Syntax Deep Explorer

José Miguel Pinheiro Correia

Dissertação para obtenção do Grau de Mestre em

Engenharia Informática e de Computadores

Orientadores: Prof. Doutor Nuno João Neves Mamede
Prof. Doutor Jorge Manuel Evangelista Baptista

Júri

Presidente:	Prof. Doutor Alberto Manuel Rodrigues da Silva
Orientador:	Prof. Doutor Nuno João Neves Mamede
Vogal:	Prof. Doutor Bruno Emanuel da Graça Martins

Outubro 2015

Agradecimentos

A realização deste projeto não seria possível sem a ajuda e apoio de pessoas importantes a quem devo agradecer.

Aos meus orientadores Prof. Nuno Mamede e Prof. Jorge Baptista, agradeço o apoio, atenção e disponibilidade dada ao longo do projeto, permitindo a realização deste.

Aos meus amigos e colegas de casa presentes nesta viagem, pelo companheirismo.

À minha namorada Marta, agradeço imenso todo o incentivo, paciência e ajuda que me deu.

Aos meus pais, irmã e avós pelo seu amor e apoio incondicional, em especial ao meu avô Leopoldo.

A todos aqueles que não referi mas que também eles contribuíram direta ou indiretamente para a realização deste projeto.

Lisboa, 16 de Outubro de 2015

José Miguel Pinheiro Correia

Resumo

A análise de padrões de co-ocorrência de palavras em textos, permite compreender as diferenças de uso (e de significado) que estão associadas às diferentes relações em que uma dada palavra participa. A quantificação desses padrões é um poderoso instrumento na lexicografia moderna bem como na construção de recursos linguísticos fundamentais para o processamento das línguas naturais, ou para a aprendizagem de uma língua.

O objetivo deste projeto é desenvolver uma ferramenta que, com base na cadeia de processamento de língua natural (PLN) STRING permita obter o acesso aos dados de co-ocorrência obtidos a partir de textos em português.

Atualmente, já existem várias ferramentas (*DeepDict*, *Sketch Engine* e *Wortschatz*) que permitem obter, para *corpora* em português, informação sobre os padrões de co-ocorrência de uma palavra, baseando-se em diferentes sistemas de PLN, adotando diferentes medidas de associação. Entre estas, encontram-se a *Mutual Information*, o Coeficiente de *Dice*, o *Log-likelihood Ratio*, ou diferentes adoções destas medidas de base.

A solução apresentada é composta pela extração das co-ocorrências e uma interface *Web*. A extração ocorre a partir de um *corpus* processado pela cadeia, que encontra e armazena as co-ocorrências numa base de dados, sendo posteriormente calculadas as diferentes medidas de associação. A aplicação *Web* fornece aos utilizadores uma interface que permite explorar esses padrões de co-ocorrência. A solução é avaliada com base no tempo consumido para extrair as co-ocorrências do *corpus* CETEMPúblico, espaço ocupado e organização da base de dados e o tempo de resposta da interface.

O projeto desenvolvido permite aceder rapidamente às co-ocorrências resultantes de *corpora* processado pela STRING, aproveitando os ricos recursos lexicais da cadeia bem como a sua sofisticada análise sintática e semântica, para produzir resultados que os sistemas referidos não permitem.

Abstract

The analysis of the co-occurrence patterns between words allows to understand the use (and meaning) of words that are associated with different relationships. The quantification of these standards is a powerful tool in modern lexicography as well as in the construction of basic linguistic resources for the processing of natural language, or learning a language.

The aim of this project is to develop a tool that, based on the STRING natural language processing chain, allows one to explore co-occurrence data obtained from Portuguese texts.

Nowadays, there are some tools like DeepDict, SketchEngine and *Wortschatz* that allow to get the information on the co-occurrence patterns of a word in Portuguese corpora. These tools are based on different natural language processing systems and adopt different measures of association. The association measures used are the Mutual Information, the Dice coefficient, the Log-likelihood ratio, or different variants of these base measures.

The presented solution consists in the extraction of co-occurrences and a web interface. The extraction occurs from a processed corpus by STRING, that finds and stores the co-occurrences in a database. Then, for each co-occurrence stored are calculated the different association measures. The web application provides to users an interface that allows to exploit these co-occurrence patterns. The solution is evaluated based on consumed time to extract the co-occurrences from CETEMPúblico corpus, the space and organization of the database and the response time of web interface.

The developed project allows the quick access to collected co-occurrences in *corpora* produced by STRING, taking advantage of the rich lexical resources in the chain, as well as its sophisticated syntactic and semantic analysis in order to produce results that the above systems don't allow.

Palavras-Chave

Palavras-Chave

Processamento de Língua Natural (PLN)

Interface gráfica

Co-ocorrência

Colocação

Medidas de associação

STRING

Keywords

Natural Language Processing (NLP)

Graphic interface

Co-occurrence

Colocation

Association measures

STRING

Índice

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas

AJAX	**A**synchronous **J**avaScript **a**nd **X**ML
API	**A**pplication **P**rogramming **I**nterface
CQL	**C**orpus **Q**uery **L**anguage
EA	Modelo **E**ntidade **A**ssociação
HTTP	**H**yper**t**ext **T**ransfer **P**rotocol
INESC-ID	**In**stituto **E**ngenharia e de **S**istemas e **C**omputadores - **I**nvestigação e **D**esenvolvimento
JDBC	**J**ava **D**atab**a**se **C**onnectivity
JSON	**J**ava**S**cript **O**bject **N**otation
LexMan	**Lex**ical **M**orfological **An**alizer
L^2F	**L**aboratório de Sistemas de **L**íngua **F**alada
MARv	**M**orphosyntactic **A**mbiguity **R**esol**v**er
PLN	**P**rocessamento de **L**íngua **N**atural
PMI	**P**ointwise **M**utual **I**nformation
POS	**P**art **o**f **S**peech - Classe de uma palavra
RI	**R**estrições de **i**ntegridade
RuDriCo	**Ru**le-**Dri**ven **Co**nverter
STRING	**St**atistical and **R**ule-based natural language process**ing**
XIP	**X**erox **I**ncremental **P**arser
XML	E**x**tensible **M**arkup **L**anguage

Capítulo 1

Introdução

Hoje em dia existe uma grande quantidade de ferramentas *online* disponíveis na *web*, como dicionários em formato eletrónico. Estas novas versões de dicionários, como o Léxico.pt[1] ou o Priberam[2], mostram mais informação que a simples definição de uma palavra, disponibilizando também sinónimos e exemplos da utilização de uma palavra em textos públicos. Isto é possível porque alguns dos dicionários *online* têm como base informação obtida a partir de *corpora*, por isso têm potencialmente uma melhor cobertura do que um dicionário tradicional [6].

1.1 Problema

Embora os dicionários mais recentes já usem informação obtida a partir de *corpora*, não deixam de ser dicionários tradicionais, pois a informação extraída é apenas usada para exemplificação e contagem de frequência de uso da palavra pesquisada. Assim, estas não mostram informação suficiente sobre o uso da língua, isto é, dados acerca de co-ocorrências entre palavras que permitem explorar padrões usados.

Os principais interessados na análise destes padrões são os linguistas, estudantes de novas línguas, tradutores e investigadores de *corpora* linguísticos que pretendam estudar o comportamento das expressões linguísticas, para permitir uma melhor compreensão do uso da língua através de exemplos concretos.

Para uma melhor análise dos padrões, é importante entender a evolução do uso da língua, para isso é necessário observar *corpora* de diferentes épocas, permitindo a comparação desses resultados. Outro meio de comparação pode ser a origem de diferentes corpora, podendo estes serem separados em diferentes géneros, tipos textuais, domínios, etc., que possibilitem obter os padrões correspondentes.

Para facilitar uma melhor compreensão das relações entre palavras, é importante fornecer uma interface que facilite ao utilizador a compreensão dos resultados apresentados, dando-lhe o controlo de tomada de decisão sobre o tipo de pesquisa a efetuar, isto é, que tipo de *corpora* pretende usar, qual a medida de associação para a co-ocorrência entre palavras, qual o valor mínimo da frequência da co-ocorrência, entre outros.

[1] http://www.lexico.pt/ (última visita a 7/10/2015).
[2] http://www.priberam.pt/DLPO/ (última visita a 7/10/2015).

1.2 Objectivos

Para extrair padrões de co-ocorrência é necessário dispor de *corpora* analisados através de uma ferramenta de Processamento de Língua Natural (PLN). Neste caso, será usada a cadeia de PLN STRING [3] (*STatistical and Rule-based Natural lanGuage processing chain*) [21].

1.2.1 STRING

A STRING é uma cadeia de PLN para o português baseada em regras e estatística e desenvolvida pelo laboratório Sistemas de Língua Falada (L^2F) do INESC-ID Lisboa [21].

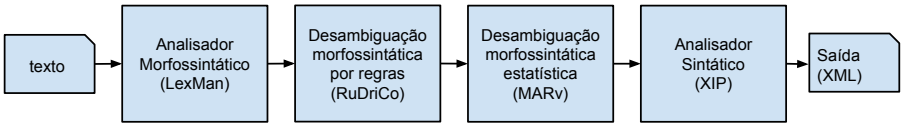


Figura 1.1: Cadeia de Processamento STRING.

Como mostra a Figura 1.1, a STRING tem uma estrutura modular. O primeiro módulo consiste num analisador morfossintático, o LexMan [32], que é responsável por segmentar o texto de entrada e de seguida atribuir a cada segmento todas as etiquetas morfológicas possíveis. Por fim, todos os segmentos são enviados para o próximo módulo.

Os próximos dois módulos são desambiguadores morfossintáticos. O RuDriCo [14] é responsável por alterar a segmentação, podendo contrair ou separar os segmentos de modo a resolver algumas ambiguidades. Este pode, por exemplo, separar as palavras para resolver as contracções ou contrair dois segmentos e assim obter palavras compostas. O MARv [26] trata as ambiguidades morfossintáticas ainda não resolvidas pelo módulo anterior, escolhendo para cada palavra a etiqueta mais provável, obtida a partir do algoritmo de Viterbi [16]. O modelo de língua usado pelo MARv é baseado em modelos de segunda ordem (trigramas), que codificam informação contextual relativa às entidades, e unigramas que codificam informação lexical. O modelo foi treinado com um *corpus* de português de 250 mil palavras e apresenta uma precisão global superior a 98% [21].

O último módulo é um analisador sintático, o XIP (*Xerox Incremental Parser*) [1] que usa as regras gramaticais da língua portuguesa para dividir o texto em constituintes sintáticos elementares (ou *chunks*, por exemplo grupo nominal (*NP*), etc.) e estabelece relações de dependência sintática (Sujeito, etc.) entre eles.

O resultado obtido na cadeia STRING pode ainda ser sujeito a processamento por outros módulos, de forma a conseguir a resolução de anáforas e a normalização de expressões temporais [21]. O processamento resulta num ficheiro XML que contém o resultado do XIP juntamente com os produzidos pelos módulos pós STRING.

1.2.1.1 Dependências

Neste projeto pretende-se usar as dependências extraídas no XIP de modo a produzir co-ocorrências entre palavras. Essas dependências são normalmente relações binárias, onde o primeiro argumento corresponde ao elemento modificado (*governed*) e o segundo corresponde ao elemento modificador (*governor*). De todas as dependências sintáticas extraídas pelo XIP, são usadas as seguintes [22]:

[3] http://string.l2f.inesc-id.pt/ (última visita a 2/1/2015).

SUBJ: dependência que liga um verbo e o seu sujeito (`"O Pedro dorme." SUBJ(dorme,Pedro)`);

CDIR: estabelece relação entre o verbo e o seu complemento directo (`"O Pedro comeu fruta." CDIR(comeu, fruta)`);

CINDIR: relaciona o verbo e o seu complemento indirecto (`"O Pedro deu um livro à Ana." CINDIR(deu, Ana)`);

MOD: representa uma relação ente um núcleo (verbal, nominal ou doutra categoria) e o seu modificador (`"O carro grande." MOD(carro,grande)`);

COMPL: dependência que liga um predicado (verbo, substantivo ou adjetivo) com um dos seus complementos essenciais (`"O Pedro está confiante na vitória." COMPL(confiante,vitória)`);

QUANTD: dependência que liga um nome a um quantificador (`"O Pedro leu 5 livros." QUANTD(livros, 5)`);

CLASSD: dependência que liga um nome a um classificador nominal (`"O Pedro lê este tipo de livros." CLASSD(livros,tipo)`).

Nas dependências `MOD`, `SUBJ` e `CDIR` pode ser adicionado um sufixo que diz qual a ordem dos elementos de uma dependência no texto. Quando o sufixo é `PRE`, o segundo elemento da dependência ocorre antes do primeiro. Por outro lado, o sufixo `POST` indica que o segundo argumento da dependência ocorre após o primeiro elemento.

Para além destas dependências sintáticas, neste projeto será aproveitado o facto de o XIP detetar as entidades mencionadas (NE, *named entities*). Esta é uma dependência semântica e unária, que classifica elementos sintáticos do texto em diversas categorias mais gerais. As entidades mencionadas podem ser classificadas em: pessoas; organizações; local ou nomes de edifícios; expressões numéricas; medidas; moedas; expressões de tempo; data; duração e frequências [22].

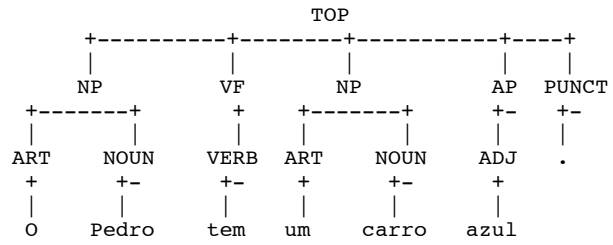


Figura 1.2: Árvore de *chunks*[4].

A Figura 1.2 mostra a árvore de *chunks* que resulta da análise na cadeia STRING da frase `"O Pedro tem um carro azul"`. As dependências aqui extraídas são `SUBJ_PRE(tem,Pedro)` que indica que "Pedro" é sujeito do verbo "ter", `CDIR_POST(tem,carro)` indica que "carro" é complemento direto do verbo "ter", `MOD_POST(carro,azul)` que a palavra "azul" modifica o nome "carro" e a entidade mencionada `NE_PEOPLE_INDIVIDUAL(Pedro)` que indica que "Pedro" é uma entidade mencionada da categoria "Pessoa".

[4]Retirado de http://string.l2f.inesc-id.pt/demo/chunktrees.pl (última visita a 8/10/2015).

1.2.2 Objetivos

Com o *corpus* processado pela cadeia STRING, pode-se desenvolver uma ferramenta que satisfaça a necessidade dos linguistas e dos restantes utilizadores para explorar o uso das palavras e explorar os padrões linguísticos.

O que se pretende com este trabalho é o desenvolvimento de uma ferramenta com os seguintes objetivos:

- produzir a partir de *corpora* processado pela STRING informação sobre os padrões de co-ocorrência;
- calcular o valor de co-ocorrência das palavras usando vários métodos estatísticos;
- armazenar a informação sobre os padrões de co-ocorrência a partir de vários *corpora* distintos;
- produzir uma interface amigável que permita o acesso à informação armazenada.

A interface gráfica deve permitir aos utilizadores, nomeadamente:

- explorar os padrões de co-ocorrência entre palavras, que possibilite recolher informação sobre o uso da língua;
- a escolha do método estatístico pretendido durante a consulta dos padrões;
- mostrar frases de exemplo para uma determinada co-ocorrência;
- a escolha entre vários *corpora* para efectuar pesquisas.

1.3 Contributos

O trabalho desenvolvido pretende contribuir e ajudar no desenvolvimento da ferramenta STRING. Este permite explorar de forma simples as diferentes relações entre as palavras processadas pela ferramenta, de modo a detetar os padrões produzidos por esta de uma forma mais intuitiva, para assim evitar o acesso a múltiplos ficheiros cujo conteúdo é o texto processado pela STRING.

Por outro lado, este trabalho também pretende que qualquer utilizador possa tirar partido das funcionalidades desenvolvidas, de forma a estudar o uso da língua portuguesa.

O trabalho realizado encontra-se armazenado num repositório do *github*[5].

1.4 Estrutura do Documento

O Capítulo 2 do documento apresenta o estado de arte onde, não só são descritas as medidas de associação entre palavras aplicadas no sistema desenvolvido, mas também são estudados os sistemas semelhantes a este.

No Capítulo 3, é descrita a arquitetura da solução e todos os passos usados durante o desenvolvimento da ferramenta.

A avaliação do trabalho realizado é apresentado no Capítulo 4, sendo expostas as conclusões deste no Capítulo 5.

[5] https://github.com/josepcorreia/DeepExplorer (última visita a 18/11/2015).

Capítulo 2

Trabalho Relacionado

Nesta Capítulo, descrevem-se as diferentes medidas de associação que podem ser usadas para estimar o grau de coesão entre duas palavras. Também se descrevem alguns trabalhos semelhantes ao desenvolvido.

2.1 Medidas de Associação

Nesta Secção são apresentadas as métricas usadas para calcular o grau de coesão entre duas palavras, nomeadamente *Pointwise Mutual Information*, coeficiente de *Dice*, *LogDice*, Chi-quadrado de *Pearson*, *Log-likelihood Ratio* e *Significance Measure*.

2.1.1 *Pointwise Mutual Information*

A primeira medida apresentada é o método *Pointwise Mutual Information* [11], que relaciona o número de co-ocorrências entre duas palavras com o número de ocorrências de cada palavra individualmente, medindo a sobreposição destas. Esta medida é definida pela equação (2.1).

$$PMI = \log\left(\frac{Nf(x,y)}{f(x)f(y)}\right) \tag{2.1}$$

onde $f(x,y)$ corresponde ao número de vezes que a palavra x co-ocorre com a palavra y, $f(x)$ o número de ocorrências de x e $f(y)$ o número de ocorrências de y num *corpus* com número total de palavras N. Muitas vezes esta medida sobre-estima a co-ocorrência entre palavras com frequência baixa.

2.1.2 Coeficiente de *Dice*

O coeficiente de *Dice* [13, 31] consiste na medição do grau de coesão que existe entre duas palavras e é definido pela equação (2.2).

$$D = \frac{2f(x,y)}{f(x)+f(y)} \tag{2.2}$$

onde $f(x,y)$ corresponde o número de co-ocorrências entre x e y, $f(x)$ e $f(y)$ corresponde o número de co-ocorrências de x e y respetivamente.

2.1.3 *LogDice*

A medida *LogDice* [28] foi criada com o objetivo de resolver o problema do coeficiente de *Dice*, que resulta do facto de os valores obtidos serem normalmente muito pequenos e de difícil leitura. O `logDice` é definido pela equação (2.3).

$$LD = 14 + \log_2(D) = 14 + \log_2\left(\frac{2f(x,y)}{f(x)+f(y)}\right) \tag{2.3}$$

onde o valor máximo do *logDice* é 14, que acontece quando todas as ocorrências de x ocorrem com y e vice-versa. Por norma o valor é inferior a 10, e é 0 se houve apenas uma co-ocorrência de x e y em 16.000 palavras. Se o número de co-ocorrências for inferior, o resultado será negativo que significa que x e y não têm nenhuma relação significante [28].

2.1.4 Chi-quadrado de Pearson

A medida chi-quadrado de *Pearson* [23] é um método estatístico de teste de hipóteses que testa a hipótese nula. Este compara as frequências observadas com as frequências esperadas de modo a verificar a independência entre os eventos, onde a frequência esperada é a probabilidades marginal para as saídas de um evento. Se a diferença entre as frequências observadas e esperadas for grande, então a hipótese nula é rejeitada. Para simplificar, é usada uma tabela de contingências, sendo o método definido pela equação (2.4).

$$\chi 2 = \sum_{i,j} \frac{(O_{ij} - E_{ij})^2}{E_{ij}} \tag{2.4}$$

onde O_{ij} representa a frequência observada e E_{ij} a frequência esperada, e i e j são as colunas e linhas da tabela.

Tabela 2.1: Tabela de contingência entre duas palavras.

	palavra1 $= x$	palavra1 $\neq x$
palavra2 $= y$	O_{11}	O_{12}
palavra2 $\neq y$	O_{22}	O_{22}

Com base na Tabela de contingências 2.1, onde as variáveis são duas palavras x e y, a hipótese nula passa a ser que as duas palavras são independentes. Caso o resultado seja próximo de 0, a hipótese nula verifica-se e as duas palavras são independentes, o que significa que estas não co-ocorrem entre elas. Caso contrário, o valor obtido será o valor de co-ocorrência entre as palavras. Assim, para simplificar, o método passa a ser definido pela equação (2.5) [23].

$$\chi 2 = \frac{N(O_{11}O_{22} - O_{12}O_{21})^2}{(O_{11}+O_{12})(O_{11}+O_{21})(O_{12}+O_{22})(O_{21}+O_{22})} \tag{2.5}$$

onde O_{ij} são os valores da frequência observada para cada posição $i, j \in \{1,2\}$ da tabela de contingências e N representa o número total de palavras no *corpus*. De acordo com [23], o teste χ^2 não deve ser usado em *corpora* de pequena dimensão.

2.1.5 *Log-likelihood Ratio*

O método *Log-likelihood Ratio* [15] é outra abordagem ao teste de hipóteses, sendo mais apropriados para dados esparsos que o método χ^2. A partir de duas hipóteses definidas no início, o método tem como resultado um número que diz qual das hipóteses é a mais provável. Na aplicação do método para descobrir se duas palavras, x e y, co-ocorrem frequentemente, as duas hipóteses são:

-Hipótese 1: $P(y|x) = p = P(y|\neg x))$, que diz que a ocorrência de y é independente da ocorrência de x;

-Hipótese 2: $P(y|x) = p1 \neq p2 = P(y|\neg x))$, que diz que a ocorrência de y é dependente duma prévia ocorrência de x.

Quanto maior for o valor calculado, maior é a probabilidade da segunda hipótese ser verdadeira. [23]. O método pode ser definido pela equação 2.6.

$$\begin{aligned} Loglikelihood &= -2\log\lambda = \\ &= -2\big(\log L(c_{12}, c_1, p) + \log L(c_2 - c_{12}, N - c_1, p) \\ &\quad - \log L(c_{12}, c_1, p1) - \log L(c_2 - c_{12}, N - c_1, p_2)\big) \end{aligned} \tag{2.6}$$

onde $L(k,n,z) = z^k(1-z)^{n-z}$, c_1 é a frequência da palavra x, c_2 é a frequência da palavra y, c_{12} a frequência das duas palavras ocorrerem juntas, $p = c_2/N$, $p_1 = c_{12}/c_1$, $p2 = (c_2 - c_{12})/(N - c_1)$ e N o número de palavras do *corpus*.

2.1.6 *Significance Measure*

A medida de Significância [7] é comparável com a medida estatística *G-Test* para distribuições de Poisson. Dadas duas palavras X e Y, cada uma ocorre x e y vezes nas frases, e k vezes em conjunto. A medida é definida pela equação (2.7).

$$sig(X,Y) = z - k \times \log(z) + \log(k!) \tag{2.7}$$

onde $z = \frac{x \times y}{N}$ e N é o número total de frases do texto.

2.2 Sistemas

Nesta Secção são apresentados os sistemas estudados, pela seguinte ordem: o *DeepDict* [6], o *Sketch Engine* [19] e, por último, o projecto *Wortschatz* [7].

2.2.1 DeepDict

O primeiro sistema que iremos apresentar, o *DeepDict*, constituiu a inspiração inicial para este projecto, razão porque o apresentamos com algum pormenor.

O *DeepDict* é uma ferramenta desenvolvida pela GrammarSoft[1] e lançada comercialmente em Setembro de 2007 [6], na plataforma Gramtrans[2] com a intenção de melhorar a informação obtida a partir de *corpora* disponíveis, tendo também a preocupação de apresentar de forma coerente essa informação, o que permite uma visão global do uso de certos padrões em que ocorre uma determinada palavra.

[1] http://grammarsoft.com/ (última visita a 23/11/2014).

[2] http://gramtrans.com/gramtrans (última visita a 21/11/2014).

2.2.1.1 Organização dos Dados

Como ponto de partida duma ferramenta deste tipo, é necessário dispor de *corpora* devidamente anotados. Neste caso, para cada língua disponível no sistema, optou-se por anotar *corpora* através de um analisador que adota o quadro teórico da Gramática Constritiva (*Constraint Grammar*) [18], devido à sua robustez, boa cobertura lexical e também devido ao facto de estabelecer as dependências sintáticas entre segmentos e não entre constituintes, o que torna o resultado obtido computacionalmente mais fácil de processar [6]. No caso da língua portuguesa, foi usado o analisador PALAVRAS [4], que se apoia num conjunto de lemas lexicais e regras gramaticais para fornecer uma análise completa, tanto morfológica como sintática, de qualquer texto. Este analisador obtém, alegadamente, uma precisão de 99% em termos de morfologia (classe de palavras e flexão), e 96-97% em termos de sintaxe [4].

Após a anotação morfossintática dos segmentos e a sua desambiguação, entra em jogo um analisador de dependências cuja saída consiste em acrescentar à informação presente nesses segmentos nova informação de ordem dependencial. É esta informação que é depois utilizada para gerar os textos anotados em forma de árvore de dependências.

Para cada segmento, é usada a informação obtida através das dependências para adicionar novas etiquetas que contêm a relação em que a palavra participa, utilizando números atribuídos a cada palavra para representar essa dependência. Na frase `"O João comeu bananas."`, considera-se que o sujeito `"João"` (palavra número 2) e o complemento `"banana"` (palavra número 5) têm uma ligação de dependência (#x→ y) com o verbo `"comeu"` (palavra número 3). Assim para a palavra `"João"` é adicionada a etiqueta (#2→3) à informação já existente.

A partir do resultado obtido são recolhidos pares de dependências, que são referidos como *"dep-grams"*. Para evitar uma explosão de *dep-grams* sem sentido, são usados os lemas de cada palavra, bem como as anotações relativas à parte do discurso (*POS*, do inglês *part-of-speech*) e às funções sintáticas. No caso dos números, decidiu-se adicionar a categoria ao invés do uso do lema e no caso dos substantivos próprios são guardados protótipos semânticos (por exemplo <humano>) como camada de abstração adicional [6]. Assim, no exemplo, alguns *dep-grams* poderiam ser:

PROP_SUBJ → comer_V

banana_ACC → comer_V

onde PROP é a etiqueta para representar nomes próprios, SUBJ o sujeito e ACC o complemento direto (acusativo) [5]. No caso de `banana` e `comer` são usados os lemas respetivos. O resultado é então gerado e apresentado numa forma de lista: {PROP, ...} → comer ←{banana_ACC, ...}.

Depois de se produzir os *dep-grams* é necessário quantificá-los, de modo a distinguir a sua importância. Assim, usou-se uma medida estatística para calcular a força de co-ocorrência, através da equação (2.8) [6],

$$C \times \log\left(\frac{p(a \rightarrow b)^2}{p(a) \times p(b)}\right) \tag{2.8}$$

em que $p(a \rightarrow b)$ representa a frequência do *dep-gram* no *corpus*, $p(a)$ e $p(b)$ a frequência das palavras a e b e C uma constante introduzida para tornar o valor obtido em apenas um dígito. Esta medida é baseada no método descrito na Secção 2.1.1, *Pointwise Mutual Information*, tendo como diferença a valorização quadrática do *deep-gram* em causa, a fim de evitar que estes se misturem com as co-ocorrências comuns [6].

Na base de dados decidiu-se armazenar juntamente com o *dep-gram*, a frequência absoluta deste, a sua força de co-ocorrência, e também o ID de frases presentes no *corpus* em que este participa.

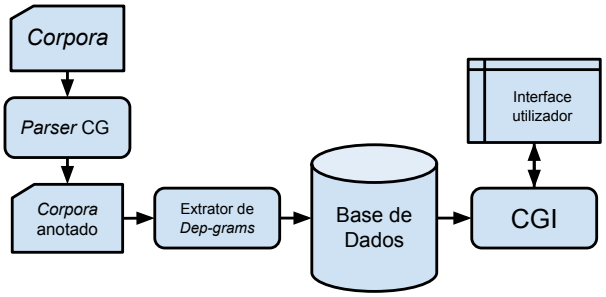


Figura 2.1: Produção de dados no *DeepDict* (adaptado de [6]).

A Figura 2.1 mostra o resumo do processo efetuado desde da anotação de *corpora* até ao armazenamento dos *dep-gram* na base de dados. Uma vez que os dados armazenados ocupam cerca de 90 Gb, foi necessário implementar algoritmos de consulta para permitir respostas num tempo aceitável. A CGI (do inglês *Common Gateway Interface*) é o módulo que permite gerar a interface.

2.2.1.2 Interface de Utilizador

Esta interface permite pesquisar a palavra pretendida de forma clara e rápida.

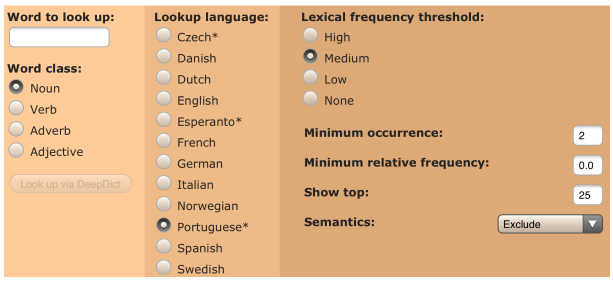


Figura 2.2: Formulário de procura[3].

Como mostra a Figura 2.2, é disponibilizado um formulário que tem vários elementos a preencher. Exceto o campo de procura, todos os outros elementos já estão preenchidos com valores por omissão, permitindo que o utilizador só os altere caso pretenda. Os elementos mais importantes presentes no formulário são:

- **Campo de procura (*Word to look up*):** caixa de texto para introduzir a palavra que se pretende procurar;
- **Classe da palavra (*Word class*):** O utilizador tem de escolher qual a classe da palavra que pretende procurar. Esta escolha resolve o problema de ambiguidade morfológica, isto é, no caso de a mesma palavra poder ser de classes diferentes (exemplo: casa_N e casa_V);
- **Língua pretendida (*Lookup language*):** qual a língua da palavra que se pretende pesquisar;
- **Limiar da frequência lexical (*Lexical frequency threshold*):** usado para filtrar palavras raras da lista de relações;

[3]Retirado de http://gramtrans.com/deepdict/ (última visita a 8/1/2015).

- **Valor mínimo de ocorrência (*Minimum occurrence*):** campo que filtra relações que raramente ocorrem. Quanto maior o valor de ocorrência introduzido, menor o número de padrões considerados;

- **Frequência relativa mínima (*Minimum relative frequency*):** estabelece um limite para o valor calculado pela medida de associação usada entre a palavra e a relação dada.

Como resultado de uma pesquisa, é apresentado um "lexicograma" (*lexicogram*) da palavra procurada (`carro`), presente na Figura 2.3, que mostra as diferentes relações de dependência que esta estabelece com outros elementos lexicais, indicando as que têm um grau de associação (ou força de co-ocorrência) por ordem decrescente. Outro aspeto importante é o facto de que, para cada classe de palavras, o lexicograma tem um modelo diferente. Os seus campos são apresentados de modo a suportar a leitura "natural" (leitura da esquerda para a direita) [6].

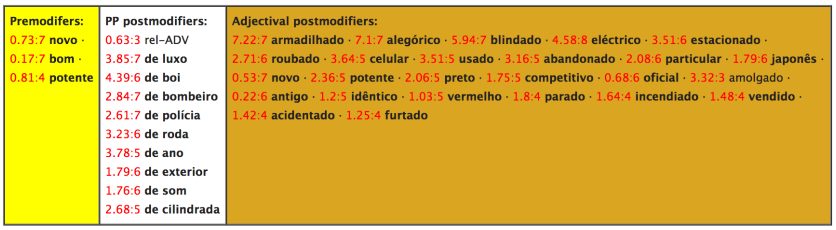


Figura 2.3: Exemplo de um lexicograma para o nome `carro`[4].

Pode-se observar que as expressões mais comuns são `novo carro` no caso de um pré-modificador, `carro armadilhado` no caso um adjetivo modificador pós-nominal e `carro de luxo` para um modificador preposicional.

Cada elemento tem o formato "$x.y : z$ palavra co-ocorrente", por exemplo 7.22 : 7 `armadilhado`, onde "$x.y : z$"$(7.22 : 7)$ aparece a vermelho, "$x.y$"(7.22) representa o grau de associação (ou valor de força de co-ocorrência) entre a palavra procurada e a "palavra co-ocorrente" e onde "z"(7) corresponde à classe da frequência absoluta, que é calculado pelo logaritmo de base 2 da frequência. Quando $z > 4$ a "palavra co-ocorrente" aparece a negrito de modo a salientar a sua importância. Os elementos possibilitam o acesso a um *link* onde aparecem exemplos de concordância extraídos do *corpus* [6].

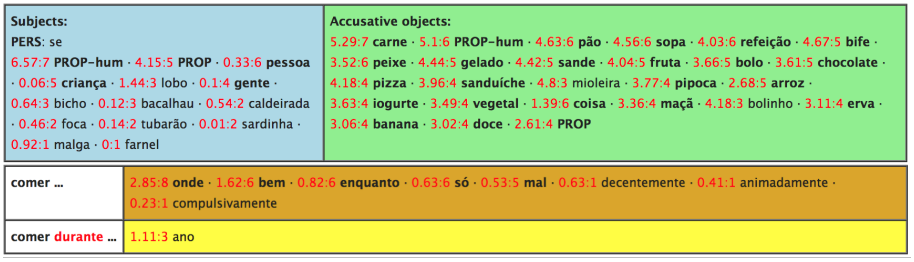


Figura 2.4: Exemplo de um lexicograma para o verbo `comer`[5].

A Figura 2.4 mostra o exemplo do lexicograma resultante da pesquisa do verbo `comer`, onde são apresentados os sujeitos bem como os complementos diretos (acusativo) e também os modificadores adverbiais que ocorrem mais frequentemente após este verbo.

[4]Retirado de http://gramtrans.com/deepdict/ (última visita a 8/1/2015).
[5]Retirado de http://gramtrans.com/deepdict/ (última visita a 8/1/2015).

2.2.2 Sketch Engine

O *Sketch Engine* é um sistema desenvolvido pela empresa Lexical Computing[6]. O ano de 2014 marcou os 10 anos do seu lançamento, sendo por isso um sistema estável e maduro. Entre as principais funcionalidades destacam-se: (i) *Word Sketches*, resumo que mostra o comportamento gramatical de uma palavra e como esta se relaciona com outras; (ii) a comparação de *Word Sketches* entre duas palavras; e (iii) *thesaurus* de uma palavra, que consiste em apresentar o grupo de palavras próximas dessa palavra [19]. Os utilizadores podem escolher *corpora* já disponíveis no sistema ou então adicionar o seu próprio *corpus* usando as ferramentas de criação e gestão fornecidas pelo sistema.

2.2.2.1 Organização dos Dados

Uma vez que o *Sketch Engine* é um sistema em que a função principal é permitir aos utilizadores o acesso a *Word Sketches*, *thesaurus* ou concordância, é necessário armazenar *corpora* devidamente preparados para permitir aos utilizadores o acesso a estas funcionalidades. O sistema usa um sistema de gestão de base de dados já existente, adicionando-lhe novas funcionalidades. A ferramenta de gestão de *corpora* usada, chamada *Manatee* [27], foi desenvolvida especialmente para lidar com *corpora* de grandes dimensões. O *Sketch Engine* já possui *corpora* armazenado em várias línguas e preparados pela sua equipa para ficarem disponíveis para os utilizadores que os quiserem explorar, mas também permite que estes submetam, criem, partilhem e/ou explorem o seu próprio *corpus*.

O autor defende que, no caso de ferramentas do tipo do *Manatee*, se devem implementar as seguintes funcionalidades [27]:

- **preparação de texto -** conversão do texto de vários formatos ou codificações;
- **gestão de metadados -** incluir informação sobre a origem dos dados, autores, género;
- **segmentador -** determinação de qual a unidade do elemento a segmentar. Normalmente essa unidade é a palavra, podendo variar com o idioma;
- **anotação de *corpora* -** anotação manual e automática para evitar ambiguidades. São adicionadas etiquetas a nível sintático, morfológico e semântico;
- **armazenamento eficiente -** as estruturas usadas no armazenamento devem permitir o rápido acesso a todos os dados;
- **concordância -** devolver fragmentos de texto correspondentes à consulta do utilizador;
- **cálculo de estatísticas -** a partir de padrões nos dados, calcular as frequências de distribuição e estatísticas de co-ocorrência.

O sistema *Manatee* é baseado numa biblioteca de indexação de texto, que fornece estruturas de armazenamento e procedimentos de recuperação para os dados baseados em *corpora*, usando uma implementação eficiente de listas invertidas (do inglês *inverted indexes)*, *array* de sufixos, compressão de palavras, entre outros [27].

Para se aceder aos dados armazenados, é necessário uma linguagem de consultas, que permite pesquisas usando restrições sobre qualquer atributo posicional, qualquer meta-informação, ou qualquer destas combinações. A

[6] http://www.sketchengine.co.uk/?page=Website/Company (última visita a 11/12/2014).

consulta pode ser refinada com a aplicação de filtros ao resultado corrente [27]. O *Sketch Engine* usa a linguagem implementada pela ferramenta *Manatee* que é uma extensão da linguagem CQL (*Corpus Query Language*) [10], especializada em consultas de *corpora* de grandes dimensões. Uma Consulta CQL é um padrão que pode coincidir com um segmento (*token*), ou com uma série de segmentos presentes no *corpus*, tendo a forma básica `[attribute="value"]`, onde *attribute* corresponde a um atributo de um segmento e o *value* uma expressão regular sobre os atributos [17].

Para armazenar um *corpus* no sistema, o ficheiro de entrada precisa de ter um formato vertical, isto é, cada palavra tem de estar numa nova linha, que contém também atributos com informação sobre a palavra, separados por *tabs*. Esses atributos podem ser etiquetas com a classe da palavra e/ou o lema da palavra. Em linhas diferentes são guardadas as marcas estruturais que identificam o início e/ou fim de frases, documentos ou parágrafos [20].

O *Sketch Engine* aceita *corpus* diretamente neste formato, o que facilita a submissão deste, mas também permite *corpora* em vários formatos, como um simples ficheiro de texto. Nestes casos, este texto deve ser inicialmente analisado para se segmentar o texto em *tokens*, o que normalmente corresponde a segmentá-lo em palavras. Depois, um analisador externo ou interno vai fornecer dois tipos de anotações: atributos posicionais que adicionam informação linguística para cada segmento, como etiquetas com a classe da palavra ou lemas, e anotações estruturais que mostram a estrutura do texto, por exemplo uma frase, parágrafo ou o fim de documento, sendo no final transformado num ficheiro vertical, que vai ser devidamente armazenado.

De modo a detetar as relações gramaticais entre as palavras de cada *corpus*, é necessário identificar as relações gramaticais de cada língua. O mecanismo usado para as detetar, é o mesmo usado pelo sistema para fazer pesquisas sobre um *corpus*. São usadas consultas CQL com expressões regulares sobre as etiquetas morfológicas [20]. No caso da língua portuguesa o sistema possui as consultas já definidas, sendo as relações logo identificadas. Mas é possível que cada utilizador (regular ou especialista) possa submeter o seu ficheiro com as consultas definidas.

A ferramenta também permite o cálculo de várias estatísticas a partir da informação obtida do *corpus*. Essas medidas são calculadas de forma a quantificar cada relação gramatical e são depois usadas na criação de *Word Sketches*. Antes do ano de 2006 o sistema usava a métrica $MI_logFreq = PMI * \log f_{xy}$, onde PMI corresponde ao valor calculado no método descrito na Secção 2.1.1, *Pointwise Mutual Information* e f_{xy} é o número de co-ocorrências das palavras x e y. A partir desse ano, a medida usada passou a ser o *LogDice*, descrito na Secção 2.1.3, devido às vantagens descritas nessa Secção.

2.2.2.2 Interface de Utilizador

O Sketch Engine usa uma versão da ferramenta Bonito [27], uma interface gráfica que permite que sejam feitas consultas aos dados disponíveis no gestor de *corpora Manatee*, permitindo que, sobre o resultado dessas consultas, sejam calculadas diversas estatísticas. Neste caso, é usada a versão web da ferramenta *Bonito2* [27], que fornece as mesmas funcionalidades que a versão original, mas neste caso é possível aceder às interfaces nas páginas *web* geradas por um *script* CGI.

A funcionalidade mais básica do *Sketch Engine* é a concordância, que mostra o conteúdo de *corpora* sem qualquer análise subjacente. É possível obter estes dados a partir de uma pesquisa, num formulário simples, onde

apenas se introduz a palavra ou a expressão pretendida. Caso os utilizadores necessitem de mais controlo sobre a pesquisa, o formulário tem a opção tipo de consulta, que permite que estes acedam a várias opções como a especificação do lema, com a hipótese de ser escolhida a classe da palavra, uma frase específica, um carácter no caso de línguas asiáticas, ou mesmo uma consulta CQL, que é interpretada directamente no *Manatee*. No caso das outras opções, o sistema trata de converter a pesquisa na linguagem CQL e faz a consulta usando a ferramenta *Bonito*, que no fim apresenta os resultados [19].

A Figura 2.5 mostra um exemplo de uma pesquisa de uma concordância para a palavra `carro`. Como se pode ver, são apresentadas frases extraídas directamente do *corpus*. Caso se clique na palavra a vermelho, é mostrado essa frase inserida num contexto mais alargado. De referir que todos os exemplos mostrados neste documento foram obtidos a partir do *corpus* `CETEMFolha - CETEMPúblico` já disponível no sistema, que contém textos de português Europeu e de português do Brasil.

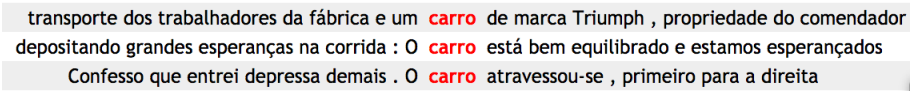


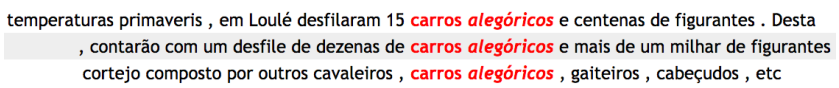
Figura 2.5: Concordância para a palavra `carro`[7].

Os *Word Sketches* são um resumo do comportamento gramatical de uma palavra e de como esta se relaciona com outras, sendo esta a principal funcionalidade do sistema. Para isso são usados padrões gramaticais para identificar as relações gramaticais em que a palavra participa. Para cada relação, são listadas as palavras que co-ocorrem mais vezes. No caso de um substantivo, as relações podem ser `sujeito de`, `complemento de`, `modificador`, complemento preposicional (`pp de`), entre outros.


carro Cetenfolha, Cetempublico Portugal freq = 5,715 (146.3 per million)

modifier	514	2.4	sujeito_de	600	3.8	objeto_de	525	3.8	pp_de	440	3.2	pp_de+o	292	2.6
alegórico	49	9.75	circular	16	6.88	estacionar	13	7.56	luxo	32	8.82	patrulha	3	6.43
eléctrico	63	9.34	parar	20	6.85	conduzir	17	7.09	boi	16	8.24	comitiva	4	6.36
idêntico	15	7.25	desfilar	7	6.38	comprar	20	6.57	roda	20	8.09	gama	3	6.21
velho	12	6.91	deslizar	5	6.15	alugar	7	6.48	aluguer	10	7.86	caravana	3	6.18
funerário	5	6.5	seguir	15	5.72	parar	11	6.05	combate	41	7.8	brigada	3	6.16
celular	6	6.38	andar	11	5.71	multar	4	5.9	bombeiro	26	7.76	polícia	42	6.11

(a)



(b)

Figura 2.6: (a) *Word Sketch* para a palavra `carro` e (b) concordância para a expressão `carros alegóricos`[8].

A Figura 2.6 mostra um exemplo de um *Word Sketch* para a palavra `carro` onde apenas estão listadas algumas relações, pois o sistema fornece mais listas de relações gramaticais do que as apresentadas no exemplo. Para cada entrada numa tabela, o utilizador pode clicar no número para ver qual o contexto onde estas duas palavras co-ocorrem. Neste caso o contexto é dado a partir de uma pesquisa de concordância com as duas palavras.

Para obter um *Word Sketch* o utilizador tem que introduzir o lema pretendido num simples formulário. Este

[7]Retirado de http://www.sketchengine.co.uk/ (última visita a 8/1/2015).
[8]Retirado de http://www.sketchengine.co.uk/ (última visita a 8/1/2015).

pode também escolher opções avançadas que permitem refinar a pesquisa. As principais opções são:

- ***Subcorpus* -** é possível escolher um *subcorpus* do *corpus* principal. No exemplo foi escolhido o subcorpus Português Europeu;
- **Frequência Mínima -** a frequência mínima para cada palavra relacionada;
- **Valor mínimo de co-ocorrência -** Introduzir qual o valor mínimo pretendido;
- **Número máximo de itens numa relação gramatical -** o número máximo de palavras na tabela da relação;
- **Ordenação -** Escolher se a ordenação é feita sobre o valor da medida de associação ou pela frequência com que as duas palavras ocorrem juntas;
- **Selecionar relações gramaticais -** escolher relações gramaticais que o utilizador quer ver no resultado.

Outra funcionalidade que o *Sketch Engine* fornece são os *thesaurus* de uma palavra. Um *thesaurus* é um conjunto de palavras vizinhas para cada palavra. Estes conjuntos são criados com base nas colocações comuns entre as palavra, isto é, se duas palavras aparecem muitas vezes juntas, com um valor de similaridade elevado, cada uma delas vai aparecer no *thesaurus* da outra [19]. O valor de similaridade entre duas palavras é calculado comparando os *Word Sketches* de cada uma, isto é, verificando se ambas partilham das mesmas colocações, onde a medida de associação tem de ser superior a 0.

Lemma	Score	Freq
veículo	0.368	6,679
automóvel	0.288	5,850
modelo	0.252	11,146
casa	0.249	31,528
produto	0.243	19,458
avião	0.241	5,690
homem	0.239	27,235
filme	0.237	24,654
equipa	0.236	19,435
time	0.235	13,754
jogador	0.234	23,944
equipe	0.232	15,191
pessoa	0.229	49,535
livro	0.227	19,194

Figura 2.7: *Thesaurus* para a palavra `carro`[9].

A Figura 2.7 mostra um exemplo de um *thesaurus*, neste caso para a palavra `carro`. À esquerda, é apresentada a tabela com todas as entradas para este *thesaurus*, e para cada entrada é apresentado o valor da frequência e o valor de similaridade entre as duas palavras, ou seja, quão próximas estão de ser sinónimas uma da outra. À direita é apresentada uma "nuvem"das palavras semelhantes, aparecendo com um tamanho maior aquelas que apresentam um grau de similaridade superior.

No caso de se clicar numa das entradas da tabela (por exemplo `veículo`), obtém-se o resultado apresentado na Figura 2.8. Esse resultado apresenta outra das funcionalidades do sistema, que é a diferença entre dois *Word Skecthes*. Para além de auxiliar a compreensão de um *thesaurus*, permite perceber o que torna as duas palavras similares, ou em que é que as palavras se diferenciam. O importante aqui não é a força de co-ocorrência entre as duas mas sim se estas são realmente similares, próximas de se considerarem sinónimos. Os autores defendem que a diferença entre *Word Skecthes* fornecem resumos úteis para pesquisadores interessados em saber como diferem pares de palavras quase sinónimas [20].

[9]Retirado de `http://www.sketchengine.co.uk/` (última visita a 8/1/2015).

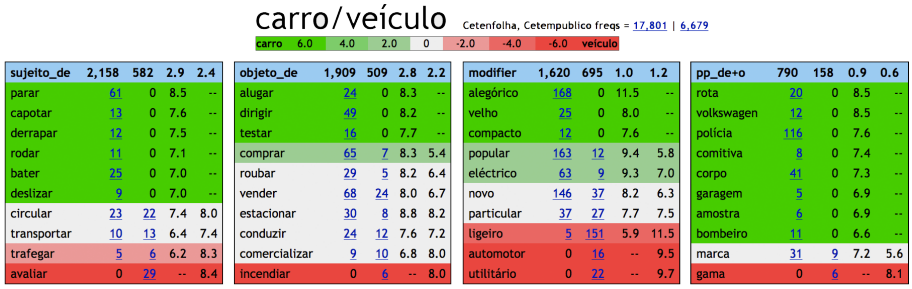


Figura 2.8: *Sketch Diference* entre as palavras `carro` e `veículo`[10].

A Figura 2.8 é um exemplo em que se mostra como são comparadas duas palavras. Em cada tabela, as entradas mais acima estão mais relacionadas com uma das palavras e as mais abaixo com a outra. Aqui podemos ver que a expressão `veículo ligeiro` é mais usada que a expressão `carro ligeiro`. Algumas das entradas adequam-se mais com uma das palavras, como por exemplo `carro da policia`, pois não é tão usual aparecer `veículo da polícia`. Na última coluna, referente aos complementos preposicionais introduzidos por `de`, observa-se que a maioria das colocações são associadas à palavra `carro`.

Uma vez que o sistema permite aos utilizadores criar e gerir o seu próprio *corpus*, é disponibilizado um formulário para auxiliar esse processo, permitindo ao utilizador submeter os seus textos através de uma caixa de texto. Os ficheiros que podem ter qualquer um dos formatos mais comuns (.doc, .html, .pdf, .txt); ficheiros comprimidos (zip, .gz, tar); ou então no formato do *Sketch Engine* (formato “vertical"). Caso o utilizador possua os seus textos já nesse último formato, o sistema não precisa de os processar, sendo logo armazenados. Por outro lado, se os dados já estiverem anotados (com etiquetas de classe da palavra, lemas, etc.), apenas é necessário convertê-los para o formato “vertical"[19]. Os restantes formatos são convertidos para texto simples (.txt), sendo depois compilados de modo a se obter o ficheiro vertical com todos os campos preenchidos, como as etiquetas de classe da palavra e os lemas.

Também é fornecido aos utilizadores um formulário do *WebBootCaT* [3] que é a versão *web* da ferramenta do *BootCaT* [2], para a construção de um *corpus*, a partir de um conjunto de “palavras-semente"definidas pelo utilizador, agrupando em conjuntos, normalmente de três palavras-semente, onde cada conjunto é enviado como uma consulta a um motor de busca, sendo em seguida extraído o conteúdo das páginas *web* encontradas [19].

2.2.3 Wortschatz

O projecto *Wortschatz* é um sistema desenvolvido na Universidade de Leipzig que cria redes de similaridades lexicais a partir de *corpora* de forma automática, com o objetivo de fornecer recursos linguísticos de extração de informação. O que leva a que este sistema seja um dos aqui estudados são as colocações geradas para cada palavra. Para os autores, a *colocação* é a ocorrência de duas ou mais palavras dentro de uma unidade de informação bem definida (frase, documento) [7]. As colocações que o sistema gera são semelhantes às que o *DeepDict* produz com os lexicogramas e o *Sketch Engine* com os *Word Skecth*.

[10]Retirado de `http://www.sketchengine.co.uk/` (última visita a 8/1/2015).

2.2.3.1 Organização dos Dados

O *corpus* é maioritariamente constituído por páginas *web*, jornais e dicionários eletrónicos. Para se obter *corpora* com informação consistente, os textos recolhidos são pré-processados para eliminar aquilo que não interessa. No caso de texto extraído de páginas *web*, é necessário eliminar as etiquetas do HTML e eliminar também todo o tipo de comentários, ficheiros CSS ou *scripts*. Depois o texto é segmentado em frases e removem-se:

- as frases que estiverem numa língua diferente daquela para qual se está a produzir o *corpus*;
- as frases que contenham caracteres especiais como `+, |, [, & e /` em abundância;
- as frases com múltiplos sinais de pontuação seguidos como `???` ou `!!!`;
- as frases com com mais de 9 vírgulas ou 5 pontos;
- as frases com longas sequências de números;
- as frases com muitos espaços em branco em relação ao seu tamanho;
- os duplicados ou quase duplicados para evitar valores errados no cálculo de estatísticas.

Por fim, o *corpus* é reduzido até um tamanho fixo e as frases são misturadas para não ser possível obter os textos originais [25]. Depois de se obterem os textos "limpos", estes são analisados de modo a identificar e atribuir etiquetas de classe de cada palavra.

Como foi dito anteriormente, a ocorrência de duas ou mais palavras numa unidade de informação definida, tal como uma frase ou um documento, é chamado de *colocação*. Para obter colocações importantes e com significado é necessário usar medidas para quantificar o grau de associação lexical dessas colocações. Para isso, o sistema usa a medida descrita na Secção 2.1.6, *Significance Measure*, para calcular o valor da significância.

O sistema identifica dois tipos de colocações: colocações baseadas na co-ocorrência de cada palavra na mesma frase e colocações baseadas nas palavras vizinhas, quer à esquerda quer à direita de cada palavra. Para permitir a análise de colocações para *corpora* completos em tempo aceitável, foram desenvolvidos algoritmos eficientes baseados em árvores de sufixos [7].

O *corpus* é armazenado numa base de dados MySQL. As bases de dados fornecem métodos eficientes de indexação, permitindo armazenar grandes quantidades de informação, bem como o acesso remoto a estes. Assim, é desta forma guardada a informação sobre as palavras em várias tabelas que têm como chave única o número da palavra no *corpus*. Os algoritmos trabalham com esse número em vez de usar a cadeia de caracteres correspondente à palavra por motivos de eficiência. As 5 tabelas que são guardadas na base de dados são:

- **Lista de Palavras -** esta tabela guarda a palavra, o número da palavra, e a sua frequência. Serve para mapear cada palavra com o seu número e fornece a frequência absoluta de cada uma;
- **Frases -** guarda a frase e o número da mesma;
- **Índex -** guarda o número da palavra e o número da frase onde cada palavra se encontra. Permite saber em quantas e em que frases uma palavra ocorre;
- **Co-ocorrência (frases) e Co-ocorrência (vizinhos) -** estas duas tabelas guardam o número de cada uma das duas palavras, a contagem da sua co-ocorrência e o valor da sua significância.

Outras tabelas são usadas para incluir informação gramatical, semântica, etiquetas com a classe da palavra [25]. Adicionalmente, também é guardada a classe de frequência de uma palavra que é calculada com base na função logarítmica relativamente à palavra com a maior frequência absoluta num *corpus*. No caso de uma palavra com classe x, significa que a palavra com maior ocorrência num *corpus* tem cerca de 2^x ocorrências a mais do que o número de ocorrências da palavra selecionada [7].

2.2.3.2 Interface de Utilizador

O projeto *Wortschatz* permite aos utilizadores a consulta das colocações de uma palavra através de uma interface disponível numa página *web*[11]. A informação disponibilizada na *web*, também pode ser integrada noutras aplicações através de *Web Services* baseados no protocolo SOAP (*Simple Object Access Protocol*) [25].

A interface permite aos utilizadores a escolha do *corpus* em que se irá fazer a pesquisa. Primeiro, é necessário escolher a língua de trabalho. Para isso, a interface fornece duas possibilidades: por um lado, a escolha pode ser feita através de uma lista com as línguas disponíveis; por outro lado, o utilizador pode usufruir de um mapa fornecido pela interface para efetuar a escolha de uma forma mais dinâmica. Depois de escolhida a língua, o sistema lista os *corpora* disponíveis, sendo possível filtrar esta consoante a sua fonte. Depois de escolhido o *corpus* basta introduzir a palavra na caixa de texto para se efetuar a consulta.

O resultado apresentado contém a seguinte informação:

- **número de ocorrências -** O número de vezes que a palavra procurada ocorreu no *corpus* escolhido;
- **classe da frequência -** A classe de frequência da palavra procurada;
- **exemplos -** São mostrados exemplos da utilização da palavra procurada. Caso o utilizador pretenda, pode pedir para ver um número maior de exemplos;
- **co-ocorrências significantes -** São listadas as colocações com maior significância em relação à palavra procurada. Esta informação é apresentada de três formas: pela listagem simples por ordem decrescente de significância, listagem dos vizinhos à esquerda e listagem dos vizinhos à direita da palavra-alvo, também nessa mesma ordem;
- **grafo de co-ocorrência -** é mostrado um grafo que descreve visualmente as associações da palavra procurada.

A Figura 2.9 mostra o exemplo das listas (abreviadas) de colocações para a palavra `carro`. As colocações são mostradas de uma forma simples, ordenadas por ordem de significância, em que as mais significantes estão no início. O número entre parênteses indica o valor da significância de cada colocação. As palavras existentes em cada lista permitem, caso o utilizador clique sobre elas, obter o mesmo resultado, mas desta vez para a palavra clicada. No *corpus* `Portuguese (Portugal) - Newscrawl, 2011`, onde foi feita a pesquisa, a colocação com um maior valor de significância é a palavra `eléctrico`. Em relação às palavras vizinhas, à esquerda encontra-se aquela com maior significância, o determinante artigo indefinido `um`. Ignorando as palavras gramaticais podemos verificar que co-ocorrem de forma significativa à esquerda `comprar` e `apreender` e adjectivos como `novo` ou `primeiro`; à direita, encontramos diversos adjectivos específicos de um nome que forma com este o composto, `carro-patrulha`.

[11] http://wortschatz.uni-leipzig.de/ws_norm/index_wm.php (última visita a 27/12/2014)

term: carro
number of occurrences: 4971
class of frequency: 9 (i.e. *de* has got about 2^9 the number of occurrences than the selected word.)

example(s):
Os presumíveis autores dos crimes acabariam por atirar a vítima dentro do **carro** para o rio Vouga, em Cacia. (source: *http://www.noticiasdeaveiro.pt/pt/16264/pj-deteve-suspeitos-de-tentativa-de-sequestro-e-homicidio/index.html*)

more examples

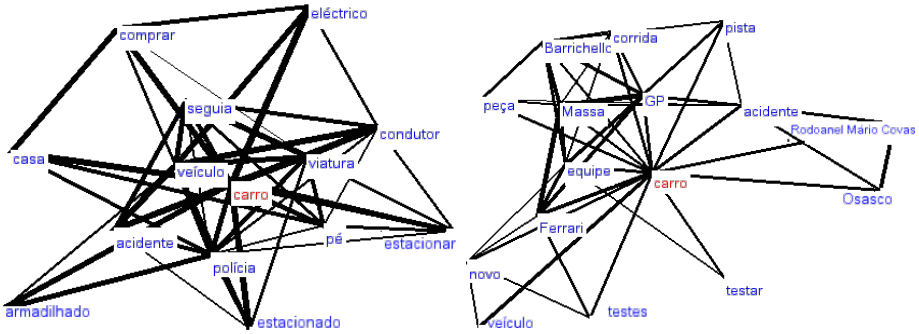
significant cooccurrences of carro:
eléctrico (1963.14), um (1428.53), armadilhado (919.85), o (585.34), casa (531.65), viatura (517.03), estacionado (495.42), comprar (473.2), num (467.85), veículo (416.39), condutor (377.29), estacionar (319.54), quando (293.67), pé (284.72), polícia (281.3), seguia (266.18), acidente (256.45), matrícula (255.04)

significant left neighbours of carro:
um (4904.91), o (3553.61), do (2030.91), num (1063.01), no (806.97), seu (767.43), meu (418.48), O (413.68), Um (259.06), comprar (252.16), novo (210.01), primeiro (172.26), de (166.14), outro (115.05), ao (71.55), cada (53.9), próprio (49.19), apreende (45.67), pelo (42.04), Comprar (40.72), nenhum (32.51), sem (29.69)

significant right neighbours of carro:
eléctrico (3249.9), armadilhado (1480.46), e (414.95), , (335.98), estacionado (287.64), alegórico (283.39), novo (222.15), patrulha (199.64), roubado (181.99), . (179.1), funerário (163.91), elétrico (154.56), híbrido (143.88), usado (136.79), em (130.51), celular (123.26), velho (116.14), alugado (114.34), ligeiro (104.62)

Figura 2.9: Colocações para a palavra `carro` no *corpus* `Portuguese (Portugal) - Newscrawl, 2011`.

Note-se que o sistema trata de forma distinta os *tokens* `comprar` (minúscula) e `Comprar` (Maiúscula inicial), ou seja, o sistema não usa os lemas para agrupar as palavras.



(a) Corpus: *Portuguese - Newscrawl, 2011*

(b) Corpus: *Portuguese - News, 2009*

Figura 2.10: Grafos de co-ocorrência de *corpora* diferente.

A Figura 2.10 mostra dois exemplos de grafos de co-ocorrência. Ambos são para a palavra `carro`, mas para *corpora* diferentes. Um é do *corpus Portuguese - Newscrawl, 2011* e o outro do *corpus Portuguese - News, 2009*. Pode-se concluir que, no caso do *corpus* do grafo (a), a palavra `carro` parece estar a ser utilizada em textos de caráter genérico, enquanto que no *corpus* do grafo (b) a palavra se encontra em textos de domínio desportivo, nomeadamente a Fórmula 1.

2.2.4 Comparação

A Tabela 2.2 apresenta uma comparação entre os sistemas estudados, em que se resumem as principais diferenças entre as funcionalidades de cada sistema. Pode-se observar que todos os sistemas são capazes de listar as colocações de uma palavra. Dos sistemas apresentados, o mais completo em termos de funcionalidades disponibilizadas é o *Sketch Engine*, no entanto, o *DeepDict* mostra a informação sobre as co-ocorrências entre palavras de uma forma mais organizada, o que torna possível ao utilizador extrair conclusões de um modo acessível.

Tabela 2.2: Comparação entre os sistemas estudados.

	DeepDict	***Sketch Engine***	***Wortschatz***
Armazenamento dos dados	*Não referido*	*Manatee*	*MySQL*
Medida de Associação usada	Variante de PMI, Equação (2.8)	*LogDice* (secção 2.1.3)	*Significance* (secção 2.1.6)
Lematização	Sim	Sim	Não
Protótipos semânticos	Sim	Não	Não
Escolha de *corpora*	Não	Sim	Sim
Definição da palavra-alvo	Lema e POS	Lema	Palavra
Exemplos/Concordância	Não	Sim	Sim
Lista de colocações	Sim (Lexicogramas)	Sim (*Word Sketches*)	Sim
Exemplos/Concordância das colocações	Sim	Sim	Não
Corpora adicionado pelo utilizador	Não	Sim	Não
Outras funcionalidades	Não tem	*Thesaurus*, *Sketch Diference*	Grafos de co-ocorrência

Capítulo 3

Arquitetura da Solução

Neste Capítulo descreve-se a arquitetura da solução implementada, não só as diferentes abordagens usadas nos vários componentes da arquitetura, mas também os problemas encontrados no desenvolvimento deste sistema.

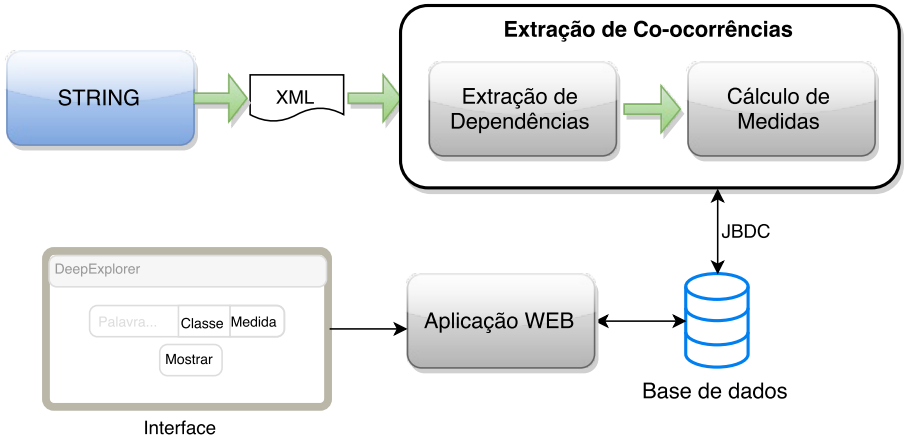


Figura 3.1: Arquitectura Solução.

A Figura 3.1 resume a arquitectura da solução, que consiste em três componentes principais: a base de dados (descrita na Secção 3.1), a extração de co-ocorrências (Secção 3.2) e a aplicação *Web* (Secção 3.3). A base de dados foi implementada de forma a armazenar grandes quantidades de informação sobre padrões de co-ocorrências extraídos a partir de *corpora* processados pela STRING. Os dados são depois fornecidos numa interface *web* para facilitar a análise da informação por parte dos utilizadores.

3.1 Base de Dados

Num sistema como este, é necessário armazenar a informação extraída a partir de *corpora* de grandes dimensões, tornando-se fundamental ter um modelo de dados consistente, bem organizado e com baixa redundância da informação. Isto permite um rápido acesso à informação pretendida.

Para criar a base de dados optou-se inicialmente por desenvolver o modelo entidade-associação (*Entity-Relationship Model*) [9], que modela os dados como um conjunto de entidades e associações entre elas, em seguida verifica quais as restrições de integridade desse modelo e no final converte-o para o modelo relacional [12], visto que este último

é o modelo usado pelas bases de dados convencionais.

3.1.1 Modelo Entidade-Associação

O modelo entidade-associação (E-A) permite uma visão mais natural de um problema e consiste na associação entre entidades. Uma `entidade` é caracterizada com um conjunto de atributos, isto é, informação sobre a entidade, onde cada um desses possui um valor atribuído. Um `conjunto de entidades` agrupa aquelas que possuem os mesmos atributos. As entidades podem ser associadas entre si, desde que pertençam a diferentes `conjunto de entidades`. Cada `associação` também pode possuir atributos[9].

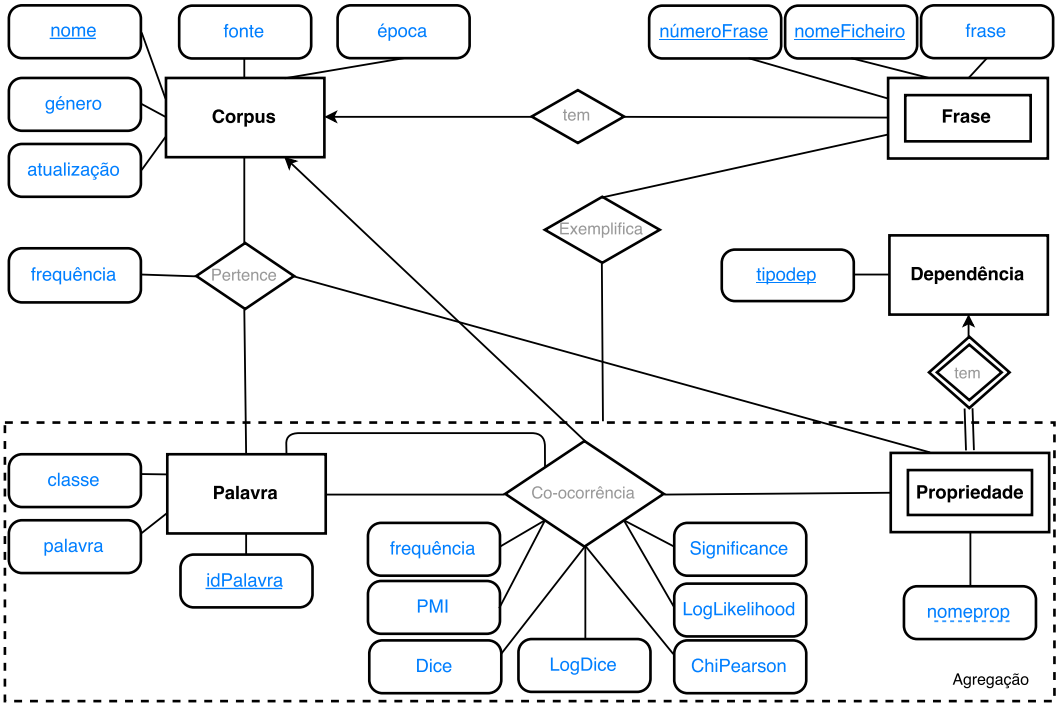


Figura 3.2: Modelo Entidade-Associação.

A base de dados é representada pelo modelo E-A presente na Figura 3.2, onde se pode observar as diferentes entidades, cada uma com atributos distintos.

A entidade `Corpus` tem como atributos o seu `nome` (chave primária da entidade) que a identifica, a `fonte`, `género` e `época` que guardam informação sobre cada *corpus* e por fim, o atributo `atualização` que indica se esse `corpus` se encontra em atualização.

A entidade `Palavra` tem como chave primária o atributo `idPalavra`, e não `palavra` e `classe` pois torna mais fácil a identificação posterior de duas palavras que co-ocorram entre si. O atributo `palavra` armazena apenas o lema dessa palavra.

A entidade `Dependência` tem apenas um atributo, o `tipodep` (tipo de dependência) que é a chave da entidade. A entidade `Propriedade` classifica-se como entidade fraca, pois os seus atributos não são suficientes para formar chave. Assim, esta entidade associa-se com a entidade `Dependência`, sendo essa uma associação com participação total.

A associação `Pertence` diz que qualquer entidade `Palavra` se pode associar a qualquer entidade `Corpus` inde-

pendentemente da sua Propriedade. Esta associação tem um atributo frequência que é o número de ocorrências que uma palavra ocorre num Corpus, com uma determinada Propriedade.

No caso da Co-ocorrência esta associa duas palavras (entidade Palavra), a Propriedade dessa co-ocorrência e o *corpus* onde ocorreu. Os restantes atributos são elementos numéricos que guardam a contagem de cada co-ocorrência e as medidas estatísticas posteriormente calculadas.

A entidade fraca Frase tem como chaves os atributos númeroFrase e nomeFicheiro e ainda a frase, que pertencem a um *corpus* (associação com uma entidade Corpus de um para muitos).

Uma vez que não é permitido duas associações se relacionarem, a associação exemplifica relaciona as frases com uma agregação que abstrai a associação Co-ocorrência. O objetivo é armazenar as frases que exemplificam uma determinada co-ocorrência.

Após se definir o Modelo E-A devem identificar-se as restrições de integridade (RI) do nosso modelo, sendo estas:

RI1: As palavras presentes na associação Co-ocorrência devem pertencer ao Corpus ao qual esta está associada;

RI2: A associação Co-ocorrência deve estar associada à mesma Propriedade com a qual as palavras se relacionam na associação Pertence;

RI3: As frases presentes na associação Exemplifica devem pertencer ao Corpus ao qual a Co-ocorrência está associada.

3.1.2 Modelo Relacional

O modelo relacional é definido por um conjunto de relações, sendo estas representadas em tabelas, onde as colunas são atributos da relação e cada linha é um conjunto n-tuplo, ou seja, uma entrada da tabela [12].

Uma vez que cada conjunto n-tuplo de uma relação é único, são usadas chaves primárias para estas se diferenciarem, garantindo a identificação de cada *n-tuplo*, sendo essas chaves um conjunto de um ou mais atributos da relação. Caso uma relação referencie outra, possui um conjunto de atributos de referência chamados chave estrangeira, onde se garante que os valores desses atributos ocorrem com o valor da chave primária de um conjunto n-tuplo na relação referenciada [30].

O modelo relacional possui uma linguagem de consulta, que define um conjunto de operações e que permite manipular as tabelas de modo a obter os resultados pretendidos numa pesquisa. Essas operações podem ser combinadas de modo a obter expressões que resultem nas consultas desejadas [30].

Uma vez que a base de dados utiliza um modelo de dados relacional, torna-se necessário converter o modelo E-A para o modelo pretendido, seguindo regras de conversão já definidas. O modelo relacional obtido deve respeitar as regras de integridade do modelo anterior. Durante a conversão é necessário transformar as entidades e associações em elementos do novo modelo. Para cada relação "muitos-para-muitos", como, por exemplo, a entidade Corpus, Palavra e Propriedade, é criada uma tabela para cada entidade e adicionada uma outra nova para a associação, neste caso para Co-ocorrência. Esta nova tabela tem como chave estrangeira, as chaves primárias de cada uma das tabela com que se relaciona. O mesmo acontece nas associações Pertence e Exemplifica.

No caso das entidades com a relação "muitos-para-um", como por exemplo na relação entre Frase e Corpus ou entre Propriedade e Dependência, cada entidade resulta numa tabela, sendo adicionado como chave estrangeira

à tabela da entidade "muitos" os atributos da chave primária da entidade "para-um".

O modelo relacional da base de dados do sistema desenvolvido é:

Corpus(nome, fonte, época, género, atualização)

not null(atualização)

Palavra(idPalavra, palavra, classe)

not null(palavra)

not null(classe)

Pertence(nomeCorpus, idPalavra, nomeprop, tipodep, frequência)

nomeCorpus: FK (Corpus)

idPalavra: FK (Palavra)

nomeprop, tipodep: FK (Propriedade)

not null(frequencia)

Dependência(tipodep)

Propriedade(nomeprop, tipodep)

tipodep: FK(Depedência)

Co-ocorrência(idPalavra1, idPalavra2, nomeprop, tipodep, nomeCorpus, frequência, PMI, Dice, LogDice, ChiPearson, LogLikelihood, Significance)

idPalavra1: FK(Palavra)

idPalavra2: FK(Palavra)

nomeprop, tipodep: FK(Propriedade)

nomeCorpus: FK(Corpus)

not null(frequência)

Frase(númeroFrase, nomeFicheiro, frase, nomeCorpus)

nomeCorpus: FK(Corpus)

not null(frase)

Exemplifica(idPalavra1, idPalavra2, nomeprop tipodep, númeroFrase, nomeFicheiro, nomeCorpus)

idPalavra1, idPalavra2, nomeprop, tipodep: FK(Co-ocorrência)

númeroFrase, nomeFicheiro, nomeCorpus: FK (Frase)

O modelo obtido para esta base de dados garante que não há dados repetidos, sendo possível armazenar toda a informação pretendida.

3.1.3 Implementação

A base de dados do nosso sistema pode ser implementada de duas formas. Num dos casos, pode ser usado um servidor remoto que gere o acesso à base de dados e que permite que esta seja acedida a partir de qualquer ponto da rede. Um exemplo é o MySQL[1]. Por outro lado, o acesso à base de dados pode funcionar localmente, isto é, no equipamento onde se pretende correr o sistema. Um exemplo é o SQLite[2].

[1] http://www.mysql.com/ (última visita a 28/8/2015).

[2] http://www.sqlite.org/about (última visita a 28/8/2015).

No nosso caso, preferiu-se o SQLite porque permite o acesso directo e local à base de dados, sem ser necessário configurar o servidor, permitindo o acesso aos dados de forma mais rápida sem nenhum intermediário. Isto é possível porque os dados presentes na base de dados serão apenas utilizados por este sistema.

Uma vez que o SQLite é uma base de dados relacional onde a linguagem implementada é SQL, linguagem padrão das base de dados relacionais, é apenas necessário implementar as tabelas criadas no modelo relacional, ficando assim a base de dados criada e pronta a ser utilizada.

3.2 Extração de Co-ocorrências

A extração dos padrões de co-ocorrência pretendidos é fundamental para o êxito deste projeto. Após se definir o modelo da base de dados, é necessário criar uma ferramenta para preenchê-la com a informação pretendida a partir de um *corpus* processado pela STRING. Após se completar o processo de carregamento da base de dados com as co-ocorrências, é necessário calcular as medidas de associação anteriormente descritas (Secção 2.1). Esta ferramenta foi desenvolvida na linguagem Java.

3.2.1 Processo de Extração das Dependências

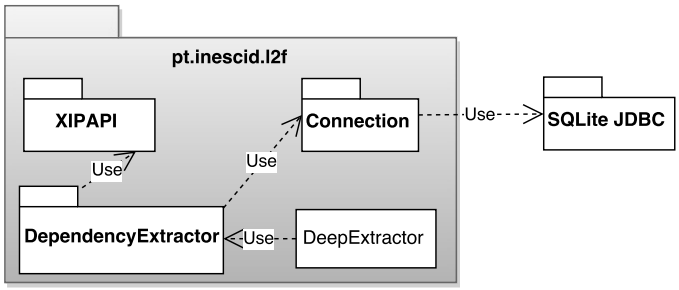


Figura 3.3: Diagrama de *packages* da extração de dependências.

A Figura 3.3 apresenta o diagrama de *packages* em UML da parte da ferramenta responsável pela extração de co-ocorrências, o `DeepExtractor`. Este diagrama identifica os principais componentes usados nesta extração.

O primeiro passo do processo é o acesso aos textos processados provenientes do XIP, uma vez que aceder diretamente ao ficheiro de XML se torna ineficiente. Para tal, foi usada uma API já existente, a XIPAPI [8, 24], que facilita o acesso a essa informação de forma mais organizada e eficiente.

A XIPAPI é uma API desenvolvida em projectos anteriores, que permite transformar o conteúdo dos ficheiros XML provenientes do Módulo XIP da STRING em estruturas Java. Esta API transforma os dados XML usando a Java DOM API (*Standard* do Java)[3] para construir uma árvore DOM, que posteriormente é organizada pela XIPAPI em estruturas próprias, originando o `XIPDocument`, formado por um conjunto de `XIPNodes` e de `XIPDependency`. Um `XIPNode` representa um nó do XIP, que é o elemento básico da estrutura da árvore de *chunks*. Pode representar o elemento raiz de uma frase, o nó `TOP`, mas também os elementos que representam folhas dessa árvore, por exemplo um nó `NP` ou `NOUN`. Este tem um conjunto de `Features` que contém propriedades do nó. Uma `XIPDependency`

[3] https://docs.oracle.com/javase/8/docs/api/org/w3c/dom/package-summary.html (última visita a 07/09/2015).

contém a informação da dependência detetada no XIP, isto é, o tipo de dependência e quais os `XIPNode` a que esta se aplica. Estas estruturas são disponibilizadas a partir de métodos existentes na API, que facilitam o acesso aos dados que inicialmente se encontravam no XML [24].

Para armazenar os dados na base de dados é necessário criar uma conexão a esta. Para isso, é usado uma API chamada JDBC (*Java Database Connectivity*), que permite ligar um programa escrito na linguagem de programação Java a um conjunto de diferentes sistemas de gestão de base de dados, possibilitando que o programa execute instruções SQL na base de dados pretendida. Neste caso, foi usado JDBC para SQLite, a SQLiteJDBC[4].

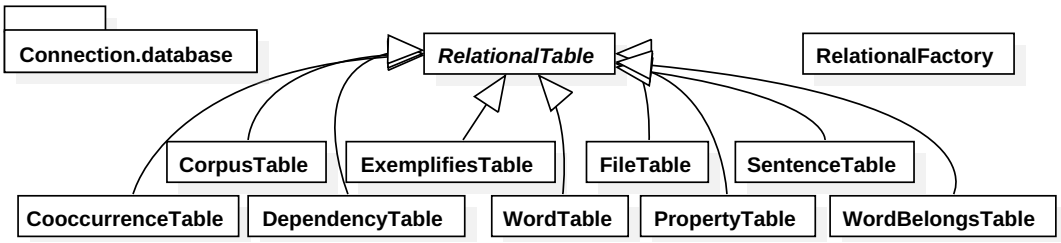


Figura 3.4: Diagrama de classes simplificado da *package* `Connection.database`.

A *package* `Connection` foi desenvolvida com o objetivo de facilitar o acesso à base de dados. Por um lado, é aqui que se cria a conexão com a base de dados usando a API acima descrita, sendo também desenvolvido um conjunto de classes onde estão os métodos utilizados para manipular a informação da base de dados. A Figura 3.4 mostra uma versão simplificada do diagrama dessas classes. Como todas as subclasses representam as diferentes tabelas da base de dados relacional, têm como superclasse `RelationalTable`. Cada subclasse tem o nome da tabela pela qual é responsável, possuindo métodos para inserir novas entradas nessa tabela e verificar se uma determinada entrada existe. No caso das subclasses `WordBelongsTable` e `CooccurrenceTable`, que são responsáveis pelas tabelas `Pertence` e `Co-ocorrência` respetivamente, estas têm métodos adicionais que não só permitem actualizar os valores das frequências de ocorrência, como também métodos que obtêm valores que depois vão ser usados para calcular as medidas de associação (ver Secção 2.1). Estes métodos foram implementados usando os formalismos que o SQLiteJDBC nos fornece para manipular os dados, usando a linguagem SQL para criar as pesquisas que os adicionam, alteram ou obtêm na nossa base de dados. A classe `Relacional factory` funciona como fábrica de objetos para as classes anteriores, criando e armazenando esses objetos no início de cada execução, o que permite o acesso a esses objetos estaticamente a partir de qualquer ponto do programa.

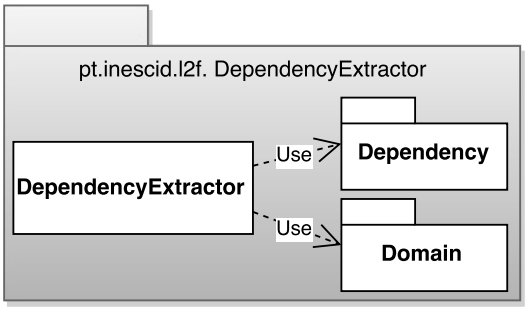


Figura 3.5: Diagrama de *packages* de `DependencyExtractor`.

[4] https://bitbucket.org/xerial/sqlite-jdbc (última visita a 07/09/2015).

A *package* `DependencyExtractor` fornece o núcleo da ferramenta de extração que, a partir da interface fornecida pela XIPAPI, analisa os textos processados pela STRING de modo a armazenar a informação que se pretende. A Figura 3.5 resume a estrutura desta *package*, onde é a classe `DeependencyExtractor` que gere o processo de extração. A partir do método `Extract`, que recebe um `XIPDocument` e recorrendo aos métodos fornecidos pela XIPAPI, é recolhido o conjunto de frases (`XIPNode TOP`) presentes nesse ficheiro, e para cada uma destas é obtido:

- o conjunto de entidades mencionadas aí presentes;
- o conjunto de `XIPDependency`, onde cada um possui dois `XIPnode`;
- a cadeia de caracteres que representa a frase.

Posteriormente, para cada `XIPDependency`, o processo de análise depende do tipo de dependência a que pertence.

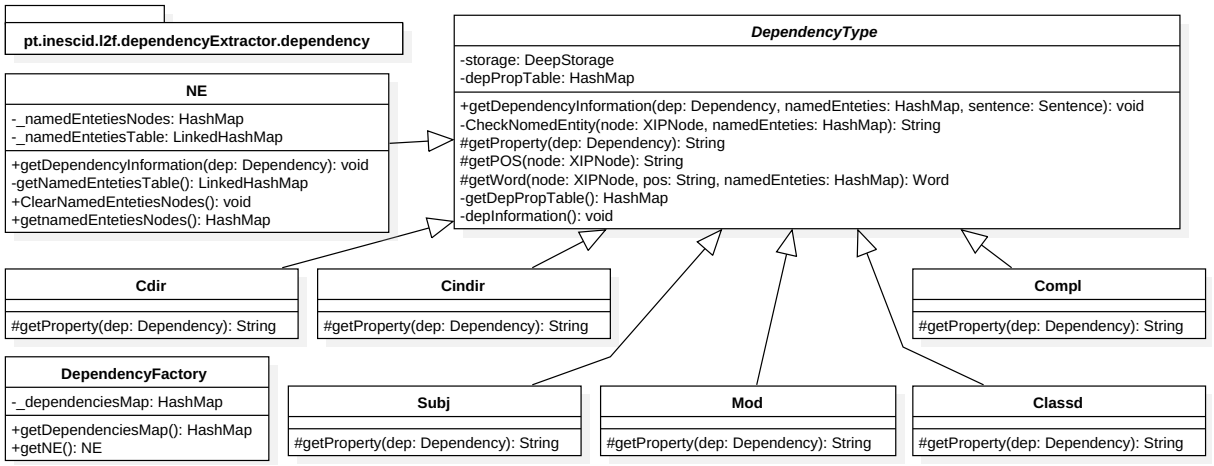


Figura 3.6: Diagrama de classes da *package* `DependencyExtractor.dependency`.

Uma vez que não interessa processar todas as dependências extraídas pelo XIP, mas unicamente aquelas descritas na Secção 1.2.1.1, foi criada uma classe para cada dependência. Assim, garante-se que apenas estas são analisadas, facilitando a diferenciação dessa análise consoante a dependência a processar. A Figura 3.6 mostra o diagrama de classes da *package* `DeependencyExtractor.dependency`. A classe abstrata `DependencyType` representa a superclasse que implementa os métodos comuns a todas as subclasses e onde cada uma destas representa as diferentes dependências e implementa os seus métodos específicos.

Os comportamentos implementados na *package* assumem um papel fundamental na extração da informação necessária pois é aqui que:

- o método `getDepInformation` que recebe uma `XIPDependency` e o conjunto das entidades mencionadas encontradas na frase, gere o processo de análise de modo a obter a propriedade da dependência, e para cada `XIPNode` o lema da palavra e a sua classe;
- para evitar que seja analisada informação de dependências com propriedades que não são relevantes para o sistema, é verificada se a propriedade da `XIPdependency` se encontra num conjunto de padrões dependência-propriedade pré-definidos. Caso esta esteja presente, a análise procede normalmente. A lista das dependências e propriedades presentes no sistema são apresentadas na Secção 3.2.1.1;

- para cada `XIPnode` é verificando se o próprio nó ou outro nó anterior está presente nalguma entidade mencionada. Caso isso aconteça, o lema da palavra passa a ser a categoria desta. As entidades mencionadas que são detectadas por nós são definidas na Secção 3.2.1.2.

Com isto, para cada dependência é armazenado o lema de cada palavra e a sua classe, o tipo de dependência e a sua propriedade, e a frase onde ocorre. Durante o processo é ainda necessário armazenar a informação que vai sendo encontrada. Para cada dependência analisada, uma forma de armazenar a informação é adicioná-la diretamente na base de dados a partir dos métodos disponibilizados pela *package* `Connection`. Uma vez que um *corpus* processado se encontra dividido em diversos ficheiros e possui um tamanho considerável, esta solução garante que toda a informação é armazenada na base de dados corretamente, embora atrase todo o processo devido aos constantes e excessivos acessos à mesma. Para evitar esta repetição, cada ficheiro do *corpus* é processado de modo a armazenar a informação em estruturas do Java. No fim, todos os dados recolhidos serão introduzidos na base de dados. Isto permite que, para os dados que se repitam, a gestão seja feita dentro do programa. Para isto foi desenvolvido um conjunto de classes que ajudam a organizar a informação até esta ser adicionada na base de dados.

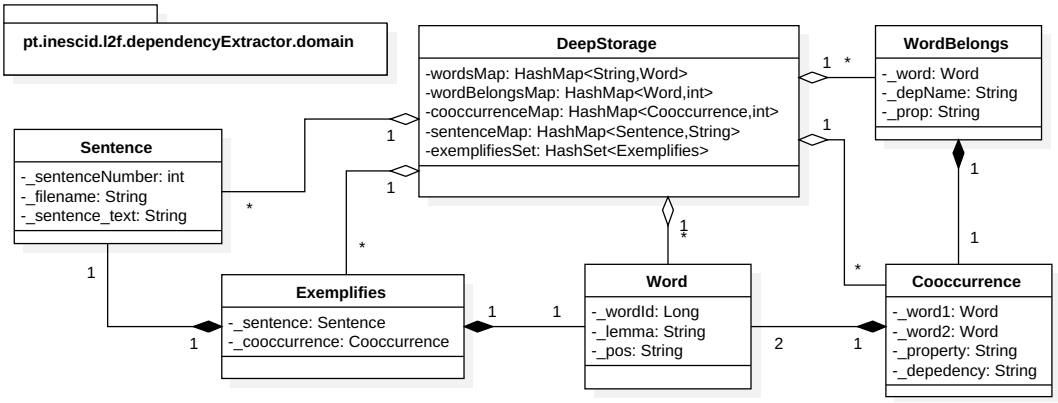


Figura 3.7: Diagrama de classes da*package* `DependencyExtractor.domain`.

A Figura 3.7 apresenta o diagrama de classes que foram desenvolvidas para solucionar o problema dos múltiplos acesso à base de dados. Como se pode observar, estas têm como objetivo organizar a informação extraída de forma a tornar mais fácil a sua manipulação. A classe `DeepStorage` funciona como base de dados local dentro da execução. Após a extração de um ficheiro do *corpus* se encontrar concluída, toda a informação armazenada é então adicionada à base de dados.

3.2.1.1 Dependências e Propriedades

Uma `XIPDependency` possui a informação sobre a dependência, as propriedades, caso estas existam, e dois `XIPNodes`. Cada `XIPNode` representa uma palavra, onde o primeiro é o elemento modificado e o segundo o elemento modificador da relação. Esta dependência pode ser uma das dependências sintáticas extraídas pelo XIP apresentadas na Secção 1.2.1.1. Para as propriedades da relação, os valores que interessam são:

- sufixo `PRE`: esta propriedade indica que o elemento modificador aparece no texto antes do elemento modificado, e é analisada nas dependências `MOD`;

- sufixo `POST`: esta propriedade indica que o elemento modificador aparece no texto após o elemento modificado, e é analisada nas dependências `MOD`;
- sufixo `FOCUS`: esta propriedade é específica para a dependência `MOD` no caso em que o elemento modificador é um advérbio de foco.

Para ajudar na seleção das co-ocorrências, decidiu-se criar um padrão para identificar cada um dos casos. Este padrão é criado juntando o tipo de dependência com a propriedade de uma determinada relação. No caso da propriedade, esta é composta pelos sufixos presentes na relação, juntamente com a classe das duas palavras que participam nesta. Posto isto, o padrão dependência-propriedade para uma co-ocorrência tem o formato `DEP PROP_POS1_POS2`, onde `POS1` e `POS2` correspondem à classe de cada uma das palavras presentes na `XIPDependency`. Assim, cada co-ocorrência possui um padrão dependência-propriedade, que durante a análise de uma `XIPDependency` é verificado se esse padrão se encontra numa lista com aqueles que se pretendem processar. Caso não esteja presente, a análise da `XIPDependency` é descartada.

Quando se pretende agrupar co-ocorrências com propriedades diferentes, tornando-as numa única co-ocorrência, é necessário indicar ao sistema quais as propriedades a guardar em conjunto. Assim, para cada padrão dependência-propriedade deve existir um novo padrão, sendo este o armazenado na base de dados.

Para tal, foi criado um ficheiro que contém uma lista, onde cada linha tem o formato "`DEP PROP_POS1_POS2,DEP NOVAPROP`", onde `PROP_POS1_POS2` corresponde à propriedade detetada pelo sistema, enquanto que `NOVAPROP` corresponde à propriedade que se pretende armazenar na base de dados, podendo ser a mesma propriedade obtida originalmente. Esse ficheiro é importado no início de cada execução da ferramenta de extração, que permite a alteração deste caso se pretenda sem ser necessário modificar o programa.

De modo a facilitar a compreensão, esta lista é apresentada de uma maneira organizada próxima da que se pretende apresentar na interface *web* (Secção 3.3). Como o objetivo é a pesquisa de co-ocorrências para uma palavra-alvo, será necessário especificar os padrões dependência-propriedade para cada uma das co-ocorrências. A palavra pode ser um nome, verbo, adjetivo ou advérbio, podendo ocorrer à esquerda ou à direita da palavra com que co-ocorre. Os conjuntos dependência-propriedade descritos abaixo, estão no formato "`DEP PROP_POS1_POS2`" que o sistema encontra originalmente.

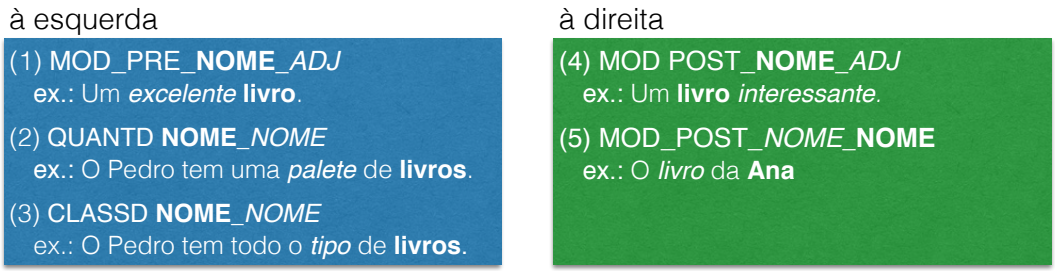


Figura 3.8: Conjuntos dependência-propriedade de uma palavra-alvo `nome`.

A Figura 3.8 apresenta o conjunto de padrões dependência-propriedade armazenados quando a palavra-alvo é um nome. Este pode ser modificado à esquerda por um adjectivo (1) (dependência `MOD`) ou por outro nome (dependências `QUANTD` (2) e `CLASSD` (3)). Por outro lado, um nome pode ser modificado à direita por um adjectivo (4) (dependência `MOD`). Quando o nome é modificador, este apenas modifica à direita outro nome (5) (dependência

MOD), ou seja, este é complemento do outro nome.

Nas situações em que a palavra-alvo é um nome, não se pretende juntar propriedades presentes na Figura 3.8. Assim, para os padrões dependência-propriedade aqui presentes quando adicionados ao ficheiro, a NOVAPROP será a repetição da PROP_POS1_POS2 presente no padrão. Por exemplo, para o padrão dependência-propriedade (2), no ficheiro é representado por "MOD PRE_NOME_ADJ,MOD PRE_NOME_ADJ".

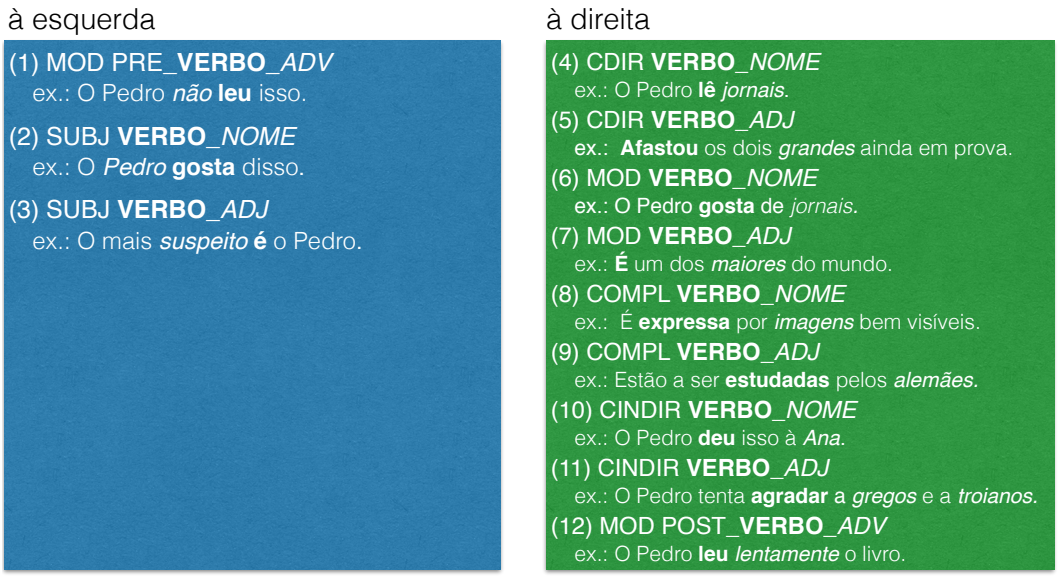


Figura 3.9: Conjuntos dependência-propriedade de uma palavra alvo verbo.

A lista dos padrões dependência-propriedade, no caso em que a palavra-alvo é um verbo, é apresentada na Figura 3.9. Apenas são analisados os casos em que o verbo é o primeiro elemento da relação. Quando este ocorre à esquerda, o elemento modificador pode ser um advérbio (1) (dependência MOD). O segundo elemento da relação pode ainda ser um nome (2) ou adjetivo (3) no caso de serem os "sujeitos" (dependência SUBJ) do verbo.

À direita, o verbo pode ser modificado (dependência MOD) por um advérbio (12). Pode ainda ter um complemento preposicional cuja cabeça é um nome (6) ou um adjetivo (7). O segundo elemento da relação pode também ser um nome (4) ou adjetivo em posição nominal (6) quando é complemento directo (dependência CDIR) do verbo, por um nome (8) ou um particípio (9) quando é um complemento essencial (dependência COMPL) do verbo; e, por fim, um nome (10) ou adjetivo (11) quando é complemento indireto (dependência CINDIR) do verbo.

No caso do sujeitos (2) e (3), complementos direto (4) e (5), complementos indireto (10) e (11), e complementos essenciais (8) e (9), é pretendido que na base de dados estes pares estejam armazenados em conjunto. Para isso, no ficheiro, cada uma das relações descritas anteriormente tem uma NOVAPROP que pretende unir os padrões das relações num único, de modo a juntar as respetivas ocorrências. Por exemplo, no caso do Sujeitos (2) e (3), estes são representados no ficheiro por "SUBJ VERBO_NOME,SUBJ SEM_PROP" e "SUBJ VERBO_ADJ,SUBJ SEM_PROP", ficando assim a representar um único padrão. Neste caso a propriedade é SEM_PROP, que significa que não tem propriedade. Os restantes padrões presentes na Figura 3.9 (1), (6), (7) e (12), não se pretendem agrupar e por isso, a NOVAPROP será a mesma propriedade presente no padrão original.

A Figura 3.10 apresenta os padrões dependência-propriedade quando a palavra-alvo é um adjetivo. No caso em que o adjetivo é modificado, o modificador é um advérbio (dependência MOD), quer à esquerda (1) quer à direita (3).

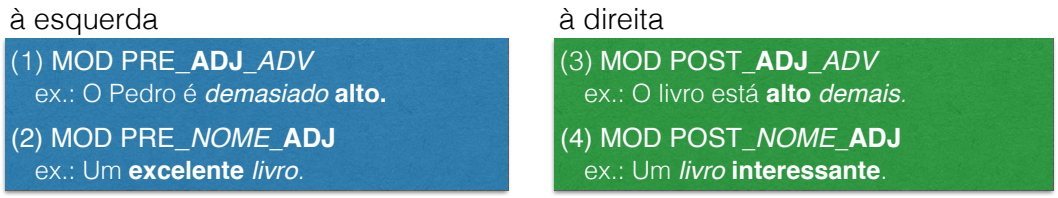


Figura 3.10: Conjuntos dependência-propriedade de uma palavra alvo `adjetivo`.

Se o adjetivo for modificador, quer à esquerda (2) quer à direita (4), este modifica (dependência `MOD`) um nome.

Para esta palavra-alvo, pretende-se guardar todos os padrões dependência-propriedade sem alterações, ou seja, no ficheiro, a `NOVAPROP` é uma das propriedades presentes nos padrões originais, descritos na figura 3.10. Por exemplo, para o padrão dependência-propriedade (1), este é representado por "`MOD PRE_ADJ_ADV,MOD PRE_ADJ_ADV`".

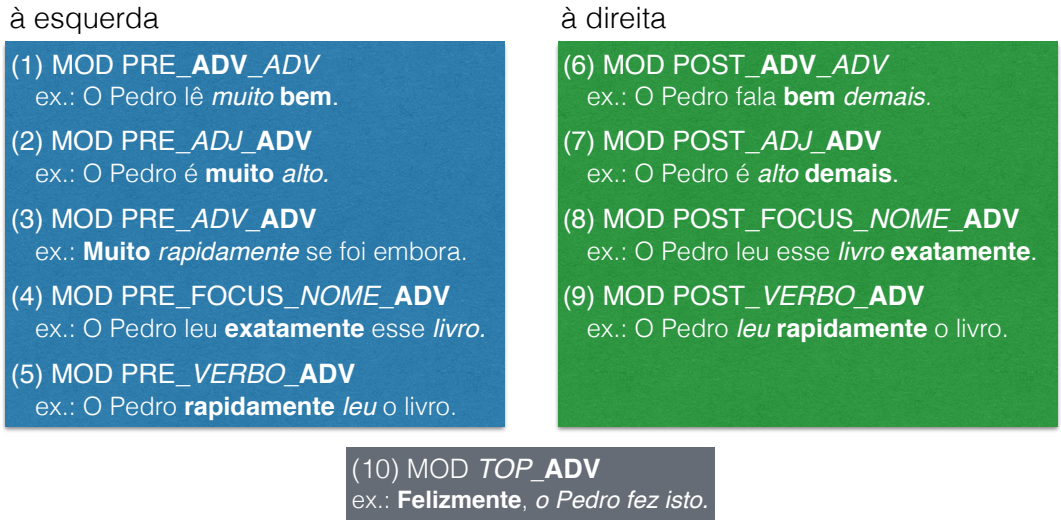


Figura 3.11: Conjuntos dependência-propriedade de uma palavra alvo `advérbio`.

A Figura 3.11 mostra a lista com os padrões dependência-propriedade para a qual a palavra-alvo é um advérbio. Um advérbio pode ser modificado (dependência `MOD`) por outro advérbio, quer à esquerda (1) quer à direita (6). Por outro lado, o advérbio modifica (dependência `MOD`) quer à esquerda, quer à direita um adjectivo (7), um nome (4)(8) e um verbo (5)(9), e pode modificar um advérbio (3) mas apenas à esquerda. No caso de um advérbio é ainda armazenada a informação sobre se este modifica toda uma frase (10). Neste caso também não se pretende juntar nenhum padrão. Por exemplo, o padrão dependência-propriedade (10) é representado no ficheiro por "`MOD TOP_ADV,MOD TOP_ADV`".

3.2.1.2 Entidades Mencionadas

Uma entidade mencionada indica para uma palavra um conjunto de atributos que a categorizam. Na ferramenta, durante o processo de extração das co-ocorrências, para cada palavra é verificado se esta se encontra presente em alguma entidade mencionada encontrada no texto. Caso isto aconteça, é necessário atribuir ao lema da palavra a categoria correspondente a um dos atributos da entidade mencionada. Para tal, existe a Tabela 3.1 com as categorias

Tabela 3.1: Lista de atributos de Entidades Mencionadas relevantes para o sistema e as suas categorias correspondentes.

Atributos da Entidade Mencionada	Categoria
CARGO	CARGO
INDIVIDUAL	PESSOA
CURR QUANT	CURR
EVENT	EVENTO
TEMPO	TDATA
T-DURATION	TDUR
T-DURATION-INTERVAL	TDUR
TEMPO T-FREQUENCY	TFREQ
VIRTUAL	LVIRT
LOCATION	LOC
INSTITUTION	INST
GROUP COLLECTIVE	HCOL
ADMINISTRATION COLLECTIVE	INST

para cada um dos atributos relevantes para o sistema que uma entidade mencionada pode ter.

Por exemplo, na frase "O Pedro tem um carro azul" é detetada a entidade mencionada Pedro com os atributos PEOPLE e INDIVIDUAL. Assim, a entidade mencionada Pedro indica que a palavra tem como categoria "PESSOA"(atributo INDIVIDUAL), sendo essa o novo lema da palavra. Ou seja, quando é atribuída uma entidade mencionada a uma palavra, partir dos atributos da entidade, procura-se para cada uma a categoria correspondente na tabela. Quando um dos atributos da entidade encontra a categoria que lhe está associada, esta é atribuída ao lema da palavra em questão. Caso não se encontre nenhuma categoria, a palavra manterá o seu lema.

Para esta verificação ser efetuada, o conteúdo da Tabela 3.1 é introduzido num ficheiro externo ao programa, carregado no início de cada execução. Uma vez que para cada palavra pode ser atribuída uma entidade mencionada com mais de um atributo, a verificação é feita pela ordem apresentada na tabela. Para esta ser mantida durante o carregamento do ficheiro, a informação presente é guardada num LinkedHashMap[5], uma implementação do HashMap com a diferença que este mantém a ordem de inserção de cada conjunto entidade mencionada-categoria, permitindo que a iteração sobre os elementos seja efectuada pela ordem apresentada.

O objetivo de adicionar as entidades mencionadas no sistema é permitir uma melhor análise dos textos processados, pois assim, palavras que representem uma mesma categoria serão consideradas a mesma.

3.2.2 Medidas de Associação

Após se completar a extração, é necessário calcular o grau de coesão entre as duas palavras a partir das co-ocorrências armazenadas. Para isso foram usadas as medidas de associação descritas na Secção 2.1.

A Figura 3.12 representa o diagrama de *packages* da parte responsável pelo cálculo das medidas de associação, permitindo perceber quais os principais componentes usados. A classe DeepMeasures inicia o processo e a

[5] https://docs.oracle.com/javase/8/docs/api/java/util/LinkedHashMap.html (última visita a 20/09/2015).

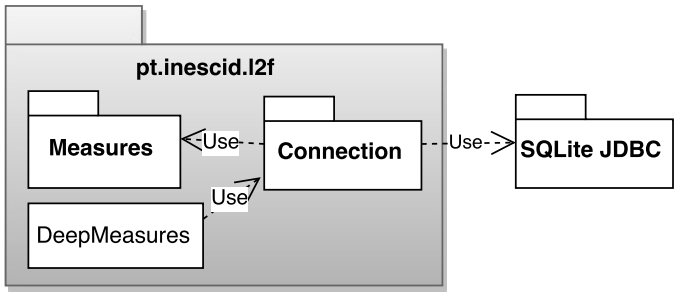


Figura 3.12: Diagrama de *packages* do cálculo das medidas de associação.

partir da *package* `Connection` descrita na Secção 3.2.1, obtêm-se todas as co-ocorrências armazenadas na tabela `Co-ocorrência` da base de dados. Uma vez que a quantidade de co-ocorrências armazenadas é de grandes dimensões, torna-se impossível calcular todas as medidas num único passo. Para isso, os cálculos são efectuados em conjuntos de duas mil co-ocorrências, sendo calculado para cada uma as seis medidas descritas anteriormente. No final, os resultados são atualizados, continuando o processo até se esgotarem as co-ocorrências armazenadas.

Os métodos que calculam cada uma das medidas descritas na Secção 2.1, encontram-se definidos na *package* `Measures`, que, a partir das contagens necessárias, calculam o valor para cada uma delas.

Quando uma medida calcula o seu valor para uma co-ocorrência, os valores das contagens necessárias são referentes apenas ao universo do padrão dependência-propriedade a que a co-ocorrência pertence, ou seja, as medidas de associação calculam o grau de coesão entre as duas palavras, sendo esta referente apenas ao universo do padrão dependência-propriedade correspondente, permitindo uma quantificação mais justa de cada uma destas, não havendo interferência entre os diferentes tipos de co-ocorrências.

3.3 Aplicação *Web*

De modo a permitir o acesso através de uma interface gráfica aos dados extraídos dos textos processados pela STRING, é disponibilizada uma interface *web*. Para isso, foi desenvolvida uma aplicação *web*, que, a partir da base de dados completa, organiza e apresenta nessa interface a informação pretendida pelo utilizador.

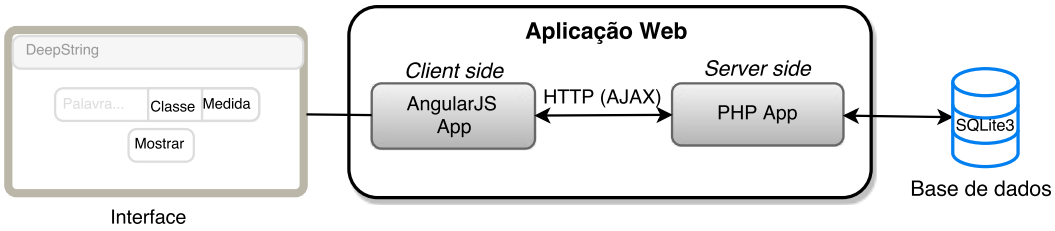


Figura 3.13: Arquitetura da aplicação *web*.

A Figura 3.13 representa a arquitetura da aplicação *web*. Esta é dividida em dois componentes principais, *server-side*, a parte que é executada no lado do servidor e *client-side*, a parte que corre no lado do cliente, isto é, no *browser*. No lado do servidor, o código executado foi desenvolvido em PHP, que é responsável pelo acesso à base de dados e que, a partir de pedidos do cliente, organiza a informação pretendida, enviando-a posteriormente para o cliente. Por outro lado, no lado do cliente, o código executado foi desenvolvido a partir da *framework*

AngularJS[6] para permitir mostrar a informação sobre as co-ocorrências de uma palavra de forma dinâmica. Isto permite dividir a execução da aplicação, pois o código que implementa a interface é executado no cliente (*browser*), e o processamento da informação presente na base de dados corre no servidor. A comunicação entre as duas partes é efectuada através de pedidos assíncronos *AJAX* (*Asynchronous Javascript and XML*) através do protocolo HTTP (*Hypertext Transfer Protocol*), sendo o formato dos dados transmitidos o *JSON* (*JavaScript Object Notation*).

3.3.1 *Server-Side*

A parte da aplicação que é executada no servidor foi implementada em PHP. Este processo inicia-se como um pedido *AJAX* do lado do cliente que contém a informação sobre o que este pretende. Caso se pretenda o conjunto de co-ocorrências de uma palavra, o pedido contém um objecto *JSON* com a palavra-alvo, a sua classe, a medida de associação, a frequência mínima das co-ocorrências e o número máximo de palavras por padrão dependência-propriedade. Caso o pedido do cliente pretenda solicitar as frases que exemplifiquem uma determinada co-ocorrência, o objecto JSON contém a informação que identifica uma co-ocorrência, isto é, o `id` de cada palavra, a dependência e propriedade desta.

Quando o pedido recebido trata o caso da procura de uma palavra-alvo, conforme a sua classe, os padrões dependência-propriedade onde estas surgem variam, sendo necessário a listagem para permitir uma mais rápida execução. Assim, tal como na ferramenta de extração, foi adicionado um ficheiro com essa informação. A principal diferença, é que neste caso, na ferramenta de extração apenas é necessário saber que uma determinada palavra participa numa dependência, sendo necessário armazená-la, enquanto que aqui é necessário saber se num determinado padrão dependência-propriedade, a palavra-alvo é o primeiro ou segundo elemento da relação, pois a pesquisa a executar na base dados difere para cada um dos casos. Normalmente, é simples identificar cada um dos elementos. O problema encontra-se nos casos em que os dois elementos da dependência são da mesma classe, tornando-se difícil saber qual é o elemento da palavra-alvo. Para isso, no ficheiro foi adicionado a cada padrão dependência-propriedade qual o papel da palavra-alvo na dependência, indicando também se esta ocorre à esquerda ou à direita da palavra, o que é representado pelos traços `PRE_WORD` (esquerda) ou `POST_WORD` (direita) e `GOVERNED` (o primeiro elemento da relação) ou `GOVERNOR` (segundo elemento da relação). Foi ainda adicionado o título do conjunto de co-ocorrências de cada padrão dependência-propriedade para permitir identificá-lo posteriormente na interface. Um exemplo do formato usado para representar esta informação no ficheiro é "`MOD PRE_NOUN_ADJ,PRE_WORD,GOVERNED,é modificado pelo adjetivo`".

```
SELECT idPalavra FROM Palavra WHERE palavra='carro' and classe='NOME' LIMIT 1;
```

Figura 3.14: Pesquisa SQL para obter o `id` do nome `carro`.

O primeiro passo para aceder à informação armazenada, é obter o `id` que a palavra pretendida apresenta na base de dados. A pesquisa SQL representada na Figura 3.14 tem como resultado o `id` do nome `carro`. No caso em que a palavra pretendida não exista na base de dados, é enviado para o lado do cliente essa informação e o processo termina. Caso a palavra esteja presente, esta pesquisa é executada de modo a não ser necessário repeti-la durante as pesquisas seguintes.

[6] https://angularjs.org (última visita a 21/09/2015).

Assim, para cada padrão dependência-propriedade, é então necessário fazer uma pesquisa SQL para obter as palavras que co-ocorrem com a palavra-alvo pretendida. Consoante esta for o primeiro ou segundo elemento da dependência, a pesquisa executada difere em cada um dos casos.

```
SELECT Palavra.palavra, Palavra.classe, Coo.LogDice, Coo.frequencia
FROM (SELECT idPalavra2, LogDice, frequencia
       FROM Coocorrencia
       WHERE idPalavra1 = 1615
         and frequencia >= 2
         and tipoDep = "MOD"
         and nomeProp = "PRE_NOUN_ADJ"
       ORDER BY LogDice DESC
       LIMIT 10) as Coo
INNER JOIN Palavra on Coo.idPalavra2 = Palavra.idPalavra;
```

(a) Pesquisa tem como resultado os adjetivos que modificam `carro`.

```
SELECT Palavra.palavra, Palavra.classe, Coo.LogDice, Coo.frequencia
FROM ( SELECT idPalavra1,Dice,frequencia
       FROM Coocorrencia
       WHERE idPalavra2 = 1615
       and frequencia >=  2
       and tipoDep = 'MOD'
       and nomeProp = 'POST_NOUN_NOUN'
       ORDER BY LogDice DESC
       LIMIT 10) as Coo
INNER JOIN Palavra on Coo.idPalavra1 = Palavra.idPalavra
```

(b) Pesquisa tem como resultado os nomes que `carro` modifica.

Figura 3.15: Pesquisas SQL para obter as co-ocorrências.

A pesquisa SQL representada na Figura 3.15 (a) tem como resultado um conjunto de dez adjetivos que modificam à esquerda o nome `carro`. Estes são ordenados de forma decrescente pelo valor da medida de associação escolhida (*LogDice*), presente em cada uma das co-ocorrências. A pesquisa representada em (b) tem como resultado um conjunto de nomes que são modificados por `carro` (ou seja, de que `carro` é complemento preposicional). A pesquisa (a) é executada quando a palavra-alvo é o primeiro elemento da relação, enquanto que a pesquisa (b) é executada quando a palavra-alvo é o segundo elemento da relação.

```
{
   "PRE_GOVERNED":{
       "MOD_PRE_NOUN_ADJ":{
         "name":"Modificado por um Adjetivo",
         "dep" :"MOD",
         "prop":"PRE_NOUN_ADJ",
         "data":[
            {
               "word":"potente",
               "word_pos":"ADJ",
               "measure":0.023,
               "frequency":14,
               "duallog":4
            },
             ...
          ]
      },
      ...
   },
   "PRE_GOVERNOR": {...},
   "POST_GOVERNED":{...},
   "POST_GOVERNOR":{...}
}
```

Figura 3.16: Exemplo do objeto *JSON* enviado para o cliente.

Durante as pesquisas, o objetivo é criar um novo objeto *JSON*, a ser preenchido com a informação da palavra e classe desejada, recolhida com as diferentes pesquisas executadas na base de dados. A Figura 3.16 representa o objeto *JSON* preenchido durante a execução, sendo posteriormente enviado para o cliente como resposta ao pedido inicial. Para cada padrão dependência-propriedade, é enviado o título a apresentar na interface e o conjunto das palavras com quem a palavra-alvo co-ocorre. Para cada uma destas, é adicionado o lema, o valor da medida de associação, a frequência e o logaritmo de base dois da frequência dessa palavra. Para as diferentes medidas de associação, o valor destas é arredondado em diferentes casas decimais consoante a medida escolhida.

```
SELECT Frase.numeroFrase, Frase.nomeFicheiro, Frase.frase
FROM (SELECT numeroFrase, nomeFicheiro
      FROM Exemplifica
      WHERE idPalavra1 = 5878
      and idPalavra2 = 1615
      and nomeProp = 'POST_NOUN_NOUN'
      and tipoDep = 'MOD'
      LIMIT 5) as ex
INNER JOIN Frase on (ex.numeroFrase = Frase.numeroFrase
                     and ex.nomeFIcheiro = Frase.nomeFicheiro);
```

Figura 3.17: Exemplo de pesquisa SQL para obter as frases para a co-ocorrência `(volante, carro)`.

Quando o pedido recebido do cliente pretende solicitar as frases que exemplificam uma co-ocorrência, é executada a pesquisa SQL representada na Figura 3.17. Esta tem como resultado um conjunto de cinco frases, contendo para cada uma delas a informação que a identifica. A partir deste é criado o objeto JSON enviado para o cliente.

3.3.2 *Client-Side*

No lado do cliente (*browser*), o código foi desenvolvido a partir da *framework AngularJS*[7], que permite usar as linguagens *HTML* (*HyperText Markup Language*) e *CSS* (*Cascading Style Sheets*) como linguagens declarativa e de apresentação, respetivamente, permitindo que a interface se torne dinâmica, fazendo-a funcionar como uma aplicação dentro do *browser*.

Figura 3.18: Formulário de pesquisa de uma palavra.

Antes de se efetuar o pedido ao servidor, é fundamental reunir os dados necessários. Para isso, na aplicação foi desenvolvido um formulário, presente na Figura 3.18, onde os utilizadores apenas têm de introduzir a palavra-alvo que pretendem pesquisar, a sua classe e a qual a medida de associação. No entanto, estes podem ainda escolher como opções adicionais qual a frequência mínima de cada co-ocorrência e o número máximo de palavras

[7] https://code.angularjs.org/1.4.6/docs/guide/introduction (última visita a 26/09/2015).

que desejam em cada padrão dependência-propriedade. Por defeito, esses valores são dois e dez respetivamente. O valor da frequência mínima das co-ocorrências foi escolhido para evitar os casos em que alguns padrões não tenham sido corretamente identificados, podendo o utilizador alterar esse número em qualquer pesquisa.

Após se reunir esta informação num objecto *JSON*, este é envidado para o lado do servidor a partir de um pedido assíncrono *AJAX* implementado pela biblioteca *jQuery*[8]. Quando a palavra pesquisada não existe ou não existem resultados para a mesma, o utilizador é informado. Em situação contrária, a informação é apresentada segundo a estrutura presente na Secção 3.2.1.1, isto é, dividida em dois grupos: palavra-alvo à esquerda e palavra-alvo à direita da palavra com quem co-ocorre.

Para cada padrão dependência-propriedade em que a palavra-alvo participa, as palavras que co-ocorrem com esta são apresentadas por ordem decrescente do grau de coesão entre estas. Cada palavra é apresentada no formato "$lema\ (l : m)$", onde l corresponde ao valor do logaritmo da base dois da frequência da co-ocorrência, e m ao valor da medida selecionada. O valor de l serve para indicar a classe da frequência desta co-ocorrência, mostrando que esta ocorreu cerca de 2^l vezes.

O utilizador tem a possibilidade de trocar rapidamente de medida de associação. De forma automática, o sistema faz um novo pedido, atualizando os resultados para esta, sem ser necessário novamente a introdução dos dados.

Figura 3.19: Co-ocorrências para o nome `país` ordenadas pela medida *LogDice*.

A Figura 3.19 apresenta um exemplo com o conjunto de co-ocorrências detetadas pelo sistema para o nome `país`. Na pesquisa efetuada que deu origem a este resultado, foi escolhida a medida de associação *LogDice* para ordenar as ocorrências.

Figura 3.20: Co-ocorrências para o adjetivo `grande` ordenadas pela medida *LogDice*.

[8] `http://api.jquery.com/jquery.ajax/` (última visita a 27/09/2015).

Para os adjetivos, o resultado apresentado é semelhante à figura 3.20. Os dados presentes no exemplo pertencem a uma procura onde a palavra-alvo escolhida é o adjetivo `grande`.

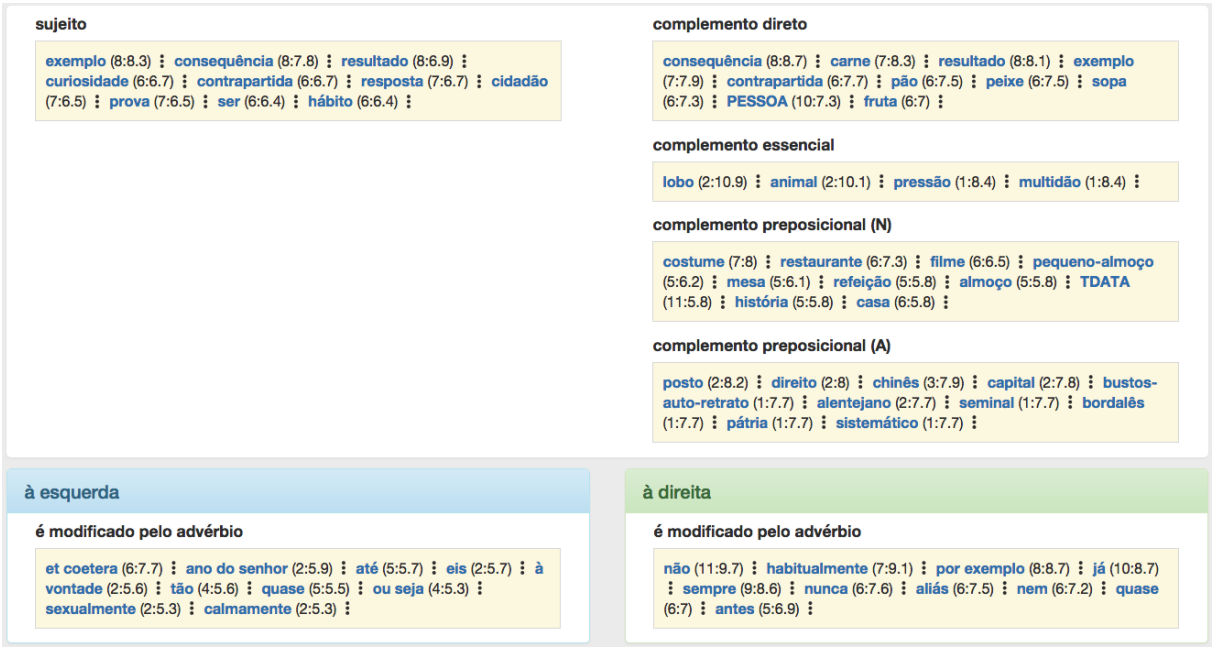


Figura 3.21: Co-ocorrências para o verbo `comer` ordenadas pela medida *LogDice*.

Quando a palavra que se pretende explorar é um verbo, o utilizador tem acesso ao resultado apresentado na Figura 3.21. Esta figura mostra o resultado para a palavra-alvo `comer`. A principal diferença observada em relação ao resultado apresentado para os nomes e adjetivos é a existência de um grupo para os padrões dependência-propriedade exclusivos dos verbos, que mostra as relações deste com o sujeito e complementos direto, essencial e preposicional.

Figura 3.22: Co-ocorrências para o advérbio `ainda` ordenadas pela medida *LogDice*.

Por fim, na Figura 3.22 é apresentado o conjunto de co-ocorrências para um advérbio, neste caso `ainda`. A principal diferença entre os exemplos anteriores e o que é apresentado para os advérbios, é a existência de um

padrão dependência-propriedade específico. Assim, indica-se se o advérbio modificou ou não uma frase, e em caso afirmativo quantas vezes o fez.

A partir dos resultados obtidos, é possível pedir mais informação sobre uma co-ocorrência específica. Ao clicar sobre uma palavra que se encontra num dos conjuntos de co-ocorrências, é solicitado ao servidor um conjunto de frases onde esta ocorreu. Após a resposta, é apresentado ao utilizador os detalhes desta co-ocorrência em particular, o conjunto de frases que exemplificam a co-ocorrência e em que parte do *corpus* esta ocorreu.

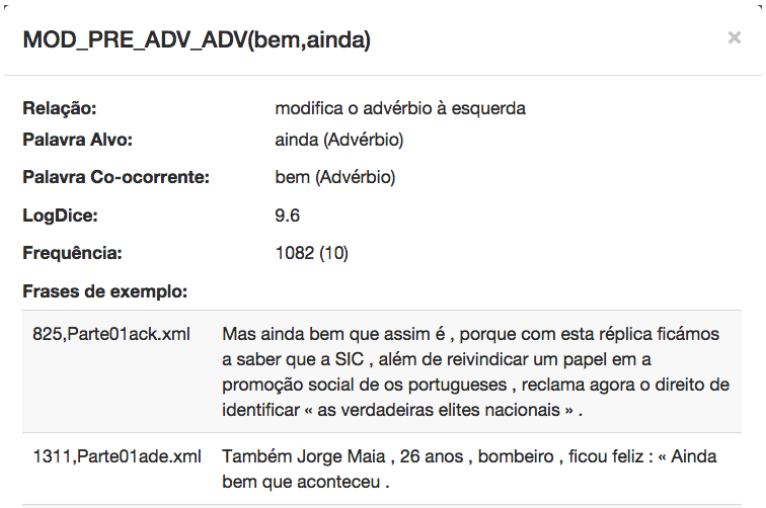


Figura 3.23: Detalhes da co-ocorrência `MOD PRE_ADV_ADV(bem,ainda)`.

A Figura 3.23 representa a informação detalhada para a co-ocorrência entre os advérbios `bem` e `ainda`, que tem como padrão dependência-propriedade `MOD_PRE_ADV_ADV`. Juntamente com a informação detalhada, também estão presentes duas frases que exemplificam a aplicação desta co-ocorrência no texto (`"ainda bem"`), sendo indicado onde se localiza essa frase dentro do *corpus*.

Capítulo 4

Avaliação

Neste Capítulo, descrevem-se quer os procedimentos utilizados para avaliar o trabalho realizado, quer os resultados obtidos. A primeira parte consiste em avaliar os componentes principais do projeto descritos no Capítulo anterior. Na Secção 4.2 é avaliada a extração das co-ocorrências (apresentada na Secção 3.2) e a evolução da base de dados (apresentada na Secção 3.1), sendo também avaliada a aplicação *web* (apresentada na Secção 3.3) na Secção 4.3.

Na segunda parte, será avaliado o resultado global do projeto, comparando os resultados obtidos nas procuras com uma das ferramentas descritas no Capítulo 2, o *DeepDict* apresentado na Secção 2.2.1.

4.1 Métodos de Avaliação

Para avaliar o desempenho do sistema desenvolvido é necessário um *corpus* que, após ser processado pela STRING, seja submetido à extração de co-ocorrências de modo a obter a base de dados com a informação devidamente armazenada. A avaliação foi realizada com base na extração das co-ocorrências a partir do *corpus* CETEMPúblico[29] processado pela STRING. Os textos deste *corpus* foram recolhidos de um jornal diário e contêm aproximadamente 190 milhões de palavras.

O *corpus* processado ocupa 237 *GigaBytes* de espaço, dividido em 20 partes com cerca de 12 *GigaBytes* cada, em que cada uma contém aproximadamente 210 ficheiros no formato *XML* que ocupam cerca de 60 *MegaBytes*.

A ferramenta de extração foi executada numa máquina Unix com arquitetura x86_64, com 16 *CPUs* Intel(R) Xeon(R)E5530 com 2.40GHz de velocidade. Esta máquina possui 48.390 *MegaBytes* de memória.

Os testes efetuados para avaliar a aplicação *web* e as pesquisas SQL foram efetuados numa máquina Unix com arquitetura x86_64, com um *CPU* Intel Core i5-2415M com velocidade 2,3 GHz, com 8.192 *MegaBytes* de memória.

4.2 Extração de co-ocorrências

Para a avaliação da ferramenta de extração de co-ocorrências será necessário medir o tempo gasto durante a execução desta, perceber qual a evolução do espaço gasto pela base de dados e analisá-la após estar preenchida de modo a tirar as conclusões necessárias acerca deste processo.

A análise consiste em executar duas versões da ferramenta e é analisada de duas formas: a versão onde apenas se extraem as co-ocorrências (versão simplificada) e a versão onde se extraem as co-ocorrências juntamente com frases de exemplo (versão completa). Esta diferenciação torna-se importante para avaliar as diferenças no desempenho que se prevêem em cada caso, quer no tempo, quer no espaço ocupado pela base de dados. Quando se diz extrair as co-ocorrências, está implícito que se armazena a informação acerca de uma co-ocorrência, as palavras que se relacionam, bem como as frequências destas.

Para tornar esta avaliação mais completa, a execução da ferramenta de extração é inicialmente analisada com apenas um ficheiro de uma das partes do *corpus* descrita acima. A ferramenta corre três vezes nas mesmas condições. Os resultados apresentados são a média aritmética dos tempos obtidos.

Assim, o ficheiro processado pelas duas versões da ferramenta de extração foi `Parte01aaa.xml` da `Parte01` que possui 60,8 *MegaBytes* de tamanho. Em primeiro lugar, este ficheiro foi processado pela versão simplificada da ferramenta, que não armazena as frases de exemplo. O processo levou em média 29,76 segundos, dos quais 9,71 segundos foram consumidos pelo XIPAPI a importar o ficheiro XML, 11,54 segundos no processamento das co-ocorrências e por fim 8,14 segundos a carregá-las na base de dados, de que resultou um ficheiro com um tamanho de 2,13 *MegaBytes*.

Em segundo lugar, o ficheiro foi processado pela versão completa da ferramenta de extração, isto é, extração de co-ocorrências juntamente com o armazenamento de frases de exemplo. Neste caso, o processo demorou em média 32,5 segundos, dos quais 9,76 segundos foram consumidos pelo XIPAPI a importar o ficheiro XML, 10,65 segundos na extração de co-ocorrências e por fim 11,69 segundos a armazenar estas co-ocorrências e as frases na base de dados, a qual após a execução ocupa 4,33 *MegaBytes* de tamanho.

Apesar de esta amostra processada ser pouco significativa quando comparada com a totalidade do *corpus* CETEMPúblico, podem-se tirar algumas conclusões. Como esperado, em ambos os casos, a XIPAPI teve um desempenho semelhante, e espera-se que se mantenha constante durante a execução dos restantes ficheiros do *corpus* processado. Para a extração de cada dependência, o tempo decorrido foi muito semelhante, porque apesar de a segunda versão da ferramenta tratar das frases, estas não terão um impacto negativo no desempenho, pois apenas se adicionam as frases num mapa existente no programa. Uma vez que este processo se aplica aos ficheiros importados pela XIPAPI, e estes têm tamanho constante, prevê-se que o desempenho se mantenha constante ao longo da totalidade dos ficheiros do *corpus*.

A principal diferença entre as duas versões da ferramenta é a fase de armazenamento da informação na base de dados, já que na versão completa, esta não só armazena a informação sobre as co-ocorrências, como também armazena as frases onde estas ocorreram, fazendo com que o processo demore mais tempo até à sua conclusão. Isto também se reflete no espaço ocupado, visto que a base de dados que não armazena exemplos de frases ocupa 49,2% do tamanho da base de dados completa. Quando se armazena a informação na base de dados, é necessário verificar se uma determinada entrada já existe numa tabela, actualizando os valores já existentes, ou em caso contrário, é

necessário inserir a nova entrada. O tempo consumido por esta fase vai depender do tempo de resposta da base de dados. Ao longo da extração, com o aumentar do espaço ocupado pela base de dados, e como esta é representada localmente por um único ficheiro, é de esperar que o tempo de resposta aumente, fazendo com que o tempo de armazenamento da base de dados seja maior. Neste caso, como a base de dados se encontrava vazia, o tempo de resposta foi mínimo.

Após a análise desta pequena amostra, que representa 0,025% da totalidade do *corpus*, pode prever-se que quando se processam apenas as co-ocorrências, o processo é ligeiramente mais rápido, no entanto, a base de dados deverá ser bastante menor do que a da ferramenta completa.

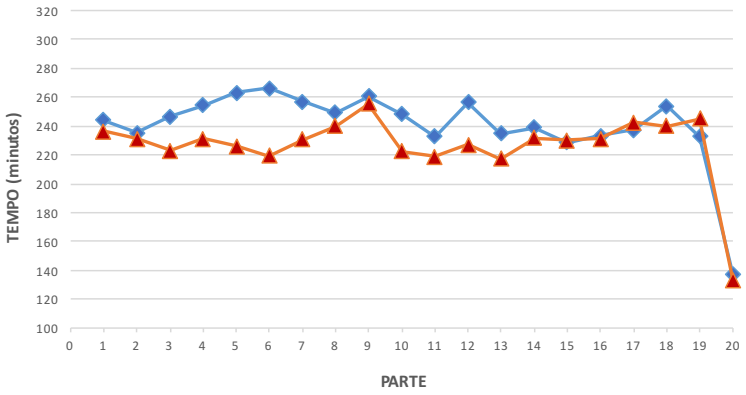


Figura 4.1: Tempo consumido por parte do *corpus* CETEMPúblico.

Posto isto, é necessário analisar o processo de extração da totalidade do *corpus*. Assim, depois de se executar as duas versões da ferramenta, é fundamental avaliar o desempenho destas. A Figura 4.1 mostra o desempenho para cada uma das versões da ferramenta, apresentando o tempo gasto em cada parte do *corpus*. A vermelho (com a marca ▲) são representados os tempos da versão simplificada e a azul (marca ♦) os tempos da versão completa. Ao observar o gráfico, pode-se verificar que ambas versões mantiveram uma execução relativamente constante. A explicação para o facto de a `Parte20` ter decorrido de forma substancialmente mais rápida é devido ao menor tamanho desta parte, que por ser a última não possui tamanho igual às anteriores. No geral, a versão simplificada corre cada parte em menos tempo em relação à versão completa, com tempos de execução aproximados para ambas na parte final. Esta aproximação é provocada por um ligeiro aumento do tempo decorrido por parte da versão simplificada e principalmente pela diminuição do tempo de execução da versão completa.

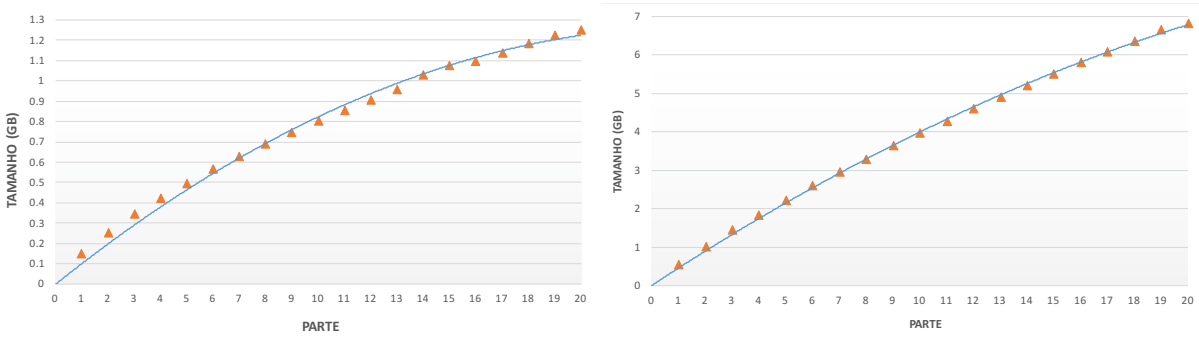
Para a versão simplificada, o tempo total de execução foi de 4.608 minutos (3 dias, 4 horas e 48 minutos), o tempo médio por parte foi de 230 minutos (3 horas e 50 minutos), e o tempo médio por ficheiro de 1 minuto e 5,7 segundos. Assim, o tempo de execução médio por ficheiro é 36 segundos mais lento que o teste inicial, mostrando que com o avançar do processo, este piora devido ao tempo de resposta da base de dados. Isto deve-se não só à informação já armazenada, onde o número de verificações aumenta, mas também ao facto de o próprio desempenho desta tender a piorar com o aumento do tamanho do ficheiro.

Após se armazenar na base de dados a informação sobre as co-ocorrências da totalidade do *corpus*, foi necessário calcular as medidas de associação para cada co-ocorrência. Para esta versão, a medida *Significance* não foi calculada pois seria necessário informação acerca das frases onde ocorreram as co-ocorrências. O cálculo das medidas foi executado em 136 minutos (2 horas e 16 minutos).

No caso da versão completa, esta processou o *corpus* em 4.878 minutos (3 dias, 9 horas e 18 minutos), o que resulta num tempo médio por parte de 243 minutos (4 horas e 3 minutos) e por ficheiro de 1 minuto e 9,4 segundos. Aqui, tal como na versão previamente analisada, o intervalo de tempo em que um ficheiro é processado aumentou 36,9 segundos. A ligeira diminuição do tempo decorrido no processamento das partes pode ser explicado pelo facto de para cada co-ocorrência apenas serem guardadas 15 frases, e para as que ocorrem mais vezes juntas, deixa de ser necessário armazená-las, o que permite diminuir ligeiramente o tempo armazenamento dos dados extraídos em cada ficheiro.

As medidas de associação foram calculadas em 146 minutos (2 horas e 26 minutos). Este tempo foi ligeiramente superior ao tempo gasto pela outra versão, pois foram calculadas todas as medidas.

No geral, pode-se dizer que cada uma das versões teve, para cada parte, um tempo decorrido constante, que permitiu que o processo de extração não se alongasse com o evoluir do processo.



(a) Versão simplificada

(b) Versão completa

Figura 4.2: Evolução do tamanho da base de dados durante o processamento do *corpus*.

Outro aspeto importante a analisar é a evolução do tamanho da base de dados. Os gráficos presentes na Figura 4.2 representam esta evolução.

No caso da versão simplificada da base de dados, após a execução da ferramenta de extração esta ocupa 1,25 *GigaBytes*. Ao executar o processo que calcula as medidas de associação, a base de dados atinge os 1,68 *GigaBytes* de tamanho. Assim verifica-se que as medidas ocupam 25,6% (0,43 *GigaBytes*) do total da base de dados.

No gráfico (a) da Figura 4.2 estão representados para cada parte do *corpus* o valor do tamanho da base de dados até atingir o tamanho total. Pode-se observar que a evolução não foi linear. A linha azul representa a função obtida a partir de uma regressão polinomial de grau dois que acompanha o crescimento. Esta mostra que inicialmente o crescimento é mais acentuado mas tende a diminuir com a evolução do processo. Esta variação explica-se porque no início a base de dados se encontra sem informação armazenada, o que faz com que sejam adicionadas inicialmente mais informações acerca das co-ocorrências (palavras, padrão dependência-propriedade a que as palavras pertencem, co-ocorrências e frequências). A sua evolução permite que, para as co-ocorrências já armazenadas, apenas seja necessário atualizar o valor da sua frequência.

Em relação à versão completa, a base de dados com a totalidade da informação extraída ocupa 6,8 *GigaBytes*, ou seja, é 5,84 vezes maior que a base de dados da versão simplificada. Após se calcularem e armazenarem as medidas de associação, a base de dados conta com 7,3 *GigaBytes*, ou seja, as medidas ocupam 6,8% (0,5 *GigaBytes*) do total da base de dados. O espaço ocupado pelas medidas de associação em ambas as base de dados

só não é exatamente igual pois, como já foi referido, na versão simplificada uma das medidas não foi calculada. O gráfico (b) da Figura 4.2 mostra, para a versão completa, que a evolução do tamanho da base de dados não é linear, apesar de aqui a curva de crescimento ser menos acentuada. A linha azul mostra a função obtida a partir de uma regressão polinomial de grau dois para os valores da figura. A evolução desta versão tem uma curva menos acentuada, ou seja, aproxima-se mais de uma função linear, porque é necessário armazenar para cada co-ocorrência a informação desta numa determinada frase. Mesmo que uma co-ocorrência já exista na base de dados, é preciso adicionar a frase onde esta aconteceu, exceto quando já foram armazenadas quinze frases de exemplo para uma determinada co-ocorrência. O armazenamento de frases faz com que o crescimento seja mais rápido e acentuado. Caso não houvesse o limite de armazenamento de quinze frases, o crescimento seria praticamente linear.

A avaliação da execução da ferramenta de extração é importante para perceber quais os aspectos que se podem melhorar. O facto de terem sido avaliadas duas versões da ferramenta, permitiu perceber que o principal factor que pode reduzir o desempenho temporal é a interação com a base de dados. Outro aspecto importante de referir é que, como a base de dados utilizada é representada num único ficheiro local, esta não permite concorrência na modificação (adição ou alteração) de dados nesta.

Após a execução da ferramenta de extração e do cálculo de medidas de associação, obtém-se a base de dados que se utiliza para analisar os padrões de co-ocorrência obtidos. Assim, é importante perceber como está organizada a informação armazenada. A base de dados analisada é a versão completa, já que a versão simplificada é semelhante, com a exceção do armazenamento das frases de exemplo.

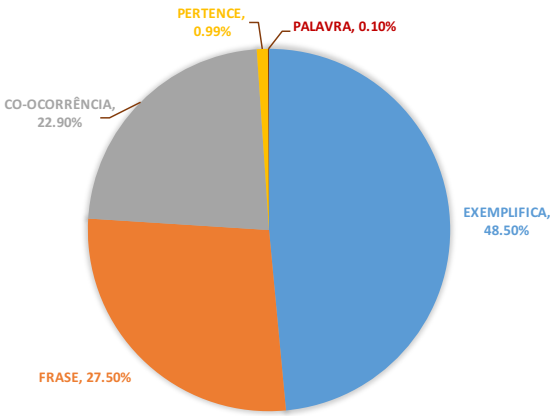


Figura 4.3: Divisão por tabela da informação armazenada na base de dados.

O gráfico da Figura 4.3 apresenta a distribuição da informação armazenada na base de dados por tabela. As tabelas `Dependência`, `Propriedade` e `Corpus` não surgem no gráfico, porque apesar da importância que estas representam no modelo relacional, o número de linhas em cada uma não é elevado, de forma que estas ocupam pouco espaço em relação às restantes, isto é, juntas apenas representam 6 *KiloBytes* do espaço total da base de dados.

Também se pode observar que as tabelas que armazenam a informação sobre as co-ocorrências, ou seja, as tabelas `Palavra`, `Pertence`, `Co-ocorrência`, `Dependência` e `Propriedade` representam 24% do total da base de dados. Como referido acima, esta tem um tamanho de 7,3 *GigaBytes*, logo as tabelas representam 1,75 *GigaBytes* do total. Assim, as tabelas representam a base de dados obtida pela versão simplificada da ferramenta de extração (1,68 *GigaBytes*), onde a diferença é a medida (*Significance*), que a base de dados completa inclui.

A Tabela `Frase` que armazena as frases e `Exemplifica` que armazena a correspondência entre uma frase e uma co-ocorrência, representam 76% do tamanho total da base de dados. Este espaço não é superior porque, como dito anteriormente, se concluiu que apenas quinze frases por co-ocorrência eram suficientes para exemplificar a relação entre as duas palavras.

Tabela 4.1: Número de entradas por tabela da base de dados.

Tabela	Número de entradas
`Corpus`	1
`Dependência`	7
`Propriedade`	18
`Palavra`	308.573
`Pertence`	761.978
`Co-ocorrência`	11.244.852
`Frase`	6.218.236
`Exemplifica`	26.735.453

A Tabela 4.1 apresenta o número de entradas para cada tabela presente na base de dados, onde foram encontradas 308.573 palavras diferentes, que ocorreram 102.643.502 vezes no *corpus*. Foram detectadas 51.321.751 co-ocorrências que resultaram em 11.244.852 co-ocorrências diferentes armazenadas na base de dados. Em média, cada co-ocorrência ocorreu 4,6 vezes.

4.3 Aplicação *Web*

Para avaliar a aplicação *Web*, foram analisados os desempenhos das pesquisas SQL, e posteriormente a análise do tempo de resposta aquando da pesquisa de uma lema duma palavra, ou seja, o tempo que demora até o servidor responder ao pedido enviado. Para isso, foram aleatoriamente selecionados dois lemas de palavras para cada classe, um com baixa e outro com grande frequência, para verificar se existem diferenças no desempenho entre eles. Assim, os lemas das palavras a avaliar são:

- `caneta` (nome): ocorreu 998 vezes;
- `país` (nome): ocorreu 196.140 vezes;
- `otimizar` (verbo): ocorreu 972 vezes;
- `ser` (verbo): ocorreu 2.533.385 vezes;
- `nitidamente` (advérbio): ocorreu 964 vezes;
- `não` (advérbio): ocorreu 1.362.286 vezes;
- `lindo` (adjetivo): ocorreu 997 vezes;
- `grande` (adjetivo): ocorreu 263.518 vezes.

As pesquisas SQL usadas para testar cada um dos lemas das palavras descritos acima, têm como medida escolhida o `Dice`, a frequência mínima de cada co-ocorrência é 2, onde cada uma tem como resultado o máximo de dez palavras com a qual se relacionam. Os resultados são apresentados nas Tabelas 4.2, 4.3 , 4.4 e 4.5,

onde se apresentam os resultados obtidos ao testar individualmente as pesquisas SQL para os diferentes padrões dependência-propriedade no qual o lema correspondente pode ocorrer. Os tempos das pesquisas apresentados na tabela foram obtidos ao executar a pesquisa SQL diretamente na base de dados.

Tabela 4.2: Tempo decorrido em cada pesquisa SQL no caso de um `nome`.

	`caneta` (nome)	`país` (nome)
DEP PROP	TEMPO (s)	TEMPO (s)
MOD_PRE_NOUN_ADJ	0,001	0,030
QUANTD_NOUN_NOUN	*0,001	*0,026
CLASSD_NOUN_NOUN	*0,001	*0,025
MOD_POST_NOUN_ADJ	0,001	0,028
MOD_POST_NOUN_NOUN	3,136	3,131
Total:	3,140	3,240

Na Tabela 4.2 pode-se observar que, para o nome que ocorreu menos vezes na base de dados (`caneta`), as pesquisas são ligeiramente mais rápidas quando comparado com o nome escolhido que ocorreu mais vezes (`país`). Isto resulta num ganho de 0,1 segundos, o que para o utilizador se torna praticamente insignificante. Tal acontece porque, apesar de cada pesquisa só pretender dez palavras como resultado, é necessário obter todas as palavras que se relacionam com a palavra pesquisada, ordenar o resultado por ordem decrescente do valor da medida de associação escolhida e, por fim, selecionar as dez entradas da tabela com os valores maiores. Como consequência, as pesquisas das palavras que ocorreram mais vezes, vão necessariamente demorar um pouco mais que as outras. O símbolo * indica que essa pesquisa não encontrou resultados.

Tabela 4.3: Tempo decorrido em cada pesquisa SQL no caso de um `verbo`.

	`otimizar` (verbo)	`ser` (verbo)
DEP PROP	**TEMPO (s)**	**TEMPO (s)**
SUBJ_SEM_PROP	0,007	1,505
MOD_PRE_VERB_ADV	0,008	0,460
CDIR_SEM_PROP	0,006	0,457
CINDIR_SEM_PROP	*0,006	0,452
COMPL_SEM_PROP	*0,006	*0,421
MOD_VERB_NOUN	0,007	0,457
MOD_VERB_ADJ	0,007	0,457
MOD_POST_VERB_ADV	0,006	0,476
Total:	0,053	4,685

A Tabela 4.3 apresenta os resultados para os verbos escolhidos, `otimizar` e `ser`. Também se pode verificar que as pesquisas com o lema da palavra que ocorreram menos vezes (`otimizar`), foram mais rápidas 4,6 segundos do que as pesquisas do lema da palavra escolhida que ocorreram mais vezes (`ser`). A diferença é superior à verificada

no caso dos nomes pois país ocorreu apenas 7,7% do número de vezes que ocorreu o verbo ser, fazendo com que, durante as pesquisas, estas tenham mais dados para procurar o que atrasa a pesquisa.

Tabela 4.4: Tempo decorrido em cada pesquisa SQL no caso de um advérbio.

	nitidamente (advérbio)	não (advérbio)
DEP PROP	**TEMPO (s)**	**TEMPO (s)**
MOD PRE_ADV_ADV	0,005	0,002
MOD PRE_ADJ_ADV	3,163	3,190
MOD PRE_ADV_ADV	3,100	3,156
MOD PRE_FOCUS_NOUN_ADV	*3,133	*3,176
MOD PRE_VERB_ADV	3,115	3,209
MOD POST_ADV_ADV	*0,001	0,002
MOD POST_ADJ_ADV	3,150	3,250
MOD POST_FOCUS_NOUN_ADV	*3,200	*3,167
MOD POST_VERB_ADV	3,274	3,186
MOD TOP_ADV	3,140	3,215
Total:	25,281	25,553

Para os advérbios testados, os resultados obtidos estão presentes na Tabela 4.4, onde o advérbio (não) escolhido entre aqueles que ocorreram mais vezes, executa as suas pesquisas SQL 0,27 segundos a mais do que a palavra menos frequente (nitidamente). A diferença verificada assemelha-se à encontrada para o caso dos nomes. O total para cada um dos advérbios é bastante superior em relação aos tempos já observados anteriormente. Uma das causas é o facto de para cada advérbio serem feitas mais pesquisas. No entanto, a maior parte destas não são executadas instantaneamente, o que, globalmente, resulta num intervalo de tempo maior.

Tabela 4.5: Tempo decorrido em cada pesquisa SQL no caso de um adjetivo.

	lindo (adjetivo)	grande (adjetivo)
DEP PROP	**TEMPO (s)**	**TEMPO (s)**
MOD PRE_ADJ_ADV	0,005	0,004
MOD PRE_NOUN_ADJ	3,249	3,294
MOD POST_ADJ_ADV	*0,001	0,005
MOD POST_NOUN_ADJ	3,354	3,276
Total:	6,609	6,579

Por fim, a Tabela 4.5 apresenta os resultados para as pesquisas SQL executadas pelos adjetivos lindo e grande. Os resultados obtidos neste caso são diferentes dos anteriores, pois é o adjetivo escolhido entre aqueles que ocorreram menos vezes, que leva mais tempo a executar as pesquisas. No entanto, este resultado deve ser específico das palavras usadas, pelo que normalmente as palavras com frequência superior deveriam ser ligeiramente mais lentas.

Após se testarem as pesquisas individualmente, chega-se à conclusão de que as pesquisas para os lemas de palavras com maior frequência demoram normalmente mais tempo a serem executadas do que para os lemas com menor frequência. No entanto, essa diferença não deve ser significativa para que o utilizador note a diferença entre os lemas escolhidos. Ainda assim, existem algumas pesquisas que deviam ser mais rápidas, para que no conjunto isso resultasse num intervalo de tempo inferior, e assim o utilizador esperasse o menor tempo possível. Contudo, com uma base de dados desta dimensão, os resultados são aceitáveis. Um dos fatores que atrasa as diferentes pesquisas é a ordenação dos resultados por ordem decrescente da medida de associação escolhida. Uma solução adequada seria efetuar uma ordenação prévia das co-ocorrências por uma das medidas, sendo essa a predefinida durante as pesquisas. Isto resultaria em pesquisas mais rápidas para a medida, não influenciando o resultado das restantes, mantendo-se estes próximos dos que foram apresentados aqui.

Tabela 4.6: Tempo de resposta de uma pesquisa na aplicação por classe.

	Nome		Verbo		Advérbio		Adjetivo	
Palavra	**caneta**	**país**	**otimizar**	**ser**	**nitidamente**	**não**	**lindo**	**grande**
Tempo (s)	8,28	8,77	0,053	11,11	66,00	67,33	16,40	16,65

A análise das pesquisas SQL é importante para perceber qual o desempenho de cada uma destas, mas também é importante perceber qual o tempo de resposta que um utilizador precisa de esperar para obter resultados na procura de um determinado lema de uma palavra. A Tabela 4.6 apresenta os resultados obtidos para o tempo de resposta de uma procura na aplicação *Web* com as palavras acima descritas. Cada uma foi repetidas três vezes, sendo o valor da média aritmética dos valores obtidos apresentado na tabela.

O tempo apresentado na tabela para cada palavra é superior ao apresentado no total das pesquisas SQL nas tabelas anteriores. Isto acontece principalmente, porque os tempos resultantes das pesquisas foram obtidos executando-as diretamente na base de dados, enquanto que aqui são obtidas a partir de métodos implementados pelo PHP que gerem o acesso à base de dados. Para este valor, ainda é necessário adicionar o tempo de envio, de resposta, e o tempo que demora a tratar os resultados obtidos nas pesquisas. Ainda assim, o principal consumidor de tempo são as pesquisas SQL. Apesar de nem todos os valores serem aceitáveis, para os advérbios, o tempo de espera atual é demasiado longo, pois são a classe de palavras que participa em mais padrões dependência-propriedade, sendo necessário reduzir este tempo de forma a conseguir tornar a resposta do sistema mais rápida.

Para testar o tempo que a aplicação demora a apresentar as frases de exemplo para uma co-ocorrência, foram escolhidas três palavras que co-ocorrem com o nome país.

A Tabela 4.7 indica qual a co-ocorrência a analisar, o número de ocorrências desta e o tempo médio que demorou até apresentar os resultados pretendidos. É possível observar que quanto maior é a frequência de uma

Tabela 4.7: Tempo decorrido para obter frases de exemplo para uma determinada co-ocorrência.

Co-ocorrência	**Frequência**	**Tempo (s)**
`(centro,país)`	789	0,301
`(país,praticamente)`	46	18,42
`(país,populoso)`	16	75

co-ocorrência, menor é o tempo de resposta. No caso da co-ocorrência `(centro, país)` com 789 ocorrências, a resposta foi praticamente instantânea, enquanto que no caso de `(país, populoso)` com 16 ocorrências já foi preciso esperar em média 75 segundos. A grande diferença observada nestes valores, deve-se ao tamanho da tabela da base de dados `Exemplifica`. Assim, quanto maior for a frequência de uma co-ocorrência, maior é a probabilidade de os quinze exemplos se encontrarem no início da tabela, permitindo o acesso a estes de forma instantânea. Por outro lado, no caso das que têm uma frequência menor, já é necessário uma procura mais aprofundada na tabela, provocando um maior gasto de tempo.

Uma vez que os resultados apresentados em alguns casos eram demasiado elevados, foram implementados três indices em duas das tabelas da base de dados de forma a permitir resultados mais rápidos. Na tabela `Co-ocorrência` foram implementados dois índices, um para as co-ocorrências onde a palavra-alvo ocorre à esquerda e outro para as co-ocorrências onde a palavra-alvo ocorre à direita. Para a tabela `Exemplifica` foi adicionado um índice para cada uma das co-ocorrências.

Tabela 4.8: Tempo de resposta de uma pesquisa na aplicação por classe após a implementação de índices na base de dados.

	Nome		Verbo		Advérbio		Adjetivo	
Palavra	**caneta**	**país**	**otimizar**	**ser**	**nitidamente**	**não**	**lindo**	**grande**
Tempo (s)	0,102	0,111	0,103	0,760	0,037	0,174	0,067	0,256

Tabela 4.9: Tempo decorrido para obter frases de exemplo para uma determinada co-ocorrência após a implementação de índices na base de dados.

Co-ocorrência	**Frequência**	**Tempo (s)**
`(centro,país)`	789	0,024
`(país,praticamente)`	46	0,025
`(país,populoso)`	16	0,035

As Tabelas 4.8 e 4.9 apresentam os resultados obtidos após a implementação dos índices. Como se pode observar, o tempo de espera por parte do utilizador foi reduzido em relação aos tempos anteriores, permitindo consultas à informação armazenada na base de dados de forma bastante mais rápidas.

4.4 Resultados Globais

Nesta Secção será comparado o resultado entre o projeto desenvolvido e o *DeepDict* (apresentado na Secção 2.2.1). O facto de se comparar o projeto desenvolvido com este sistema em específico deve-se em parte à importância e influência que o *DeepDict* teve na definição deste projeto. Uma vez que o utilizador vai usar a aplicação *Web*, que permite explorar os padrões de co-ocorrência para uma determinada palavra-alvo, as principais comparações serão ao nível das funcionalidades e informação disponibilizada pelas interfaces de ambos os sistemas.

Na comparação dos sistemas, é importante definir para ambos as principais diferenças e semelhanças entre as funcionalidades que estes disponibilizam aos utilizadores. Assim, os sistemas partilham:

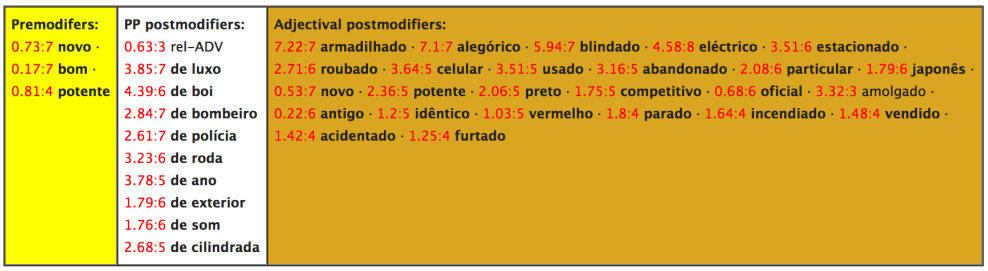
- pesquisa de padrões de co-ocorrência para uma determinada palavra-alvo;

- pesquisa definida a partir do lema da palavra-alvo e da sua classe;
- em cada classe da palavra-alvo os resultados são organizados de forma a permitir a leitura natural de cada co-ocorrência;
- possibilidade de mostrar exemplos de frases para cada co-ocorrência;
- informação armazenada em ambos os ficheiros proveniente do *corpus* CETEMPúblico.

Por outro lado, as principais diferenças entre os sistemas são:

- o *DeepDict* utiliza alguns protótipos semânticos para agrupar palavras que representam diferentes categorias, enquanto que no projecto desenvolvido se usam as categorias das entidades mencionadas (Secção 3.2.1.2) para agrupar os lemas;
- o *DeepDict* apenas ordena as co-ocorrências pela medida de associação presente no sistema (Variante do PMI) enquanto que no projeto aqui desenvolvido é possível escolher a medida de associação que mais se adequa ao que se pretende, de entre as medidas implementadas e descritas na Secção 2.1;

Para além das funcionalidades que os sistemas partilham ou em que diferem, existem ainda outras diferenças nos resultados que advêm do facto de os dados armazenados nas bases de dados de ambos os sistemas terem sido recolhidos de textos processados por diferentes ferramentas de PLN (STRING e PALAVRAS), o que faz com que o processo origine uma diferente análise sintática e semântica destes.



(a) Lexicograma para o nome `carro`.

(b) Co-ocorrências para o nome `carro` ordenadas pela medida *PMI*.

Figura 4.4: Co-ocorrências para o nome `carro`.

Na Figura 4.4 (a) é apresentado o resultado da pesquisa para o nome `carro` no *DeepDict*. A partir de uma procura no sistema desenvolvido com o nome `carro`, obteve-se os resultados apresentados na Figura 4.4 (b). Pode-se observar que apesar da organização dos resultados ser diferente, a finalidade é a mesma: permitir ao

utilizador que rapidamente entenda o que é apresentado. É possível observar que os resultados apresentados não diferem muito.

No que diz respeito ao facto de os textos terem sido processados por diferentes ferramentas PLN, pode-se observar que no caso dos adjetivos que modificam o nome, no projeto desenvolvido a co-ocorrência (`carro,alegórico`), não aparece no resultado, enquanto que no *DeepDict* é a segunda co-ocorrência com maior grau de coesão entre as palavras. Esta diferença verifica-se porque no caso da STRING, esta trata estas duas palavras como um nome composto, o que aparentemente não foi o caso do PALAVRAS. Assim, no sistema desenvolvido as palavras compostas são tratadas como uma única palavra, sendo armazenadas como tal.

Capítulo 5

Conclusão e Trabalhos Futuros

5.1 Conclusão

Ao longo deste documento, apresentou-se a pertinência do uso de ferramentas semelhantes à desenvolvida. Sabendo da importância da exploração de padrões de co-ocorrência para a compreensão (e modelação) do funcionamento da língua, é necessário permitir o fácil acesso a este tipo de dados, de modo a que possam ser analisados tanto por linguistas como por estudantes da língua portuguesa, sobretudo enquanto língua estrangeira. Atualmente, existem para o português várias ferramentas que permitem obter as colocações de uma palavra, dos quais se distinguem o *DeepDict*, o *Sketch Engine* e *Wortschatz*. No caso dos dois primeiros, estes permitem uma análise mais detalhada dos padrões de co-ocorrência, enquanto que a última apenas lista aqueles que são mais relevantes para uma determinada palavra.

A solução aqui apresentada pretende ser distinta dos sistemas acima referidos por se basear no processamento realizado pela STRING. Nesse sentido, os recursos lexicais, bem como algumas diferenças no processamento sintático, deverão permitir extrair informação relevante que aqueles sistemas não permitem. Assim, foi criada uma ferramenta que extrai as co-ocorrências presentes em textos processados por esta, sendo detetadas com base nas dependências sintáticas extraídas pela cadeia. Estas são posteriormente armazenadas numa base de dados. O grau de associação de cada co-ocorrência entre duas palavras é quantificado com base num conjunto de medidas estatísticas de associação.

É disponibilizada uma interface *web* para permitir o acesso a esta informação, a fim de permitir uma análise dos padrões armazenados a partir dos textos processados pela STRING. Isto permite obter um conjunto de co-ocorrências para uma palavra-alvo, sendo possível observar informação detalhada acerca de cada co-ocorrência, juntamente com algumas das frases onde esta ocorreu.

A avaliação do projeto foi dividida em duas partes: armazenamento de co-ocorrências e avaliação do desempenho da aplicação *web* que fornece a interface. Os resultados foram obtidos a partir da extração de co-ocorrências do *corpus* CETEMPúblico processado pela STRING. O tempo de execução da ferramenta de extração manteve-se constante ao longo do *corpus*, enquanto que o tamanho da base de dados não teve crescimento linear, o que indica que não se está a repetir informação. Em relação ao tempo de resposta da aplicação *web*, os utilizadores podem

aceder aos resultados de forma praticamente instantânea. No geral, os resultados apresentam-se bons.

Em relação ao sistema desenvolvido e às metas que foram propostas no início, pode-se observar que estas foram praticamente todas atingidas, sendo por isso possível aceder rapidamente às co-ocorrências resultantes de *corpora* processado pela STRING de modo a produzir resultados que os sistemas acima referidos não permitem, como por exemplo a análise de nomes compostos.

5.2 Trabalho Futuro

Nesta secção são apontadas algumas pistas para o futuro deste projeto. As principais sugestões para trabalho são:

- Aumentar a informação disponível sobre uma palavra, isto é, para além de se armazenar apenas o lema, armazenar também a a forma flexionada em que esta ocorreu, de forma a perceber quais as diferentes utilizações de palavras associadas ao mesmo lema;
- Aumentar o número de *corpora* presente na base de dados para permitir uma procura mais ampla, com resultados distintos consoante o *corpus*;
- Com base nos metadados disponíveis para alguns *corpora* (data, tipo textual/género/domínio ou a sua origem), permitir que a ferramenta possibilite comparar os padrões de co-ocorrência da mesma palavra em *corpora* de natureza diferentes, a fim de ser possível estudar melhor os contextos que determinam padrões específicos (em função, por exemplo, do domínio) bem como a evolução desses padrões ao longo do tempo;
- Criar *thesauri* para cada palavra a partir da informação sobre as co-ocorrências armazenadas na base de dados, adicionando essa funcionalidade à aplicação *Web*. Assim, seria possível analisar as palavras mais próximas (sinónimos) e contrárias (antónimos) produzidas a partir da análise da STRING;
- Permitir que o utilizador possa introduzir os seus textos, sendo estes processados pela STRING, extraídas as co-ocorrências e armazená-las na base de dados, para posteriormente os explorar na interface.

Referências

[1] Aıt-Mokhtar, S., Chanod, J.-P., and Roux, C. (2002). Robustness beyond shallowness: Incremental deep parsing. *Natural Language Engineering*, 8:121–144.

[2] Baroni, M. and Bernardini, S. (2004). Bootcat: Bootstrapping corpora and terms from the web. In *In Proceedings of LREC 2004*, pages 1313–1316.

[3] Baroni, M., Kilgarriff, A., Pomikálek, J., and Rychly, P. (2006). WebBootCaT: instant domain-specific corpora to support human translators. *Proceedings of EAMT*, pages 247–252.

[4] Bick, E. (2000). *The Parsing System "Palavras". Automatic Grammatical Analysis of Portuguese in a Constraint Grammar Framework*. Aarhus University Press, Arhus.

[5] Bick, E. (2005). Gramática Constritiva na Análise Automática de Sintaxe Portuguesa. In Berber Sardinha, T., editor, *A Língua Portuguesa no Computador [The Portuguese Language on the Computer]*, Campinas: Mercado de Letras, São Paulo:FAPESP.

[6] Bick, E. (2009). DeepDict - A Graphical Corpus-based Dictionary of Word Relations. In *Proceedings of NODALIDA 2009. NEALT Proceedings Series*, volume 4, pages 268–271. Tartu: Tartu University Library.

[7] Biemann, C., Bordag, S., Heyer, G., Quasthoff, U., and Wolff, C. (2004). Language-independent methods for compiling monolingual lexical data. In *Proceedings of CicLING*, pages 215–228. Springer.

[8] Carapinha, F. (2013). Extração automática de conteúdos documentais. Master's thesis, Instituto Superior Técnico, Universidade Técnica de Lisboa.

[9] Chen, P. (1976). The entity-relationship model—toward a unified view of data. *ACM Transactions on Database Systems*, 1(1):9–36.

[10] Christ, O., Schulze, B., and König, E. (1999). *Corpus Query Processor (CQP). User's Manual*. Institut für Maschinelle Sprachverarbeitung, Universität Stuttgart, Stuttgart, Germany.

[11] Church, K. and Hanks, P. (1990). Word Association Norms, Mutual Information, and Lexicography. *Computational Linguistics*, 16(1):22–29.

[12] Codd, E. (1983). A relational model of data for large shared data banks. *Commun. ACM*, 26(6):64–69.

[13] Dice, L. (1945). Measures of the Amount of Ecologic Association Between Species. *Ecology*, 26(3):297–302.

[14] Diniz, C., Mamede, N., and Pereira, J. (2010). RuDriCo2 - A Faster Disambiguator and Segmentation Modifier. In *INFORUM II*, pages 573–584.

[15] Dunning, T. (1993). Accurate methods for the statistics of surprise and coincidence. *Computational Linguistics*, 19(1):61–74.

[16] Forney, G. (1973). The Viterbi Algorithm. In *Proceedings of the IEEE*, volume 61, pages 268–278. IEEE.

[17] Jakubíček, M., Kilgarriff, A., McCarthy, D., and Rychly, P. (2010). Fast Syntactic Searching in Very Large Corpora for Many Languages. In *Proceedings of the 24th PACLIC*, pages 741–747.

[18] Karlsson, F., Voutilainen, A., Heikkila, J., and Anttila, A. (1995). *Constraint Grammar, A Language-independent System for Parsing Unrestricted Text*. Mouton de Gruyter.

[19] Kilgarriff, A., Baisa, V., Bušta, J., Jakubíček, M., Kovář, V., Michelfeit, J., Rychly, P., and Suchomel, V. (2014). The Sketch Engine: ten years on. *Lexicography*, 1(1):7–36.

[20] Kilgarriff, A., Rychly, P., Tugwell, D., and Smrz, P. (2004). The Sketch Engine. In *Proceedings of Euralex*, volume Demo Session, pages 105–116, Lorient, France.

[21] Mamede, N., Baptista, J., Diniz, C., and Cabarrão, V. (2012). STRING: An Hybrid Statistical and Rule-Based Natural Language Processing Chain for Portuguese. In *PROPOR 2012*, volume Demo Session.

[22] Mamede, N., Baptista, J., and Hagège, C. (2013). Nomenclature of Chunks and Dependencies in Portuguese XIP Grammar 4.0. Technical report, L2F/INESC-ID, Lisboa.

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

[24] Nobre, N. (2011). Resolução de expressões anafóricas. Master's thesis, Instituto Superior Técnico, Universidade Técnica de Lisboa.

[25] Quasthoff, U., Richter, M., and Biemann, C. (2006). Corpus Portal for Search in Monolingual Corpora. In *Proceedings of the 5th LREC*, pages 1799–1802.

[26] Ribeiro, R. (2003). Anotação morfossintática desambiguada do português. Master's thesis, Instituto Superior Técnico, Universidade Técnica de Lisboa.

[27] Rychly, P. (2007). Manatee/Bonito – A Modular Corpus Manager. In P. Sojka and A. Horák, editors, *RASLAN 2008*, pages 65–70, Brno. Masaryk University.

[28] Rychly, P. (2008). A Lexicographer-friendly Association Score. In *RASLAN 2008*, pages 6–9, Brno. Masarykova Univerzita.

[29] Santos, D. and Rocha, P. (2001). Evaluating CETEMPúblico, a Free Resource for Portuguese. In *Proceedings of the 39th Annual Meeting of ACL*, ACL '01, pages 450–457, Stroudsburg, PA, USA. Association for Computational Linguistics.

[30] Silberschatz, A., Korth, H., and Sudarshan, S. (2010). *Database System Concepts*. Connect, learn, succeed. McGraw-Hill Education.

[31] Smadja, F., McKeown, K., and Hatzivassiloglou, V. (1996). Translating collocations for bilingual lexicons: A statistical approach. *Computational Linguistics*, 22(1):1–38.

[32] Vicente, A. (2013). LexMan: um Segmentador e Analisador Morfológico com Transdutores. Master's thesis, Instituto Superior Técnico, Universidade Técnica de Lisboa.