor de confiança desse padrão é subestimado.

Quanto à geração de novos padrões é realizada à custa do processamento do texto em torno dos pares identificados. Este processamento consiste em identificar sequências de palavras e sinais de pontuação

antes, entre e depois dos pares. Os padrões gerados, são vetores com o seguinte formato: <*esquerda*, entidade1, *meio*, entidade2, *direita*>, em que *esquerda*, *meio* e *direita* são segmentos de texto (tal como o nome indica são segmentos de texto que se situam à esquerda, entre e à direita dos pares de entidades). Para um determinado par, é possível e desejável, que existam várias instâncias destes vetores, pelo que o passo final da construção de um padrão passa por agrupar várias instâncias semelhantes (método de cálculo da semelhança e agregação de vetores descrito detalhadamente em [Agichtein and Gravano, 2000]).

Todos os componentes e métodos descritos até agora já faziam parte do sistema desenvolvido para a TAC10. Para a TAC11 foi desenvolvido um conjunto de classificadores de máxima entropia, treinados através de um método de supervisão distante (*distant supervision* [Mintz et al., 2009]). O processo de treino destes classificadores usa a Freebase e o *corpus* fornecido para a TAC11 e foram utilizadas cerca de 4.1 milhões de instâncias de relações para o treino dos classificadores. Estas relações da Freebase foram mapeadas em 29 *slots* distintos. Quanto à definição de exemplo positivo e negativo, foi feita da seguinte forma:

Dado um par de nomes <a,b>:

-se <a,b> for uma instância de uma relação presente na Freebase, então, <a,b> é um exemplo positivo.

-se <a,b> não for uma instância de uma relação presente na Freebase mas <a,b'> for, para algum b' ! = b, então, <a,b> é um exemplo negativo.

O critério de seleção das *features* utilizadas para treinar os classificadores, é baseado no trabalho de [Sun et al., 2011a]. As *features* são extraídas a partir das sequências de *tokens* e das árvores de dependências.

Devido ao método de classificação de exemplos negativos e positivos, são gerados casos de falsos negativos e falsos positivos, pelo que os autores aplicam um método de refinamento de etiquetagem das classes. Este método passa pela extração de padrões de dependência, a partir dos exemplos, e por estimar a precisão de um padrão para uma classe baseando-se nas estatísticas das classes de etiquetagem geradas. A precisão de um padrão p para uma classe c_i é então definida da seguinte forma:

$$prec(p, c_i) = \frac{count(p, c_i)}{\sum_j count(p, c_j)}$$

Na qual c_j representa qualquer uma das classes definidas.

O método de supervisão distante produz uma distribuição de classes desequilibrada. Os autores afirmam que, através de métodos empíricos, concluíram que um classificador de n classes para todos os tipos de pares de entidades tende a estar enviesado para um pequeno número de classes que representam a maioria, ignorando por isso as classes que representam minorias. Para minimizar esta situação, foi introduzido um método de *undersampling* das classes maiores que consiste em treinar classificadores de n classes diferentes para cada tipo de pares de entidades (o classificador para o par PESSOA e ORGANIZAÇÃO é diferente do classificador para o par PESSOA e PESSOA). Este método minimiza o problema inicial, no entanto, continuam a existir classes de maioria para cada um destes classificadores, por isso, o número destas classes é reduzido, selecionando aleatoriamente subgrupos de exemplos a partir delas.

A última das melhorias introduzidas no sistema para a TAC11 foi a utilização da informação proveniente do módulo de correferências (mencionado no início da descrição deste sistema). Aquando do preenchimento de determinado *slot* relativo a uma entidade, são utilizados também os nomes que foram considerados como correferentes dessa mesma entidade.

Por fim, o componente de agregação é responsável pelo tratamento das respostas. Respostas diferentes, para o mesmo *slot*, podem ser provenientes de várias passagens dentro do mesmo documento ou de documentos diferentes. Estas respostas podem ser diferentes e, portanto, todas relevantes; ou podem ser equivalentes e, neste caso, apenas uma é devolvida. Para os *slots single-valued*, a resposta preferida é a que for devolvida mais vezes; caso não exista nenhuma que satisfaça este critério, é devolvida a mais longa. Também para o caso de *slots single-valued*, é feito o tratamento necessário para obedecer às regras gerais definidas para a TAC11. A especificação deste tratamento não é relevante para esta dissertação.

Este sistema obteve 25,5% de cobertura, 35% de precisão e uma medida F de 29,5%.

2.3.2 UCD IIRG

O sistema [Byrne and Dunnion, 2011] desenvolvido pelo *Intelligent Information Retrieval Group* da *University College Dublin* foi o que apresentou melhores resultados a nível de precisão. Tal como o sistema descrito anteriormente, este foi construído com base no sistema apresentado na TAC10.

A arquitetura deste sistema baseia-se numa arquitetura típica de um sistema de Resposta automática a Perguntas - RAP (*Question Answering*) descrita em [Pasca, 2003] sendo esta constituída por 3 módulos principais: o Módulo de Processamento de *Queries* (*Query Processing*); o Módulo de Recolha de Documentos (*Passage Retrieval*); e o Módulo de Seleção de Valor-*Slot* (*Slot-Value Selection*). Segundo os autores, é possível abordar o problema do preenchimento de *slots* pois esta pode ser vista como uma especialização da tarefa de resposta a perguntas. A *query* fornecida corresponde à pergunta ou conjunto de perguntas e o valor de cada *slot* corresponde a uma resposta única a essa ou essas perguntas.

Numa primeira fase do desenvolvimento do sistema, os dados da TAC09[19] e da TAC10 foram usados como dados de treino para a criação de padrões. Estes dados de treino foram construídos anotando as ocorrências da entidade-alvo e os valores dos *slots*, tendo sido feita esta etiquetação ao nível da frase. Durante a extração dos dados de treino, os autores, detetaram que era frequente as respostas relevantes para o preenchimento de um dado *slot* ocorrerem a curta distância da entidade-alvo a que se referiam, ou seja, toda a informação relevante estava presente num curto segmento de texto. Devido a este facto, os padrões candidatos foram gerados em volta do valor do *slot* presente nestes segmentos. Adicionalmente, e devido à similaridade das tarefas de PS e de RAP, os *slots* foram convertidos em questões equivalentes usadas num sistema de RAP. Para o *slot* per:age foram definidas as seguintes perguntas: *"When was <entidade-alvo> born"*; *"How old is <entidade-alvo>"*; *"What age is <entidade-alvo>"*; entre outras. Estas perguntas foram usadas para gerar mais padrões candidatos. Os autores identificaram ainda um parâmetro adicional, o tipo de valor esperado para um *slot*, que foi associado aos padrões candidatos (*slot* do tipo per:age deve ser preenchido com um número, um *slot* do tipo org:website deve ser preenchido com um URL, etc.).

O módulo de Recolha de Documentos funciona em duas fases. Na primeira, identificam-se passagens (segmentos de texto ou mesmo documentos inteiros) nas quais é mais provável existir a resposta que se procura. Na segunda fase, faz-se uma redução do conjunto de passagens gerado anteriormente. Esta redução é feita processando o conjunto de passagens com o *Stanford NER core pipeline*[20]. Em cada frase são anotadas as ocorrências da entidade-alvo. De seguida, são selecionadas as frases onde ocorre mais vezes a entidade alvo. As frases selecionadas são passadas como *input* ao módulo seguinte.

O último módulo deste sistema (Módulo de Seleção de Valor-*Slot*) seleciona e extrai o segmento de texto que tem a maior probabilidade de ser a resposta para o preenchimento do *slot*. Este módulo seleciona as passagens que têm o tipo de valor esperado para o *slot*, previamente identificado no módulo de Processamento de *Queries*. Depois, seleciona e adiciona a um conjunto de resposta as passagens mais

[19] Text Analysis Conference 2009

[20] http://nlp.stanford.edu/software/corenlp.shtml (data de acesso: 30/12/2011)

pequenas que emparelhem com os padrões candidatos. O conjunto de resposta é, então, ordenado por frequência, sendo preferidas as respostas que surgem num maior número de documentos distintos. Por fim é feito algum tratamento às respostas por forma a obedecerem às regras da TAC11 (este tratamento, no entanto, não é relevante para esta dissertação).

Este sistema obteve 12,6% de cobertura, 49,2% de precisão e uma medida F de 20,1%.

Capítulo 3

Arquitetura da Solução

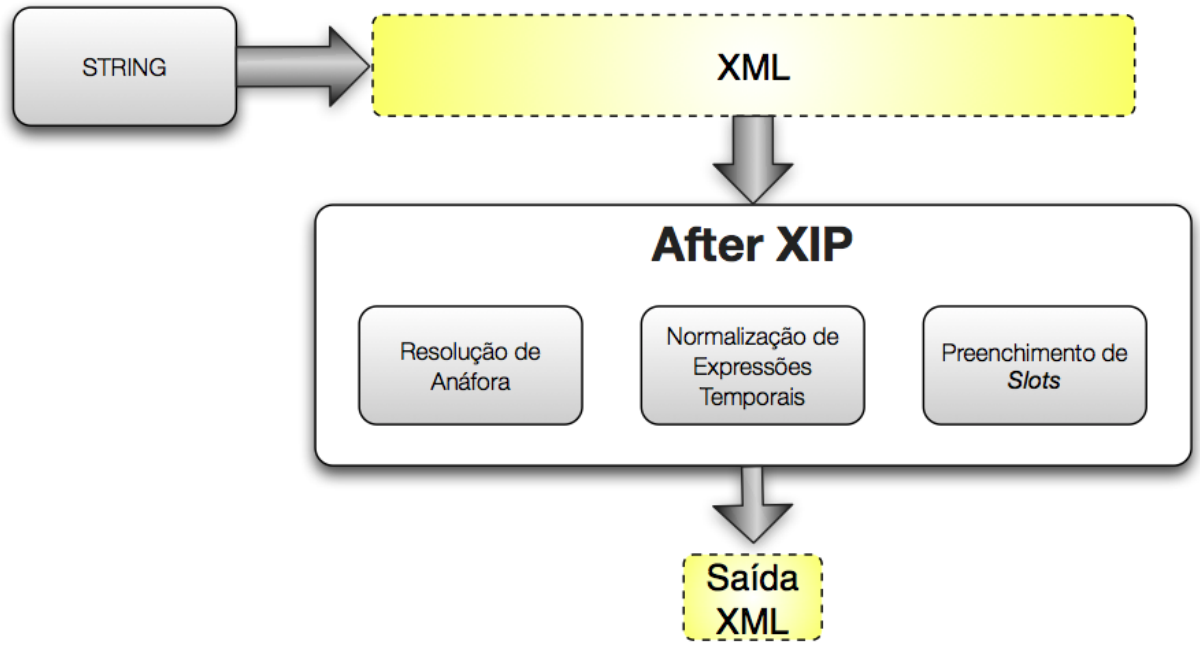

Figura 3.1: Módulos de processamento Pós-XIP

Este capítulo descreve a solução implementada para resolver o problema de extração de informação estruturada a partir de textos escritos em língua portuguesa. A figura 3.1 apresenta um diagrama da solução implementada, que contém os três sub-módulos envolvidos nas tarefas de extração de informação (módulos de Resolução de Anáfora, Normalização de Expressões Temporais e Preenchimento de *Slots*).

A secção 3.1 descreve os tipos de eventos que já são detetados pela STRING [Baptista et al., 2012]. Estes eventos vêm expressos no XML de saída da cadeia e esta saída é utilizada como entrada dos seguintes módulos pertencentes ao AfterXIP. Na secção seguinte, são descritos os sub-módulos que compõem o módulo AfterXIP. O módulo desenvolvido para esta dissertação é o módulo de Preenchimento de *Slots* - *slot-filling* (descrito na secção 3.2.3), tendo sido no entanto integrados os outros dois módulos anteriormente desenvolvidos, o módulo de resolução de anáfora (ARM - Anafora Resolution Module) e o módulo de normalização de expressões temporais (Time Normalization). As secções 3.2.1 e 3.2.2 descrevem sucintamente os módulos de normalização de expressões temporais e de resolução de anáfora e as alterações que foram necessárias por forma a possibilitar a sua integração e utilização para o preenchimento de *slots*.

3.1 Eventos da STRING

A saída do XIP é constituída por um ficheiro XML contendo informação morfossintática, sintática e semântica referente ao texto processado. Este ficheiro XML contém os seguintes elementos principais:

XIPRESULT - Nó raiz do documento XML que contém um conjunto de LUNITs.

LUNIT - Nó que representa uma frase do texto de entrada. É constituído por uma árvore com um nó raiz do tipo NODE e um conjunto de DEPENDENCYs.

DEPENDENCY - Representa o resultado da análise linguística feita pelo XIP sobre a árvore de NODEs. Contém nós do tipo FEATURE e PARAMETER.

NODE - Contém o resultado da análise morfológica. Pode ser um nó pai de outros elementos do tipo NODE e FEATURE entre outros.

FEATURE - Guarda as propriedades (*features*) associadas às DEPENDENCYs e aos NODEs.

PARAMETER - Contém informação acerca dos nós que estão envolvidos numa DEPENDENCY.

Para além destes constituintes principais existem ainda outros que não são relevantes para o preenchimento de *slots* pelo que não se encontram aqui descritos. Os constituintes descritos apresentam diversos tipos de atributos como *nome*, *valor*, *número*, entre outros. Nas dependências está contida a informação acerca das relações e é esta a informação que é utilizada pelo módulo de preenchimento de *slots*. A listagem 3.1 mostra um excerto do ficheiro XML de saída da STRING.

Listagem 3.1: Excerto do XML devolvido pela STRING

```
<XIPRESULT math="0">
    <LUNIT language="Portuguese">
        <NODE end="101" num="44" start="0" tag="TOP">
            <FEATURE attribute="CAT" value="0" />
            <NODE end="11" num="51" start="0" tag="NP">
                <FEATURE attribute="START" value="+" />
                <FEATURE attribute="NOUN" value="+" />
                ...
            </NODE>
            ...
        </NODE>
        <DEPENDENCY name="MAIN">
            <PARAMETER ind="0" num="47" word="Pedro Paix&#xE3;o" />
        </DEPENDENCY>
        ...
        <DEPENDENCY name="NE">
            <FEATURE attribute="T-REF-SIMULT" value="+" />
            <FEATURE attribute="T-ABSOLUT" value="+" />
            <FEATURE attribute="TEMPO" value="+" />
            <FEATURE attribute="T-DATE" value="+" />
            <PARAMETER ind="0" num="64"
                word="em o dia 7 de Fevereiro de 1956" />
        </DEPENDENCY>
        ...
    </LUNIT>
    ...
</XIPRESULT>
```

Neste trabalho pretende-se agregar informação relativa às relações entre entidades. Esta informação está contida em eventos. Estes são representados através de dependências que possuem o atributo *name* preenchido com o valor **EVENT** no XML proveniente do XIP. A definição e implementação destes

eventos está fora do âmbito destes trabalho e os seus detalhes estão descritos em [Baptista et al., 2012]. A estrutura básica destes eventos é a seguinte:

EVENT_LEX_(<*palavra*>, <*palavra-chave*>)
EVENT_<**EVENT_TYPE**>_<*features*>(<*palavra*>)
EVENT_<**RELATION_TYPE**>_<*features*>(<*palavra*>,<*arg1*>)

A dependência **EVENT_LEX** associa o elemento predicativo (*predicado*) do evento a uma palavra normalizada (*subtipo do evento*) e esta dependência existe em todos os eventos definidos. A dependência **EVENT**_<**EVENT_TYPE**> é a que define a categoria do evento, existindo 4 categorias principais (**LIFETIME, BUSINESS, LOCATION** e **PUBLIC-EVENT**; as duas últimas não são utilizadas neste trabalho). Esta dependência possui apenas um parâmetro, o elemento predicativo, e uma propriedade que é a categoria do evento. Por fim as dependências **EVENT**_<**RELATION_TYPE**> são as que contêm a informação que se pretende recolher. Estas dependências têm sempre dois parâmetros e o primeiro é o elemento predicativo e o segundo a informação a agregar. A tabela 3.1 ilustra as 2 categorias de eventos que são utilizados neste trabalho, e os tipos de dependências **EVENT**_<**RELATION_TYPE**> pertencentes a cada uma delas. A listagem 3.2 mostra a saída XML que representa uma relação de casamento entre duas pessoas.

Listagem 3.2: Excerto do ficheiro XML de saída da STRING que contem um evento do tipo **SPOUSE**.

```
<DEPENDENCY name="EVENT">
  <FEATURE attribute="LEX" value="+"/>
  <PARAMETER ind="0" num="8" word="casar"/>
  <PARAMETER ind="1" num="50" word="NOUN"/>
</DEPENDENCY>
<DEPENDENCY name="EVENT">
  <FEATURE attribute="LIFETIME" value="+"/>
  <PARAMETER ind="0" num="8" word="casar"/>
</DEPENDENCY>
<DEPENDENCY name="EVENT">
  <FEATURE attribute="2M" value="+"/>
  <FEATURE attribute="SPOUSE" value="+"/>
  <PARAMETER ind="0" num="8" word="casar"/>
  <PARAMETER ind="1" num="33" word="Jo&amp;#xE3;o Carlos Silva"/>
</DEPENDENCY>
<DEPENDENCY name="EVENT">
  <FEATURE attribute="2F" value="+"/>
  <FEATURE attribute="SPOUSE" value="+"/>
  <PARAMETER ind="0" num="8" word="casar"/>
  <PARAMETER ind="1" num="34" word="Ana Maria"/>
</DEPENDENCY>
<DEPENDENCY name="EVENT">
  <FEATURE attribute="DATE" value="+"/>
  <PARAMETER ind="0" num="8" word="casar"/>
  <PARAMETER ind="1" num="49" word="a 20 de dezembro de 1990"/>
</DEPENDENCY>
```

3.1.1 Eventos LIFETIME

A categoria de eventos **LIFETIME** está relacionada com eventos que representem marcos na vida de uma pessoa. Existem 6 tipos de eventos desta categoria, eventos de nascimento (**BIRTH**), morte (**DEATH**), idade (**AGE**), parentesco (**FAMILY**), residência (**RESIDENCE**) e educação (**EDUCATION**). Podem ser estabelecidos vários tipos de relações para estes tipos de eventos. Nas secções seguintes serão detalhados cada um deles[1].

[1]Nas tabelas que descrevem os eventos, o símbolo * indica que essa entidade participante é obrigatória.

EVENT_TYPE	Subtipo	Descrição
LIFETIME		eventos e relações associadas ao percurso de vida de uma pessoa
	nascimento	evento que associa um pessoa ao seu local e/ou data de nascimento
	morte	evento que associa um pessoa ao seu local e/ou data de morte
	idade	evento que associa uma pessoa à sua idade
	parentesco	eventos que representam relações familiares
	residência	evento que associa uma pessoa ao seu local de residência
	educação	evento que associa uma pessoa a diversas informações sobre a sua formação académica
BUSINESS		eventos e relações associadas a organizações
	trabalho	evento que associa uma pessoa a diversas informações acerca da sua vida profissional
	fundação	evento que associa uma organização ao seu fundador
	propriedade	evento que associa uma organização ao seu proprietário
	cliente	evento que associa uma pessoa a uma organização através da relação de cliente
	afiliação	evento que associa uma pessoa a uma organização através de uma relação de afiliação

Tabela 3.1: Eventos

3.1.1.1 Eventos BIRHT e DEATH

Os eventos de nascimento (**BIRHT**) e de morte (**DEATH**) relacionam uma pessoa com o seu local e data de nascimento ou morte respetivamente. É sempre necessário que exista pelo menos uma destas entidades (local ou data) para que a relação exista. A tabela 3.2 apresenta as entidades participantes nestes eventos. Seguem-se dois exemplos destes tipos de relações:

- O João nasceu no dia 21 de Janeiro de 1918 em Lisboa.

```
EVENT_LEX(nasceu,nascimento)
EVENT_LIFETIME(nasceu)
EVENT_PARTICIPANT(nasceu,João)
EVENT_DATE(nasceu,21 de Janeiro de 1918)
EVENT_PLACE(nasceu,Lisboa)
```

- O João morreu em Dezembro de 1990 no Porto.

```
EVENT_LEX(morreu,morte)
EVENT_LIFETIME(morreu)
EVENT_PARTICIPANT(morreu,João)
EVENT_DATE(morreu,Dezembro de 1990)
EVENT_PLACE(morreu,Porto)
```

3.1.1.2 Evento de AGE

O evento **AGE** (idade) relaciona uma pessoa com a expressão temporal que designa a sua idade. Esta relação é apenas aplicável a pessoas e não é aplicável a outros eventos como um aniversário ou a duração de uma empresa desde a sua fundação. A tabela 3.3 mostra as entidades participantes neste evento. Segue-se um exemplo deste tipo de relação:

Subtipo	**Nascimento** ou **Morte**
Relação	Descrição
PARTICIPANT*	Pessoa que nasce ou morre
DATE	Data de nascimento ou morte
PLACE	Local de nascimento ou morte

Tabela 3.2: Eventos **BIRTH** e **DEATH**

- O Pedro comemora 25 anos no dia 10 de Março.

```
EVENT_LEX(anos,idade)
EVENT_LIFETIME(anos)
EVENT_PARTICIPANT(anos,Pedro)
EVENT_AGE(anos,25 anos)
```

Subtipo	**Idade**
Relação	Descrição
PARTICIPANT*	Pessoa a quem se refere a idade
AGE*	Idade da pessoa

Tabela 3.3: Evento **AGE**

3.1.1.3 Eventos FAMILY

Os eventos **FAMILY** capturam uma relação de parentesco entre duas ou mais pessoas. A tabela 3.4 apresenta das diferentes variações deste evento, sendo que cada uma representa um grau de parentesco diferente. Todas as relações de parentesco apresentadas são semelhantes e contêm apenas informação sobre as pessoas envolvidas na relação, com exceção da relação de **casamento**. Este caso é abordado na secção 3.1.1.4. Segue-se um exemplo destes tipos de relações.

- A Maria é mãe do Pedro.

```
EVENT_LEX(mãe,parentesco)
EVENT_LIFETIME(mãe)
EVENT_PARENT_2F(mãe,Maria)
EVENT_CHILD_2M(mãe,Pedro)
```

3.1.1.4 Evento SPOUSE

A relação de **casamento** é diferente das restantes relações de parentesco pois é a única à qual podem surgir associadas datas. Uma pessoa pode casar-se e a partir desse momento existe a relação de casado, no entanto, pode divorciar-se posteriormente, deixando assim de existir esta relação. Nesta relação é obrigatória a existência da identificação da pessoa com quem o sujeito está casado, sendo as restantes entidades opcionais. A tabela 3.5 apresenta todas as entidades participantes numa relação de casamento. Segue-se um exemplo deste tipo de relação:

- O Pedro e a Maria casaram-se no Mosteiro de Alcobaça no dia 14 de Fevereiro.

```
EVENT_LEX(casaram-se,parentesco)
EVENT_LIFETIME(casaram-se)
```

Subtipo	**Parentesco**
Relação	Descrição
SPOUSE	esposos
PARENT	pais e padrastos
CHILD	filhos, enteados e filhos adotados
GRANDPARENT	avôs, bisavôs, etc.
GRANDCHILD	netos
UNCLE	tios
NEPHEW	sobrinhos
PARENT-IN-LAW	sogros
SON-IN-LAW	genros
GODFATHER	padrinhos
GODSON	afilhados
GRANDUNCLE	tios-avôs
GRANDNEPHEW	sobrinhos-netos
SIBLING	irmãos
COUSIN	primos
BROTHER-IN-LAW	cunhados

Tabela 3.4: Eventos **FAMILY**

```
EVENT_SPOUSE_2M(casaram-se,Pedro)
EVENT_SPOUSE_2F(casaram-se,Maria)
EVENT_DATE(casaram-se,14 de Fevereiro)
EVENT_PLACE(casaram-se,Mosteiro de Alcobaça)
```

Subtipo	**Parentesco**
Relação	Descrição
SPOUSE*	o conjugue
DATE-START	data de início do casamento
DATE-END	data de fim do casamento
DURATION	duração do casamento

Tabela 3.5: Relação **SPOUSE**

3.1.1.5 Evento RESIDENCE

O evento **RESIDENCE** (residência) associa uma pessoa ao seu local de residência, podendo adicionalmente ser associadas datas de início e fim a este evento. A tabela 3.6 mostra todos os possíveis constituintes deste evento. Segue-se um exemplo deste tipo de relação:

- O João vive na Avenida da Liberdade há 15 anos.

```
EVENT_LEX(vive,residência)
EVENT_LIFETIME(vive)
EVENT_PARTICIPANT(vive,João)
EVENT_DURATION(vive,há 15 anos)
EVENT_PLACE(vive,Avenida da Liberdade)
```

Subtipo	**Residência**
Relação	Descrição
PARTICIPANT*	pessoa cuja residência está a ser identificada
PLACE*	local de residência
DATE-START	data de início de residência
DATE-END	data de fim de residência
DURATION	duração de residência

Tabela 3.6: Evento **RESIDENCE**

3.1.1.6 Evento EDUCATION

O evento **EDUCATION** relaciona uma pessoa com um grau académico. Adicionalmente pode ser extraído outro tipo de informação como local, datas de início e fim, organização que atribui o grau académico e domínio associado a esse grau. Segue-se um exemplo deste tipo de relação:

- O João tirou o doutoramento em Física em 2005 no IST em Lisboa.

```
EVENT_LEX(lexeducação,educação)
EVENT_LIFETIME(lexeducação)
EVENT_ACADEMIC-DEGREE(lexeducação, doutoramento)
EVENT_PARTICIPANT(lexeducação,João)
EVENT_DATE-END(lexeducação,2005)
EVENT_ORG(lexeducação,IST)
EVENT_PLACE(lexeducação,Lisboa)
EVENT_DOMAIN(lexeducação,Física)
```

Subtipo	**Educação**
Relação	Descrição
ACADEMIC-DEGREE*	grau académico obtido
PARTICIPANT*	pessoa que obteve o grau académico
DATE-START	data de início do processo para obtenção do grau académico
DATE-END	data de fim do processo para obtenção do grau académico
ORG	instituição que atribuiu o grau académico
PLACE	localização da instituição que atribuiu o grau académico
DOMAIN	domínio cientifico do grau académico (física, línguas, etc.)

Tabela 3.7: Evento **EDUCATION**

3.1.2 Eventos BUSINESS

A categoria de eventos **BUSINESS** está relacionada com eventos que associam pessoas a organizações. Existem 5 tipos de eventos desta categoria, eventos de trabalho (**WORK**), fundação (**FOUNDATION**), propriedade (**OWNERSHIP**), cliente (**WORK**) e filiação (**AFFILIATION**). Podem ser estabelecidos vários tipos de relações para estes tipos de eventos, nas secções seguintes serão detalhados cada um deles.

3.1.2.1 Evento WORK

O evento **WORK** associa uma pessoa à organização para a qual a mesma trabalha ou trabalhou. Neste evento existem 2 entidades obrigatórias, a entidade **EMPLOYEE** e a entidade **ORG**, sendo as

restantes facultativas. A tabela 3.8 mostra as relações envolvidas neste evento. Segue-se um exemplo para este tipo de relação:

- O João é engenheiro na Microsoft desde 2005.

```
EVENT_LEX(lextrabalho,trabalho)
EVENT_BUSINESS(lextrabalho)
EVENT_EMPLOYEE(lextrabalho,João)
EVENT_PROFESSION(lextrabalho,engenheiro)
EVENT_DATE-START(lextrabalho,2005)
EVENT_ORG(lextrabalho,Microsoft)
```

Subtipo	**Trabalho**
Relação	Descrição
EMPLOYEE*	pessoa que trabalha para determinada organização
ORG*	organização onde a pessoa trabalha
PROFESSION	profissão ou função da pessoa
DATE-START	data de início do evento
DATE-END	data de fim do evento
DURATION	duração do evento
PLACE	local do evento

Tabela 3.8: Evento **WORK**

3.1.2.2 Evento FOUNDATION

O evento **FOUNDATION** associa uma organização à pessoa que a fundou, sendo as entidades **ORG** e **FOUNDER** de carácter obrigatório e as restantes facultativas. A tabela 3.9 mostra todas as relações possíveis para este evento. Segue-se um exemplo para este tipo de relação:

- Bill Gates fundou a Microsoft a 4 de abril de 1975 nos Estados Unidos.

```
EVENT_LEX(fundou,fundação)
EVENT_BUSINESS(fundou)
EVENT_FOUNDER(fundou,Bill Gates)
EVENT_DATE(fundou,4 de abril de 1975)
EVENT_ORG(fundou,Microsoft)
EVENT_PLACE(fundou,Estados Unidos)
```

Subtipo	**Fundação**
Relação	Descrição
ORG*	organização que foi fundada
FOUNDER	pessoa que fundou a organização
PLACE	local de fundação da organização
DATE	data de fundação da organização

Tabela 3.9: Evento **FOUNDATION**

3.1.2.3 Evento OWNERSHIP

O evento **OWNERSHIP** associa uma pessoa a uma organização como dona da organização independentemente da relação de pertença (acionista, sócio, etc.), sendo as entidades **ORG** e **OWNER** de carácter obrigatório e as restantes facultativas. A tabela 3.10 mostra todas as relações possíveis para este evento. Segue-se um exemplo para este tipo de relação:

- O João foi accionista da EDP entre 2005 e 2007.

```
EVENT_LEX(accionista,propriedade)
EVENT_BUSINESS(accionista)
EVENT_OWNER(accionista,João)
EVENT_DATE-START(accionista,2005)
EVENT_DATE-END(accionista,2007)
EVENT_ORG(accionista,EDP)
```

Subtipo	**Propriedade**
Relação	Descrição
OWNER*	pessoa dona da organização
ORG*	organização envolvida na relação
DATE-START	data de início do evento
DATE-END	data de fim do evento
DURATION	duração do evento
PLACE	localização da organização

Tabela 3.10: Evento **OWNERSHIP**

3.1.2.4 Evento CLIENT

O evento **CLIENT** associa uma pessoa a uma organização da qual ela é cliente, sendo as entidades **ORG** e **CLIENT** de carácter obrigatório e as restantes facultativas. A tabela 3.11 mostra todas as entidades possíveis para este evento. Segue-se um exemplo para este tipo de evento:

- O João é cliente do Continente de Telheiras desde 2011.

```
EVENT_LEX(cliente,cliente)
EVENT_BUSINESS(cliente)
EVENT_CLIENT(cliente,João)
EVENT_DATE-START(cliente,2011)
EVENT_ORG(cliente,Continente)
EVENT_PLACE(cliente,Telheiras)
```

3.1.2.5 Evento AFFILIATION

O evento **AFFILIATION** associa uma pessoa a uma organização da qual ela é afiliada, sendo as entidades **ORG** e **MEMBER** de carácter obrigatório e as restantes facultativas. A tabela 3.12 mostra todas as relações possíveis para este evento. Segue-se um exemplo para este tipo de eventos:

- O João é socialista há 50 anos.

```
EVENT_LEX(afiliação,afiliação)
EVENT_BUSINESS(afiliação)
EVENT_MEMBER(afiliação,João)
EVENT_DURATION(afiliação,há 50 anos)
EVENT_ORG(afiliação,socialista)
```

Subtipo	**Cliente**
Relação	Descrição
CLIENT*	pessoa cliente da organização
ORG*	organização da qual a pessoa é cliente
DATE-START	data de início do evento
DATE-END	data de fim do evento
DURATION	duração do evento
PLACE	localização da organização

Tabela 3.11: Evento **CLIENT**

Subtipo	**afiliação**
Relação	Descrição
MEMBER*	pessoa que é afiliada da organização
ORG*	organização da qual a pessoa é afiliada
PLACE	localização da organização
DATE-START	data de início do evento
DATE-END	data de fim do evento
DURATION	duração do evento

Tabela 3.12: Evento **AFFILIATION**

3.1.3 BUSINESS_PROFESSION

Para além das relações definidas como eventos, existe ainda uma relação adicional que não se enquadra nesta definição. Esta relação assume a forma **BUSINESS_PROFESSION**(*pessoa*,*profissão*) e é remanescente de uma anterior terminologia para as relações. Esta relação associa uma pessoa à sua profissão e serve apenas para casos muito específicos em que o conceito de evento não faz sentido. O exemplo seguinte mostra um desses casos, em que a frase apresenta informação profissional sobre a pessoa mas não faz sentido que esta seja considerada um evento:

- O Eng. António está feliz.

```
BUSINESS_PROFESSION(António,Eng.)
```

3.2 Módulos AfterXIP

No início desta dissertação, existiam já implementados dois módulos, o módulo de normalização de expressões temporais (TimeNormalization) e o módulo de resolução de expressões anafóricas (ARM). Em ambos os casos estes recebiam como entrada um ficheiro XML com o processamento efetuado pela cadeia de processamento do L^2F. Estes módulos foram desenvolvidos em separado e não têm qualquer dependência entre eles, ambos processavam o ficheiro de saída da STRING e devolviam o respetivo resultado na forma de texto. Tomou-se a opção de normalizar as saídas destes módulos por forma a manterem o formato de *output* da STRING, portanto, foi necessário implementar mecanismos de escrita para XML para ambos os módulos. Esta normalização facilitou a utilização da saída destes módulos na tarefa de preenchimento de *slots*.

Ambos os módulos já existentes utilizam a XIPAPI [Nobre, 2011] desenvolvida simultaneamente com o módulo de resolução de anáfora. Esta API é responsável por transformar os dados XML em estruturas Java para facilitar o processamento destes dados. A XIPAPI utiliza por sua vez outra API que é standard

do Java, a Java DOM API [2] para construir uma árvore DOM que posteriormente é convertida pela XIPAPI em estruturas que são utilizadas pelos módulos do *AfterXIP*. No início deste trabalho, esta API apenas convertia os dados XML para Java não permitindo o processamento inverso. A conversão inversa foi desenvolvida no decurso deste trabalho e passou pela conversão da representação interna para uma árvore DOM que posteriormente é transformada em XML através de métodos standard do Java fornecidos pela Java DOM API.

Outra das decisões tomadas foi garantir que os módulos poderiam ser invocados de diversas formas. Estes passaram a poder ser invocados separadamente, apenas dois dos módulos, os três módulos e por qualquer ordem. Este funcionamento foi conseguido através da implementação de um *handler* (classe *AfterXIP* na figura 3.4) responsável por gerir as invocações dos módulos e da XIPAPI. As escolhas são feitas pelo utilizador que pode indicar quais os módulos a ser executados e a ordem pela qual são executados, bem como o tipo de saída que deve ser produzido (XML ou textual). Com a introdução do *handler* o ficheiro XML de saída da STRING passou a ser transformado na representação Java apenas uma vez. Esta representação interna é comum aos três módulos (normalização de expressões temporais, resolução de expressões anafóricas e preenchimento de *slots*). Este *handler* pode receber vários parâmetros de entrada que alteram o comportamento do *AfterXIP*; podem ser escolhidos quais os módulos a executar e qual a ordem de execução, o tipo de saída; e, ainda, podem ser definidos parâmetros de *debug*, que permitem analisar tempos de execução dos três sub-módulos.

3.2.1 Normalização de Expressões Temporais

O módulo de normalização de expressões temporais utiliza informação contida no XML para identificar nós que contêm uma expressão temporal e adicionar ao mesmo uma representação normalizada dessa expressão. Os detalhes deste trabalho estão descritos em [Maurício, 2011]. As alterações efetuadas neste módulo prendem-se com a adição da informação devolvida pelo mesmo à representação interna dos dados, sendo esta informação guardada sob a forma de uma ou várias *features* no nó a que corresponde. A Figura 3.2 mostra a árvore gerada pela STRING ao processar a frase "O João nasceu em 1984.". Nesta pode verificar-se que o nó PP é composto por dois sub nós, PREP e NUM. Estes constituem uma expressão temporal identificada pelo módulo de normalização de expressões temporais. O resultado do processamento efetuado por este módulo é então adicionado ao nó-pai dos nós que contêm a expressão temporal, neste caso é adicionado ao nó PP.

A listagem 3.3 mostra um excerto de um ficheiro XML de saída do módulo TimeNormalization, este contem uma expressão temporal "em 1984"e a respetiva *feature* que representa a normalização "+CM———D1984—-T———E–LM-". Como se pode verificar, a *feature* com o atributo *attribute*=VAL_NORM está associada ao nó PP.

[2] http://docs.oracle.com/javase/6/docs/api/org/w3c/dom/package-summary.html (data de acesso: 26/04/2013)

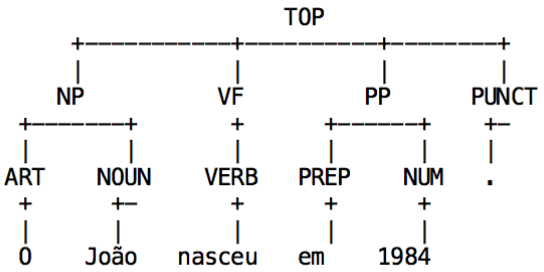

Figura 3.2: Árvore gerada

Listagem 3.3: Excerto do ficheiro XML com uma normalização de uma expressão temporal.

```
<NODE end="20" num="18" start="14" tag="PP">
  <FEATURE attribute="QUANT" value="+" />
  <FEATURE attribute="TIME" value="+" />
  <FEATURE attribute="ENT4" value="+" />
  <FEATURE attribute="PP" value="+" />
  <FEATURE attribute="T-YEAR" value="+" />
  <FEATURE attribute="T-TEMPREF" value="ABSOLUT" />
  <FEATURE attribute="VAL-NORM"
          value="+CM------D1984-----T----------E--LM--" />
  <FEATURE attribute="VAL-DELTA" value="" />
  <FEATURE attribute="UMED" value="" />
  <NODE end="15" num="6" start="14" tag="PREP">
    <FEATURE attribute="PREP" value="+" />
    <FEATURE attribute="HMMSELECTION" value="+" />
    <FEATURE attribute="FIRST" value="+" />
    <TOKEN end="15" pos="PREP" start="14">em
          <READING lemma="em" pos="PREP" />
    </TOKEN>
  </NODE>
  <NODE end="20" num="8" start="17" tag="NUM">
    <FEATURE attribute="TIME" value="+" />
    <FEATURE attribute="DIG" value="+" />
    <FEATURE attribute="LAST" value="+" />
    <FEATURE attribute="T-YEAR" value="+" />
    <FEATURE attribute="HMMSELECTION" value="+" />
    <FEATURE attribute="NUM" value="+" />
    <FEATURE attribute="CARD" value="+" />
    <TOKEN end="20" pos="NUM" start="17">1984
          <READING lemma="1984" pos="NUM" />
    </TOKEN>
  </NODE>
</NODE>
```

3.2.2 Resolução de Expressões Anafóricas

O módulo de Resolução de Anáfora (ARM) [Nobre, 2011], processa anáforas pronominais guardando a informação resultante em **FEATURES** nos nós que representa a expressão anafórica. Tomando como exemplo as frases: *"O Carlos é pai do João e da Maria. Ele é casado com a Carla desde 1994."*, e tendo como resultado do processamento efetuado pela STRING as árvores da figura 3.3, verifica-se que o ARM identifica uma anáfora entre "Ele"e "O Carlos". A listagem 3.4 mostra o resultado do processamento efetuado pelo ARM.

O módulo ARM adiciona, ao nó que contém o pronome "Ele", as *features* com os atributos *attri-*

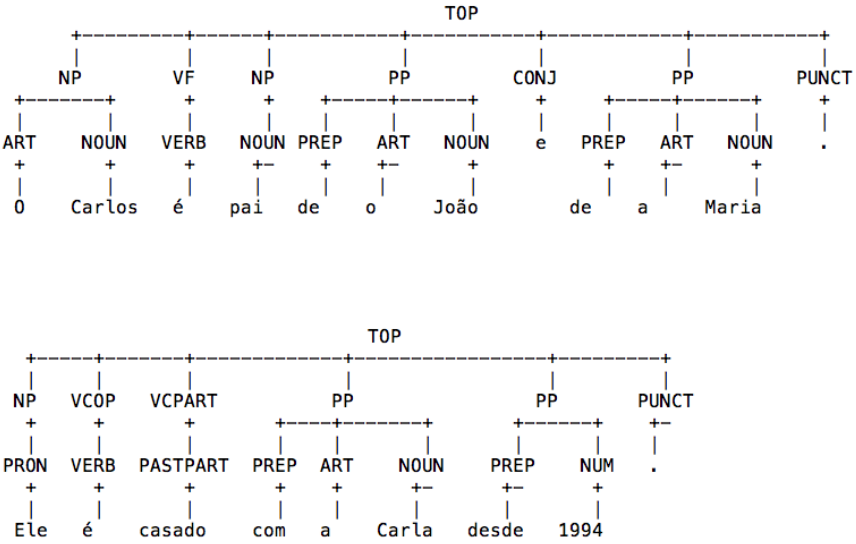

Figura 3.3: Excerto do ficheiro XML com uma dependência que contem uma anáfora.

bute="ANAPHORA"e *attribute*="ANTECEDENT". No início deste trabalho, o ARM já tinha este comportamento implementado e portanto já possibilitava a utilização destes resultados para o preenchimento de *slots*. No entanto, foram ainda introduzidas algumas alterações para facilitar a sua utilização.

Listagem 3.4: Excerto do ficheiro XML com o resultado do processamento efetuado pelo ARM.

```
<NODE end="37" num="0" start="35" tag="PRON">
  <FEATURE attribute="MAJ" value="+" />
  <FEATURE attribute="SG" value="+" />
  <FEATURE attribute="NOM" value="+" />
  <FEATURE attribute="ANAPHORA" value="pronominal" />
  <FEATURE attribute="MASC" value="+" />
  <FEATURE attribute="HMMSELECTION" value="+" />
  <FEATURE attribute="HEAD" value="+" />
  <FEATURE attribute="START" value="+" />
  <FEATURE attribute="NOPPHEAD" value="+" />
  <FEATURE attribute="3P" value="+" />
  <FEATURE attribute="LAST" value="+" />
  <FEATURE attribute="ANTECEDENT" value="29" />
  <FEATURE attribute="PERS" value="+" />
  <FEATURE attribute="PRON" value="+" />
  <FEATURE attribute="FIRST" value="+" />
  <TOKEN end="37" pos="PRON" start="35">Ele
      <READING lemma="eu" pos="PRON" />
  </TOKEN>
</NODE>
```

A listagem 3.5 mostra uma das dependências onde o pronome "Ele"surge como parâmetro. A alteração efetuada foi a adição de um novo tipo de nó denominado **ANAPHOR**, este nó é adicionado às dependências que têm como **PARAMETER** a expressão anafórica. Este novo tipo de nó tem dois atributos, o *val* e o *nodeNumber* que representam, respetivamente, a expressão antecedente e o número do nó da mesma. O atributo *nodeNumber* serve para que seja possível decidir qual dos **PARAMETER** da dependência pode ver o valor do seu campo *word* substituído pelo valor do atributo *val* do **ANAPHOR**. Esta substituição é apenas utilizada para o preenchimento de *slots* e não se trata de uma substituição efetiva no XML. A introdução deste novo nó permite que o módulo de preenchimento de *slots* tenha

apenas de inspecionar as dependências para utilizar a informação de processamento do ARM, ao invés de pesquisar os nós que constituem a frase em análise.

Listagem 3.5: Excerto do ficheiro XML com uma dependência que contem uma anotação referente a uma anáfora.

```
<DEPENDENCY name="EVENT">
  <FEATURE attribute="SPOUSE" value="+" />
  <PARAMETER ind="0" num="4" word="casado" />
  <PARAMETER ind="1" num="0" word="Ele" />
  <ANAPHOR nodeNumber="0" value="Carlos" />
</DEPENDENCY>
```

3.2.3 Preenchimento de *Slots*

O sistema implementado processa o *output* da cadeia de processamento de língua natural do L^{2}F (STRING). Este módulo pode utilizar ainda as anotações feitas pelos módulos de Resolução de Anáfora e de Normalização de Expressões Temporais por forma a aumentar a quantidade de informação que consegue agrupar.

A figura 3.4 mostra um diagrama UML da aplicação desenvolvida. Foram criadas duas classes para representar as entidades alvo de extração de informação (*Person* e *Organization*) e ainda duas classes auxiliares, o *GenericContainer* e o *TimeNorm*.

O *GenericContainer* guarda informação como localizações, nomes de pessoas, nomes de organizações etc. Esta classe foi criada para agrupar todas as informações que são comuns a vários eventos de modo a evitar replicação de código. Com finalidade semelhante, foi criada a classe *TimeNorm* para guardar uma expressão temporal e a respetiva normalização.

Quando às classes *Person* e *Organization*, são as representações internas para entidades dos tipos pessoa e organização respetivamente. Um objeto de um destes tipos guarda toda a informação recolhida acerca de uma instância de uma entidade do respetivo tipo, isto é, se num texto surgirem várias referências e eventos associados a uma pessoa, toda a informação extraída acerca dessa pessoa é agrupada num único objeto do tipo *Person*. Nestas classes foram ainda implementados mecanismos que permitem fazer as comparações necessárias por forma a ser implementado o mecanismo resolução de correferências que permita juntar entidades semelhantes. Este mecanismo é descrito em detalhe na secção seguinte. Cada um dos objetos do tipo *Person* e *Organization* contém um conjunto de *slots* (atributos das classes apresentados na figura 3.1) e este conjunto é aquilo que até aqui se denominou *frame*. Um *slot* mapeia diretamente um subtipo de evento (normalmente com o mesmo nome que o *slot*) e é preenchido com o valor do atributo *word* do nó **PARAMETER** que tem como nó-pai um nó do tipo **EVENT** correspondente ao subtipo a que o *slot* se refere. Existem, no entanto, alguma exceções como o caso do *slot alternateNames*, este não tem mapeamento direto com nenhum evento, é preenchido apenas quando é detetado um novo nome para uma entidade já existente e serve para guardar um histórico do processo de resolução de correferências. A sua utilização é descrita mais adiante na secção 3.2.3.3. Na secção 3.2.3.2 cada um dos atributos das classes *Person* e *Organization* é descrito em detalhe.

O *CoreferenceAnalyzer* é a classe responsável por guardar e gerir as listas de entidades identificadas pelo sistema. Esta classe guarda duas listas que contêm as entidades do tipo *Person* e *Organization*. Para cada evento é feita uma pesquisa pelas entidades já existentes; se a entidade já existir em alguma das listas, são adicionados os novos campos ou completados caso já tenham sido preenchidos; caso a entidade não exista, é criada uma nova entidade e adicionada à respetiva lista (se a nova entidade for do tipo *Person* vai ser guardada na lista de entidades desse tipo, se for do tipo *Organization* vai para a lista que contém entidades do tipo *Organization*) do *CoreferenceAnalyzer*. Na secção 3.2.3.3 serão apresentados

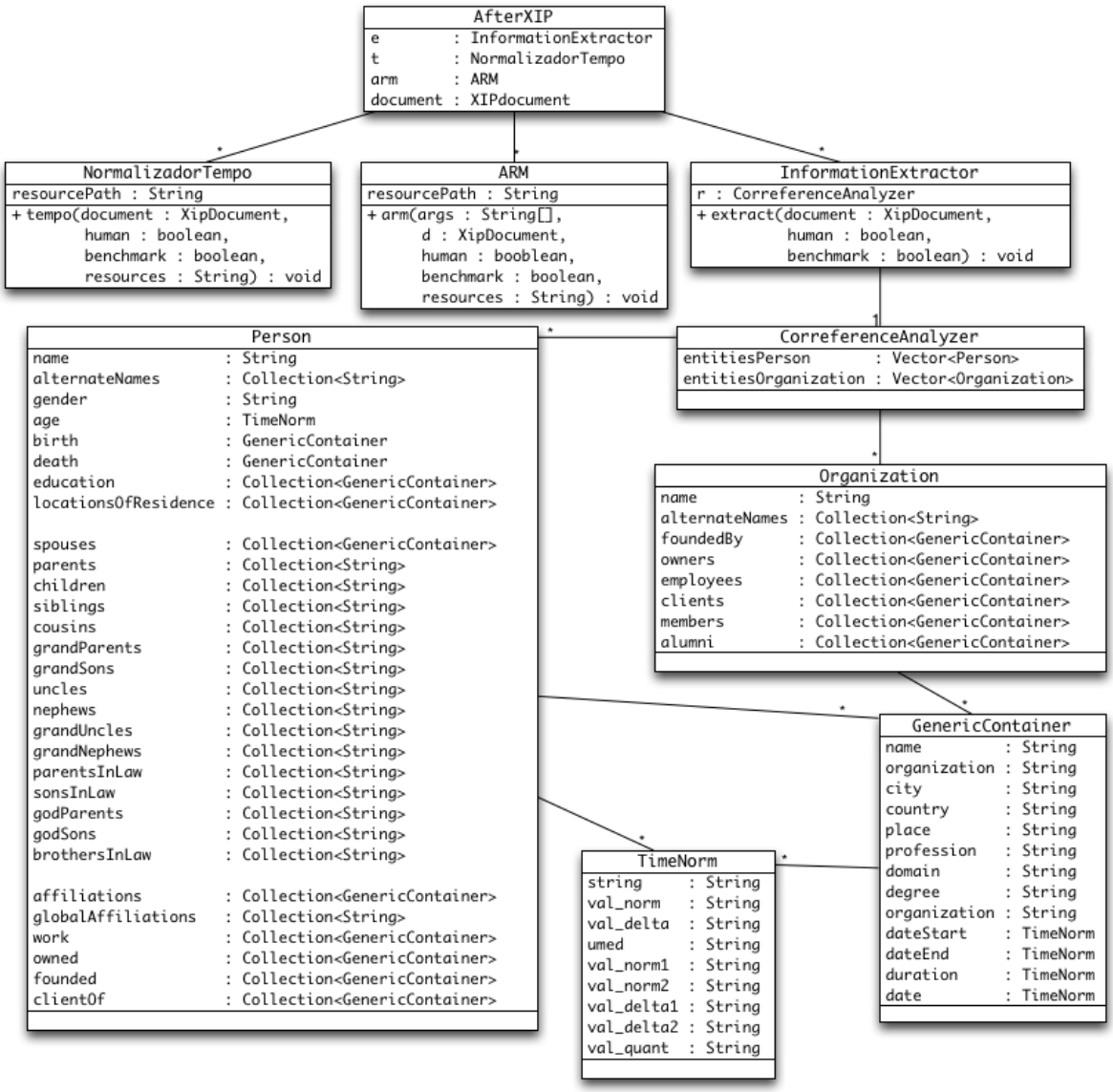

Figura 3.4: UML AfterXIP

os pormenores sobre o funcionamento destas classes e sobre a implementação do mecanismo de resolução de correferências.

O *InformationExtractor* é a classe principal do sistema sendo responsável por processar todos os eventos presentes no XML proveniente do XIP ou de um dos outros módulos do *AfterXIP*. Os eventos são percorridos por frase, ou seja, são agrupados os eventos de cada frase e só aí é percorrido o ciclo principal do programa, que processa apenas os eventos agrupados. Nesta classe, existe uma instância de um *CoreferenceAnalyzer*. O *InformationExtractor* é responsável por identificar o tipo de relação em questão e respetivos parâmetros, criar as novas entidades ou preencher campos de entidades já existentes.

A classe *AfterXIP* foi desenvolvida durante a elaboração deste trabalho e serve para coordenar as chamadas aos módulos de normalização de expressões temporais, resolução de expressões anafóricas e ao módulo de preenchimento de *slots*. As funcionalidades fornecidas por esta classe foram anteriormente descritas na secção 3.2.

3.2.3.1 Propriedades das Relações

Os *slots* definidos para este sistema baseiam-se nos eventos definidos pela equipa do L^2F para o XIP. Esta decisão permite o mapeamento direto entre eventos e a informação que irá ser agrupada. Seguem-se duas definições importantes no contexto das relações entre entidades:

- RELAÇÃO SIMÉTRICA - É uma relação entre duas entidades em que a ordem dos argumentos pode ser invertida e o seu papel semântico é exatamente o mesmo, sendo a expressão que veicula a relação exatamente a mesma, salvas as diferenças de acordo em género/número:

- O Pedro casou com a Ana. = A Ana casou com o Pedro. **(SPOUSE)**
- O Pedro é irmão da Ana. = A Ana é irmã do Pedro. **(SIBLING)**

Na tabela 3.13 pode verificar-se que não existe nenhuma diferença na saída da cadeia quando processados os dois exemplos anteriores. A tabela 3.14 apresenta os eventos que se enquadram na definição de Relação Simétrica.

O Pedro casou com a Ana.	A Ana casou com o Pedro.
EVENT_LEX(casou,parentesco)	EVENT_LEX(casou,parentesco)
EVENT_LIFETIME(casou)	EVENT_LIFETIME(casou)
EVENT_2M_SPOUSE(casou,Pedro)	EVENT_2M_SPOUSE(casou,Pedro)
EVENT_2F_SPOUSE(casou,Ana)	EVENT_2F_SPOUSE(casou,Ana)

Tabela 3.13: Exemplo de uma relação simétrica.

PESSOA	PESSOA
SPOUSE	**SPOUSE**
SIBLING	**SIBLING**
COUSIN	**COUSIN**
BROTHER-IN-LAW	**BROTHER-IN-LAW**

Tabela 3.14: Relações Simétricas

- SIMETRIA DE RELAÇÕES - São pares de relações entre entidades em que cada relação apresenta os mesmos argumentos mas estes estão invertidos (numa relação relativamente à outra):

- O Pedro é tio da Ana. **(UNCLE)** <> A Ana é sobrinha do Pedro. **(NEPHEW)**

O Pedro é tio da Ana.	A Ana é sobrinha do Pedro.
EVENT_LEX(tio,parentesco)	EVENT_LEX(sobrinha,parentesco)
EVENT_LIFETIME(tio)	EVENT_LIFETIME(sobrinha)
EVENT_2M_UNCLE(tio,Pedro)	EVENT_2M_UNCLE(sobrinha,Pedro)
EVENT_2F_NEPHEW(tio,Ana)	EVENT_2F_NEPHEW(sobrinha,Ana)

Tabela 3.15: Exemplo de simetria de relações em eventos UNCLE e NEPHEW

Na tabela 3.15 pode verificar-se que não existe nenhuma diferença na saída da cadeia quando processados os dois exemplos anteriores. Neste caso são geradas 2 relações quando, na frase, só vem explícita uma delas. No entanto, existem alguns casos em que isso não acontece, portanto essas situações têm de ser tratadas de outra forma. Quando a STRING não gera a relação inversa é a classe *InformationExtractor* que,

quando deteta uma relação num sentido, constrói o significado inverso da mesma como se pode verificar no exemplo na tabela 3.16: o resultado de processamento vem explícita a relação EVENT_EMPLOYEE no entanto, pode assumir-se a existência de uma relação simétrica do tipo WORK. Neste caso, serão adicionadas duas relações às entidades identificadas. À entidade do tipo pessoa (*João*) será adicionado um *slot* WORK preenchido com o nome da organização (*Optimus*) e à entidade do tipo organização (*Optimus*) será adicionado o *slot* EMPLOYEE preenchido com o nome da pessoa (*João*), tal como se vê na listagem 3.6. Na tabela 3.17 apresenta os eventos que se enquadram na definição de Simetria de Relações.

O João trabalha na Optimus desde 2001.
EVENT_LEX(trabalha,trabalho)
EVENT_BUSINESS(trabalha)
EVENT_DATE-START(trabalha,desde 2001)
EVENT_ORG(trabalha,Optimus)
EVENT_EMPLOYEE_2M(trabalha,João)

Tabela 3.16: Exemplo de simetria de relações em evento EMPLOYEE

Listagem 3.6: Resultado do processamento do exemplo da tabela 3.16

```
********* PERSONS *********
———— PERSON ————
NAME: Joao
GENDER: MASC
WORK:
        ORGANIZATION: Optimus
        START-DATE:
                STRING: desde 2001
                VAL-NORM: +CM———D2001——T———E—LMP


********* ORGANIZATIONS *********
———— ORGANIZATION ————
NAME: Optimus
EMPLOYEE:
        NAME: Joao
        START-DATE:
                STRING: desde 2001
                VAL-NORM: +CM———D2001——T———E—LMP
```

ORGANIZAÇÃO	PESSOA
FOUNDED-BY	**FOUNDER**
OWNER	**OWNER-OF**
AFFILIATED	**MEMBER**
EMPLOYEE	**WORK**
CLIENT	**CLIENT-OF**
ALUMNI	**EDUCATION**

Tabela 3.17: Subtipos de eventos que apresentam simetria de relações organização-pessoa

PESSOA	PESSOA
PARENT	**CHILD**
GRANDPARENT	**GRANDCHILD**
UNCLE	**NEPHEW**
PARENT-IN-LAW	**SON-IN-LAW**
GODFATHER	**GODSON**
GRANDUNCLE	**GRANDNEPHEW**

Tabela 3.18: Subtipos de eventos que apresentam simetria de relações pessoa-pessoa

3.2.3.2 *Slots*

Os eventos identificados pelo XIP são processados e guardados em *slots* no módulo de preenchimento de *slots*. Foram definidos quatro tipos distintos de *slots*, simples, compostos, simples não únicos e compostos não únicos. Seguem-se as definições de cada um destes tipos.

***Slots* simples:** Os *slots* simples destinam-se a guardar informação de eventos que são únicos para determinada entidade. Estes guardam informação referente a apenas um subtipo de evento. A listagem 3.7 mostra a estrutura de um *slot* deste tipo. Na listagem 3.8 o *slot* GENDER é uma instância de um *slot* simples. O género é um caso especial de *slot* simples pois é extraído não de um evento mas a partir dos traços atribuídos pela STRING ao nome da entidade, (traço 'masc' ou 'fem').

Listagem 3.7: Estrutura dos *slot* simples.

```
<NOME_DO_SLOT attr1=valor1 . . . attrn=valorn>
```

Listagem 3.8: Exemplo de um *slot* simples.

```
        <NAME val="Carlos"/>
```

***Slots* simples não únicos:** Os *slots* simples não únicos foram definidos para guardar informação proveniente de eventos que se podem repetir para determinada entidade. Estes *slots* guardam informações de apenas um subtipo de eventos, verificando a listagem 3.9 a etiqueta **SLOT** é sempre a mesma para cada **NOME_DO_SLOT** distinto. Utilizando a listagem 3.10, tem-se um exemplo de um *slot* simples não único. O evento **CHILDREN** da STRING pode surgir várias vezes para determinada entidade pois, neste caso, uma pessoa pode ter vários filhos. A etiqueta correspondente à **SLOT** é a etiqueta **PERSON** e é sempre extraída a partir do subtipo de evento **CHILDREN**. Todas as relações familiares são guardadas na forma de *slots* simples não únicos (existe uma exceção: as relações de casamento que devido à existência de datas associadas são representadas como *slots* compostos não únicos).

Listagem 3.9: Estrutura dos *slot* simples não únicos.

```
<NOME_DO_SLOT>
    <SLOT att1=valor1 attr2=valor2 . . . attrn=valorn>
    <SLOT att1=valor1 attr2=valor2 . . . attrn=valorn>
        .
        .
        .
    <SLOT att1=valor1 attr2=valor2 . . . attrn=valorn>
<NOME_DO_SLOT>
```

Listagem 3.10: Exemplo de um *slot* simples não único.

```
<CHILDREN>
  <PERSON name="Carla"/>
  <PERSON name="Armando"/>
</CHILDREN>
```

***Slots* compostos:** Este tipo de *slot*, à semelhança dos *slots* simples, é único para cada entidade, não podendo haver repetições. A listagem 3.11 mostra a estrutura dos *slots* compostos. A diferença para os simples está no facto de que um *slot* composto pode ser preenchido com informação proveniente de diversos subtipos de eventos (PLACE, DATE, CITY, etc.). Tomando como exemplo a listagem 3.12, pode verificar-se que o *slot* BIRTH (que representa o evento da STRING com o mesmo nome) é composto por dois *slots*, CITY e DATE (que na saída da STRING são subtipos de eventos distintos).

Listagem 3.11: Estrutura dos *slot* compostos.

```
<TIPO_DO_SLOT>
    <NOME_DO_SLOT att1=valor1 attr2=valor2 . . . attrn=valorn>
    <NOME_DO_SLOT att1=valor1 attr2=valor2 . . . attrn=valorn>
    .
    .
    .
    <NOME_DO_SLOT att1=valor1 attr2=valor2 . . . attrn=valorn>
</TIPO_DO_SLOT>
```

Listagem 3.12: Exemplo de um *slot* composto.

```
<BIRTH>
  <CITY val="Porto"/>
  <DATE string="em 1965" umed=""
        valdelta="" valdelta1="" valdelta2=""
        valnorm="+CM———D1965——T————E—LM—"
        valnorm1="" valnorm2="" valquant=""/>
</BIRTH>
```

***Slots* compostos não únicos:** Por fim os *slots* compostos não únicos servem para representar eventos que se podem repetir para determinada entidade. Recorrendo à listagem 3.14 pode verificar-se que o evento LOCATIONS-OF-RESIDENCE é um exemplo de um evento composto e não único. Este evento é composto por várias relações acerca da localização de residência de uma pessoa, pode conter locais, cidades, países e datas. No exemplo, este evento é apenas composto por um local e uma data de início.

Listagem 3.13: Estrutura dos *slot* compostos não únicos.

```
<TIPO_DO_SLOT>
  <ELEM>
    <NOME_DO_SLOT att1=valor1 attr2=valor2 . . . attrn=valorn>
    <NOME_DO_SLOT att1=valor1 attr2=valor2 . . . attrn=valorn>
        .
        .
        .
    <NOME_DO_SLOT att1=valor1 attr2=valor2 . . . attrn=valorn>
  </ELEM>
  <ELEM>
  .
  .
  .
  </ELEM>
<TIPO_DO_SLOT>
```

Listagem 3.14: Exemplo de um *slot* composto não único.

```
<LOCATIONS-OF-RESIDENCE>
  <ELEM>
    <CITY val="Coimbra"/>
  </ELEM>
  <ELEM>
    <CITY val="Lisboa"/>
    <DATE-START string="desde 1985" umed=""
                valdelta="" valdelta1="" valdelta2=""
                valnorm="+CM------D1985----T----------E--LMP"
                valnorm1="" valnorm2="" valquant=""/>
  </ELEM>
</LOCATIONS-OF-RESIDENCE>
```

Mapeamentos entre eventos da STRING e *slots*: Quanto ao mapeamento dos eventos da STRING em *slots*, adotou-se, na aplicação, uma abordagem que consistiu em tentar o fazer o mapeamento da forma mais direta possível. No entanto, foram necessárias algumas adaptações. Na tabela 3.19 constam os mapeamentos para os *slots* simples (únicos e não únicos) de entidades do tipo *Person*. Os *slots name* e *alternateNames* são preenchidos recorrendo a todos os eventos que se encontram nesta tabela. Isto deve-se à organização da saída da STRING e à forma como esta é processada. Utilizando o seguinte exemplo de saída da STRING:

- O Pedro é pai do Carlos.

```
EVENT_LEX(pai,parentesco)
EVENT_LIFETIME(pai)
EVENT_PARENT_2M(pai,Pedro)
EVENT_CHILD_2M(anos,Carlos)
```

Quando esta saída é processada pelo módulo de preenchimento de *slots* obtém-se o resultado expresso na listagem 3.15. Nesta, pode verificar-se que na primeira entidade o *slot name* é preenchido pelo evento EVENT_CHILD e o *slot parents* é preenchido com o valor do EVENT_PARENT. Na segunda entidade acontece o oposto.

Listagem 3.15: Saída do módulo de preenchimento de *slots*.

```
****************** Slot-Filling ******************

********* PERSONS *********
------- PERSON -------
NAME: Carlos
GENDER: MASC
PARENTS:
        Pedro

_____________________

------- PERSON -------
NAME: Pedro
GENDER: MASC
CHILDREN:
        Carlos

_____________________
```

Relativamente à tabela 3.20, esta contém os mapeamentos entre *slots* compostos (únicos e não únicos) de entidades do tipo *Person* e os eventos da STRING correspondentes. Como descrito anteriormente, os

slots compostos são constituídos por diversas informações e são guardados na aplicação sob a forma de um objeto da classe *GenericContainer*. Na primeira parte da tabela 3.20 tem-se os nomes das relações e na segunda parte o mapeamento entre *slots* que fazem parte dessas relações (atributos da *GenericContainer*) e os eventos da STRING. Todas as relações indicadas são constituídas por uma combinação dos *slots* indicados.

Slots simples *Person*	Evento da STRING
name alternateNames	EVENT_PARTICIPANT, EVENT_FOUNDER, EVENT_FOUNDER, EVENT_EMPLOYEE, EVENT_FOUNDER, EVENT_OWNER, EVENT_CLIENT, EVENT_MEMBER, EVENT_FORMAL-MEMBER
parent	EVENT_PARENT
child	EVENT_CHILD
grandParent	EVENT_GRANDPARENT
grandChild	EVENT_GRANDCHILD
uncle	EVENT_UNCLE
nephew	EVENT_NEPHEW
parentInLaw	EVENT_PARENT-IN-LAW
sonInLaw	EVENT_SON-IN-LAW
godParent	EVENT_GODFATHER
godSon	EVENT_GODSON
grandUncle	EVENT_GRANDUNCLE
grandNephew	EVENT_GRANDNEPHEW
sibling	EVENT_SIBLING
cousin	EVENT_COUSIN
brotherInLaw	EVENT_BROTHER-IN-LAW
age	EVENT_AGE

Tabela 3.19: Mapeamentos entre os *slots* simples de uma entidade do tipo pessoa e os eventos da STRING

Relações	
birth	
death	
education	
locationsOfResidence	
spouse	
affiliations	
globalAffiliations	
work	
owned	
founded	
clientOf	
Slots compostos *Person*	Eventos STRING
name	EVENT_SPOUSE
city	EVENT_PLACE com *feature* CITY
country	EVENT_PLACE com *feature* COUNTRY
place	EVENT_PLACE
profession	EVENT_PROFESSION
domain	EVENT_DOMAIN
degree	EVENT_DEGREE
organization	EVENT_ORG
dateStart	EVENT_DATE-START
dateEnd	EVENT_DATE-END
duration	EVENT_DURATION
date	EVENT_DATE

Tabela 3.20: Mapeamentos entre os *slots* compostos de uma entidade do tipo pessoa e os eventos da STRING

Na tabela 3.21 estão mapeados os dois *slots* simples referentes a entidades do tipo *Organization*. Como se pode verificar, e ao contrário do que acontece com as entidades do tipo *Person*, existe apenas um evento que permite o preenchimento do nome de uma organização. Os restantes mapeamentos para estas entidades estão listados na tabela 3.22, sendo os critérios de preenchimento semelhantes aos dos *slots* compostos das entidades do tipo *Person*.

Por fim, na tabela 3.22, estão mapeados os *slots* compostos das entidades do tipo *Organization*. A tabela está estruturada da mesma forma que a tabela 3.20.

Slots simples *Organization*	Evento da STRING
name alternateNames	EVENT_ORG

Tabela 3.21: Mapeamentos entre os *slots* simples de uma entidade do tipo organização e os eventos da STRING

Slots compostos *Organization*	Eventos da STRING
foundedBy owners employees clients members alumni	
(Armazenados num *GenericContainer*)	
name	EVENT_FOUNDER, EVENT_EMPLOYEE, EVENT_FOUNDER, EVENT_OWNER, EVENT_CLIENT, EVENT_MEMBER, EVENT_FORMAL-MEMBER
city	EVENT_PLACE com *feature* CITY
country	EVENT_PLACE com *feature* COUNTRY
place	EVENT_PLACE
profession	EVENT_PROFESSION
domain	EVENT_DOMAIN
degree	EVENT_DEGREE
organization	EVENT_ORG
dateStart	EVENT_DATE-START
dateEnd	EVENT_DATE-END
duration	EVENT_DURATION
date	EVENT_DATE

Tabela 3.22: Mapeamentos entre os *slots* compostos de uma entidade do tipo organização e os eventos da STRING

3.2.3.3 Agregação de Entidades

A informação relativa a eventos que está contida no output do XIP encontra-se organizada por frase, ou seja, todos os eventos contidos numa frase ficam agrupados, juntamente com o restante resultado de processamento da STRING, no nó referente a essa mesma frase. Desta forma não existe qualquer ligação entre os eventos contidos em 2 frases distintas. Como já foi referido anteriormente, um dos objetivos deste trabalho é criar uma ligação entre as instâncias do texto que referenciam a mesma entidade do mundo real e com isso agrupar os eventos por entidade. Com esta finalidade, foi criado um mecanismo

que, em primeiro lugar, estabelece uma relação de equivalência entre as entidades presentes no texto e, de seguida, agrupa as informações provenientes de eventos nos quais essas entidades estão presentes.

Neste trabalho, definiu-se que seriam agrupados eventos relativos a pessoas e organizações. Para guardar a informação referente a uma pessoa foi criada a classe *Person*. Esta é a representação interna de uma entidade do tipo pessoa e é nela que são guardados os *slots* (*slots* simples sob a forma de uma *String* ou listas de *Strings* e *slots* compostos sob a forma de um *GenericContainer* ou uma lista de *GenericContainers*). De forma análoga, foi definida a classe *Organization* para guardar a informação relativa a entidades do tipo organização. Para além de atributos, estas classes possuem ainda os métodos utilizados para comparar classes dos respetivos tipos.

Implementação: A implementação da comparação entre entidades passou pela definição de um novo método *equals* em cada uma destas duas classes. Estes novos métodos recebem uma *string* como *input*, *string* esta que contém o nome da entidade a comparar. Passaram a ser utilizadas as *strings* contidas nos *slots name* (nome principal) e *alternateNames* (nomes alternativos). O *slot name* é sempre preenchido com o nome mais longo encontrado até ao momento. Sempre que se utiliza o método *equals* este atualiza a informação contida na lista de nomes alternativos (*alternateNames*) caso o novo nome satisfaça os critérios de comparação. Os novos nomes só são adicionados à lista de nomes alternativos caso ainda não exista nenhum exatamente igual nessa mesma lista nem como nome principal. Seguem-se os critérios utilizados para a comparação de entidades do tipo PESSOA:

- Dado o nome A e uma entidade E, considera-se que o nome A é um nome de E:

  - Se A é exatamente igual ao nome principal (*string* guardada no *slot name*) ou a algum dos nomes alternativos (*strings* guardadas no *slot alternateNames*) da entidade E.
  - Se a primeira e última palavras do nome A são iguais a alguma das primeiras ou últimas palavras de algum dos nomes da entidade E.
  - Se o nome A é constituído por uma única palavra e é igual à primeira palavra de um dos nomes de E.

No caso das entidades do tipo ORGANIZAÇÃO os critérios utilizados são os seguintes:

- Dado o nome A e uma entidade E, considera-se que o nome A é um nome de E:

  - Se A é exatamente igual ao nome principal ou a algum dos nomes alternativos da entidade E.
  - Se a concatenação das primeiras letras de todas as palavras do nome mais longo de E é igual a A para o caso do nome mais longo de E ser maior do que A.
  - Se a concatenação das primeiras letras de todas as palavras de A for igual a um dos nomes de E no caso de A ser mais longo do que o nome mais longo de E.

Para além destas duas classes foi criada ainda uma classe responsável pelo tratamento das correferências, o *CorreferenceAnalyzer*. Como foi descrito anteriormente, o módulo de preenchimento de *slots* processa o XML de *input* evento a evento. Todos os eventos identificados pela STRING têm associados, pelo menos, uma entidade (pessoa ou organização). Essa entidade vem já marcada com o seu tipo (pessoa ou organização) portanto, a *string* que representa essa entidade é passada ao *CorreferenceAnalyzer* que por sua vez pesquisa nas suas listas de entidades, fazendo chamadas aos métodos *equals* de cada entidade, em busca de uma que satisfaça os critérios definidos. Se encontrar, devolve o índice da lista que corresponde à entidade encontrada, caso contrário devolve um índice inválido. Quando a entidade já existe, as novas informações presentes no evento são adicionadas à mesma. No caso do *slot* referente ao evento encontrado já se encontrar preenchido, e de forma a evitar informações repetidas, a nova informação tem de obedecer a alguns critérios para que possa ser guardada:

- Para o caso dos *slots* únicos a nova informação só substitui a antiga caso a *string* que a representa seja mais longa (em número de carateres) que a informação que o *slot* já continha, caso contrário não é adicionada.
- Para o caso dos *slots* simples não únicos, a nova informação só é adicionada se não existir na lista que representa o *slot*.
- No caso dos *slots* compostos únicos, é feito um *merge* da informação já existente com a nova só sendo adicionados campos que ainda não estiverem preenchidos no *GenericContainer* que guarda a informação do *slot*.
- Por fim, para o caso dos *slots* compostos não únicos, as novas informações são sempre adicionadas na forma de um novo *GenericContainer*, exceto no caso de já existir um *GenericContainer* que contenha a mesma informação que aquela que se está a tentar adicionar.

Este mecanismo de agregação de entidades foi desenvolvido para o processamento de cada documento de forma independente, não sendo adequado ao processamento de múltiplos documentos em simultâneo. Este facto permitiu que se assumisse a existência de *coerência semântica* (tema abordado na secção 2.2) relativamente aos assuntos presentes no documento. O mecanismo de agregação de entidades está relacionado com desambiguação de entidades mencionadas, no entanto, o que se pretende obter no final é ligeiramente diferente. Enquanto a desambiguação de entidades mencionadas tem como objetivo mapear uma referência presente num texto com uma entidade canónica (no caso da desambiguação de entidades mencionadas com recurso a bases de conhecimento, descrita na secção 2.2, a entidade canónica é um conceito da base de conhecimento) o objetivo neste trabalho é decidir se duas referências presentes no mesmo documento se referem à mesma entidade, ou seja, se são o mesmo conceito, não sendo necessário estabelecer nenhuma relação com elementos externos ao documento. Se for possível tomar esta decisão, é possível agrupar as informações que estão associadas a cada uma das referências.

3.2.3.4 Saída do módulo de preenchimento de *slots*

Como referido anteriormente, a saída dos módulos do AfterXIP, em particular do módulo de *Slot-Filling*, é feita no formato de XML. Tanto para as entidades do tipo PERSON como para as do tipo ORGANIZATION, existe uma estrutura básica que é comum a ambas. A *tag* FRAMES é única e aparece após as *tags* LUNITs. Esta *tag* só é gerada caso tenha sido identificada e extraída alguma informação pelo módulo de Preenchimento de *Slots*. A *tag* FRAMES contém vários nós com a *tag* FRAME, e cada um destes representa uma entidade. Os nós FRAME têm sempre dois atributos: o *name*, que é preenchido com o nome mais longo da entidade a que a FRAME se refere; e o *type*, que indica o tipo de entidade a que a FRAME se refere (PERSON ou ORGANIZATION).

Foi criado um tipo alternativo de saída para o módulo de preenchimento de *slots* por forma a facilitar a leitura. A listagem 3.17 contém a mesma informação que a listagem 3.16 mas desta vez neste formato alternativo mais facilmente legível.

Listagem 3.16: Excerto do ficheiro XML que contem uma *frame* do tipo PERSON

```
<RESULT>
  <LUNIT language="Portuguese">
    .
    .
    .
  </LUNIT>
  <FRAMES>
    <FRAME name="Joana Almeida" type="PERSON">
      <ALTERNATE-NAMES>
        <NAME val="Joana"/>
      </ALTERNATE-NAMES>
      <GENDER val="FEM"/>
      <BIRTH>
        <CITY val="Porto"/>
        <DATE string="a 25 de Agosto de 1975" umed=""
              valdelta="" valdelta1="" valdelta2=""
              valnorm="+CM------D19750825T----------E--LM--"
              valnorm1="" valnorm2="" valquant=""/>
      </BIRTH>
      <LOCATIONS-OF-RESIDENCE>
        <ELEM>
          <PLACE val="Rua Catilho"/>
          <DATE-START string="desde 2001" umed=""
                      valdelta="" valdelta1="" valdelta2=""
                      valnorm="+CM------D2001----T----------E--LMP"
                      valnorm1="" valnorm2="" valquant=""/>
        </ELEM>
      </LOCATIONS-OF-RESIDENCE>
      <CHILDREN>
        <PERSON name="Ant&#xF3;nio"/>
        <PERSON name="Maria"/>
      </CHILDREN>
    </FRAME>
    .
    .
    .
  </FRAMES>
</RESULT>
```

Listagem 3.17: Resultado do processamento do exemplo da tabela

```
------- PERSON -------
NAME: Joana Almeida
        ALTERNATIVE NAMES:
                Joana
GENDER: FEM
BIRTH:
        CITY: Porto
        DATE:
                STRING: a 25 de Agosto de 1975
                VAL-NORM: +CM------D19750825T----------E--LM--

RESIDENCE:
        CITY: Lisboa
        START-DATE:
                STRING: desde 2001
                VAL-NORM: +CM------D2001----T----------E--LMP
CHILDREN:
        Antonio
        Maria
```

Capítulo 4

Avaliação

A introdução dos novos módulos na cadeia de processamento de língua natural do L^2F implica que estes sejam testados segundo 2 perspetivas. Foi feita uma avaliação de desempenho e de qualidade de resultados. Em ambos os casos foi necessário definir uma metodologia de avaliação. Foram definidas métricas para avaliar os resultados e um conjunto de testes a executar. Depois de obtidos os resultados dos testes, procedeu-se uma análise detalhada dos mesmos. A secção 4.1 descreve a metodologia de avaliação e apresenta os resultados da avaliação bem como a análise dos mesmos.

4.1 Metodologia de Avaliação e Resultados

Para este trabalho foram definidos testes de desempenho e de qualidade. A fim de ser possível avaliar a qualidade do sistema de preenchimento de *slots*, é necessário distinguir entre erros provenientes de módulos anteriores ao *AfterXIP* e erros introduzidos pelos novos módulos. Este facto condicionou a metodologia utilizada para avaliar a qualidade do sistema. Esta metodologia é descrita detalhadamente na secção 4.1.1.

Com a finalidade de testar o desempenho dos novos módulos, foram medidos os tempos de execução dos sub-módulos do *AfterXIP*, e os resultados foram comparados com os tempos de execução dos restantes módulos da cadeia de processamento, por forma a ser verificado o impacto que a introdução destes novos módulos tinha no desempenho global da cadeia. A avaliação do desempenho encontra-se descrita na secção 4.1.2.

Em ambos os casos, foi utilizado um corpus de avaliação já existente no L^2F. Este é composto por 28 documentos extraídos de diversas fontes de notícias online. Em conjunto estes documentos contêm cerca de 700 frases, nas quais a cadeia identificou 651 entidades, 380 relações pertencentes a 146 eventos (28 da categoria LIFETIME e 96 da categoria BUSINESS). Foram identificadas ainda 38 dependências BUSINESS_PROFESSION.

4.1.1 Avaliação da Qualidade

Para avaliação da qualidade dos resultados do sistema foram processados os 28 documentos que compõem o corpus de avaliação e verificados manualmente os resultados obtidos. Os módulos de resolução de expressões anafóricas e de normalização de expressões temporais foram também executados e o ficheiro XML de saída da STRING enriquecido com os resultados destes módulos foi utilizado como entrada do módulo de preenchimento de *slots*. Foram contabilizados os seguintes tipos de erros:

Omissão de *Frame* - Considera-se que existe uma omissão de *frame* quando existe um evento que permite gerar determinada *frame* e o mesmo não acontece. ex: Pessoa A casada com Pessoa B e

o resultado apenas apresenta a *frame* da Pessoa A considera-se que existiu omissão da *frame* da Pessoa B.

Omissão de *Slot* - Quando existe a *frame* à qual o *slot* pertence mas este é omitido. ex: Pessoa A casada com Pessoa B e no resultado, na *frame* da Pessoa A não aparece o *slot* SPOUSE com o valor B.

***Frame* errada** - Considera-se que existe uma *frame* errada quando um evento que deveria gerar ou ser associado à *frame* A gera uma *frame* B.

***Slot* com valor errado** - Quando determinado *slot* é preenchido com um valor incorreto ou com valor vazio. ex: Pessoa A casada com Pessoa B e no resultado, na *frame* da Pessoa A, o *slot* SPOUSE surge preenchido com o valor C.

Foram empregues as métricas comummente utilizadas na avaliação de sistemas de PLN: precisão, cobertura e *F-measure*, todas calculadas tanto ao nível da *frame* e do *slot*. As respetivas fórmulas são apresentadas abaixo.

$$\textbf{Cobertura} = \frac{NRC}{NRC + NRE + NRO}$$

$$\textbf{Precisão} = \frac{NRC}{NRC + NRE}$$

$$\textbf{\textit{F-measure}} = \frac{2.Precisão.Cobertura}{Precisão + Coertura}$$

NRC = Número de respostas corretas.

NRE = Número de respostas erradas.

NRO = Número de respostas omitidas.

Os tipos de erro anteriormente mencionados são considerados recorrendo à observação das dependências provenientes do XIP e considerando que estas estão corretas. Desta forma assume-se como 100% correto o *output* devolvido pelo XIP e portanto verificam-se apenas os erros que são introduzidos pelo novo módulo. De forma a conseguir fazer esta verificação foi comparado manualmente a entrada do *AfterXIP* (mais precisamente as dependências do tipo **EVENT** que são geradas pelo XIP) com o seu *output* (*frames* geradas pelo sistema), e contabilizados os erros dos tipos anteriormente descritos. Os erros dos módulos de resolução de expressões anafóricas e de normalização de expressões temporais foram tratados da mesma forma que os erros em eventos, ou seja, foram tomados como respostas corretas. As métricas apresentadas foram utilizadas tanto para avaliar o sistema a nível dos *slots* como a nível das *frames* geradas. Foram contabilizados um total de 517 *slots* dos quais 422 foram preenchidos corretamente, 77 com erro e 18 omitidos. Quanto às *frames* foram identificadas 148 das quais 119 foram geradas corretamente, 25 com erro e 4 omitidas. A Tabela 4.1 mostra os resultados finais dos testes efetuados.

Relativamente aos resultados, verificou-se que a grande maioria dos erros encontrados deve-se, no caso dos *slots*, a repetições de valores para os mesmos *slots* e ainda a preenchimentos com valores vazios. Existem ainda situações que não estão refletidas nesta avaliação e estão relacionadas com o módulo de resolução de expressões anafóricas. Verificou-se que, por diversas vezes nos testes, surgiam *frames* de entidades em que o valor do *slot name* era um pronome. Nestes casos não foi considerado erro do módulo

Métrica	*Slot*	*Frame*
Precisão	84,57%	82,64%
Cobertura	81,62%	80,41%
F-measure	83,07%	81,51%

Tabela 4.1: Resultados das métricas de qualidade.

de preenchimento de *slots* pois o pronome surgia nos eventos. Isto deve-se ao facto de que o módulo de resolução de expressões anafóricas não resolveu a anáfora e portanto trata-se de um erro anterior ao módulo desenvolvido.

4.1.2 Avaliação do Desempenho

Teste	Input	LexMan	RuDriCo	MARv	XIP	Input	TimeNorm	ARM	SF
1	0.16	6.71	1.93	0.64	7.94	1.05	0.11	0.14	0.24
2	0.17	7.71	0.85	0.49	5.79	1.09	0.11	0.17	0.03
3	0.17	7.91	0.39	0.37	3.03	0.62	0.04	0.05	0.02
4	0.14	7.85	1.28	0.51	5.51	1.1	0.11	0.28	0.04
5	0.17	7.26	0.55	0.42	3.84	0.81	0.05	0.02	0.05
6	0.17	8.37	1.19	0.36	3.77	1.1	0.12	0.12	0.05
7	0.14	8.32	1.74	0.62	6.23	1.21	0.11	0.57	0.02
8	0.15	7.54	0.34	0.37	3.43	0.44	0.03	0.03	0.01
9	0.15	7.59	0.25	0.32	3.28	0.45	0.05	0.01	0.01
10	0.17	7.84	0.65	0.42	3.85	0.93	0.06	0.05	0.02
11	0.15	7.58	0.47	0.39	3.79	0.93	0.05	0.05	0.01
12	0.17	6.92	0.69	0.33	4.27	1.09	0.07	0.06	0.02
13	0.14	8.26	1.36	0.46	5.13	1.07	0.09	0.22	0.03
14	0.15	7.22	0.34	0.37	3.35	0.46	0.04	0.02	0.01
15	0.16	6.59	0.69	0.41	3.98	0.91	0.07	0.08	0.01
16	0.15	7.84	0.74	0.38	4.34	0.87	0.05	0.05	0.02
17	0.13	7.84	0.78	0.42	4.3	0.89	0.07	0.14	0.01
18	0.15	7.28	0.54	0.39	3.55	0.89	0.06	0.04	0.01
19	0.16	7.19	0.81	0.45	4.47	1.0	0.08	0.11	0.02
20	0.17	7.78	0.63	0.43	3.4	0.88	0.06	0.13	0.02
21	0.16	8.01	0.73	0.43	4.23	0.94	0.06	0.09	0.01
22	0.16	6.07	0.18	0.32	2.58	0.25	0.03	0.02	0.01
23	0.17	7.9	0.52	0.39	3.79	0.91	0.05	0.17	0.01
24	0.16	8.04	1.47	0.6	5.81	1.1	0.1	0.15	0.03
25	0.16	7.7	1.98	0.58	7.41	1.3	0.11	0.15	0.06
26	0.17	8.47	1.42	0.52	11.05	1.23	0.11	0.21	0.09
27	0.16	8.55	1.33	0.48	5.86	1.18	0.11	0.23	0.08
28	0.16	8.41	1.36	0.49	4.62	0.79	0.07	0.21	0.02
Total	4,42	214,75	25,21	12,36	132,6	25,49	2,07	3,57	0,96

Tabela 4.2: Tempos de processamento (em segundos) dos diversos módulos da STRING.

Para avaliar o impacto que a introdução dos novos módulos teve na cadeia de processamento, foram processados os 28 documentos do corpus de teste e anotados os tempos de processamento de cada uma das etapas tendo sido obtidos os resultados apresentados na Tabela 4.2. As execução destas tarefas foi feita de forma encadeada e sempre na mesma máquina, que apresenta as seguintes especificações:

- Processador: Intel® Xeon® CPU E5530 @ 2.40GHz

- RAM: 48GB

Como se pode verificar, o impacto da introdução dos módulos do *AfterXIP* na STRING é mínimo sendo os tempos de execução deste módulos substancialmente mais baixos do que os restantes módulos da cadeia. Relativamente ao tempo do módulo de preenchimento de *Slots*, verifica-se que este representa aproximadamente 0,23% do tempo total de processamento, podendo, portanto, considerar-se que o impacto da introdução deste módulo na cadeia é mínimo. Quando ao impacto da introdução do *AfterXIP* como um todo, verifica-se que o seu tempo de execução representa aproximadamente 7,62% do tempo total de execução. Este resultado é bastante satisfatório, uma vez que neste tempo se inclui a leitura do ficheiro XML produzido pelo XIP e 3 processamentos do mesmo ficheiro efetuados pelos módulos de normalização de expressões temporais, resolução de anáfora e preenchimento de *slots*.

Capítulo 5

Conclusões e Trabalho Futuro

Neste trabalho estudou-se a tarefa de Preenchimento de *Slots* para língua portuguesa. Foram apresentados vários sistemas que abordam esta e outras tarefas relacionadas e os métodos que utilizam para o fazer. Com base neste estudo, partiu-se para a resolução do problema de extração e informação estruturada a partir de documentos em língua portuguesa.

Foram integrados na cadeia de processamento de língua natural do L^2F (STRING) os módulos de resolução de expressões anafóricas e o módulo de normalização de expressões temporais (desenvolvidos no contexto de trabalhos anteriores) por forma a que o resultado de processamento dos mesmos pudesse ser utilizado e incluído neste trabalho. Esta integração foi importante para que o sistema de preenchimento de *slots* desenvolvido pudesse atuar sobre o processamento efetuado por estes módulos.

O módulo de preenchimento de *slots* desenvolvido neste trabalho processa a saída do XIP (incluindo as anotações feitas pelos módulos de normalização de expressões temporais e de resolução de expressões anafóricas). Este módulo possui os mecanismos necessários para realizar resolução de correferências para um único documento através de regras, o que possibilita a agregação (por entidade) e reestruturação de informação proveniente dos eventos da STRING. Por fim, este módulo foi integrado na STRING e foram desenvolvidos os mecanismos que permitem que o resultado do preenchimento de *slots* fosse integrado no XML de saída da STRING.

Os resultados deste trabalho foram avaliados manualmente, recorrendo aos métodos e métricas descritos no capítulo 4. Estes resultados, avaliando apenas a qualidade do módulo de preenchimento de *Slots*, revelaram-se muito satisfatórios, tanto ao nível dos *slots* como das *frames*, com valores próximos de precisão e cobertura, e na *F-measure* de 83,07% para os *slots* e 81,51% para as *frames*

Como trabalho futuro, e a fim de melhorar os resultados já obtidos, sugere-se o investimento na melhoria do modulo de resolução de expressões anafóricas. Como evolução e, para que o sistema passe a ter capacidade para processar múltiplos documentos de forma simultânea, será necessário o desenvolvimento de métodos mais sofisticados de resolução de correferências, sugerindo-se como exemplo a introdução de um sistema de desambiguação de entidades mencionadas, recorrendo a bases de conhecimento, semelhante aos abordados neste documento.

Bibliografia

[Agichtein and Gravano, 2000] Agichtein, E. and Gravano, L. (2000). Snowball: extracting relations from large plain-text collections. In *Proceedings of the fifth ACM Conference on Digital Libraries*, DL '00, pages 85–94. ACM.

[Anastácio et al., 2011] Anastácio, I., Martins, B., and Calado, P. (2011). Supervised learning for linking named entities to knowledge base entries. In *Proceedings of Text Analysis Conference 2011*, KBP '11. National Institute of Standards and Technology.

[Baptista et al., 2012] Baptista, J., Cabarrão, V., and Mamede, N. (2012). *Classification Directives for Events and Relations Extraction between Named Entities in Portuguese Texts.* Instituto Superior Técnico.

[Baptista et al., 2011] Baptista, J., Mamede, N., Oliveira, D., and Santos, D. (2011). *Classification Directives for Named Entities in Portuguese Texts.* L2F - INESC-ID, Lisbon.

[Bunescu and Pasca, 2006] Bunescu, R. and Pasca, M. (2006). Using encyclopedic knowledge for named entity disambiguation. In *11th Conference of the European Chapter of the Association for Computational Linguistics*, pages 9–16.

[Byrne and Dunnion, 2011] Byrne, L. and Dunnion, J. (2011). Ucd iirg at tac 2011. In *Proceedings of Text Analysis Conference 2011*, KBP '11. National Institute of Standards and Technology.

[Cao et al., 2007] Cao, Z., Quin, T., Liu, T.-Y., Tsai, M.-F., and Li, H. (2007). Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, pages 129–136. ACM.

[Collins and Duffy, 2002] Collins, M. and Duffy, N. (2002). New ranking algorithms for parsing and tagging: kernels over discrete structures, and the voted perception. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 263–270.

[Diniz, 2010] Diniz, C. (2010). RuDriCo2 - *Um Conversor Baseado em Regras de Transformação Declarativas.* Master's Thesis, Instituto Superior Técnico, Lisboa.

[Forney, 1973] Forney, G. D. (1973). The viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278.

[Grishman and Min, 2010] Grishman, R. and Min, B. (2010). New York University KBP 2010 Slot-Filling System. In *Proceedings of Text Analysis Conference 2010*, KBP '10. National Institute of Standards and Technology.

[Hoffart et al., 2011] Hoffart, J., Yosef, M. A., Bordino, I., Fürstenau, H., Pinkal, M., Spaniol, M., Taneva, B., Thater, S., and Weikum, G. (2011). Robust disambiguation of named entities in text. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 782–792. Association for Computational Linguistics.

[Joachims, 1999] Joachims, T. (1999). *Advantages in Kernel Methods - Support Vector Learning*, chapter Making large-scale SVM learning practical, pages 169–184. MIT Press, Cambridge, USA.

[Joachims, 2002] Joachims, T. (2002). Optimizing search engines using clickthrough data. In *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '02, pages 133–142. ACM.

[Kulkarni et al., 2009] Kulkarni, S., Singh, A., Ramakrishnan, G., and Chakrabarti, S. (2009). Collective annotation of Wikipedia entities in web text. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 457–465.

[Liu, 2009] Liu, T.-Y. (2009). Learning to rank for information retrieval. In *Foundations and Trends in Information Retrival*, volume 3, pages 225–331.

[Maurício, 2011] Maurício, A. S. B. (2011). *Identificação, Classificação e Normalização de Expressões Temporais*. Master's Thesis, Instituto Superior Técnico.

[Metzler and Croft, 2007] Metzler, D. and Croft, W. B. (2007). Linear feature-based models for information retrieval. *Inf. Retr.*, 10:257–274.

[Milne and Witten, 2008] Milne, D. and Witten, I. H. (2008). Learning to link with Wikipedia. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management*, pages 509–518.

[Mintz et al., 2009] Mintz, M., Bills, S., Snow, R., and Jurafsky, D. (2009). Distant supervision for relation extraction without labeled data. In *Proceedings ACL-IJCNLP*, pages 1003–1011, Singapore. Association for Computational Linguistics.

[Nobre, 2011] Nobre, N. R. P. (2011). *Resolução de Expressões Anafóricas*. Master's Thesis, Instituto Superior Técnico, Lisboa.

[Pasca, 2003] Pasca, M. (2003). *Open Domain Question Answering from Large Text Collections*. Center for the Study of language and Information.

[Rabiner and Juang, 2003] Rabiner, L. and Juang, B. (2003). An introduction to hidden markov models. *ASSP Magazine, IEEE*, 3(1):4–16.

[Ribeiro, 2003] Ribeiro, R. (2003). *Anotação Morfossintáctica Desambiguada em Português*. Master's Thesis, Instituto Superior Técnico, Lisboa.

[Sun et al., 2011a] Sun, A., Grishman, R., and Skekine, S. (2011a). Semisupervised relation extraction with large-scale word clustering. In *Proceedings of ACL 2011*.

[Sun et al., 2011b] Sun, A., Grishman, R., Xu, W., and Min, B. (2011b). New York University 2011 System for KBP Slot Filling. In *Proceedings of Text Analysis Conference 2011*, KBP '11. National Institute of Standards and Technology.

[Vicente, 2013] Vicente, A. M. F. (2013). *LexMan: um Segmentador e Analisador Morfológico com transdutores*. Master's Thesis, Instituto Superior Técnico.

[Xerox, 2003] Xerox. Xerox incremental parser – reference guide [online]. (2003). Available from: https://open.xerox.com/Repo/service/XIPParser/public/XIPReferenceGuide.pdf.

[Xu and Li, 2007] Xu, J. and Li, H. (2007). Adarank: a boosting algorithm for information retrieval. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Reasearch and Development in Information Retrieval*, SIGIR '07, pages 391–389. ACM.